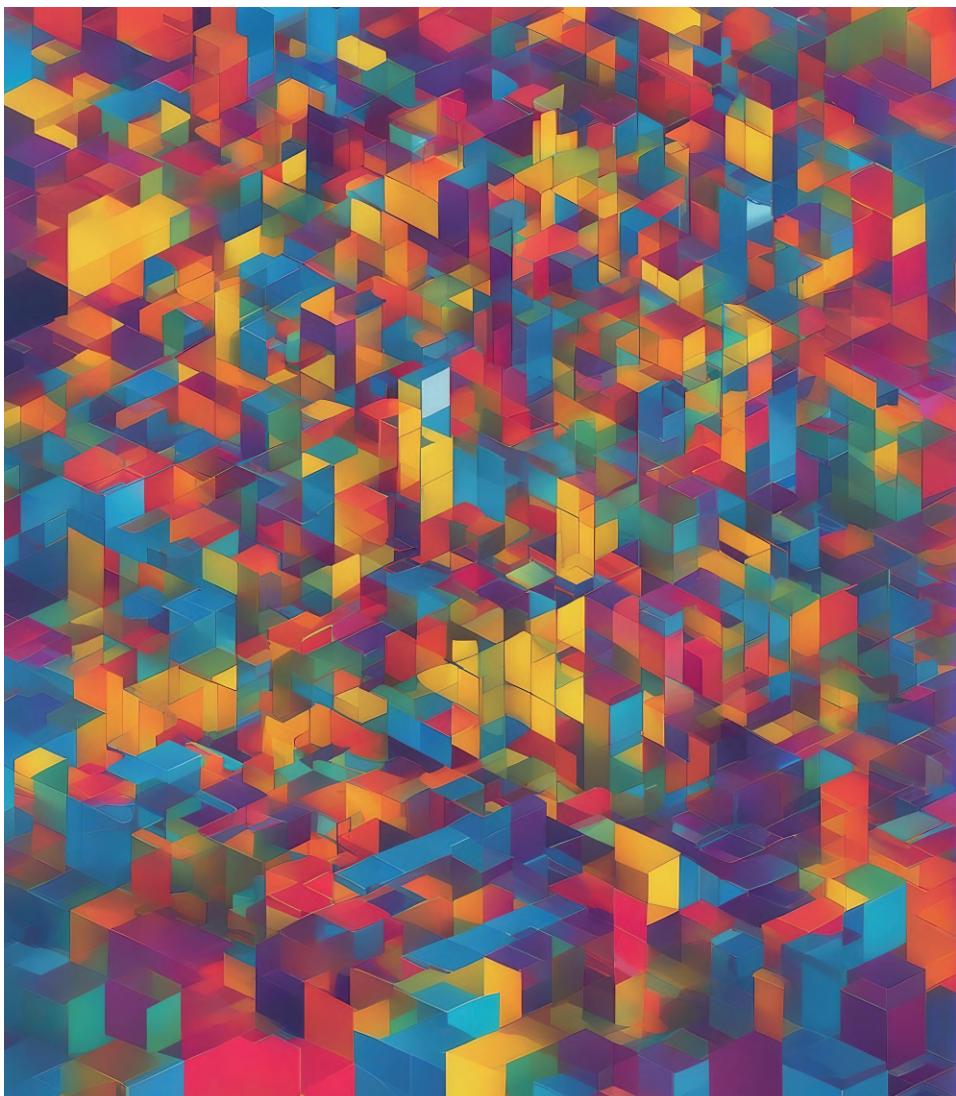


# Towards Accurate and Reliable Deep Regression Models



Fredrik K. Gustafsson

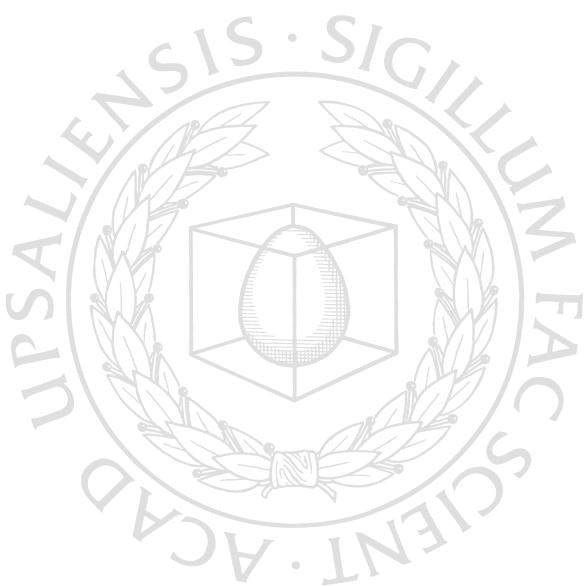


UPPSALA  
UNIVERSITET

*Digital Comprehensive Summaries of Uppsala Dissertations  
from the Faculty of Science and Technology 2320*

# Towards Accurate and Reliable Deep Regression Models

FREDRIK K. GUSTAFSSON



ACTA UNIVERSITATIS  
UPSALIENSIS  
2023

ISSN 1651-6214  
ISBN 978-91-513-1925-4  
urn:nbn:se:uu:diva-513727



UPPSALA  
UNIVERSITET

Dissertation presented at Uppsala University to be publicly examined in 101121 Sonja Lyttkens, Ångströmlaboratoriet, Lägerhyddsvägen 1, Uppsala, Thursday, 30 November 2023 at 09:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Søren Hauberg (Technical University of Denmark, Section for Cognitive Systems).

## Abstract

Gustafsson, F. K. 2023. Towards Accurate and Reliable Deep Regression Models. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 2320. 61 pp. Uppsala: Acta Universitatis Upsaliensis. ISBN 978-91-513-1925-4.

Regression is a fundamental machine learning task with many important applications within computer vision and other domains. In general, it entails predicting continuous targets from given inputs. Deep learning has become the dominant paradigm within machine learning in recent years, and a wide variety of different techniques have been employed to solve regression problems using deep models. There is however no broad consensus on how deep regression models should be constructed for best possible accuracy, or how the uncertainty in their predictions should be represented and estimated.

These open questions are studied in this thesis, aiming to help take steps towards an ultimate goal of developing deep regression models which are both accurate and reliable enough for real-world deployment within medical applications and other safety-critical domains.

The first main contribution of the thesis is the formulation and development of energy-based probabilistic regression. This is a general and conceptually simple regression framework with a clear probabilistic interpretation, using energy-based models to represent the true conditional target distribution. The framework is applied to a number of regression problems and demonstrates particularly strong performance for 2D bounding box regression, improving the state-of-the-art when applied to the task of visual tracking.

The second main contribution is a critical evaluation of various uncertainty estimation methods. A general introduction to the problem of estimating the predictive uncertainty of deep models is first provided, together with an extensive comparison of the two popular methods ensembling and MC-dropout. A number of regression uncertainty estimation methods are then further evaluated, specifically examining their reliability under real-world distribution shifts. This evaluation uncovers important limitations of current methods and serves as a challenge to the research community. It demonstrates that more work is required in order to develop truly reliable uncertainty estimation methods for regression.

**Keywords:** Machine Learning, Deep Learning, Regression, Probabilistic Models, Energy-Based Models, Uncertainty Estimation

*Fredrik K. Gustafsson, Department of Information Technology, Division of Systems and Control, Box 337, Uppsala University, SE-75105 Uppsala, Sweden. Department of Information Technology, Artificial Intelligence, Box 337, Uppsala University, SE-75105 Uppsala, Sweden.*

© Fredrik K. Gustafsson 2023

ISSN 1651-6214

ISBN 978-91-513-1925-4

URN urn:nbn:se:uu:diva-513727 (<http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-513727>)

*To all hard-working and kind-hearted people out in the world.*



# Populärvetenskaplig sammanfattning

Maskininlärning går ut på att man samlar in någon typ av data, att man anpassar en modell till den insamlade datan, och att man slutligen använder denna modell för att förutse och prediktera diverse saker. Man kan säga att maskininlärning handlar om att skapa *data-drivna prediktionsmodeller*, som sedan kan användas för att automatiskt lösa olika typer av praktiska problem.

De flesta problem som studeras inom maskininlärning kan kategoriseras som antingen klassificerings- eller regressionsproblem. I ett klassificeringsproblem kan modellen enbart välja bland en diskret mängd av olika alternativ när den skapar sina prediktioner. Att ge en binär prediktion (ja eller nej) för huruvida det kommer att regna imorgon är ett typiskt exempel på ett klassificeringsproblem. Ett annat exempel är att avgöra om en viss given bild föreställer antingen en katt, hund eller fågel.

I ett regressionsproblem är prediktionerna istället kontinuerliga värden. Det kan röra sig om att prediktera hur mycket ett visst hus kommer kosta när det säljs, att prediktera hur stor elförbrukning ett visst hushåll kommer ha under nästa år, eller att prediktera avståndet till de fotgängare som befinner sig i närheten av en självkörande bil.

Denna avhandling studerar hur maskininlärning ska användas för att på bästa möjliga sätt kunna lösa just olika typer av regressionsproblem. Mer specifikt så studeras hur en särskild typ av maskininlärningsmodeller, som kallas djupa neurala nätverk, kan användas i detta syfte. Ett djupt neutralt nätverk är egentligen bara en helt vanlig matematisk funktion, men med en speciell struktur och ett stort antal parametrar som kan justeras för att anpassa funktionen till insamlad data.

Detta kan liknas vid en rät linje  $y = kx + m$ . Denna räta linje beskriver en funktion från  $x$  till  $y$ , och har två parametrar i form av  $k$  (linjens lutning) och  $m$  (vid vilket värde som linjen skär y-axeln). Genom att ändra värdet på dessa parametrar  $k$  och  $m$  så får helt olika räta linjer, och därmed även helt olika funktioner från  $x$  till  $y$ . Ett djupt neutralt nätverk har istället tusentals eller till och med miljontals parametrar, men grundprincipen är fortfarande

densamma: om värdet på dessa parametrar justeras så fås helt olika funktioner, och om de justeras på ett smart sätt så kan funktionen fås att passa bra till insamlad data.

Djupa neurala nätverk kan användas på många olika sätt för att lösa regressionsproblem, och tidigare forskning har också skapat många olika typer av ”djupa” regressionsmodeller på detta vis. Det råder dock ingen bred enighet kring hur dessa djupa regressionsmodeller bör skapas för att generera så bra prediktioner som möjligt, och inte heller kring hur osäkerheten i dessa prediktioner bör uppskattas eller anges.

Dessa specifika frågor är vad som studeras i denna avhandling. Dess syfte är att bidra till att steg tas mot en framtid där djupa regressionsmodeller är tillräckligt säkra och tillförlitliga för att kunna användas inom sjukvården och andra säkerhetskritiska områden.

Avhandlingens första huvudsakliga bidrag är utvecklingen av en generell och konceptuellt enkel metod för att skapa djupa regressionsmodeller. Denna metod kallas *energibaserad probabilistisk regression*. Metoden tillämpas för att lösa ett antal olika typer av regressionsproblem, och visar sig fungera särskilt bra för att automatiskt spåra olika objekt i bildsekvenser.

Det andra huvudsakliga bidraget är en *kritisk utvärdering av olika metoder som används för att uppskatta osäkerheten* i de prediktioner som skapas av djupa regressionsmodeller. Först ges en övergripande introduktion till hur osäkerheten i djupa modellers prediktioner kan uppskattas, tillsammans med en noggrann empirisk jämförelse av två särskilt populära metoder.

Ett antal andra metoder för osäkerhetsuppskattning utvärderas sedan ytterligare, där deras tillförlitlighet testas särskilt utförligt. Denna utvärdering blottlägger viktiga brister och begränsningar hos nuvarande metoder, och utgör därmed en sorts utmaning för andra forskare. Utvärderingen visar nämligen att mer forskningsarbete kommer krävas för att kunna skapa verkligt säkra och tillförlitliga djupa regressionsmodeller.

## Acknowledgements

First of all, I want to thank my supervisor Thomas Schön. Your guidance and feedback throughout this entire journey has been incredibly valuable. Sending you that email in April 2018 has turned out to be a fantastic decision. Secondly, I want to express how grateful I am for my co-supervisor Martin Danelljan. Thank you for all your hands-on support with the writing of manuscripts and rebuttals, the planning, implementation and analysis of various experiments, and all the other practical things one encounters along the years of a PhD. Thomas and Martin, you have been great mentors and played a huge role in my development from student to researcher. I will always be in debt to the both of you, all I can do is try to pay it forward to the next generation of PhD students.

I also want to thank all my other co-authors, with a special thanks to the first authors Johannes H and Philipp VB. Although the papers are not included in this thesis, Thomas and I have also had a great collaboration with Ziwei L, Zheng Z and Jens S over the past year. Ziwei, I am very impressed by your drive, creativity and technical skills. Thank you also to Taro L, Melanie D and Jordan M for other fun collaborations. It has been a pleasure working together with all of you. Moreover, I want to thank the anonymous reviewers who have provided feedback on different versions of our papers. While the review process can be frustrating at times, their constructive feedback has undoubtedly helped strengthen the papers and improve the clarity of our writing.

The division of Systems and Control has been a really nice place to work throughout the years – thank you to all the current and former colleagues who have contributed to creating a friendly and welcoming environment. Daniel G, thank you for being a great co-author, teaching assistant colleague, reading group co-organizer and office roommate. Andreas L, Calle A, Calle J, Anna W, David W, Antônio R, Niklas W and Fredrik O, thank you for making me feel at home right from the beginning when I joined the division. Ludvig H, thanks for the interesting and helpful discussions, in particular during the work on Paper VII. Håkan R, thank you for all the fun running-related discussions. Hans R and Per M, I always really enjoyed teaching in your courses. Paul H, Linnéa G, David S, Philipp P, and everyone who joined the reading group at some point over the years, thank you for the interesting papers and discussions. M Osama

and Bernhard W, thank you for being friendly office roommates. Thank you to Bengt C for the work as head of division, Sofia E for handling the seminars, and Sebastian M for the newsletters. To Niklas G, Anna F, Ruoqi Z, Maria B, Jennifer A and all other PhD students at the division, best of luck finishing your degrees and in your future endeavours. Lastly, thank you to the administration and support staff at the department for making things run smoothly.

I also want to thank all the friendly runners I have met during my years in Uppsala. You have helped turn running into an important part of my life, crucial in order to keep me productive and in a good mental state throughout the work days and weeks (also, I mentally “wrote” most of this section during a couple of morning runs). Special thanks for the organized Sunday long runs.

I would probably not have even pursued a PhD in the first place had it not been for my thoroughly enjoyable years as a student in Linköping. Thus, thank you to all lecturers who contributed to the high-quality education, and to all the good friends I made along the way. Daniel A, what a pleasant surprise that our paths crossed again at least for a couple of months here in Uppsala. Also, thank you for the help with putting together this thesis. Beth, thank you for taking such incredibly good care of me and the rest of “the Four Swedes” during our exchange year. I am also greatly indebted to many teachers from my early school years. Special thanks to Eric Windhede at Nils Ericsonsgymnasiet, Frank Bergström at Lyrfägelskolan, and to everyone who helped make Åsaka skola such a safe, happy and welcoming place.

Finally, I want to thank my parents Lena and Sture. Your unwavering support and encouragement throughout all these years has been invaluable. This thesis would definitely not have been possible without you.

*Uppsala, October 2023  
Fredrik K. Gustafsson*

## Funding

My research was financially supported by the Swedish Foundation for Strategic Research via the project *ASSEMBLE* (contract number: RIT15-0012); by the Swedish Research Council via the projects *Learning Flexible Models for Nonlinear Dynamics* (contract number: 2017-03807), *NewLEADS – New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) and *Deep Probabilistic Regression – New Models and Learning Algorithms* (contract number: 2021-04301); and by the *Kjell & Märta Beijer Foundation*.

# List of Papers

This thesis includes the following eight papers:

- I **Energy-Based Models for Deep Probabilistic Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, Thomas B. Schön.* The European Conference on Computer Vision (ECCV), 2020.
- II **How to Train Your Energy-Based Model for Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Radu Timofte, Thomas B. Schön.* The British Machine Vision Conference (BMVC), 2020.
- III **Learning Proposals for Practical Energy-Based Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The International Conference on Artificial Intelligence and Statistics (AISTATS), 2022.
- IV **Accurate 3D Object Detection using Energy-Based Models.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2021.
- V **Deep Energy-Based NARX Models.** *Johannes Hendriks, Fredrik K. Gustafsson, Antônio Ribeiro, Adrian Wills, Thomas B. Schön.* The 19th IFAC Symposium on System Identification (SYSID), 2021.
- VI **Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2020.
- VII **How Reliable is Your Regression Model's Uncertainty Under Real-World Distribution Shifts?** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* Transactions on Machine Learning Research (TMLR), 2023.
- VIII **ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.** *Philipp Von Bachmann, Daniel Gedon, Fredrik K. Gustafsson, Antônio H. Ribeiro, Erik Lampa, Stefan Gustafsson, Johan Sundström, Thomas B. Schön.* In Preparation, 2023.



# Contents

1	Introduction .....	13
1.1	Motivating Example .....	14
1.2	Contributions & Thesis Outline .....	15
1.3	Papers Included in the Thesis .....	17
1.4	Related but not Included Papers .....	26
1.5	Chronology of the Included Papers .....	26
2	Learning to Solve Supervised Regression Problems .....	29
2.1	Supervised Regression Problems .....	29
2.2	A First Simple Machine Learning Method .....	29
2.3	Machine Learning with Parametric Models .....	31
2.4	Deep Learning Approaches .....	33
3	Energy-Based Probabilistic Regression .....	37
3.1	Background on Energy-Based Models .....	37
3.2	Formulation .....	38
3.3	Prediction .....	39
3.4	Training .....	39
3.5	Practical Limitations .....	42
4	Uncertainty Estimation .....	45
4.1	Predictive Uncertainty Estimation using Bayesian Deep Learning	45
4.2	Aleatoric Uncertainty .....	45
4.3	Epistemic Uncertainty .....	46

4.4	Illustrative Example .....	48
5	Concluding Remarks .....	51
5.1	Conclusion .....	51
5.2	Future Work .....	52
	References .....	53

# 1

## Introduction

Supervised machine learning problems entail collecting examples of how an input  $x$  relates to some target  $y$ , fitting a model to the collected data  $\{(x_i, y_i)\}_{i=1}^N$ , and then using this model to output predicted targets for other, previously unseen inputs  $x$  [1, 2]. The model often contains parameters  $\theta$  which explicitly determine how inputs  $x$  are mapped onto predicted targets  $y$ . Fitting this model to the collected data then means finding the parameter values that minimize a certain loss function, thereby “learning” a model from data. If accurate enough, such data-driven prediction models can be utilized to automatically perform various types of practical tasks.

To create accurate data-driven prediction models, deep learning has emerged as the dominant paradigm within machine learning during the past decade [3, 4]. It involves the use of so-called neural networks, which are models containing a large number of parameters  $\theta$  with a hierarchical structure of multiple hidden layers. These “deep models” are capable of extracting predictive yet compact feature representations even for high-dimensional inputs  $x$ , for example images and text files, enabling various applications within complex tasks such as image recognition and natural language processing.

All supervised machine learning problems can be further categorized into either classification or regression problems. In classification, the targets  $y$  only take on values in a discrete set of labels. Simple examples include making a binary prediction of whether or not it will rain tomorrow, or classifying if a given image depicts either a cat, dog or a bird. In regression problems, the targets  $y$  are instead continuous values, e.g.  $y \in \mathbb{R}$ . Examples include predicting house prices (e.g. given features such as house size, location and construction year), predicting electric power consumption based on historical data, or predicting the 3D position of all pedestrians in the surroundings of an autonomous vehicle (e.g. given camera and radar sensor measurements).

Regression is thus a fundamental machine learning task, with many important applications within computer vision and other domains, but still remains some-

what understudied compared to classification. While classification problems generally are addressed using standardized target representations and loss functions, these are not directly applicable to regression. Consequently, a wide variety of different techniques have been employed to solve regression problems, and there is no broad consensus on how to construct deep regression models for best possible accuracy, or how to represent and estimate the uncertainty in their predictions.

This thesis studies these open questions, aiming to take steps towards an ultimate goal of developing deep regression models which are both *accurate* and *reliable* enough for deployment within medical applications and other safety-critical domains<sup>1</sup>.

## 1.1 Motivating Example

Within medical imaging, a number of tasks are naturally formulated as regression problems, including brain age estimation [7, 8, 9], prediction of cardiovascular volumes and risk factors [10, 11] and body composition analysis [12, 13]. If machine learning models could be deployed to automatically regress various such properties within real-world clinical practice, this would ultimately help lower costs and improve patient outcomes across the medical system [14].

As a specific example that will be studied later in the thesis (in Paper VIII), let us consider the problem of ECG-based electrolyte prediction. Given an electrocardiogram (ECG), which measures the electrical activity of the heart of a person, the task is to predict the potassium concentration level in the person’s body. Abnormal potassium levels can lead to serious heart conditions, and if the concentration level could be monitored using an ECG-based regression model, potentially life-threatening conditions could thus be avoided.

For such a regression model to be useful in practice, it first of all needs to be capable of producing highly *accurate* predictions. The predicted concentration levels must be close to the true values, and even very small changes in the true concentration must be reflected in a corresponding change in the prediction. While a model that produces correct but coarse predictions, e.g. a model that classifies whether the concentration level is above or below a certain threshold, would be somewhat useful, it would fail to enable crucial use-cases such as early detection of abnormal changes, or monitoring that the concentration decreases rapidly enough after administration of medication for hyperkalemia (high potassium).

---

<sup>1</sup>The image on the cover of this thesis was generated by a diffusion-based text-to-image generative model [5, 6], from the text input *A beautifully colorful image to put on the cover of a PhD thesis titled “Towards Accurate and Reliable Deep Regression Models”*.

Moreover, the regression model needs to be *reliable*. If the model would start to output completely incorrect predictions for some patients once it is deployed in a certain hospital (e.g., a patient’s true concentration level is abnormally high, but the model predicts a value well within the normal range), this could have catastrophic consequences. During such real-world deployment, the model is however bound to occasionally encounter input ECGs  $x$  for which the model can not be expected to output entirely correct predictions. For example, it would be inherently difficult for any model (or human) to accurately predict the potassium level from an ECG that is severely corrupted by measurement noise. A reliable model must be able to detect such cases (enabling doctors to take appropriate action, e.g. acquiring a new ECG), instead of “failing silently” and simply output a prediction that is far from the true value.

This issue would be solved by a regression model that, along with a predicted target  $y$ , also outputs an estimate of the *uncertainty* in its prediction for each input  $x$ . When encountering inputs for which an accurate prediction can not be expected, the model could then “fail loudly” by outputting a large uncertainty. While such models still would output incorrect predictions sometimes, it would also indicate to users that those predictions should not be trusted. These uncertainty estimates themselves must however also be accurate and reliable. Otherwise, if the model occasionally becomes overconfident and outputs highly confident (low uncertainty) yet incorrect predictions, providing these uncertainty estimates might just instill users with a false sense of security – arguably making the model even less suitable for practical deployment.

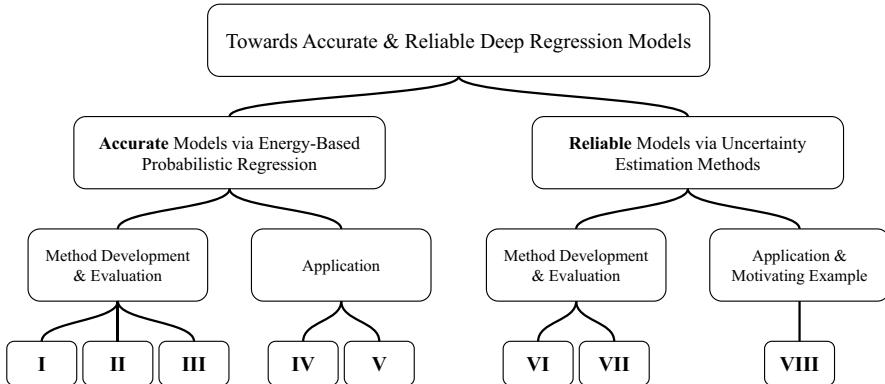
In summary, real-world deployment within safety-critical domains puts very high requirements on deep regression models, in terms of a range of different aspects related to both accuracy and reliability. In this thesis, some of these aspects are studied in detail.

## 1.2 Contributions & Thesis Outline

This thesis is divided into two parts. The first part contains five chapters (including the current one), which are intended to provide background information and an overview for the eight included papers. These eight papers constitute the second part of the thesis, representing the scientific contributions.

The main general contribution of this work is method development and evaluation that helps to take steps towards the ultimate goal of developing deep regression models which are *accurate* and *reliable* enough for real-world deployment within safety-critical domains.

The eight included papers can in turn be divided into two main tracks. The first track consists of **Paper I - Paper V**, and focuses on how to develop *accurate*



**Figure 1.1:** Conceptual overview of the eight papers included in the thesis (Paper I - Paper VIII), which can be divided into two main tracks. The first track consists of Paper I - Paper V, and focuses on how to develop *accurate* deep regression models. The second track consists of Paper VI - Paper VIII, and focuses on how to develop *reliable* deep regression models. Both tracks include both application-oriented papers, and papers entailing more fundamental method development and evaluation.

deep regression models. Specifically, it focuses on energy-based probabilistic regression – using energy-based models to accurately represent the true conditional target distribution  $p(y|x)$ . Paper I - Paper III propose a general regression framework and evaluate different method variations. This approach is then applied to two specific applications in Paper IV & V.

The second track consists of **Paper VI - Paper VIII**, and focuses on how to develop *reliable* deep regression models via uncertainty estimation. A critical evaluation of various uncertainty estimation methods is conducted in Paper VI & VII, uncovering important limitations. Paper VIII applies some of the evaluated methods to the task of ECG-based electrolyte prediction, and also serves as the primary motivating example for much of the work in Paper VI & VII.

Both tracks thus include both application-oriented papers, and papers entailing more fundamental method development and evaluation. The two tracks of papers are conceptually illustrated in Figure 1.1.

The remainder of the current chapter presents a summary of the eight included papers, along with some background on how they originated and evolved into their final form. Next, Chapter 2 provides a general introduction to how various machine learning techniques can be utilized to solve regression problems from data. It constitutes relevant background for all eight included papers. Chapter 3 then provides a further introduction specifically for the first track of Paper I - Paper V, whereas Chapter 4 introduces the second track of Paper VI - Paper VIII. Chapter 5, which is meant to be read after the papers, finally

### 1.3. Papers Included in the Thesis

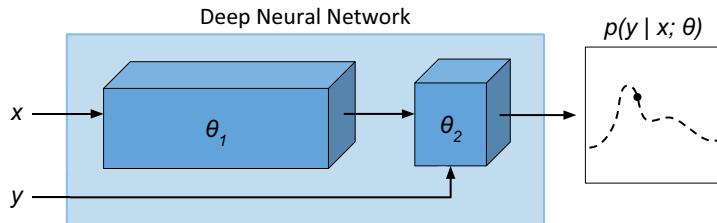
contains some concluding remarks and reflections, along with an outlook on possible future work.

## 1.3 Papers Included in the Thesis

The eight papers included in this thesis are briefly summarized next, together with a short explanation of my individual contributions.

## Paper I

**Energy-Based Models for Deep Probabilistic Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, Thomas B. Schön.* The European Conference on Computer Vision (ECCV), 2020.

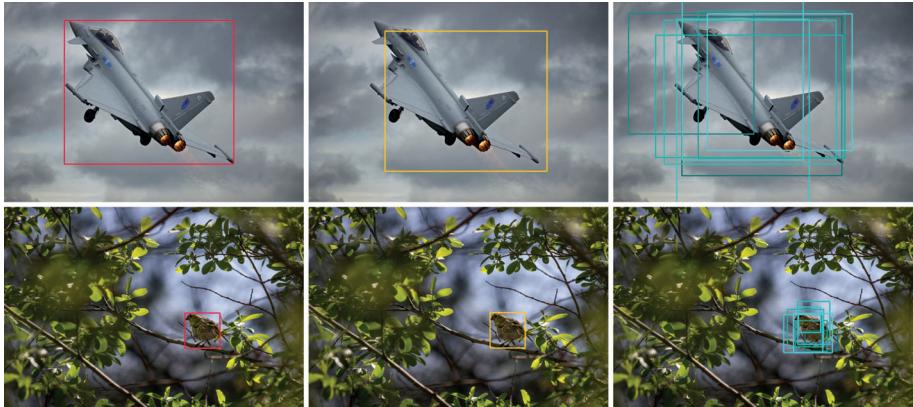


**Summary** We propose a general and conceptually simple regression method with a clear probabilistic interpretation. We create an energy-based model of the conditional target density  $p(y|x)$ , using a deep neural network to predict the un-normalized density from the input-target pair  $(x, y)$ . This model of  $p(y|x)$  is trained by directly minimizing the associated negative log-likelihood, approximated using Monte Carlo sampling. Notably, our model achieves a 2.2% average precision (AP) improvement over Faster-RCNN for object detection on the COCO dataset, and sets a new state-of-the-art on visual tracking when applied for bounding box regression.

**Statement of Contribution** Martin came up with the original underlying idea of the paper. I refined the original idea into the energy-based model formulation in discussions with Martin and Thomas (using the feedback provided by anonymous reviewers on a previous version of the paper). Martin conducted the visual tracking experiments. Goutam conducted the object detection experiments. I conducted the remaining experiments and did the majority of the writing, with a lot of feedback from Martin and Thomas.

## Paper II

**How to Train Your Energy-Based Model for Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Radu Timofte, Thomas B. Schön.* The British Machine Vision Conference (BMVC), 2020.

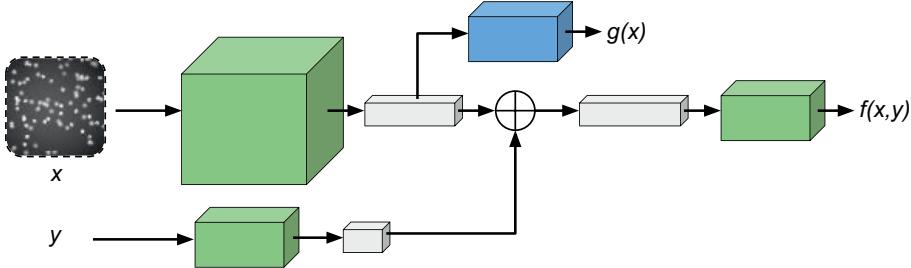


**Summary** We propose a simple yet highly effective extension of noise contrastive estimation (NCE) to train energy-based models  $p(y|x; \theta)$  for regression tasks. Our proposed method NCE+ can be understood as a direct generalization of NCE, accounting for noise in the annotation process of real-world datasets. We provide a detailed comparison of NCE+ and six popular methods from the literature, the results of which suggest that NCE+ should be considered the go-to training method. We also apply NCE+ to the task of visual tracking, achieving state-of-the-art performance on five commonly used datasets. Notably, our tracker achieves 63.7% AUC on LaSOT and 78.7% success on TrackingNet.

**Statement of Contribution** I came up with the underlying idea of the paper. Martin conducted the visual tracking experiments. I conducted the remaining experiments and did the majority of the writing, with feedback from the other authors.

**Paper III**

**Learning Proposals for Practical Energy-Based Regression.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The International Conference on Artificial Intelligence and Statistics (AISTATS), 2022.

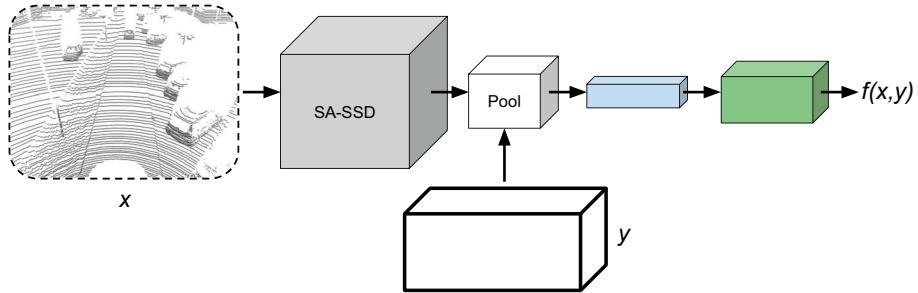


**Summary** We derive an efficient and convenient objective that can be employed to train a parameterized distribution  $q(y|x; \phi)$  by directly minimizing its KL divergence to a conditional energy-based model (EBM)  $p(y|x; \theta)$ . We then employ the proposed objective to jointly learn an effective mixture density network (MDN) proposal distribution during EBM training, thus addressing the main practical limitations of energy-based regression. Furthermore, we utilize our derived training objective to learn MDNs with a jointly trained energy-based teacher, consistently outperforming conventional MDN training on four real-world regression tasks within computer vision.

**Statement of Contribution** I came up with the underlying idea of the paper in discussions with Martin and Thomas, after having found initial empirical evidence of Result 1. Martin came up with the first mathematical derivation of Result 1. I conducted all experiments and did the majority of the writing, with feedback from Martin and Thomas.

**Paper IV**

**Accurate 3D Object Detection using Energy-Based Models.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2021.

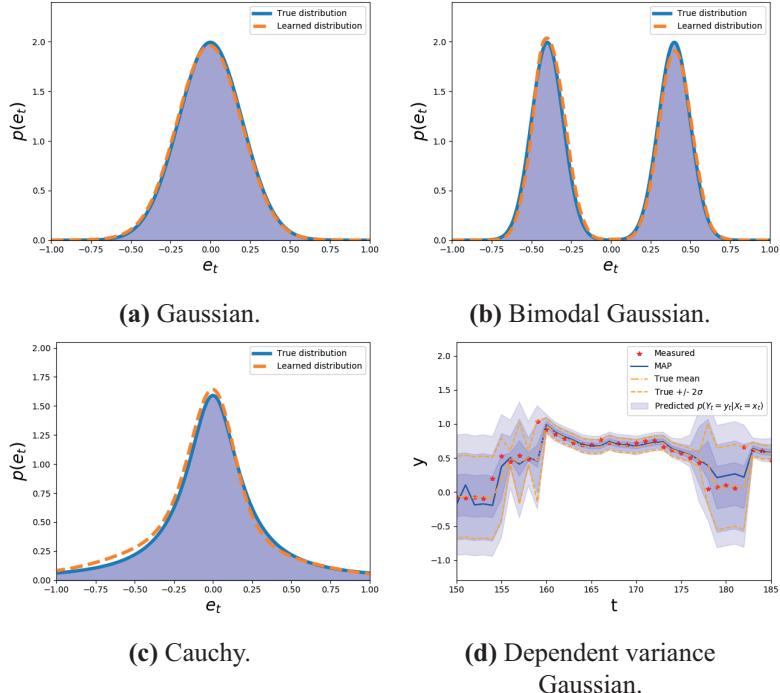


**Summary** We apply energy-based models  $p(y|x; \theta)$  to the task of 3D bounding box regression, extending the recent energy-based regression approach from 2D to 3D object detection. This is achieved by designing a differentiable pooling operator for 3D bounding boxes  $y$ , and adding an extra network branch to the state-of-the-art 3D object detector SA-SSD. We evaluate our proposed detector on the KITTI dataset and consistently outperform the SA-SSD baseline, demonstrating the potential of energy-based models for 3D object detection.

**Statement of Contribution** I came up with the underlying idea of the paper, conducted all experiments and did the majority of the writing, with feedback from Martin and Thomas throughout the entire process.

## Paper V

**Deep Energy-Based NARX Models.** Johannes Hendriks, Fredrik K. Gustafsson, Antônio Ribeiro, Adrian Wills, Thomas B. Schön. The 19th IFAC Symposium on System Identification (SYSID), 2021.

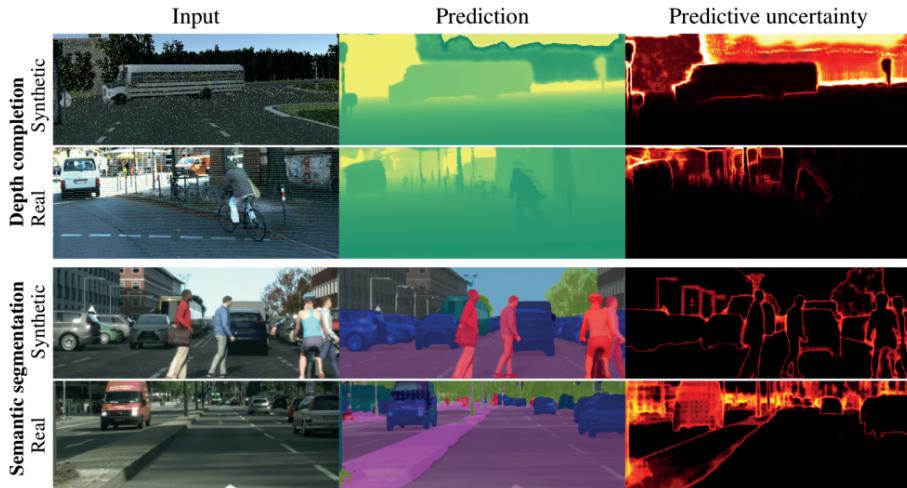


**Summary** We study the problem of learning nonlinear ARX models based on observed input-output data. In particular, we want to learn a conditional distribution of the current output based on a finite window of past inputs and outputs. To achieve this, we consider the use of energy-based models. This energy-based model relies on a general function to describe the distribution, and here we consider a neural network for this purpose. The primary benefit of our approach is that it is capable of learning both simple and highly complex noise models, which we demonstrate on simulated and experimental data.

**Statement of Contribution** Johannes came up with the underlying idea of the paper, conducted all experiments and did the majority of the writing. I discussed the methods and experiments in meetings with Johannes and the other authors. I contributed to the writing of Section 3, and provided detailed feedback on the rest of the manuscript.

## Paper VI

**Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2020.



**Summary** We propose a comprehensive evaluation framework for scalable epistemic uncertainty estimation methods in deep learning. It is specifically designed to test the robustness required in real-world computer vision applications. We also apply our proposed framework to provide the first properly extensive and conclusive comparison of the two current state-of-the-art scalable methods: ensembling and MC-dropout. Our comparison demonstrates that ensembling consistently provides more reliable and practically useful uncertainty estimates.

**Statement of Contribution** I came up with the underlying idea of the paper in discussions with Martin and Thomas. I conducted all experiments and did the majority of the writing, with a lot of feedback from Martin and Thomas.

## Paper VII

**How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?** *Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön.* Transactions on Machine Learning Research (TMLR), 2023.

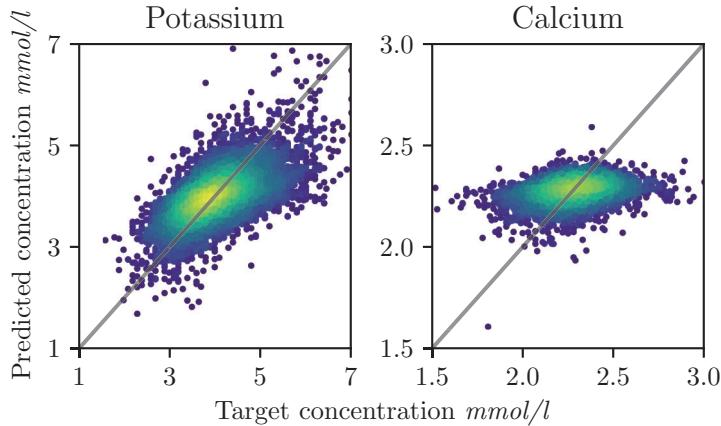
	Cells-Tails	ChairAngle-Gap	AssetWealth	Ventricular Volume	BrainTumour Pixels	SkinLesion Pixels	Histology NucleiPixels	AerialBuilding Pixels
Train								
Test								
	y = 100	y = 80.5	y = 1.594	y = 40.38	y = 252	y = 500	y = 1257	y = 1097
	y = 198	y = 43.8	y = 1.314	y = 85.93	y = 273	y = 516	y = 1156	y = 433

**Summary** We propose a benchmark for testing the reliability of regression uncertainty estimation methods under real-world distribution shifts. It consists of eight image-based regression datasets with different types of challenging distribution shifts. We use our benchmark to evaluate many of the most common uncertainty estimation methods, as well as two state-of-the-art uncertainty scores from OOD detection. While methods are well calibrated when there is no distribution shift, they all become highly overconfident on many of the benchmark datasets. This uncovers important limitations of current uncertainty estimation methods, and our benchmark thus serves as a challenge to the research community.

**Statement of Contribution** I came up with the underlying idea of the paper in discussions with Martin and Thomas. I conducted all experiments and did the majority of the writing, with feedback from Martin and Thomas. The analysis of the results was also extended quite substantially based on feedback by the anonymous reviewers (Figure 4, Figure A1-A3 and Figure A6-A17 were added).

## Paper VIII

**ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.** *Philipp Von Bachmann, Daniel Gedon, Fredrik K. Gustafsson, Antônio H. Ribeiro, Erik Lampa, Stefan Gustafsson, Johan Sundström, Thomas B. Schön.* In Preparation, 2023.



**Summary** Imbalances of the electrolyte concentration levels in the body can lead to catastrophic consequences, but accurate and accessible measurements could improve patient outcomes. While blood tests provide accurate measurements, they are invasive and the laboratory analysis can be slow or inaccessible. In contrast, an ECG is a widely adopted tool which is quick and simple to acquire. However, the problem of estimating continuous electrolyte concentrations directly from ECGs is not well-studied. We therefore investigate if regression methods can be used for ECG-based prediction of electrolyte concentrations.

**Statement of Contribution** Johan and Thomas came up with the initial underlying idea of the paper. The ECG-electrolyte data was provided by Stefan, Johan and Erik, and Antônio helped create the train/test datasets. Philipp conducted all experiments, and did the majority of the writing for the initial manuscript. Daniel and I acted as supervisors for Philipp. Daniel, Philipp and I came up with the detailed problem formulation and planned the experiments, with feedback from the other authors. Daniel and I revised the manuscript based on feedback by anonymous reviewers, after discussions with the other authors.

## 1.4 Related but not Included Papers

In addition to the eight included papers, the following research (to which I have contributed) is also relevant to this thesis:

**Uncertainty-Aware Body Composition Analysis with Deep Regression Ensembles on UK Biobank MRI.** *Taro Langner, Fredrik K. Gustafsson, Benny Avelin, Robin Strand, Håkan Ahlström, Joel Kullberg.* Computerized Medical Imaging and Graphics, Volume 93, 2021.

**Image Restoration with Mean-Reverting Stochastic Differential Equations.** *Ziwei Luo, Fredrik K. Gustafsson, Zheng Zhao, Jens Sjölund, Thomas B. Schön.* The International Conference on Machine Learning (ICML), 2023.

**Refusion: Enabling Large-Size Realistic Image Restoration with Latent-Space Diffusion Models.** *Ziwei Luo, Fredrik K. Gustafsson, Zheng Zhao, Jens Sjölund, Thomas B. Schön.* The IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2023.

**Controlling Vision-Language Models for Universal Image Restoration.** *Ziwei Luo, Fredrik K. Gustafsson, Zheng Zhao, Jens Sjölund, Thomas B. Schön.* arXiv preprint arXiv:2310.01018, 2023.

The first paper is closely connected to the second track of Paper VI - Paper VIII. It applies the approach of Paper VI to a certain medical imaging application, which also serves as an additional motivating example for the included work on reliable deep regression models.

The three remaining papers study the problem of image restoration, which is an example of an interesting regression problem in which clean, high-quality images  $y$  should be predicted from corrupted, low-quality versions  $x$  (e.g. image deblurring, denoising or dehazing).

## 1.5 Chronology of the Included Papers

The eight included papers are not entirely in chronological order, but are instead listed according to the structure illustrated in Figure 1.1. Here, I provide some background on how and in what order the papers originated.

**Paper VI** is the oldest of the included papers. The underlying idea originated already during the work of my MSc thesis [15] on automotive 3D object detection. This application naturally raises questions related to uncertainty estimation, as the predicted 3D position for distant or partially occluded surrounding

vehicles should be inherently more uncertain than for clearly visible vehicles nearby. Thus, we set out to study different uncertainty estimation methods in the context of automotive applications.

Next, we started working on **Paper I**. The original idea came from Martin, which entailed exploring if the IoU-Net [16] approach could be extended to a general probabilistic regression framework. The first version of the paper was entitled “Deep Conditional Target Densities for Accurate Regression” and did not make any references to energy-based models. Based on feedback by anonymous reviewers, and following a more extensive review of previous work, the paper was then refined into its current form.

**Paper II** then followed as a quite natural extension of Paper I. From the review of previous work on energy-based models, it was clear that a wide variety of alternative training methods had been explored. This previous work had however almost exclusively focused on generative modelling, and how to best train energy-based models specifically for the regression setting was therefore an open question.

Next, we applied this work on energy-based regression to the task of 3D object detection in **Paper IV**. I was still interested in this task since my MSc thesis, and was curious to explore if the good performance achieved on object detection and visual tracking (bounding box regression) in Paper I & II also could be extended from 2D to 3D. At roughly the same time, we also explored if the energy-based regression approach could be applied to system identification problems, resulting in **Paper V**.

Following this more application-oriented work, a methodological improvement was proposed in **Paper III**. The underlying idea originated from empirical observations made during the development of Paper I & II, which seemed to indicate that a proposal distribution could be jointly trained together with the energy-based model. We were curious to explore if these observations could be properly understood, and also saw potential practical benefits (not having to manually specify a proposal for training the energy-based model).

Next, I got involved in the work on **Paper VIII**. The underlying idea was to combine previous work by colleagues and collaborators on ECG-based classification tasks [17, 18] with our work on regression. Paired ECG-electrolyte data could be extracted from a previously utilized large-scale dataset, and automatic electrolyte prediction was deemed a clinically impactful application.

Lastly, **Paper VII** was completed shortly after the first version Paper VIII, and much of the work on these two papers was conducted in parallel. I had remained interested in uncertainty estimation techniques and reliability-related issues ever since Paper VI, as they always had struck me as important and intriguing problems. After completing Paper III, I thus decided that it finally was time to return to these topics. Discussions during the work on Paper VIII

## Chapter 1. Introduction

then also triggered a number of questions regarding what requirements such real-world clinical deployment would put on deep regression models. To a large extent, these questions shaped the problem formulation of Paper VII.

# 2

## Learning to Solve Supervised Regression Problems

This chapter provides a general introduction to supervised machine learning, and discusses some of the specific challenges associated with solving regression problems. It is relevant background material for all eight included papers.

### 2.1 Supervised Regression Problems

In a supervised regression problem, the task is to predict a target value  $y^* \in \mathcal{Y}$  for any given input  $x^* \in \mathcal{X}$ . To solve this, we are also given a training set of  $N$  i.i.d. input-target pairs,  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ . What separates regression from classification is that the target space  $\mathcal{Y}$  is continuous,  $\mathcal{Y} = \mathbb{R}^K$ .

In this thesis, I will mostly consider regression problems from the computer vision domain, meaning that the input space  $\mathcal{X}$  often will correspond to the space of images. Other examples will however also be studied, including the space of 3D point clouds (Paper IV) and 12-lead ECGs (Paper VIII). The target space  $\mathcal{Y}$  will vary with the specific regression problem. I will often focus on the 1D case, i.e. when  $\mathcal{Y} = \mathbb{R}$ , but multidimensional target spaces such as  $\mathcal{Y} = \mathbb{R}^4$  and  $\mathcal{Y} = \mathbb{R}^7$  (bounding box regression in 2D and 3D, respectively) will also be considered.

### 2.2 A First Simple Machine Learning Method

One conceptually simple approach for solving supervised regression problems is the  $k$  nearest neighbours (kNN) method. Given a new input  $x^*$ , kNN constructs a prediction  $\hat{y}(x^*)$  by explicitly utilizing the training data  $\{(x_i, y_i)\}_{i=1}^N$ . First, the distance  $\|x_i - x^*\|$  to  $x^*$  is computed for all training inputs  $\{x_i\}_{i=1}^N$ . Next, the  $k$  training inputs  $\{x_{j_1}, \dots, x_{j_k}\}$  with the smallest distance to  $x^*$  are

extracted. Finally, a prediction  $\hat{y}(x^*)$  is computed as the average of the  $k$  corresponding training targets  $\{y_{j_1}, \dots, y_{j_k}\}$ .

Using this simple algorithm, a computer program could then be constructed that is capable of taking any previously unseen  $x^*$  as input and outputting a corresponding prediction  $\hat{y}(x^*)$ . Crucially, what value  $\hat{y}(x^*)$  this computer program outputs for a given input  $x^*$  is entirely determined by the training data  $\{(x_i, y_i)\}_{i=1}^N$ . If a different set of input-target pairs  $\{(x_l, y_l)\}_{l=1}^M$  instead had been provided as training data, the predicted value  $\hat{y}(x^*)$  would also change.

This is perhaps the simplest possible example of supervised machine learning. Instead of writing a conventional computer program, taking  $x^*$  as input and returning  $\hat{y}(x^*)$  based on an explicit set of rules or formulas, the kNN method automatically defines a mapping from input  $x^*$  to prediction  $\hat{y}(x^*)$  based on the provided training data  $\{(x_i, y_i)\}_{i=1}^N$ . Importantly, the kNN method is also generally applicable to any regression task for which such training data can be collected.

As long as examples  $\{(x_i, y_i)\}_{i=1}^N$  of how an input  $x$  relates to some target  $y$  can be collected, for each of the different considered tasks, the same general kNN method could directly be applied to predict both potassium concentration levels from ECGs *and* ventricular volumes from echocardiogram images, for instance. Designing conventional computer programs for two such varied regression problems would instead require significant time and effort to explicitly incorporate highly problem-specific domain knowledge into two entirely different set of rules.

For some regression problems, the relationship between input  $x$  and target  $y$  might also be virtually impossible to describe explicitly, even for highly skilled domain experts. For example, exactly how ECGs relate to electrolyte concentration levels is not fully understood. It is however still possible to pair an ECG  $x$  with a corresponding concentration level  $y$ , by using an alternative measurement method (laboratory analysis of blood tests provides accurate measurements, but is slow and inaccessible compared to ECG-based methods). The general underlying idea of the machine learning approach is thus that by collecting a large number of observed input-target pairs  $\{(x_i, y_i)\}_{i=1}^N$ , and constructing a computer program that defines a mapping from  $x$  to  $y$  entirely based on this data, the relationship between ECG and concentration level will hopefully be “learned” automatically.

Collecting input-target pairs  $\{(x_i, y_i)\}_{i=1}^N$  can of course also be difficult, or at least very time-consuming and expensive, for certain regression problems. Moreover, the number of collected pairs  $N$  might have to be very large in order for the kNN method (or other more advanced machine learning approaches) to perform well in terms of predictive accuracy. All data-driven methods have some inherent limitations, and machine learning should definitely not be ex-

pected to perfectly solve every imaginable regression problem. The general idea, of using observed data to automatically “learn” relationships between inputs  $x$  and targets  $y$ , has however turned out to be extremely powerful. The kNN method is an illustrative example of this general idea.

## 2.3 Machine Learning with Parametric Models

The kNN method described above is an example of a so-called nonparametric machine learning method, since the training data  $\{(x_i, y_i)\}_{i=1}^N$  is explicitly used every time a prediction  $\hat{y}(x^*)$  is computed for a given  $x^*$ . While the kNN method is conceptually very simple, it therefore quickly becomes impractical when the size  $N$  of the training set increases.

A second general machine learning approach is to instead fit a parametric model to the training data. The data  $\{(x_i, y_i)\}_{i=1}^N$  is used once to estimate the parameters of the model in an initial training stage, but can then be discarded. To output a prediction  $\hat{y}(x^*)$  for a new input  $x^*$ , only the estimated parameters are required.

Machine learning can now be described in terms of three major cornerstones: 1) *Training data*. 2) *Model*. 3) *Learning algorithm*. First, training data in the form of observed input-target pairs  $\{(x_i, y_i)\}_{i=1}^N$  is collected. Second, a model containing parameters  $\theta$  is specified. Finally, a learning algorithm is used to find the parameter values  $\theta$  that make the model fit the data as well as possible.

Specifying the model in turn entails two steps. First, a function  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  that is parameterized by  $\theta \in \mathbb{R}^P$  is specified. This function  $f_\theta$  maps inputs  $x \in \mathcal{X}$  to outputs  $f_\theta(x) \in \mathcal{O}$  in some output space  $\mathcal{O}$ . Secondly, a loss function  $\ell(f_\theta(x_i), y_i)$  is chosen, determining what is meant by a “good data fit”.

With a specified function  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  and loss  $\ell(f_\theta(x_i), y_i)$ , a learning algorithm is then utilized to find the optimal parameters  $\theta^*$ ,

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^N \ell(f_\theta(x_i), y_i), \quad (2.1)$$

i.e. to find the parameter values  $\theta$  that minimize the loss function  $\ell(f_\theta(x_i), y_i)$  over the training data  $\{(x_i, y_i)\}_{i=1}^N$ . The learning algorithm is thus used to numerically solve the optimization problem in (2.1).

The simplest and most commonly used approach is to let the function  $f_\theta$  directly output predicted targets,  $\hat{y}(x) = f_\theta(x)$ , and choosing the L2 loss function,  $\ell(f_\theta(x_i), y_i) = \|f_\theta(x_i) - y_i\|_2^2$ . In this case, solving (2.1) to find the optimal parameters  $\theta^*$  thus corresponds to finding the parameters  $\theta$  that minimize the discrepancy between the predictions  $f_\theta(x_i)$  and the true target values  $y_i$ .

Once  $\theta^*$  has been computed, the training data  $\{(x_i, y_i)\}_{i=1}^N$  can be discarded and a prediction  $\hat{y}(x^*) = f_{\theta^*}(x^*)$  can be output for any given  $x^*$ .

This simple approach can however be modified in numerous ways, all leading to different models. First of all, one could replace the L2 loss function with for instance the L1 loss,  $\ell(f_\theta(x_i), y_i) = \|f_\theta(x_i) - y_i\|_1$ , or the Huber loss [19]. This small modification changes the optimization objective in (2.1), resulting in different optimal parameters  $\theta^*$  and thus also different predictions  $\hat{y}(x^*) = f_{\theta^*}(x^*)$ . Secondly, the internal structure of the parameterized function  $f_\theta$  allows for a range of customization options (e.g., changing the number of layers if  $f_\theta$  is defined in terms of a neural network), all of which also result in different predictions  $\hat{y}(x^*) = f_{\theta^*}(x^*)$ . Finally, instead of letting  $f_\theta$  directly output predicted targets, this function could be designed to, for example, map inputs  $x$  to the mean and variance of a Gaussian distribution. That is, the output space  $\mathcal{O}$  of the function  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  could also be modified. Just this second machine learning cornerstone of specifying a model thus entails a large number of different possible design choices.

Both of the two other cornerstones – collecting training data  $\{(x_i, y_i)\}_{i=1}^N$ , and using a learning algorithm to find  $\theta^*$  in (2.1) – can also significantly impact the performance of machine learning methods when deployed in real-world applications. They are both active research topics and I definitely do not consider them to be solved problems. Many of the related issues (for example, how to best construct large and high-quality training datasets, or how to efficiently find good approximate solutions to the optimization problem (2.1)) are however common to both regression and classification problems.

In contrast, the *2) Model* cornerstone involves a number of open questions which are specific for regression. While classification problems generally are addressed using a standardized approach, letting  $f_\theta$  map inputs  $x$  to class logit scores and using the cross-entropy loss function, previous work has explored a range of different output spaces  $\mathcal{O}$  and loss functions  $\ell(f_\theta(x_i), y_i)$  for regression problems [20, 21, 22, 23, 16, 24]. In this thesis, I have focused almost exclusively on this second cornerstone of how to specify the model.

For the *1) Training data*, I have in this thesis either directly used or modified existing public datasets, or created simple synthetic datasets for illustrative examples. The only exception is Paper VIII in which we utilize internal datasets of real-world ECG-electrolyte pairs. I was however not directly involved in the collection or curation of these datasets. For the *3) Learning algorithm*, I have always just used existing stochastic gradient-based optimizers such as ADAM [25].

In summary, I have in this thesis focused on how to best specify the model, i.e. on how the function  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  and corresponding loss function  $\ell(f_\theta(x_i), y_i)$  should be designed. The training dataset  $\{(x_i, y_i)\}_{i=1}^N$  has mostly been consid-

ered given, and common gradient-based optimizers have then been utilized to (approximately) solve the resulting optimization problem in (2.1). Specifically, I have focused on approaches for specifying the model which utilize deep learning, as described next.

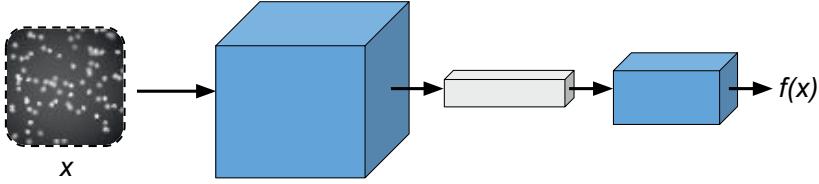
## 2.4 Deep Learning Approaches

Deep learning entails using a certain general class of parameterized functions to specify  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$ , so-called deep neural networks (DNNs). These DNN functions have a hierarchical structure with multiple layers, and typically contain a large number of parameters  $\theta$  (often in the range of, at least, millions of parameters). They are capable of modelling intricate relationships in the data and of extracting predictive yet compact feature representations, also for high-dimensional inputs  $x$  such as images, 3D point clouds or 12-lead ECGs.

DNNs can thus be used to automatically extract features from the inputs  $x$ , i.e. to “learn” predictive feature representations directly from the data. During the past decade or so, this entirely learning-based approach has been shown to significantly outperform the traditional method – building machine learning models on top of various hand-crafted feature representations – across a wide range of applications within computer vision, natural language processing and other domains.

The exact internal structure or architecture of the DNNs  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  can vary considerably, and numerous variations have been presented in the literature. In this thesis, however, I have not put much focus on what specific DNN architecture one should use. Instead, I have mostly just viewed DNNs as high-level parameterized functions  $f_\theta$  that map inputs  $x$  to some type of output  $f_\theta(x)$ . The regression methods which are studied in this thesis are in general entirely independent of the specific choice of low-level DNN architecture.

Thus, I simply view a DNN as a function  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$ . I typically also divide the DNN  $f_\theta$  into a *backbone feature extractor*, and one or more smaller *network heads*. The feature extractor takes  $x$  as input and outputs a feature vector  $g(x) \in \mathbb{R}^L$  of length  $L$ . This feature vector  $g(x)$  is then fed into the network heads (typically consisting of a few fully-connected layers), producing the final function output  $f_\theta(x) \in \mathcal{O}$ . If the inputs  $x$  are images, the backbone feature extractor can be specified by, e.g., taking a DNN used for image classification and removing its final softmax layer. In the thesis I have often used ResNet [26] variants, as they are widely used across various applications, yet simple to implement and quite computationally inexpensive. The modularity of this approach, separating the backbone feature extractor from the network heads, does however enable the use of any other backbone architecture (that takes  $x$  as input and extracts a feature vector  $g(x)$ ) as well.



**Figure 2.1:** Illustration of the most straightforward, deep direct regression approach. The input  $x$  is fed into a backbone DNN (the first blue box) that extracts a feature vector  $g(x) \in \mathbb{R}^L$  (the gray box). This feature vector is then fed into a single small network head (the second blue box), directly outputting a predicted target  $\hat{y}(x) = f_\theta(x)$  for the input  $x$ . The visualized input is from a regression problem where, given a synthetic microscopy image  $x$ , the task is to predict the number of cells  $y$  in the image.

The most common and straightforward approach, called *direct regression* [20], is to let the DNN directly output predicted targets,  $\hat{y}(x) = f_\theta(x)$ . This entails feeding the input  $x$  to the feature extractor, and then feeding the feature vector into a single network head with a linear final layer, outputting  $\hat{y}(x) = f_\theta(x)$ . This direct regression approach is illustrated in Figure 2.1. The DNN is “trained” by (approximately) finding the optimal model parameters  $\theta^*$  in (2.1), for example using the L2 or L1 loss function for  $\ell(f_\theta(x_i), y_i)$ .

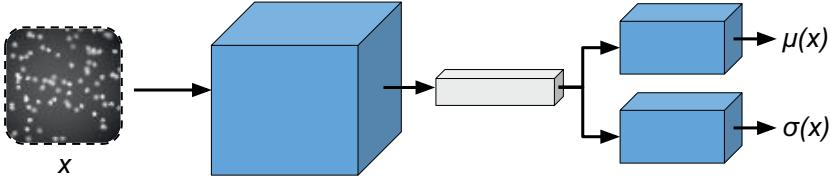
From a probabilistic perspective, the choice of loss function corresponds to minimizing the negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$  for a specific model  $p(y|x; \theta)$  of the conditional target distribution  $p(y|x)$ . For example, the L2 loss  $\ell(f_\theta(x_i), y_i) = \|f_\theta(x_i) - y_i\|_2^2$  is derived from a fixed-variance Gaussian model,  $p(y|x; \theta) = \mathcal{N}(y; f_\theta(x), \sigma^2 I)$ . For the 1D case, this can be seen from,

$$\begin{aligned}
 \theta^* &= \arg \min_{\theta} -\log p(y_i|x_i; \theta) \\
 &= \arg \min_{\theta} -\log \mathcal{N}(y_i; f_\theta(x_i), \sigma^2) \\
 &= \arg \min_{\theta} -\log \left( \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(f_\theta(x_i) - y_i)^2}{2\sigma^2} \right) \right) \\
 &= \arg \min_{\theta} (f_\theta(x_i) - y_i)^2.
 \end{aligned} \tag{2.2}$$

Similarly, the L1 loss can be derived from a fixed-variance Laplace distribution. By using the L2 or L1 loss functions, one is thus implicitly using quite restrictive models  $p(y|x; \theta)$ , which might fail to accurately represent the true target distribution  $p(y|x)$  in various scenarios.

## Probabilistic Regression

This probabilistic perspective can be extended and used to define a general approach for deep regression: Use a DNN  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  to specify a model



**Figure 2.2:** Illustration of the deep probabilistic regression approach with a general 1D Gaussian model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ . The DNN outputs both the Gaussian mean and variance as  $f_\theta(x) = [\mu_\theta(x) \ \sigma_\theta^2(x)]^\top \in \mathbb{R}^2$ , by feeding the feature vector into two separate network heads.

$p(y|x; \theta)$  of the conditional target distribution, and minimize the corresponding negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$  in order to train the DNN. In this thesis, I have called this approach *probabilistic regression*.

Previous work has utilized this probabilistic regression approach to achieve more flexible parametric models  $p(y|x; \theta) = p(y; \phi_\theta(x))$ , by letting the DNN  $f_\theta$  output the parameters  $\phi$  of a family of probability distributions  $p(y; \phi)$  [27, 28, 29, 21, 22, 30, 31]. For example, a general 1D Gaussian model can be realized as  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , where the DNN outputs both the mean and variance as  $f_\theta(x) = \phi_\theta(x) = [\mu_\theta(x) \ \sigma_\theta^2(x)]^\top \in \mathbb{R}^2$ . Minimizing the negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$  for this model is equivalent to minimizing the following loss  $J(\theta)$ ,

$$J(\theta) = \sum_{i=1}^N \ell(f_\theta(x_i), y_i) = \sum_{i=1}^N \frac{(y_i - \mu_\theta(x_i))^2}{\sigma_\theta^2(x_i)} + \log \sigma_\theta^2(x_i), \quad (2.3)$$

which is used to train the DNN  $f_\theta$ . To output both the mean  $\mu_\theta(x)$  and variance  $\sigma_\theta^2(x)$ , a second network head can be added to the DNN, as illustrated in Figure 2.2. The output variance  $\sigma_\theta^2(x)$  can also be taken as a natural estimate of the aleatoric uncertainty (inherent noise and ambiguities in the data itself) in the prediction, a topic I will return to in more detail in Chapter 4.

A general Gaussian model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$  is however still quite restrictive, in the sense that it is unable to capture e.g. multi-modal or asymmetric true distributions  $p(y|x)$ . To address this, previous work has for example used mixture density networks (MDNs) [32] to specify the model  $p(y|x; \theta)$  [23, 33, 34]. An MDN is a mixture of  $K$  components of a certain base distribution. Specifically for a Gaussian MDN, the model  $p(y|x; \theta)$  is defined according to,

$$p(y|x; \theta) = \sum_{k=1}^K \pi_\theta^{(k)}(x) \mathcal{N}(y; \mu_\theta^{(k)}(x), \sigma_\theta^{(k)}(x)), \quad (2.4)$$

where the set of Gaussian mixture parameters  $\{\pi_\theta^{(k)}, \mu_\theta^{(k)}, \sigma_\theta^{(k)}\}_{k=1}^K$  is output by the DNN  $f_\theta$ . For example, a third network head can be added to the DNN in order to output the mixture weights  $\{\pi_\theta^{(k)}\}_{k=1}^K$ .

One potential issue with using MDNs to specify the model  $p(y|x; \theta)$  is that the number of mixture components  $K$  needs to be pre-determined before training. Previous work has thus also used infinite mixture models by applying the conditional VAE (cVAE) framework [35, 36]. A latent variable model,

$$p(y|x; \theta) = \int p(y; \phi_\theta(x, z))p(z; \phi_\theta(x))dz, \quad (2.5)$$

is then employed to specify  $p(y|x; \theta)$ , where  $p(y; \phi_\theta(x, z))$  and  $p(z; \phi_\theta(x))$  typically are Gaussian distributions.

Alternatively, one can instead use energy-based models (EBMs) [37] to specify the model  $p(y|x; \theta)$ . EBMs are not restricted to the functional form of any specific distribution (e.g. Gaussian or Laplace) and, in contrast to MDNs and cVAEs, are not even limited to distributions which are simple to generate samples from. How EBMs can be used for probabilistic regression is described in detail in Chapter 3.

### Alternative Approaches

Another category of approaches reformulates the regression problem as  $\hat{y}(x) = \arg \max_y f_\theta(x, y)$ , where  $f_\theta(x, y) \in \mathbb{R}$  is a scalar confidence value predicted by the DNN [38, 39, 40, 41, 16, 42]. The idea is thus to predict a quantity  $f_\theta(x, y)$ , depending on both input  $x$  and target  $y$ , that can be maximized over  $y$  to obtain the final prediction  $\hat{y}(x)$  for a given  $x$ . This line of research was the main inspiration for our work on energy-based probabilistic regression.

A regression problem can also be treated as a classification problem by first discretizing the target space  $\mathcal{Y}$  into a finite set of  $C$  classes. Standard techniques from classification, such as softmax and the cross-entropy loss, can then be employed [43, 44, 45, 46, 24, 47, 48].

For a more detailed description of these two categories of alternative regression approaches, and how they relate to energy-based probabilistic regression, I refer to Section 2 in Paper I.

# 3

## Energy-Based Probabilistic Regression

This chapter provides a further introduction specifically for the first track of **Paper I - Paper V**, which focuses on how to develop *accurate* deep regression models via energy-based probabilistic regression. This entails using energy-based models to accurately represent the true conditional target distribution  $p(y|x)$ . The chapter mostly focuses on Paper I - Paper III, as they propose the general regression approach which then is applied to two specific applications in Paper IV & V.

### 3.1 Background on Energy-Based Models

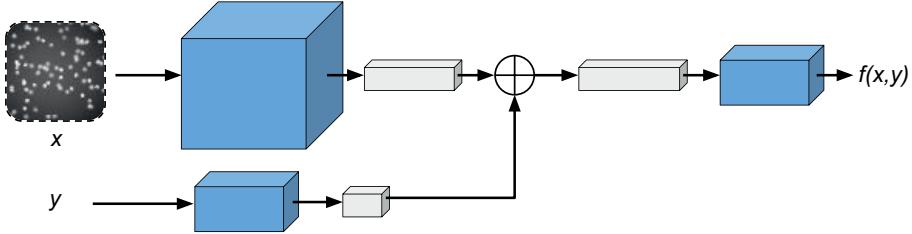
Energy-based models (EBMs) [37] have a rich history within the field of machine learning [49, 50, 51, 52, 53]. In general, an EBM specifies a probability distribution  $p(x; \theta)$  over inputs  $x \in \mathcal{X}$  directly via a parameterized scalar function  $f_\theta : \mathcal{X} \rightarrow \mathbb{R}$ ,

$$p(x; \theta) = \frac{e^{f_\theta(x)}}{Z(\theta)}, \quad Z(\theta) = \int e^{f_\theta(\tilde{x})} d\tilde{x}, \quad (3.1)$$

where  $Z(\theta)$  is the so-called normalizing partition function.

By defining the scalar energy function  $f_\theta(x)$  using a DNN [54], the EBM  $p(x; \theta)$  becomes expressive enough to learn practically any distribution from observed data  $\{x_i\}_{i=1}^N$ . EBMs have therefore experienced a significant resurgence in recent years, commonly being employed for various generative modelling tasks [55, 57, 58, 59, 60, 61, 62, 63, 64, 56].

EBMs  $p(x; \theta) = e^{f_\theta(x)} / \int e^{f_\theta(\tilde{x})} d\tilde{x}$  are highly expressive models, designed to put minimally restricting assumptions on the true distribution  $p(x)$ . However, they come with the significant drawback that the partition function  $Z(\theta) = \int e^{f_\theta(\tilde{x})} d\tilde{x}$  generally is intractable, which complicates evaluating or sampling



**Figure 3.1:** Illustration of a DNN  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  used for energy-based probabilistic regression, specifying the conditional EBM  $p(y|x; \theta)$  in (3.2). The DNN maps any input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  to a scalar  $f_\theta(x, y) \in \mathbb{R}$ . A feature vector is extracted for both the input  $x$  and target  $y$ , these feature vectors are then combined into a single vector and fed into a network head, that finally outputs  $f_\theta(x, y) \in \mathbb{R}$ .

from the EBMs  $p(x; \theta)$ . It can therefore be illustrative to contrast EBMs with normalizing flows [65, 66, 67], which is another class of generative models. While normalizing flows are specifically designed to be simple to both evaluate and sample, EBMs instead prioritize maximum model expressivity.

## 3.2 Formulation

While EBMs recently had become increasingly popular within computer vision when we started working on Paper I, they were basically only being employed for generative modeling tasks. In Paper I, we therefore explored if EBMs could be applied also to solve supervised regression problems. Specifically, we formulated the *energy-based probabilistic regression* approach, which entails using a conditional EBM to specify a model  $p(y|x; \theta)$  of the conditional target distribution,

$$p(y|x; \theta) = \frac{e^{f_\theta(x, y)}}{Z(x, \theta)}, \quad Z(x, \theta) = \int e^{f_\theta(x, \tilde{y})} d\tilde{y}, \quad (3.2)$$

where  $Z(x, \theta)$  is an input-dependent partition function. The conditional EBM  $p(y|x; \theta)$  is thus directly specified via  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ , a DNN that maps any input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  to a scalar  $f_\theta(x, y) \in \mathbb{R}$ .

For general regression problems, the DNN  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  can be defined by extending the basic architecture illustrated in Figure 2.1. A backbone feature extractor is still used to produce a feature vector  $g(x) \in \mathbb{R}^L$  for the input  $x$ . A small network head is then added to extract a feature vector  $h(y) \in \mathbb{R}^{L'}$  also for the target  $y$ . These two feature vectors are then combined into one (e.g. via concatenation) and fed into a second network head, that finally outputs the scalar  $f_\theta(x, y) \in \mathbb{R}$ . This DNN architecture is illustrated in Figure 3.1.

### 3.3 Prediction

At test-time, the problem of predicting a target value  $y^*$  from an input  $x^*$  corresponds to finding a point estimate of the predicted conditional distribution  $p(y|x^*; \theta)$ . In Paper I & II, a prediction  $y^*$  is output by finding the most likely target under the model,  $y^* = \arg \max_y p(y|x^*; \theta)$ :

$$y^* = \arg \max_y p(y|x^*; \theta) = \arg \max_y \frac{e^{f_\theta(x^*, y)}}{Z(x^*, \theta)} = \arg \max_y f_\theta(x^*, y). \quad (3.3)$$

The prediction  $y^*$  is thus obtained by directly maximizing the DNN scalar output  $f_\theta(x^*, y)$  w.r.t.  $y$ . This does not require the partition function  $Z(x^*, \theta)$  to be evaluated, nor any samples from  $p(y|x^*; \theta)$  to be generated.

In practice,  $y^* = \arg \max_y f_\theta(x^*, y)$  is approximated by refining an initial estimate  $\hat{y}$  via  $T$  steps of gradient ascent,

$$y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y), \quad (3.4)$$

thus finding a local maximum of  $f_\theta(x^*, y)$ . This prediction procedure is further detailed in Algorithm 1, where  $\lambda$  denotes the gradient ascent step-length,  $\eta$  is a decay of the step-length and  $T$  is the number of iterations.

---

**Algorithm 1** Prediction via gradient-based refinement.

---

**Input:**  $x^*, \hat{y}, T, \lambda, \eta$ .

```

1:  $y \leftarrow \hat{y}$ .
2: for  $t = 1, \dots, T$  do
3:    $\text{PrevValue} \leftarrow f_\theta(x^*, y)$ .
4:    $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ .
5:    $\text{NewValue} \leftarrow f_\theta(x^*, \tilde{y})$ .
6:   if  $\text{NewValue} > \text{PrevValue}$  then
7:      $y \leftarrow \tilde{y}$ .
8:   else
9:      $\lambda \leftarrow \eta \lambda$ .
10:  Return  $y$ .

```

---

### 3.4 Training

The DNN  $f_\theta(x, y)$  that specifies the conditional EBM  $p(y|x; \theta) = e^{f_\theta(x, y)} / \int e^{f_\theta(x, \tilde{y})} d\tilde{y}$  can in principle be trained using the standard probabilistic regression approach, i.e. by minimizing the corresponding negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$ . For the conditional EBM  $p(y|x; \theta)$ ,

the negative log-likelihood  $\mathcal{L}(\theta)$  is given by,

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{i=1}^N -\log p(y_i|x_i; \theta) \\ &= \sum_{i=1}^N -\log \frac{e^{f_\theta(x_i, y_i)}}{\int e^{f_\theta(x_i, y)} dy} \\ &= \sum_{i=1}^N \log \left( \int e^{f_\theta(x_i, y)} dy \right) - f_\theta(x_i, y_i).\end{aligned}\tag{3.5}$$

While the integral in (3.5) is intractable, preventing exact evaluation of  $\mathcal{L}(\theta)$ , it can be approximated using importance sampling. Specifically,  $\mathcal{L}(\theta)$  can be approximated using  $M$  samples  $\{y_i^{(m)}\}_{m=1}^M \sim q(y)$  drawn from a proposal distribution  $q(y)$ ,

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{i=1}^N \log \left( \int e^{f_\theta(x_i, y)} dy \right) - f_\theta(x_i, y_i) \\ &= \sum_{i=1}^N \log \left( \int \frac{e^{f_\theta(x_i, y)}}{q(y)} q(y) dy \right) - f_\theta(x_i, y_i) \\ &\approx \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{q(y_i^{(m)})} \right) - f_\theta(x_i, y_i).\end{aligned}\tag{3.6}$$

This approximate negative log-likelihood in (3.6) was used to train the DNN  $f_\theta(x, y)$  in Paper I.

Various alternative techniques had however been utilized to train EBMs for generative modeling tasks in previous work, including noise contrastive estimation (NCE) [68, 69], score matching [70, 71, 72] and Markov chain Monte Carlo (MCMC) [73, 60, 59, 74]. How EBMs best should be trained specifically for regression problems remained an open question, which therefore was studied in detail in Paper II. There, six different training methods were compared on the task of 2D bounding box regression in images, concluding that a simple extension of NCE should be considered the go-to approach.

## Noise Contrastive Estimation

As its general principle, NCE entails learning to discriminate between observed data examples and samples drawn from a noise distribution. NCE had been used to train EBMs for classification tasks in the past [75, 76, 77, 78], and had recently also become highly utilized within self-supervised representation learning [79, 80, 81, 82].



**Figure 3.2:** Illustration of the NCE loss  $J_{\text{NCE}}(\theta)$  in (3.7) for a 2D bounding box regression example. NCE here entails learning to discriminate between the true bounding box target  $y_i^{(0)} \triangleq y_i$  (left image) and  $M$  bounding boxes  $\{y_i^{(m)}\}_{m=1}^M$  (right image) which have been sampled from a noise distribution  $q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I)$ .

Using NCE to train the DNN  $f_\theta(x, y)$  of the conditional EBM  $p(y|x; \theta)$  entails minimizing the loss  $J_{\text{NCE}}(\theta)$ ,

$$\begin{aligned} J_{\text{NCE}}(\theta) &= -\frac{1}{N} \sum_{i=1}^N J_{\text{NCE}}^{(i)}(\theta), \\ J_{\text{NCE}}^{(i)}(\theta) &= \log \frac{\exp \{f_\theta(x_i, y_i^{(0)}) - \log q(y_i^{(0)})\}}{\sum_{m=0}^M \exp \{f_\theta(x_i, y_i^{(m)}) - \log q(y_i^{(m)})\}}, \end{aligned} \quad (3.7)$$

where  $y_i^{(0)} \triangleq y_i$ , and  $\{y_i^{(m)}\}_{m=1}^M$  are  $M$  samples drawn from a noise distribution  $q(y)$ . Effectively,  $J_{\text{NCE}}(\theta)$  in (3.7) is the softmax cross-entropy loss for a classification problem with  $M + 1$  classes: Among the  $M + 1$  values  $\{y_i^{(m)}\}_{m=0}^M$ , correctly classify  $y_i^{(0)}$  as the true target  $y_i$ .

A simple choice for the noise distribution  $q(y)$ , that was shown effective in Paper II, is using a mixture of  $K$  Gaussians centered at the true target  $y_i$ ,

$$q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I), \quad (3.8)$$

where  $K$  (the number of mixture components) and the variances  $\{\sigma_k^2\}_{k=1}^K$  are hyperparameters. The NCE loss  $J_{\text{NCE}}(\theta)$  thus entails learning to discriminate between the true target value  $y_i$  and  $M$  values  $\{y_i^{(m)}\}_{m=1}^M \sim q(y)$  sampled around  $y_i$ . This is illustrated for a 2D bounding box regression example in Figure 3.2, showing  $y_i$  (red bounding box) in the left image and  $\{y_i^{(m)}\}_{m=1}^M$  (blue boxes) in the right image.

### 3.5 Practical Limitations

In Paper I, the DNN  $f_\theta(x, y)$  of the conditional EBM  $p(y|x; \theta)$  was trained by approximating the negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$  using importance sampling according to (3.6). This corresponds to minimizing the following loss  $J(\theta)$ ,

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{q(y_i^{(m)})} \right) - f_\theta(x_i, y_i), \quad (3.9)$$

$$\{y_i^{(m)}\}_{m=1}^M \sim q(y) \text{ (proposal distribution).}$$

In Paper II, the DNN was instead trained by minimizing the NCE loss  $J_{\text{NCE}}(\theta)$ ,

$$J_{\text{NCE}}(\theta) = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp \{f_\theta(x_i, y_i^{(0)}) - \log q(y_i^{(0)})\}}{\sum_{m=0}^M \exp \{f_\theta(x_i, y_i^{(m)}) - \log q(y_i^{(m)})\}}, \quad (3.10)$$

$$y_i^{(0)} \triangleq y_i, \quad \{y_i^{(m)}\}_{m=1}^M \sim q(y) \text{ (noise distribution).}$$

In both cases, training thus requires samples  $\{y_i^{(m)}\}_{m=1}^M$  to be drawn from a proposal/noise distribution  $q(y)$ . In both cases, the distribution  $q(y)$  was also set to a mixture of  $K$  Gaussian components centered at the true target  $y_i$ ,

$$q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I). \quad (3.11)$$

Consequently, the distribution  $q(y)$  contains hyperparameters  $K$  and  $\{\sigma_k^2\}_{k=1}^K$ , which need to be tuned for each specific regression problem. Moreover,  $q(y)$  depends on the true target  $y_i$  and can therefore only be utilized during training. While the refinement-based method for producing predictions  $y^*$  at test-time that is employed in Paper I & II (Algorithm 1) is accurate, it does require access to good initial estimates  $\hat{y}$ .

Paper III aims to address both of these limitations, by jointly learning a parameterized proposal/noise distribution  $q(y|x; \phi)$  during EBM training. This is achieved by deriving a convenient objective that can be employed to train  $q(y|x; \phi)$  by directly minimizing its Kullback–Leibler (KL) divergence to the EBM  $p(y|x; \theta)$ .

Since  $q(y|x; \phi)$  is conditioned only on the input  $x$ , it can be utilized also at test-time. Moreover, as  $q(y|x; \phi)$  has been trained to approximate the EBM  $p(y|x; \theta)$ , it can be used with self-normalized importance sampling [83] to ef-

ficiently approximate expectations  $\mathbb{E}_p$  w.r.t. the EBM  $p(y|x; \theta)$ ,

$$\begin{aligned}\mathbb{E}_p[\xi(y)] &= \int \xi(y)p(y|x; \theta)dy \approx \sum_{m=1}^M w^{(m)}\xi(y^{(m)}), \\ w^{(m)} &= \frac{e^{f_\theta(x, y^{(m)})}/q(y^{(m)}|x; \phi)}{\sum_{l=1}^M e^{f_\theta(x, y^{(l)})}/q(y^{(l)}|x; \phi)}.\end{aligned}\tag{3.12}$$

By setting  $\xi(y) = y$ , (3.12) can be used to approximately compute the mean value of the EBM  $p(y|x; \theta)$ . In this manner, a stand-alone prediction  $y^*$  for the EBM can thus be produced.



# 4

## Uncertainty Estimation

This chapter provides a further introduction specifically for the second track of **Paper VI - Paper VIII**, which focuses on how to develop *reliable* deep regression models via uncertainty estimation.

### 4.1 Predictive Uncertainty Estimation using Bayesian Deep Learning

DNNs  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  have become the go-to approach within computer vision and many other domains due to their impressive predictive power compared to previous approaches. However, they generally fail to properly capture the uncertainty inherent in their predictions. Estimating this predictive uncertainty can be crucial, for example in medical and automotive applications.

The approach of *Bayesian deep learning* aims to address this issue in a principled manner [84, 27]. It deals with predictive uncertainty by decomposing it into the distinct types of aleatoric and epistemic uncertainty. *Aleatoric* uncertainty captures inherent and irreducible ambiguity in the data, while *epistemic* uncertainty accounts for uncertainty in the DNN model parameters  $\theta$ .

### 4.2 Aleatoric Uncertainty

Given an input  $x$ , it is not always obvious what the correct target value  $y$  should be. For example, if a DNN has been trained to classify if images  $x$  contain either a cat, dog or a bird, how should it classify an image that contains *both* a cat and a dog? How about images with very low brightness, in which it is difficult to recognize any objects at all? Or, what prediction  $\hat{y}(x)$  should a DNN trained for ECG-based electrolyte regression output for an ECG  $x$  that is



**Figure 4.1:** Example of *aleatoric* uncertainty in the task of street-scene semantic segmentation [85, 86]. Image pixels right at object boundaries are inherently more difficult to classify than pixels in the middle of objects.

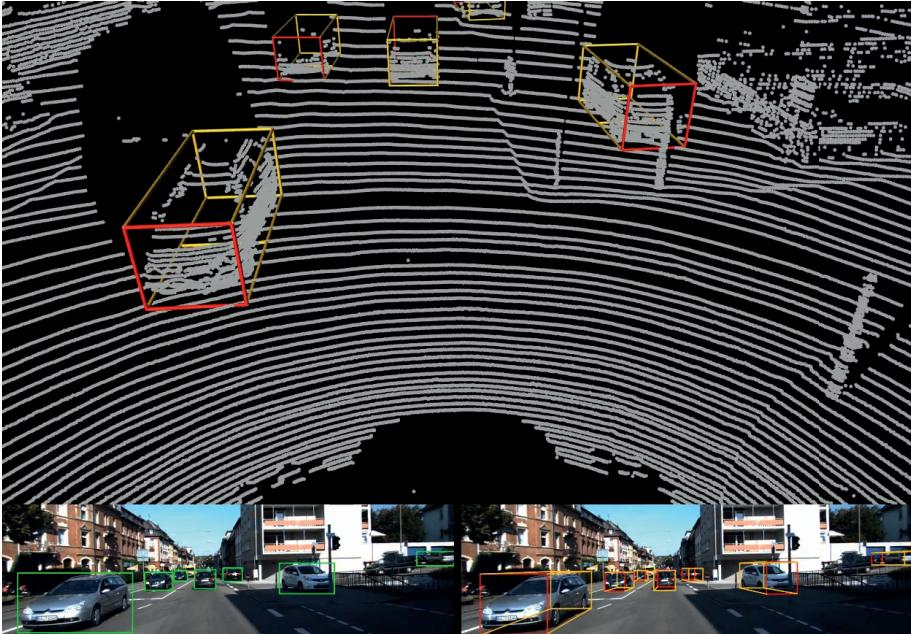
severely corrupted by measurement noise? Aleatoric uncertainty captures this type of irreducible ambiguity that can be present in the inputs  $x$ .

Input-dependent aleatoric uncertainty arises whenever the target  $y$  is expected to be inherently more uncertain for some inputs  $x$  than others. This is true e.g. in street-scene semantic segmentation, where image pixels at object boundaries are inherently ambiguous. For the example image shown in Figure 4.1, pixels right at the border between a vehicle and the road, for instance, will simply be more difficult to classify for any model (or human) than pixels in the middle of the road. This is true also in automotive 3D object detection, as illustrated in Figure 4.2 which shows example sensor data in the form of a 3D point cloud. Due to the limited sensor resolution, the estimated 3D position and size will be inherently more uncertain for distant or partially occluded vehicles than for clearly visible vehicles nearby.

To estimate input-dependent aleatoric uncertainty, the DNN  $f_\theta : \mathcal{X} \rightarrow \mathcal{O}$  can be used to specify a model  $p(y|x; \theta)$  of the conditional target distribution. That is, one can employ the probabilistic regression approach. For example if a Gaussian model is used,  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , the DNN outputs both a mean  $\mu_\theta(x)$  and variance  $\sigma_\theta^2(x)$  for each input  $x$ . The mean can be taken as a prediction,  $\hat{y}(x) = \mu_\theta(x)$ , whereas the variance  $\sigma_\theta^2(x)$  naturally can be interpreted as a measure of aleatoric uncertainty for this prediction.

### 4.3 Epistemic Uncertainty

Using DNNs to specify models  $p(y|x; \theta)$  of the conditional target distribution does however not capture epistemic uncertainty, as information about the uncertainty in the model parameters  $\theta$  is disregarded. Large epistemic uncertainty



**Figure 4.2:** Example of *aleatoric* uncertainty in the task of automotive 3D object detection [87, 15]. Based on given sensor data in the form of a 3D point cloud, it is inherently more difficult to estimate the 3D position and size of distant or partially occluded vehicles than for clearly visible vehicles nearby.

is present whenever a large set of model parameters explains the given training data (approximately) equally well. This is often the case for DNNs, since the corresponding optimization landscapes are highly multi-modal [88, 89].

Disregarding this epistemic model uncertainty often leads to highly confident yet incorrect DNN predictions, especially for inputs  $x$  which are not well-represented by the training distribution [90, 28]. For instance, a DNN can fail to generalize to unfamiliar weather conditions or environments in automotive applications, but still generate confident predictions.

Epistemic uncertainty can be estimated in a principled manner by performing Bayesian inference. Instead of just finding a single point estimate  $\theta^*$  of the model parameters  $\theta$ , by minimizing the negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$  over the training set  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ , Bayesian inference entails estimating the full posterior distribution  $p(\theta|\mathcal{D})$ . This posterior is obtained from the data likelihood  $\prod_{i=1}^N p(y_i|x_i; \theta)$  and a chosen prior  $p(\theta)$  by applying Bayes' theorem. The posterior  $p(\theta|\mathcal{D})$  is then utilized to obtain the

predictive posterior distribution  $p(y|x, \mathcal{D})$ ,

$$\begin{aligned} p(y|x, \mathcal{D}) &= \int p(y|x; \theta)p(\theta|\mathcal{D})d\theta \\ &\approx \frac{1}{M} \sum_{m=1}^M p(y|x; \theta^{(m)}), \quad \theta^{(m)} \sim p(\theta|\mathcal{D}), \end{aligned} \tag{4.1}$$

which captures both aleatoric and epistemic uncertainty. In practice, obtaining samples from the true posterior  $p(\theta|\mathcal{D})$  is virtually impossible for DNNs, requiring an approximate posterior  $q(\theta) \approx p(\theta|\mathcal{D})$  to be used instead.

## 4.4 Illustrative Example

To provide intuition for how predictive uncertainty can be estimated using DNNs, let us consider the simple 1D problem of regressing a sinusoid corrupted by input-dependent Gaussian noise. The true conditional target distribution  $p(y|x)$  is given by,

$$\begin{aligned} p(y|x) &= \mathcal{N}(y; \mu(x), \sigma^2(x)), \\ \mu(x) &= \sin(x), \quad \sigma(x) = 0.15(1 + e^{-x})^{-1}, \end{aligned} \tag{4.2}$$

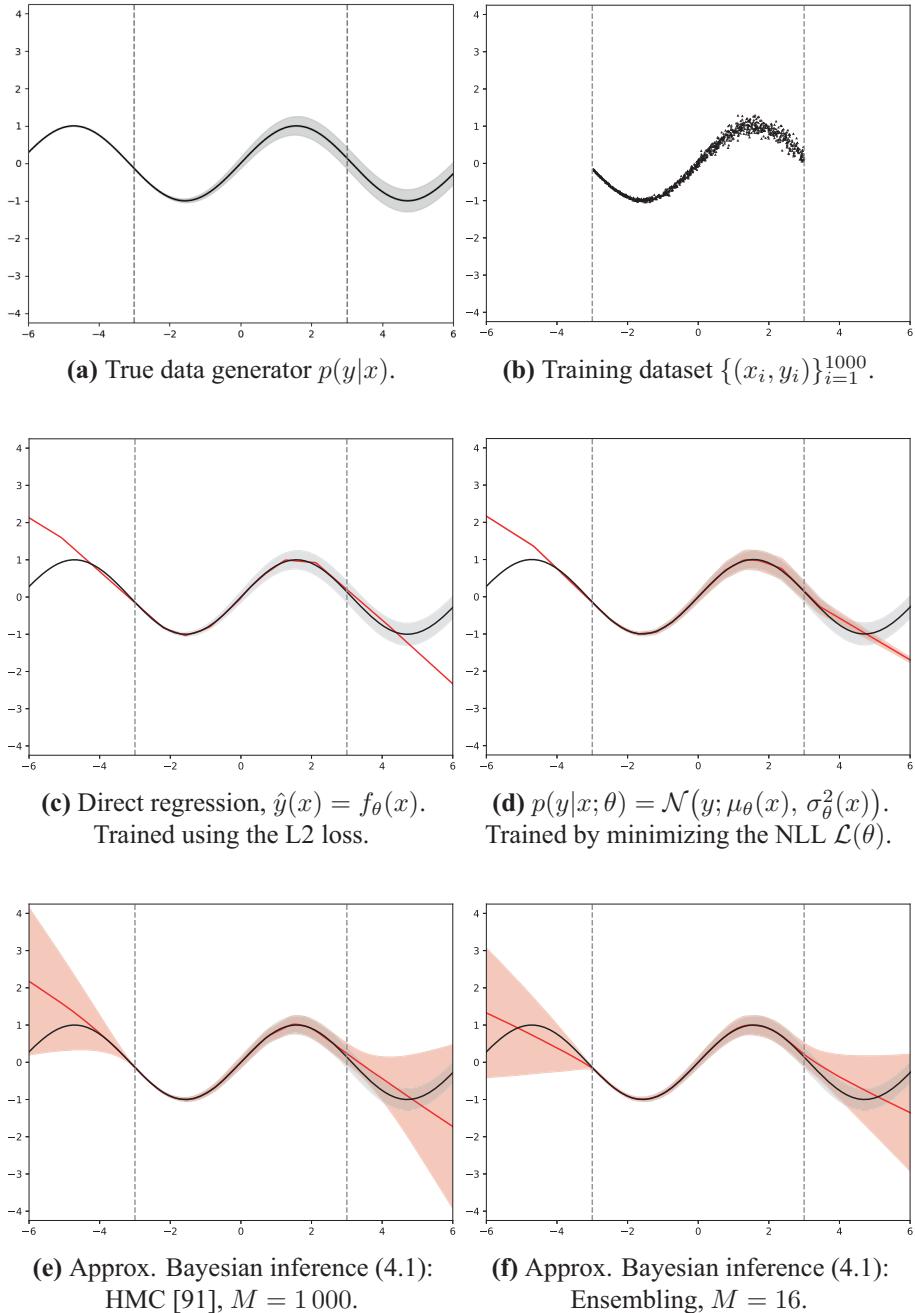
which is visualized in Figure 4.3a. There, the mean  $\mu(x) = \sin(x)$  is given by the solid black line and the variance  $\sigma^2(x)$  is represented in shaded gray. Training data  $\{(x_i, y_i)\}_{i=1}^{1000}$  of  $N = 1000$  input-target pairs is only generated for the interval  $x \in [-3, 3]$ , as visualized in Figure 4.3b.

A DNN trained to directly output predicted targets,  $\hat{y}(x) = f_\theta(x)$ , is able to accurately regress the mean  $\mu(x) = \sin(x)$  for  $x \in [-3, 3]$ , as shown in Figure 4.3c. However, this model fails to capture any notion of uncertainty.

Instead, the DNN can be used to specify a Gaussian model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , trained by minimizing the negative log-likelihood (NLL)  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$ . As shown in Figure 4.3d, the model  $p(y|x; \theta)$  closely matches the true  $p(y|x)$  for  $x \in [-3, 3]$  and thus correctly accounts for *aleatoric* uncertainty. For inputs  $|x| > 3$  not seen during training, however, the estimated mean  $\mu_\theta(x)$  deviates significantly from the true  $\mu(x) = \sin(x)$ , while the estimated uncertainty  $\sigma_\theta^2(x)$  remains very small. That is, the model becomes *overconfident* for inputs  $|x| > 3$ .

The Gaussian DNN model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$  can instead be estimated via approximate Bayesian inference (4.1), in order to account for both *aleatoric* and *epistemic* uncertainty. Specifically, a prior distribution  $p(\theta) = \mathcal{N}(0, I_P)$  and  $M = 1000$  samples  $\{\theta^{(m)}\}_{m=1}^M$  obtained via Hamiltonian Monte Carlo (HMC) [91] is used in (4.1). In this case, the model  $p(y|x; \theta)$

#### 4.4. Illustrative Example



**Figure 4.3:** Simple 1D regression problem that illustrates how predictive uncertainty can be estimated using DNNs. The predictive mean and variance of the DNN model are given by the solid red line and the shaded red area, respectively.

predicts a more reasonable uncertainty  $\sigma_\theta^2(x)$  in the region with no available training data, as shown in Figure 4.3e. While the estimated mean  $\mu_\theta(x)$  still deviates from the true  $\mu(x) = \sin(x)$  for  $|x| > 3$ , the uncertainty  $\sigma_\theta^2(x)$  also increases accordingly – the model does *not* become overconfident.

While HMC is considered a “gold standard” method for approximate Bayesian inference, it does not scale well to the large DNNs used in real-world applications. In practice, among scalable alternatives, it has been shown difficult to beat the simple approach of ensembling [28, 92]. This entails training  $M$  identical DNNs by repeatedly minimizing the negative log-likelihood  $\mathcal{L}(\theta) = \sum_{i=1}^N -\log p(y_i|x_i;\theta)$  with *random initialization*. This gives  $M$  point estimates  $\{\hat{\theta}^{(m)}\}_{m=1}^M$  of the DNN model parameters, which can be used as approximate samples in (4.1). As can be observed in Figure 4.3f, ensembling provides a good approximation of HMC in this illustrative example, even for relatively small values of  $M$ .

In Paper VI, the ensembling approach is extensively compared with another commonly used scalable method for epistemic uncertainty estimation: MC-dropout [84, 27, 93]. The results of this comparison demonstrate that ensembling consistently provides more reliable and useful estimates of predictive uncertainty. In Paper VII, ensembling and other uncertainty estimation methods are then further evaluated, critically examining their reliability under real-world distribution shifts.

# 5

## Concluding Remarks

This chapter, which is meant to be read after the included papers, contains concluding reflections along with an outlook on possible future work.

### 5.1 Conclusion

This thesis studied open questions related to how deep regression models should be constructed for best possible accuracy, and how the uncertainty in their predictions should be represented and estimated. By studying these issues, it aimed to help take steps towards an ultimate goal of developing deep regression models which are both *accurate* and *reliable* enough for real-world deployment within medical applications and other safety-critical domains. The two aspects of accuracy and reliability were studied in two different tracks of papers, each constituting one main contribution.

The first main contribution of the thesis is the formulation and development of *energy-based probabilistic regression* in Paper I - Paper III. This is a general and conceptually simple regression framework with a clear probabilistic interpretation, using EBMs to model the true conditional target distribution  $p(y|x)$ . The framework was formulated and initially evaluated in Paper I. A comprehensive study of how the EBMs should be trained for best possible regression performance was then conducted in Paper II, and some practical limitations of the approach were finally addressed in Paper III. The framework has been applied to a number of regression problems, demonstrating particularly strong performance for 2D bounding box regression – improving the state-of-the-art when applied to the task of visual tracking.

The second main contribution of the thesis is the *critical evaluation of various uncertainty estimation methods* conducted in Paper VI & VII. A general introduction to the problem of estimating the predictive uncertainty of deep models was provided in Paper VI, together with the first extensive compar-

son of ensembling and MC-dropout – demonstrating that ensembling consistently produces uncertainty estimates of higher quality. In Paper VII, ensembling and other regression uncertainty estimation methods were then further evaluated, specifically examining their reliability under real-world distribution shifts. This evaluation uncovered important limitations of current methods and serves as a challenge to the research community. It demonstrates that more work is required in order to develop truly reliable uncertainty estimation methods for regression.

## 5.2 Future Work

For energy-based probabilistic regression, a problem that could be further studied is how to produce accurate predictions at test-time without requiring the use of relatively time-consuming gradient-based refinement. While the technique presented in Paper III (approximately computing the EBM mean value using importance sampling) produces stand-alone predictions, they are generally not as accurate as those produced by the refinement-based approach.

Another interesting direction is to further study why energy-based probabilistic regression has achieved particularly strong performance specifically for bounding box regression. The fact that functions  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  naturally can be defined via pooling operations for these problems is likely a contributing factor. It would thus be interesting to explore if the quite straightforward DNN architecture used for general regression problems, as illustrated in Figure 3.1, perhaps could be modified in order to improve the performance. It is not obvious how the input images  $x$  should be combined with the targets  $y$ , defining a joint function  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ , in this general setting.

As demonstrated by both Paper VII & VIII, more work is still required in order to develop truly reliable regression uncertainty estimation methods. One particularly interesting direction is to study the clear performance difference between synthetic and real-world datasets that was observed for selective prediction methods in Paper VII. For instance, alternative uncertainty scores from the out-of-distribution detection literature (e.g., reconstruction-based approaches) could be evaluated within the selective prediction framework. Another interesting direction is to explore if self-supervised learning techniques or vision-language models perhaps could be utilized to learn feature representations which are more robust to real-world distribution shifts.

## References

- [1] M. I. Jordan and T. M. Mitchell. “Machine learning: Trends, perspectives, and prospects.” In: *Science* (2015).
- [2] A. Lindholm, N. Wahlström, F. Lindsten, and T. B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022.
- [3] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning.” In: *Nature* (2015).
- [4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-resolution image synthesis with latent diffusion models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.
- [6] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. “SDXL: Improving latent diffusion models for high-resolution image synthesis.” In: *arXiv preprint arXiv:2307.01952* (2023).
- [7] J. H. Cole, R. P. Poudel, D. Tsagkrasoulis, M. W. Caan, C. Steves, T. D. Spector, and G. Montana. “Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker.” In: *NeuroImage* (2017).
- [8] B. A. Jónsson, G. Bjornsdottir, T. Thorgeirsson, L. M. Ellingsen, G. B. Walters, D. Gudbjartsson, H. Stefansson, K. Stefansson, and M. Ulfarsson. “Brain age prediction using deep learning uncovers associated sequence variants.” In: *Nature Communications* (2019).
- [9] W. Shi, G. Yan, Y. Li, H. Li, T. Liu, C. Sun, G. Wang, Y. Zhang, Y. Zou, and D. Wu. “Fetal brain age estimation and anomaly detection using attention-based deep ensembles with uncertainty.” In: *NeuroImage* (2020).

- [10] W. Xue, A. Islam, M. Bhaduri, and S. Li. “Direct multitype cardiac indices estimation via joint representation and regression learning.” In: *IEEE Transactions on Medical Imaging* (2017).
- [11] R. Poplin, A. V. Varadarajan, K. Blumer, Y. Liu, M. V. McConnell, G. S. Corrado, L. Peng, and D. R. Webster. “Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning.” In: *Nature Biomedical Engineering* (2018).
- [12] T. Langner, R. Strand, H. Ahlström, and J. Kullberg. “Large-scale biometry with interpretable neural network regression on UK Biobank body MRI.” In: *Scientific Reports* (2020).
- [13] T. Langner, F. K. Gustafsson, B. Avelin, R. Strand, H. Ahlström, and J. Kullberg. “Uncertainty-aware body composition analysis with deep regression ensembles on UK Biobank MRI.” In: *Computerized Medical Imaging and Graphics* (2021).
- [14] E. J. Topol. “High-performance medicine: the convergence of human and artificial intelligence.” In: *Nature Medicine* (2019).
- [15] F. K. Gustafsson and E. Linder-Norén. “Automotive 3D Object Detection without Target Domain Annotations.” Master of Science Thesis in Electrical Engineering. 2018.
- [16] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. “Acquisition of localization confidence for accurate object detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [17] A. H. Ribeiro, M. H. Ribeiro, G. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr, et al. “Automatic diagnosis of the 12-lead ECG using a deep neural network.” In: *Nature communications* (2020).
- [18] S. Gustafsson, D. Gedon, E. Lampa, A. H. Ribeiro, M. J. Holzmann, T. B. Schön, and J. Sundström. “Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients.” In: *Scientific Reports* (2022).
- [19] P. J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* (1964).
- [20] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [21] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

- [22] E. Ilg, O. Cicek, S. Galessos, A. Klein, O. Makansi, F. Hutter, and T. Brox. “Uncertainty estimates and multi-hypotheses networks for optical flow.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [23] O. Makansi, E. Ilg, O. Cicek, and T. Brox. “Overcoming limitations of mixture density networks: A sampling and fitting framework for multi-modal future prediction.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [24] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. “FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation from a Single Image.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [25] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [26] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [27] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [28] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [29] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [30] D. Feng, L. Rosenbaum, F. Timm, and K. Dietmayer. “Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3D object detection.” In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019.
- [31] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. “Bounding box regression with uncertainty for accurate object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [32] C. M. Bishop. *Mixture density networks*. 1994.
- [33] C. Li and G. H. Lee. “Generating multiple hypotheses for 3d human pose estimation with mixture density network.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

- [34] A. Varamesh and T. Tuytelaars. “Mixture Dense Regression for Object Detection and Human Pose Estimation.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [35] K. Sohn, H. Lee, and X. Yan. “Learning structured output representation using deep conditional generative models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.
- [36] S. Prokudin, P. Gehler, and S. Nowozin. “Deep directional statistics: Pose estimation with uncertainty quantification.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [37] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).
- [38] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. “Hand keypoint detection in single images using multiview bootstrapping.” In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [39] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. “Convolutional pose machines.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [40] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. “DeepCut: Joint subset partition and labeling for multi person pose estimation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [41] B. Xiao, H. Wu, and Y. Wei. “Simple baselines for human pose estimation and tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [42] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. “ATOM: Accurate tracking by overlap maximization.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [43] R. Rothe, R. Timofte, and L. Van Gool. “Deep expectation of real and apparent age from a single image without facial landmarks.” In: *International Journal of Computer Vision (IJCV)* (2016).
- [44] H. Pan, H. Han, S. Shan, and X. Chen. “Mean-variance loss for deep age estimation from a face.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [45] T.-Y. Yang, Y.-H. Huang, Y.-Y. Lin, P.-C. Hsiu, and Y.-Y. Chuang. “SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018.

- [46] N. Ruiz, E. Chong, and J. M. Rehg. “Fine-grained head pose estimation without keypoints.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.
- [47] W. Cao, V. Mirjalili, and S. Raschka. “Rank-consistent Ordinal Regression for Neural Networks.” In: *arXiv preprint arXiv:1901.07884* (2019).
- [48] R. Diaz and A. Marathe. “Soft Labels for Ordinal Regression.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [49] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. “Energy-based models for sparse overcomplete representations.” In: *Journal of Machine Learning Research* (2003).
- [50] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. “A neural probabilistic language model.” In: *Journal of machine learning research* (2003).
- [51] A. Mnih and G. Hinton. “Learning nonlinear constraints with contrastive backpropagation.” In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE. 2005.
- [52] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. “Unsupervised discovery of nonlinear structure using contrastive backpropagation.” In: *Cognitive science* (2006).
- [53] M. Osadchy, M. L. Miller, and Y. L. Cun. “Synergistic face detection and pose estimation with energy-based models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.
- [54] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. “A theory of generative convnet.” In: *International Conference on Machine Learning (ICML)*. 2016.
- [55] J. Xie, S.-C. Zhu, and Y. Nian Wu. “Synthesizing dynamic patterns by spatial-temporal generative convnet.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [56] Y. Du, S. Li, J. Tenenbaum, and I. Mordatch. “Improved contrastive divergence training of energy based models.” In: *International Conference on Machine Learning (ICML)*. 2021.
- [57] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. “Learning generative convnets via multi-grid modeling and sampling.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [58] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu. “Learning descriptor networks for 3d shape synthesis and analysis.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

- [59] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. “Learning non-convergent non-persistent short-run MCMC toward energy-based model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [60] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [61] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. “Your classifier is secretly an energy based model and you should treat it like one.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [62] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. “Flow contrastive estimation of energy-based models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [63] B. Pang, T. Han, E. Nijkamp, S.-C. Zhu, and Y. N. Wu. “Learning latent space energy-based prior model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [64] F. Bao, C. Li, K. Xu, H. Su, J. Zhu, and B. Zhang. “Bi-level Score Matching for Learning Energy-based Latent Variable Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [65] L. Dinh, D. Krueger, and Y. Bengio. “Nice: Non-linear independent components estimation.” In: *arXiv preprint arXiv:1410.8516* (2014).
- [66] L. Dinh, J. Sohl-Dickstein, and S. Bengio. “Density estimation using Real NVP.” In: *International Conference on Learning Representations (ICLR)*. 2017.
- [67] D. P. Kingma and P. Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [68] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010.
- [69] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. “Flow Contrastive Estimation of Energy-Based Models.” In: *arXiv preprint arXiv:1912.00589* (2019).
- [70] A. Hyvärinen. “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research* (2005).
- [71] P. Vincent. “A connection between score matching and denoising autoencoders.” In: *Neural computation* (2011).

- [72] Y. Song and S. Ermon. “Generative modeling by estimating gradients of the data distribution.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [73] G. E. Hinton. “Training products of experts by minimizing contrastive divergence.” In: *Neural computation* (2002).
- [74] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. “On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models.” In: *Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.
- [75] A. Mnih and Y. W. Teh. “A fast and simple algorithm for training neural probabilistic language models.” In: *International Conference on Machine Learning (ICML)*. 2012.
- [76] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013.
- [77] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the limits of language modeling.” In: *arXiv preprint arXiv:1602.02410* (2016).
- [78] Z. Ma and M. Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [79] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning deep representations by mutual information estimation and maximization.” In: *International Conference on Learning Representations (ICLR)*. 2019.
- [80] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning representations by maximizing mutual information across views.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [81] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations.” In: *International Conference on Machine Learning (ICML)*. 2020.
- [82] T. Han, W. Xie, and A. Zisserman. “Self-supervised Co-training for Video Representation Learning.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [83] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.
- [84] Y. Gal. “Uncertainty in Deep Learning.” PhD thesis. University of Cambridge, 2016.

- [85] M. Wrenninge and J. Unger. “Syncscapes: A photorealistic synthetic dataset for street scene parsing.” In: *arXiv preprint arXiv:1810.08705* (2018).
- [86] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. “The cityscapes dataset for semantic urban scene understanding.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [87] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. “Vision meets Robotics: The KITTI Dataset.” In: *International Journal of Robotics Research (IJRR)* (2013).
- [88] P. Auer, M. Herbster, and M. K. Warmuth. “Exponentially many local minima for single neurons.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1996.
- [89] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. “The loss surfaces of multilayer networks.” In: *Artificial Intelligence and Statistics*. 2015.
- [90] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks.” In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.
- [91] R. M. Neal. “MCMC using Hamiltonian dynamics.” In: *Handbook of Markov chain Monte Carlo* (2011).
- [92] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [93] A. Kendall, V. Badrinarayanan, and R. Cipolla. “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2017.
- [94] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [95] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. “How to Train Your Energy-Based Model for Regression.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2020.
- [96] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Learning Proposals for Practical Energy-Based Regression.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2022.

- [97] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Accurate 3D Object Detection using Energy-Based Models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2021.
- [98] J. N. Hendriks, F. K. Gustafsson, A. H. Ribeiro, A. G. Wills, and T. B. Schön. “Deep Energy-Based NARX Models.” In: *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*. 2021.
- [99] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2020.
- [100] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?” In: *Transactions on Machine Learning Research (TMLR)*. 2023.
- [101] P. Von Bachmann, D. Gedon, F. K. Gustafsson, A. H. Ribeiro, E. Lampa, S. Gustafsson, J. Sundström, and T. B. Schön. “ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.” In Preparation. 2023.

# Acta Universitatis Upsaliensis

*Digital Comprehensive Summaries of Uppsala Dissertations from  
the Faculty of Science and Technology 2320*

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)



# Paper I

## Title

Energy-Based Models for Deep Probabilistic Regression

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020





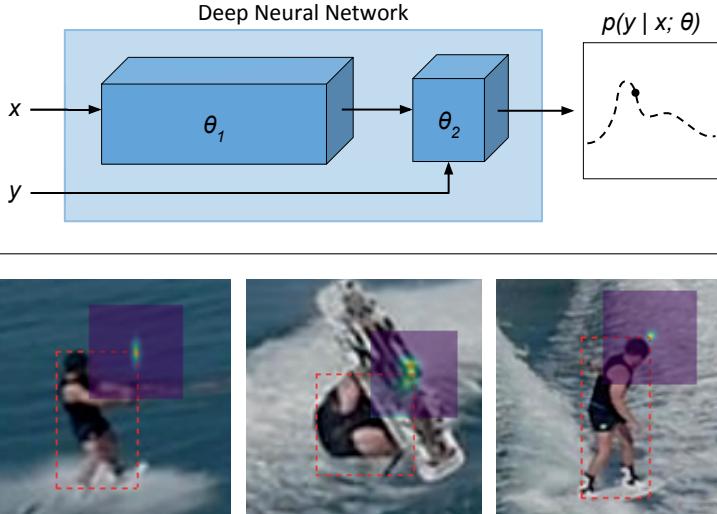
# Energy-Based Models for Deep Probabilistic Regression

## Abstract

While deep learning-based classification is generally tackled using standardized approaches, a wide variety of techniques are employed for regression. In computer vision, one particularly popular such technique is that of confidence-based regression, which entails predicting a confidence value for each input-target pair  $(x, y)$ . While this approach has demonstrated impressive results, it requires important task-dependent design choices, and the predicted confidences lack a natural probabilistic meaning. We address these issues by proposing a general and conceptually simple regression method with a clear probabilistic interpretation. In our proposed approach, we create an energy-based model of the conditional target density  $p(y|x)$ , using a deep neural network to predict the un-normalized density from  $(x, y)$ . This model of  $p(y|x)$  is trained by directly minimizing the associated negative log-likelihood, approximated using Monte Carlo sampling. We perform comprehensive experiments on four computer vision regression tasks. Our approach outperforms direct regression, as well as other probabilistic and confidence-based methods. Notably, our model achieves a 2.2% AP improvement over Faster-RCNN for object detection on the COCO dataset, and sets a new state-of-the-art on visual tracking when applied for bounding box estimation. In contrast to confidence-based methods, our approach is also shown to be directly applicable to more general tasks such as age and head-pose estimation. Code is available at [https://github.com/fregu856/ebms\\_regression](https://github.com/fregu856/ebms_regression).

## 1 Introduction

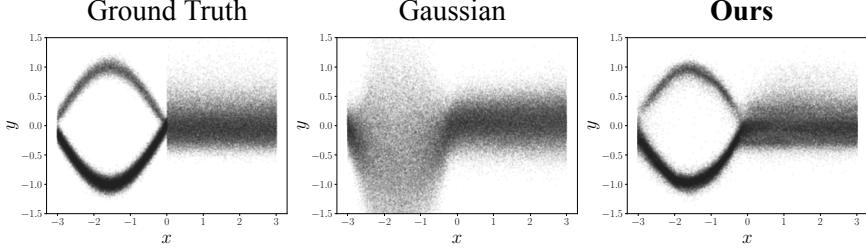
Supervised regression entails learning a model capable of predicting a continuous target value  $y$  from an input  $x$ , given a set of paired training examples. It is a fundamental machine learning problem with many important applications within computer vision and other domains. Common regression tasks



**Figure 1:** An overview of the proposed regression method (top). We train an energy-based model  $p(y|x; \theta) \propto e^{f_\theta(x,y)}$  of the conditional target density  $p(y|x)$ , using a DNN  $f_\theta$  to predict the un-normalized density directly from the input-target pair  $(x, y)$ . Our approach is capable of predicting highly flexible densities and produce highly accurate estimates. This is demonstrated for the problem of bounding box regression (bottom), visualizing the marginal density for the top right box corner as a heatmap.

within computer vision include object detection [1, 2, 3, 4], head- and body-pose estimation [5, 6, 7, 8], age estimation [9, 10, 11], visual tracking [12, 13, 14, 15] and medical image registration [16, 17], just to mention a few. Today, such regression problems are commonly tackled using Deep Neural Networks (DNNs), due to their ability to learn powerful feature representations directly from data.

While classification is generally addressed using standardized losses and output representations, a wide variety of different techniques are employed for regression. The most conventional strategy is to train a DNN to directly predict a target  $y$  given an input  $x$  [18]. In such *direct regression* approaches, the model parameters of the DNN are learned by minimizing a loss function, for example the  $L^2$  or  $L^1$  loss, penalizing discrepancy between the predicted and ground truth target values. From a probabilistic perspective, this approach corresponds to creating a simple parametric model of the conditional target density  $p(y|x)$ , and minimizing the associated negative log-likelihood. The  $L^2$  loss, for example, corresponds to a fixed-variance Gaussian model. More recent work [19, 20, 21, 22, 23, 24] has also explored learning more expressive models of  $p(y|x)$ , by letting a DNN instead output the full set of parameters of a certain family of probability distributions. To allow for straightforward implementation and training, many of these *probabilistic regression* approaches



**Figure 2:** An illustrative 1D regression problem. The training data  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{2000}$  is generated by the ground truth conditional target density  $p(y|x)$ . Our energy-based model  $p(y|x; \theta) \propto e^{f_\theta(x, y)}$  of  $p(y|x)$  is trained by directly minimizing the associated negative log-likelihood, approximated using Monte Carlo importance sampling. In contrast to the Gaussian model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , our energy-based model can learn multimodal and complex target densities directly from data.

however restrict the parametric model to unimodal distributions such as Gaussian [20, 21] or Laplace [19, 22, 25], still severely limiting the expressiveness of the learned conditional target density. While these methods benefit from a clear probabilistic interpretation, they thus fail to fully exploit the predictive power of the DNN.

The quest for improved regression accuracy has also led to the development of more specialized methods, designed for a specific set of tasks. In computer vision, one particularly popular approach is that of *confidence-based regression*. Here, a DNN instead predicts a scalar confidence value for input-target pairs  $(x, y)$ . The confidence can then be maximized w.r.t.  $y$  to obtain a target prediction for a given input  $x$ . This approach is commonly employed for image-coordinate regression tasks within e.g. human pose estimation [5, 6, 7] and object detection [3, 4], where a 2D heatmap over image pixel coordinates  $y$  is predicted. Recently, the approach was also applied to the problem of bounding box regression by Jiang et al. [2]. Their proposed method, IoU-Net, obtained state-of-the-art accuracy on object detection, and was later also successfully applied to the task of visual tracking [15]. The training of such confidence-based regression methods does however entail generating additional pseudo ground truth labels, e.g. by employing a Gaussian kernel [26, 6], and selecting an appropriate loss function. This both requires numerous design choices to be made, and limits the general applicability of the methods. Moreover, confidence-based regression methods do not allow for a natural probabilistic interpretation in terms of the conditional target density  $p(y|x)$ . In this work, we therefore set out to develop a method combining the general applicability and the clear interpretation of probabilistic regression with the predictive power of the confidence-based approaches.

**Contributions** We propose a general and conceptually simple regression method with a clear probabilistic interpretation. Our method employs an

energy-based model [27] to predict the un-normalized conditional target density  $p(y|x)$  from the input-target pair  $(x, y)$ . It is trained by directly minimizing the associated negative log-likelihood, exploiting tailored Monte Carlo approximations. At test time, targets are predicted by maximizing the conditional target density  $p(y|x)$  through gradient-based refinement. Our energy-based model is straightforward both to implement and train. Unlike commonly used probabilistic models, it can however still learn highly flexible target densities directly from data, as visualized in Figure 2. Compared to confidence-based approaches, our method requires no pseudo labels, benefits from a clear probabilistic interpretation, and is directly applicable to a variety of computer vision applications. We evaluate the proposed method on four diverse computer vision regression tasks: object detection, visual tracking, age estimation and head-pose estimation. Our method is found to significantly outperform both direct regression baselines, and popular probabilistic and confidence-based alternatives, including the state-of-the-art IoU-Net [2]. Notably, our method achieves a 2.2% AP improvement over FPN Faster-RCNN [28] when applied for object detection on COCO [29], and sets a new state-of-the-art on standard benchmarks [30, 31] when applied for bounding box estimation in the recent ATOM [15] visual tracker. Our method is also shown to be directly applicable to the more general tasks of age and head-pose estimation, consistently improving performance of a variety of baselines.

## 2 Background & Related Work

In supervised regression, the task is to learn to predict a target value  $y^* \in \mathcal{Y}$  from a corresponding input  $x^* \in \mathcal{X}$ , given a training set of i.i.d. input-target examples,  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ . As opposed to classification, the target space  $\mathcal{Y}$  is a continuous set, e.g.  $\mathcal{Y} = \mathbb{R}^K$ . In computer vision, the input space  $\mathcal{X}$  often corresponds to the space of images, whereas the output space  $\mathcal{Y}$  depends on the task at hand. Common examples include  $\mathcal{Y} = \mathbb{R}^2$  in image-coordinate regression [6, 3],  $\mathcal{Y} = \mathbb{R}_+$  in age estimation [9, 10], and  $\mathcal{Y} = \mathbb{R}^4$  in object bounding box regression [1, 2]. A variety of techniques have previously been applied to supervised regression tasks. In order to motivate and provide intuition for our proposed method, we here describe a few popular approaches.

**Direct Regression** Over the last decade, DNNs have been shown to excel at a wide variety of regression problems. Here, a DNN is viewed as a function  $f_\theta : \mathcal{U} \rightarrow \mathcal{O}$ , parameterized by a set of learnable weights  $\theta \in \mathbb{R}^P$ . The most conventional regression approach is to train a DNN to directly predict the targets,  $y^* = f_\theta(x^*)$ , called *direct regression*. The model parameters  $\theta$  are learned by minimizing a loss  $\ell(f_\theta(x_i), y_i)$  that penalizes discrepancy between the prediction  $f_\theta(x_i)$  and the ground truth target value  $y_i$  on training examples

$(x_i, y_i)$ . Common choices include the  $L^2$  loss,  $\ell(\hat{y}, y) = \|\hat{y} - y\|_2^2$ , the  $L^1$  loss,  $\ell(\hat{y}, y) = \|\hat{y} - y\|_1$ , and their close relatives [32, 18]. From a probabilistic perspective, the choice of loss corresponds to minimizing the negative log-likelihood  $-\log p(y|x; \theta)$  for a specific model  $p(y|x; \theta)$  of the conditional target density. For example, the  $L^2$  loss is derived from a fixed-variance Gaussian model,  $p(y|x; \theta) = \mathcal{N}(y; f_\theta(x), \sigma^2)$ .

**Probabilistic Regression** More recent work [19, 20, 21, 22, 25, 33, 23] has explicitly taken advantage of this probabilistic perspective to achieve more flexible parametric models  $p(y|x; \theta) = p(y; \phi_\theta(x))$ , by letting the DNN output the parameters  $\phi$  of a family of probability distributions  $p(y; \phi)$ . For example, a general 1D Gaussian model can be realized as  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , where the DNN outputs the mean and log-variance as  $f_\theta(x) = \phi_\theta(x) = [\mu_\theta(x) \quad \log \sigma_\theta^2(x)]^\top \in \mathbb{R}^2$ . The model parameters  $\theta$  are learned by minimizing the negative log-likelihood  $-\sum_{i=1}^N \log p(y_i|x_i; \theta)$  over the training set  $\mathcal{D}$ . At test time, a target estimate  $y^*$  is obtained by first predicting the density parameter values  $\phi_\theta(x^*)$  and then, for instance, taking the expected value of  $p(y; \phi_\theta(x))$ . Previous work has applied simple Gaussian and Laplace models on computer vision tasks such as object detection [34, 35] and optical flow estimation [22, 25], usually aiming to not only achieve accurate predictions, but also to provide an estimate of aleatoric uncertainty [19, 36]. To allow for multimodal models  $p(y; \phi_\theta(x))$ , mixture density networks (MDNs) [37] have also been applied [33, 23]. The DNN then outputs weights for  $K$  mixture components along with  $K$  sets of parameters, e.g.  $K$  sets of means and log-variances for a mixture of Gaussians. Previous work has also applied *infinite* mixture models by utilizing the conditional VAE (cVAE) framework [38, 24]. A latent variable model  $p(y|x; \theta) = \int p(y; \phi_\theta(x, z))p(z; \phi_\theta(x))dz$  is then employed, where  $p(y; \phi_\theta(x, z))$  and  $p(z; \phi_\theta(x))$  typically are Gaussian distributions. Our proposed method also entails predicting a conditional target density  $p(y|x; \theta)$  and minimizing the associated negative log-likelihood. However, our energy-based model  $p(y|x; \theta)$  is not limited to the functional form of any specific probability density (e.g. Gaussian or Laplace), but is instead directly defined via a learned scalar function of  $(x, y)$ . In contrast to MDNs and cVAEs, our model  $p(y|x; \theta)$  is not even limited to densities which are simple to generate samples from. This puts *minimal restricting assumptions* on the true  $p(y|x)$ , allowing it to be efficiently learned directly from data.

**Confidence-Based Regression** Another category of approaches reformulates the regression problem as  $y^* = \arg \max_y f_\theta(x, y)$ , where  $f_\theta(x, y) \in \mathbb{R}$  is a scalar confidence value predicted by the DNN. The idea is thus to predict a quantity  $f_\theta(x, y)$ , depending on both input  $x$  and target  $y$ , that can be maximized over  $y$  to obtain the final prediction  $y^*$ . This maximization-based formulation is inherent in Structural SVMs [39], but has also been adopted for DNNs. We term this family of approaches *confidence-based regression*. Compared

to direct regression, the predicted confidence  $f_\theta(x, y)$  can encapsulate multiple hypotheses and other ambiguities. Confidence-based regression has been shown particularly suitable for image-coordinate regression tasks, such as hand keypoint localization [40] and body-part detection [26, 41, 6]. In these cases, a CNN is trained to output a 2D heatmap over the image pixel coordinates  $y$ , thus taking full advantage of the translational invariance of the problem. In computer vision, confidence prediction has also been successfully employed for tasks other than pure image-coordinate regression. Jiang et al. [2] proposed the IoU-Net for bounding box regression in object detection, where a bounding box  $y \in \mathbb{R}^4$  and image  $x$  are both input to the DNN to predict a confidence  $f_\theta(x, y)$ . It employs a pooling-based architecture that is differentiable w.r.t. the bounding box  $y$ , allowing efficient gradient-based maximization to obtain the final estimate  $y^* = \arg \max_y f_\theta(x, y)$ . IoU-Net was later also successfully applied to target object estimation in visual tracking [15].

In general, confidence-based approaches are trained using a set of *pseudo label* confidences  $c(x_i, y_i, y)$  generated for each training example  $(x_i, y_i)$ , and by employing a loss  $\ell(f_\theta(x_i, y), c(x_i, y_i, y))$ . One strategy [41, 3] is to treat the confidence prediction as a binary classification problem, where  $c(x_i, y_i, y)$  represents either the class,  $c \in \{0, 1\}$ , or its probability,  $c \in [0, 1]$ , and employ cross-entropy based losses  $\ell$ . The other approach is to treat the confidence prediction as a direct regression problem itself by applying standard regression losses, such as  $L^2$  [40, 15, 26] or the Huber loss [2]. In these cases, the pseudo label confidences  $c$  can be constructed using a similarity measure  $S$  in the target space,  $c(x_i, y_i, y) = S(y, y_i)$ , for example defined as the Intersection over Union (IoU) between two bounding boxes [2] or simply by a Gaussian kernel [26, 6, 7].

While these methods have demonstrated impressive results, confidence-based approaches thus require important design choices. In particular, the strategy for constructing the pseudo labels  $c$  and the choice of loss  $\ell$  are often crucial for performance and highly *task-dependent*, limiting general applicability. Moreover, the predicted confidence  $f_\theta(x, y)$  can be difficult to interpret, since it has no natural connection to the conditional target density  $p(y|x)$ . In contrast, our approach is directly trained to predict  $p(y|x)$  itself, and importantly it does *not* require generation of pseudo label confidences or choosing a specific loss.

**Regression-by-Classification** A regression problem can also be treated as a classification problem by first discretizing the target space  $\mathcal{Y}$  into a finite set of  $C$  classes. Standard techniques from classification, such as softmax and the cross-entropy loss, can then be employed. This approach has previously been applied to both age estimation [9, 10, 42] and head-pose estimation [43, 8]. The discretization of the target space  $\mathcal{Y}$  however complicates exploiting its inherent neighborhood structure, an issue that has been addressed by exploring ordinal regression methods for 1D problems [11, 44]. While our energy-based

approach can be seen as a generalization of the softmax model for classification to the continuous target space  $\mathcal{Y}$ , it does not suffer from the aforementioned drawbacks of regression-by-classification. On the contrary, our model naturally allows the network to exploit the full structure of the continuous target space  $\mathcal{Y}$ .

**Energy-Based Models** Our approach is of course also related to the theoretical framework of energy-based models, which often has been employed for machine learning problems in the past [45, 46, 27]. It involves learning an energy function  $\mathcal{E}_\theta(x) \in \mathbb{R}$  that assigns low energy to observed data  $x_i$  and high energy to other values of  $x$ . Recently, energy-based models have been used primarily for unsupervised learning problems within computer vision [47, 48, 49, 50, 51], where DNNs are directly used to predict  $\mathcal{E}_\theta(x)$ . These models are commonly trained by minimizing the negative log-likelihood, stemming from the probabilistic model  $p(x; \theta) = e^{-\mathcal{E}_\theta(x)} / \int e^{-\mathcal{E}_\theta(x)} dx$ , for example by generating approximate image samples from  $p(x; \theta)$  using Markov Chain Monte Carlo [48, 49, 51]. In contrast, we study the application of energy-based models for  $p(y|x)$  in *supervised* regression, a mostly overlooked research direction in recent years, and obtain state-of-the-art performance on four diverse computer vision regression tasks.

### 3 Proposed Regression Method

We propose a general and conceptually simple regression method with a clear probabilistic interpretation. Our method employs an energy-based model within a probabilistic regression formulation, defined in Section 3.1. In Section 3.2, we introduce our training strategy which is designed to be simple, yet highly effective and applicable to a wide variety of regression tasks within computer vision. Lastly, we describe our prediction strategy for high accuracy in Section 3.3.

#### 3.1 Formulation

We take the probabilistic view of regression by creating a model  $p(y|x; \theta)$  of the conditional target density  $p(y|x)$ , in which  $\theta$  is learned by minimizing the associated negative log-likelihood. Instead of defining  $p(y|x; \theta)$  by letting a DNN predict the parameters of a certain family of probability distributions (e.g. Gaussian or Laplace), we construct a versatile energy-based model that can better leverage the predictive power of DNNs. To that end, we take inspiration from confidence-based regression approaches and let a DNN directly predict a scalar value for any input-target pair  $(x, y)$ . Unlike confidence-based methods

however, this prediction has a clear probabilistic interpretation. Specifically, we view a DNN as a function  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ , parameterized by  $\theta \in \mathbb{R}^P$ , that maps an input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  to a scalar value  $f_\theta(x, y) \in \mathbb{R}$ . Our model  $p(y|x; \theta)$  of the conditional target density  $p(y|x)$  is then defined according to,

$$p(y|x; \theta) = \frac{e^{f_\theta(x, y)}}{Z(x, \theta)}, \quad Z(x, \theta) = \int e^{f_\theta(x, \tilde{y})} d\tilde{y}, \quad (1)$$

where  $Z(x, \theta)$  is the input-dependent normalizing partition function. We train this energy-based model (1) by directly minimizing the negative log-likelihood  $-\log p(\{y_i\}_i | \{x_i\}_i; \theta) = \sum_{i=1}^N -\log p(y_i|x_i; \theta)$ , where each term is given by,

$$-\log p(y_i|x_i; \theta) = \log \left( \int e^{f_\theta(x_i, y)} dy \right) - f_\theta(x_i, y_i). \quad (2)$$

This direct and straightforward training approach thus requires the evaluation of the generally intractable  $Z(x, \theta) = \int e^{f_\theta(x, y)} dy$ . Many fundamental computer vision tasks, such as object detection, keypoint estimation and pose estimation, however rely on regression problems with a low-dimensional target space  $\mathcal{Y}$ . In such cases, effective finite approximations of  $Z(x, \theta)$  can be applied. In some tasks, such as image-coordinate regression, this is naturally performed by a grid approximation, utilizing the dense prediction obtained by fully-convolutional networks. In this work, we however investigate a more *generally applicable* technique, namely Monte Carlo approximations with importance sampling. This procedure, when employed for training the network, is detailed in Section 3.2.

At test time, given an input  $x^*$ , our model in (1) allows evaluating the conditional target density  $p(y|x^*; \theta)$  for any target  $y$  by first approximating  $Z(x^*, \theta)$ , and then predicting the scalar  $f_\theta(x^*, y)$  using the DNN. This enables the computation of, e.g., the mean and variance of the target value  $y$ . In this work, we take inspiration from confidence-based regression and focus on finding the most likely prediction,  $y^* = \arg \max_y p(y|x^*; \theta) = \arg \max_y f_\theta(x^*, y)$ , which does not require the evaluation of  $Z(x^*, \theta)$  during inference. Thanks to the auto-differentiation capabilities of modern deep learning frameworks, we can apply gradient-based techniques to find the final prediction by simply maximizing the network output  $f_\theta(x^*, y)$  w.r.t.  $y$ . We elaborate on this procedure for prediction in Section 3.3.

## 3.2 Training

Our energy-based model  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$  of the conditional target density is trained by directly minimizing the negative log-likelihood

$\sum_{i=1}^N -\log p(y_i|x_i; \theta)$ . To evaluate the integral in (2), we employ Monte Carlo importance sampling. Each term  $-\log p(y_i|x_i; \theta)$  is therefore approximated by sampling values  $\{y^{(k)}\}_{k=1}^M$  from a proposal distribution  $q(y|y_i)$  that depends on the ground truth target value  $y_i$ ,

$$-\log p(y_i|x_i; \theta) \approx \log \left( \frac{1}{M} \sum_{k=1}^M \frac{e^{f_\theta(x_i, y^{(k)})}}{q(y^{(k)}|y_i)} \right) - f_\theta(x_i, y_i). \quad (3)$$

The final loss  $J(\theta)$  used to train the DNN  $f_\theta$  is then obtained by averaging over all training examples  $\{(x_i, y_i)\}_{i=1}^n$  in the current mini-batch,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y^{(i,m)})}}{q(y^{(i,m)}|y_i)} \right) - f_\theta(x_i, y_i), \quad (4)$$

where  $\{y^{(i,m)}\}_{m=1}^M$  are  $M$  samples drawn from  $q(y|y_i)$ . Qualitatively, minimizing  $J(\theta)$  encourages the DNN to output large values  $f_\theta(x_i, y_i)$  for the ground truth target  $y_i$ , while minimizing the predicted value  $f_\theta(x_i, y)$  at all other targets  $y$ . In ambiguous or uncertain cases, the DNN can output small values everywhere or large values at multiple hypotheses, but at the cost of a higher loss.

As can be seen in (4), the DNN  $f_\theta$  is applied both to the input-target pair  $(x_i, y_i)$ , and all input-sample pairs  $\{(x_i, y^{(i,m)})\}_{m=1}^M$  during training. While this can seem inefficient, most applications in computer vision employ network architectures that first extract a deep feature representation for the input  $x_i$ . The DNN  $f_\theta$  can thus be designed to combine this input feature with the target  $y$  at a late stage, as visualized in Figure 1. The input feature extraction process, which becomes the main computational bottleneck, therefore needs to be performed only once for each  $x_i$ . In practice, we found our training strategy to not add any significant overhead compared to the direct regression baselines, and the computational cost to be *identical* to that of the confidence-based methods.

Compared to confidence-based regression, a significant advantage of our approach is however that there is no need for generating task-dependent pseudo label confidences or choosing between different losses. The *only* design choice of our training method is the proposal distribution  $q(y|y_i)$ . Note however that since the loss  $J(\theta)$  in (4) explicitly adapts to  $q(y|y_i)$ , this choice has no effect on the overall behaviour of the loss, only on the quality of the sampled approximation. We found a mixture of a few equally weighted Gaussian components, all centered at the target label  $y_i$ , to consistently perform well in our experiments across all four diverse computer vision applications. Specifically,  $q(y|y_i)$  is set to,

$$q(y|y_i) = \frac{1}{L} \sum_{l=1}^L \mathcal{N}(y; y_i, \sigma_l^2 I), \quad (5)$$

where the standard deviations  $\{\sigma_l\}_{l=1}^L$  are hyperparameters selected based on a validation set for each experiment. We only considered the simple Gaussian proposal in (5), as this was found sufficient to obtain state-of-the-art experimental results. Full ablation studies for the number of components  $L$  and  $\{\sigma_l\}_{l=1}^L$  are provided in the supplementary material. Figure 2 illustrates that our model  $p(y|x; \theta)$  can learn complex conditional target densities, containing both multi-modalities and asymmetry, directly from data using the described training procedure. In this illustrative example, we use (5) with  $L = 2$  and  $\sigma_1 = 0.1, \sigma_2 = 0.8$ .

### 3.3 Prediction

Given an input  $x^*$  at test time, the trained DNN  $f_\theta$  can be used to evaluate the full conditional target density  $p(y|x^*; \theta) = e^{f_\theta(x^*, y)} / Z(x^*, \theta)$ , by employing the aforementioned techniques to approximate the constant  $Z(x^*, \theta)$ . In many applications, the most likely prediction  $y^* = \arg \max_y p(y|x^*; \theta)$  is however the single desired output. For our energy-based model, this is obtained by directly maximizing the DNN output,  $y^* = \arg \max_y f_\theta(x^*, y)$ , thus not requiring  $Z(x^*, \theta)$  to be evaluated. By taking inspiration from IoU-Net [2] and designing the DNN  $f_\theta$  to be differentiable w.r.t. the target  $y$ , the gradient  $\nabla_y f_\theta(x^*, y)$  can be efficiently evaluated using the auto-differentiation tools implemented in modern deep learning frameworks. An estimate of  $y^* = \arg \max_y f_\theta(x^*, y)$  can therefore be obtained by performing gradient ascent to find a local maximum of  $f_\theta(x^*, y)$ .

The gradient ascent refinement is performed either on a single initial estimate  $\hat{y}$ , or on a set of random initializations  $\{\hat{y}_k\}_{k=1}^K$  to obtain a final accurate prediction  $y^*$ . Starting at  $y = \hat{y}_k$ , we thus run  $T$  gradient ascent iterations,  $y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ , with step-length  $\lambda$ . In our experiments, we fix  $T$  (typically,  $T = 10$ ) and select  $\lambda$  using grid search on a validation set. As noted in Section 3.2, this prediction procedure can be made highly efficient by extracting the feature representation for  $x^*$  only once. Back-propagation is then performed only through a few final layers of the DNN to evaluate the gradient  $\nabla_y f_\theta(x^*, y)$ . The gradient computation for a set of candidates  $\{\hat{y}_k\}_{k=1}^K$  can also be parallelized on the GPU by simple batching, requiring no significant overhead. Overall, the inference speed is somewhat decreased compared to direct regression baselines, but is *identical* to confidence-based methods such as IoU-Net [2]. An algorithm detailing this prediction procedure is found in the supplementary material.

**Table 1:** Impact of  $L$  and  $\{\sigma_l\}_{l=1}^L$  in the proposal distribution  $q(y|y_i)$  (5), for the object detection task on the *2017 val* split of the COCO [29] dataset. For  $L = 2$ ,  $\sigma_1 = \sigma_2/4$ . For  $L = 3$ ,  $\sigma_1 = \sigma_3/4$  and  $\sigma_2 = \sigma_3/2$ .  $L = 3$  with  $\sigma_L = 0.15$  is selected.

Number of components $L$	1	2	3						
Base proposal st. dev. $\sigma_L$	0.02	0.04	0.08	0.1	0.15	0.2	0.1	0.15	0.2
AP (%)	38.1	38.5	37.5	39.0	<b>39.1</b>	39.0	39.0	<b>39.1</b>	38.8

**Table 2:** Results for the object detection task on the *2017 test-dev* split of the COCO [29] dataset. Our proposed method significantly outperforms the baseline FPN Faster-RCNN [28] and the state-of-the-art confidence-based IoU-Net [2].

Formulation Approach	Direct Faster-RCNN	Gaussian	Gaussian Mixt. 2	Gaussian Mixt. 4	Gaussian Mixt. 8	Gaussian cVAE	Laplace	Confidence IoU-Net	Confidence IoU-Net*	Ours
AP (%)	37.2	36.7	37.1	37.0	36.8	37.2	37.1	38.3	38.2	<b>39.4</b>
AP <sub>50</sub> (%)	<b>59.2</b>	58.7	59.1	59.1	59.1	<b>59.2</b>	59.1	58.3	58.4	58.6
AP <sub>75</sub> (%)	40.3	39.6	40.0	39.9	39.7	40.0	40.2	41.4	41.4	<b>42.1</b>
FPS	12.2	<b>12.2</b>	<b>12.2</b>	12.1	12.1	9.6	<b>12.2</b>	5.3	5.3	5.3

## 4 Experiments

We perform comprehensive experiments on four different computer vision regression tasks: object detection, visual tracking, age estimation and head-pose estimation. Our proposed approach is compared both to baseline regression methods and to state-of-the-art models. Notably, our method significantly outperforms the confidence-based IoU-Net [2] method for bounding box regression in direct comparisons, both when applied for object detection on the COCO dataset [29] and for target object estimation in the recent ATOM [15] visual tracker. On age and head-pose estimation, our approach is shown to consistently improve performance of a variety of baselines. All experiments are implemented in PyTorch [52]. For all tasks, further details are also provided in the supplementary material.

### 4.1 Object Detection

We first perform experiments on object detection, the task of classifying and estimating a bounding box for each object in a given image. Specifically, we compare our regression method to other techniques for the task of bounding box regression, by integrating them into an existing object detection pipeline. To this end, we use the Faster-RCNN [1] framework, which serves as a popular baseline in the object detection field due to its strong state-of-the-art performance. It employs one network head for classification and one head for regressing the bounding box using the direct regression approach. We also include various probabilistic regression baselines and compare with simple Gaussian and Laplace models, by modifying the Faster-RCNN regression head to predict both the mean and log-variance of the distribution, and adopting the as-

sociated negative log-likelihood loss. Similarly, we compare with mixtures of  $K = \{2, 4, 8\}$  Gaussians by duplicating the modified regression head  $K$  times and adding a network head for predicting  $K$  component weights. Moreover, we compare with an infinite mixture of Gaussians by training a cVAE. Finally, we also compare our approach to the state-of-the-art confidence-based IoU-Net [2]. It extends Faster-RCNN with an additional branch that predicts the IoU overlap between a target bounding box  $y$  and the ground truth. The IoU prediction branch uses differentiable region pooling [2], allowing the initial bounding box predicted by the Faster-RCNN to be refined using gradient-based maximization of the predicted IoU confidence.

For our approach, we employ an *identical architecture* as used in IoU-Net for a fair comparison. Instead of training the network to output the IoU, we predict the exponent  $f_\theta(x, y)$  in (1), trained by minimizing the negative log-likelihood in (4). We parametrize the bounding box as  $y = (c_x/w_0, c_y/h_0, \log w, \log h) \in \mathbb{R}^4$ , where  $(c_x, c_y)$  and  $(w, h)$  denote the center coordinate and size, respectively. The reference size  $(w_0, h_0)$  is set to that of the ground truth during training and the initial box during prediction. Based on the ablation study found in Table 1, we employ  $L = 3$  isotropic Gaussians with standard deviation  $\sigma_l = 0.0375 \cdot 2^{l-1}$  for the proposal distribution (5). In addition to the standard IoU-Net, we compare with a version (denoted IoU-Net\*) employing the same proposal distribution and inference settings as in our approach. For both our method and IoU-Net\*, we set the refinement step-length  $\lambda$  using grid search on a separate validation set.

Our experiments are performed on the large-scale COCO benchmark [29]. We use the *2017 train* split ( $\approx 118\,000$  images) for training and the *2017 val* split ( $\approx 5\,000$  images) for setting our hyperparameters. The results are reported on the *2017 test-dev* split ( $\approx 20\,000$  images), in terms of the standard COCO metrics AP, AP<sub>50</sub> and AP<sub>75</sub>. We also report the inference speed in terms of frames-per-second (FPS) on a single NVIDIA TITAN Xp GPU. We initialize all networks in our comparison with the pre-trained Faster-RCNN weights, using the ResNet50-FPN [28] backbone, and re-train *only* the newly added layers for a fair comparison. The results are shown in Table 2. Our proposed method obtains the best results, significantly outperforming Faster-RCNN and IoU-Net by 2.2% and 1.1% in AP, respectively. The Gaussian model is outperformed by the mixture of 2 Gaussians, but note that adding more components does *not* further improve the performance. In comparison, the cVAE achieves somewhat improved performance, but is still clearly outperformed by our method. Compared to the probabilistic baselines, we believe that our energy-based model offers a more direct and effective representation of the underlying density via the scalar DNN output  $f_\theta(x, y)$ . The inference speed of our method is somewhat lower than that of Faster-RCNN, but identical to IoU-Net. How the number of iterations  $T$  in the gradient-based refinement affects inference speed and per-

**Table 3:** Results for the visual tracking task on the two common datasets TrackingNet [30] and UAV123 [31]. The symbol  $\dagger$  indicates an approximate value ( $\pm 1$ ), taken from the plot in the corresponding paper. Our proposed method significantly outperforms the baseline ATOM and other recent state-of-the-art trackers.

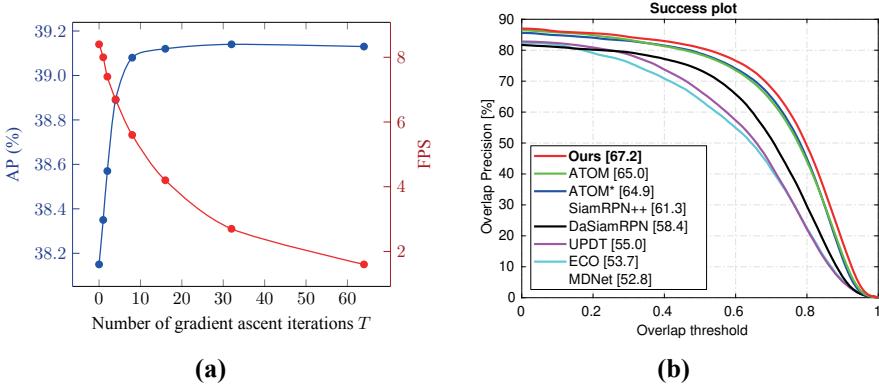
Dataset	Metric	ECO [53]	SiamFC [54]	MDNet [12]	UPDT [55]	DaSiamRPN [13]	SiamRPN++ [14]	ATOM [15]	ATOM* [15]	Ours
TrackingNet	Precision (%)	49.2	53.3	56.5	55.7	59.1	69.4	64.8	66.6	<b>69.7</b>
	Norm. Prec. (%)	61.8	66.6	70.5	70.2	73.3	80.0	77.1	78.4	<b>80.1</b>
	Success (%)	55.4	57.1	60.6	61.1	63.8	73.3	70.3	72.0	<b>74.5</b>
UAV123	OP <sub>0.50</sub> (%)	64.0	-	-	66.8	73.6	75 <sup>†</sup>	78.9	79.0	<b>80.8</b>
	OP <sub>0.75</sub> (%)	32.8	-	-	32.9	41.1	56 <sup>†</sup>	55.7	56.5	<b>60.2</b>
	AUC (%)	53.7	-	52.8	55.0	58.4	61.3	65.0	64.9	<b>67.2</b>

formance is analyzed in Figure 3a, showing that our choice  $T = 10$  provides a reasonable trade-off.

## 4.2 Visual Tracking

Next, we evaluate our approach on the problem of generic visual object tracking. The task is to estimate the bounding box of a target object in every frame of a video. The target object is defined by a given box in the first video frame. We employ the recently introduced ATOM [15] tracker as our baseline. Given the first-frame annotation, ATOM trains a classifier to first roughly localize the target in a new frame. The target bounding box is then determined using an IoU-Net-based module, which is also conditioned on the first-frame target appearance using a modulation-based architecture. We train our network to predict the conditional target density through  $f_\theta(x, y)$  in (1), using a network architecture *identical* to the baseline ATOM tracker. In particular, we employ the same bounding box parameterization as for object detection (Section 4.1) and sample  $M = 128$  boxes during training from a proposal distribution (5) generated by  $L = 2$  Gaussians with standard deviations  $\sigma_1 = 0.05$ ,  $\sigma_2 = 0.5$ . During tracking, we follow the same procedure as in ATOM, sampling 10 boxes in each frame followed by gradient ascent to refine the estimate generated by the classification module. The inference speed of our approach is thus identical to ATOM, running at over 30 FPS on a single NVIDIA GT-1080 GPU.

We demonstrate results on two standard tracking benchmarks: TrackingNet [30] and UAV123 [31]. TrackingNet contains challenging videos sampled from YouTube, with a test set of 511 videos. The main metric is the Success, defined as the average IoU overlap with the ground truth. UAV123 contains 123 videos captured from a UAV, and includes small and fast-moving objects. We report the overlap precision metric ( $OP_H$ ), defined as the percentage of frames having bounding box IoU overlap larger than a threshold  $H$ . The final AUC score is computed as the average OP over all thresholds  $H \in [0, 1]$ . Hyperpa-



**Figure 3:** (a) Impact of the number of gradient ascent iterations  $T$  on performance (AP) and inference speed (FPS), for the object detection task on the *2017 val* split of the COCO [29] dataset. (b) Success plot on the UAV123 [31] visual tracking dataset, showing the overlap precision  $OP_H$  as a function of the overlap threshold  $H$ .

rameters are set on the OTB [56] and NFS [57] datasets, containing 100 videos each. Due to the significant challenges imposed by the limited supervision and generic nature of the tracking problem, there are no competitive baselines employing direct bounding box regression. Current state-of-the-art employ either confidence-based regression, as in ATOM, or anchor-based bounding box regression techniques [13, 14]. We therefore only compare with the ATOM baseline and include other recent state-of-the-art methods in the comparison. As in Section 4.1, we also compare with a version (denoted ATOM\*) of the IoU-Net-based ATOM employing the same training and inference settings as our final approach. The results are shown in Table 3, and the success plot on UAV123 is shown in Figure 3b. Our approach achieves a significant 2.5% and 2.2% absolute improvement over ATOM on the overall metric on TrackingNet and UAV123, respectively. Note that the improvements are most prominent for high-accuracy boxes, as indicated by  $OP_{0.75}$ . Our approach also outperforms the recent SiamRPN++ [14], which employs anchor-based bounding box regression [1, 58] and a much deeper backbone network (ResNet50) compared to ours (ResNet18). Figure 1 (bottom) visualizes an illustrative example of the target density  $p(y|x; \theta) \propto e^{f_\theta(x,y)}$  predicted by our approach during tracking. As illustrated, it predicts flexible densities which qualitatively capture meaningful uncertainty in challenging cases.

### 4.3 Age Estimation

To demonstrate the general applicability of our proposed method, we also perform experiments on regression tasks not involving bounding boxes. In age

**Table 4:** Results for the age estimation task on the UTKFace [59] dataset. Gradient-based refinement using our proposed method consistently improves MAE (lower is better) for the age predictions produced by a variety of different baselines.

+Refine	Niu et al. [60]	Cao et al. [11]	Direct	Gaussian	Laplace	Softmax (CE, $L^2$ )	Softmax (CE, $L^2$ , Var)
✓	$5.74 \pm 0.05$	$5.47 \pm 0.01$	$4.81 \pm 0.02$	$4.79 \pm 0.06$	$4.85 \pm 0.04$	$4.78 \pm 0.05$	$4.81 \pm 0.03$

**Table 5:** Results for the head-pose estimation task on the BIWI [62] dataset. Gradient-based refinement using our proposed method consistently improves the average MAE (lower is better) for yaw, pitch and roll for the predicted pose produced by our baselines.

+Refine	Gu et al. [63]	Yang et al. [8]	Direct	Gaussian	Laplace	Softmax (CE, $L^2$ )	Softmax (CE, $L^2$ , Var)
✓	3.66	3.60	$3.09 \pm 0.07$	$3.12 \pm 0.08$	$3.21 \pm 0.06$	$3.04 \pm 0.08$	$3.15 \pm 0.07$

estimation, we are given a cropped image  $x \in \mathbb{R}^{h \times w \times 3}$  of a person’s face, and the task is to predict his/her age  $y \in \mathbb{R}_+$ . We utilize the UTKFace [59] dataset, specifically the subset of 16 434 images used by Cao et al. [11]. We also utilize the dataset split employed in [11], with 3 287 test images and 11 503 images for training. Additionally, we use 1 644 of the training images for validation. Methods are evaluated in terms of the Mean Absolute Error (MAE). The DNN architecture  $f_\theta(x, y)$  of our model first extracts ResNet50 [61] features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The age  $y$  is processed by four fully-connected layers, generating  $g_y \in \mathbb{R}^{128}$ . The two feature vectors are then concatenated and processed by two fully-connected layers, outputting  $f_\theta(x, y) \in \mathbb{R}$ . We apply our proposed method to refine the age predicted by baseline models, using the gradient ascent maximization of  $f_\theta(x, y)$  detailed in Section 3.3. All baseline DNN models employ a similar architecture, including an identical ResNet50 for feature extraction and the same number of fully-connected layers to output either the age  $y \in \mathbb{R}$  (*Direct*), mean and variance parameters for Gaussian and Laplace distributions, or to output logits for  $C$  discretized classes (*Softmax*). The results are found in Table 4. We observe that age refinement provided by our method consistently improves the accuracy of the predictions generated by the baselines. For *Direct*, e.g., this refinement marginally decreases inference speed from 49 to 36 FPS.

#### 4.4 Head-Pose Estimation

Lastly, we evaluate our method on the task of head-pose estimation. In this case, we are given an image  $x \in \mathbb{R}^{h \times w \times 3}$  of a person, and the aim is to predict the orientation  $y \in \mathbb{R}^3$  of his/her head, where  $y$  is the yaw, pitch and roll angles. We utilize the BIWI [62] dataset, specifically the processed dataset provided by Yang et al. [8], in which the images have been cropped to faces

detected using MTCNN [64]. We also employ protocol 2 as defined in [8], with 10 613 images for training and 5 065 images for testing. Additionally, we use 1 010 training images for validation. The methods are evaluated in terms of the average MAE for yaw, pitch and roll. The network architecture of the DNN  $f_\theta(x, y)$  defining our model takes the image  $x \in \mathbb{R}^{h \times w \times 3}$  and orientation  $y \in \mathbb{R}^3$  as inputs, but is otherwise identical to the age estimation case (Section 4.3). Our model is again evaluated by applying the gradient-based refinement to the predicted orientation  $y \in \mathbb{R}^3$  produced by a number of baseline models. We use the same baselines as for age estimation, and apart from minor changes required to increase the output dimension from 1 to 3, identical network architectures are also used. The results are found in Table 5, and also in this case we observe that refinement using our proposed method consistently improves upon the baselines.

## 5 Conclusion

We proposed a general and conceptually simple regression method with a clear probabilistic interpretation. It models the conditional target density  $p(y|x)$  by predicting the un-normalized density through a DNN  $f_\theta(x, y)$ , taking the input-target pair  $(x, y)$  as input. This energy-based model  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$  of  $p(y|x)$  is trained by directly minimizing the associated negative log-likelihood, employing Monte Carlo importance sampling to approximate the partition function  $Z(x, \theta)$ . At test time, targets are predicted by maximizing the DNN output  $f_\theta(x, y)$  w.r.t.  $y$  via gradient-based refinement. Extensive experiments performed on four diverse computer vision tasks demonstrate the high accuracy and wide applicability of our method. Future directions include exploring improved architectural designs, studying other regression applications, and investigating our proposed method’s potential for aleatoric uncertainty estimation.

**Acknowledgments** This research was supported by the Swedish Foundation for Strategic Research via *ASSEMBLE*, the Swedish Research Council via *Learning flexible models for nonlinear dynamics*, the ETH Zürich Fund (OK), a Huawei Technologies Oy (Finland) project, an Amazon AWS grant, and Nvidia.

## References

- [1] S. Ren, K. He, R. B. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2015).

- [2] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. “Acquisition of localization confidence for accurate object detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [3] H. Law and J. Deng. “CornerNet: Detecting objects as paired keypoints.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [4] X. Zhou, J. Zhuo, and P. Krahenbuhl. “Bottom-up object detection by grouping extreme and center points.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [5] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. “Realtime multi-person 2d pose estimation using part affinity fields.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [6] B. Xiao, H. Wu, and Y. Wei. “Simple baselines for human pose estimation and tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [7] K. Sun, B. Xiao, D. Liu, and J. Wang. “Deep high-resolution representation learning for human pose estimation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [8] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. “FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation from a Single Image.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [9] R. Rothe, R. Timofte, and L. Van Gool. “Deep expectation of real and apparent age from a single image without facial landmarks.” In: *International Journal of Computer Vision (IJCV)* (2016).
- [10] H. Pan, H. Han, S. Shan, and X. Chen. “Mean-variance loss for deep age estimation from a face.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [11] W. Cao, V. Mirjalili, and S. Raschka. “Rank-consistent Ordinal Regression for Neural Networks.” In: *arXiv preprint arXiv:1901.07884* (2019).
- [12] H. Nam and B. Han. “Learning multi-domain convolutional neural networks for visual tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [13] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. “Distractor-aware siamese networks for visual object tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [14] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. “SiamRPN++: Evolution of siamese visual tracking with very deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

- [15] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. “ATOM: Accurate tracking by overlap maximization.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [16] M. Niethammer, Y. Huang, and F.-X. Vialard. “Geodesic regression for image time-series.” In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2011.
- [17] C.-R. Chou, B. Frederick, G. Mageras, S. Chang, and S. Pizer. “2D/3D image registration using regression learning.” In: *Computer Vision and Image Understanding* (2013).
- [18] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [19] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [20] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [21] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [22] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [23] A. Varamesh and T. Tuytelaars. “Mixture Dense Regression for Object Detection and Human Pose Estimation.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [24] S. Prokudin, P. Gehler, and S. Nowozin. “Deep directional statistics: Pose estimation with uncertainty quantification.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [25] E. Ilg, O. Cicek, S. Galessio, A. Klein, O. Makansi, F. Hutter, and T. Bro. “Uncertainty estimates and multi-hypotheses networks for optical flow.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [26] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. “Convolutional pose machines.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [27] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).

- [28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature pyramid networks for object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. “Microsoft COCO: Common objects in context.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.
- [30] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem. “TrackingNet: A large-scale dataset and benchmark for object tracking in the wild.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [31] M. Mueller, N. Smith, and B. Ghanem. “A benchmark and simulator for UAV tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.
- [32] P. J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* (1964).
- [33] O. Makansi, E. Ilg, O. Cicek, and T. Brox. “Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [34] D. Feng, L. Rosenbaum, F. Timm, and K. Dietmayer. “Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3D object detection.” In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019.
- [35] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. “Bounding box regression with uncertainty for accurate object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [36] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2020.
- [37] C. M. Bishop. *Mixture density networks*. 1994.
- [38] K. Sohn, H. Lee, and X. Yan. “Learning structured output representation using deep conditional generative models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.
- [39] I. Tschantzidis, T. Joachims, T. Hofmann, and Y. Altun. “Large margin methods for structured and interdependent output variables.” In: *Journal of Machine Learning Research (JMLR)* (2005).

- [40] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. “Hand keypoint detection in single images using multiview bootstrapping.” In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [41] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele. “DeepCut: Joint subset partition and labeling for multi person pose estimation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [42] T.-Y. Yang, Y.-H. Huang, Y.-Y. Lin, P.-C. Hsiu, and Y.-Y. Chuang. “SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation.” In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018.
- [43] N. Ruiz, E. Chong, and J. M. Rehg. “Fine-grained head pose estimation without keypoints.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.
- [44] R. Diaz and A. Marathe. “Soft Labels for Ordinal Regression.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [45] A. Mnih and G. Hinton. “Learning nonlinear constraints with contrastive backpropagation.” In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE. 2005.
- [46] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. “Unsupervised discovery of nonlinear structure using contrastive backpropagation.” In: *Cognitive science* (2006).
- [47] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. “A theory of generative convnet.” In: *International Conference on Machine Learning (ICML)*. 2016.
- [48] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. “Learning generative convnets via multi-grid modeling and sampling.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [49] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [50] D. Lawson, G. Tucker, B. Dai, and R. Ranganath. “Energy-Inspired Models: Learning with Sampler-Induced Distributions.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [51] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. “On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models.” In: *Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.

- [52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [53] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg. “ECO: Efficient convolution operators for tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [54] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. “Fully-Convolutional Siamese Networks for Object Tracking.” In: *European Conference on Computer Vision (ECCV) Workshops*. 2016.
- [55] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg. “Unveiling the power of deep tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [56] Y. Wu, J. Lim, and M.-H. Yang. “Object tracking benchmark.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. 2015.
- [57] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. “Need for speed: A benchmark for higher frame rate object tracking.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
- [58] J. Redmon and A. Farhadi. “YOLO9000: better, faster, stronger.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [59] Z. Zhang, Y. Song, and H. Qi. “Age progression/regression by conditional adversarial autoencoder.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [60] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. “Ordinal regression with multiple output cnn for age estimation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [61] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [62] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. “Random forests for real time 3d face analysis.” In: *International Journal of Computer Vision (IJCV)* (2013).
- [63] J. Gu, X. Yang, S. De Mello, and J. Kautz. “Dynamic facial analysis: From Bayesian filtering to recurrent neural network.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

- [64] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. “Joint face detection and alignment using multitask cascaded convolutional networks.” In: *IEEE Signal Processing Letters* (2016).
- [65] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [66] R. B. Girshick. “Fast R-CNN.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015).
- [67] F. Massa and R. Girshick. *maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch*. <https://github.com/facebookresearch/maskrcnn-benchmark>. Accessed: 04/09/2019. 2018.
- [68] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. “Mask R-CNN.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).
- [69] M. Danelljan and G. Bhat. *PyTracking: Visual tracking library based on PyTorch*. <https://github.com/visionml/pytracking>. Accessed: 30/04/2020. 2019.
- [70] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. “LaSOT: A high-quality benchmark for large-scale single object tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [71] L. Huang, X. Zhao, and K. Huang. “GOT-10k: A large high-diversity benchmark for generic object tracking in the wild.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).

## Supplementary Material

In this supplementary material, we provide additional details and results. It consists of Appendix A - Appendix F. Appendix A contains a detailed algorithm for our employed prediction procedure. Appendix B contains details on the illustrative 1D regression problem in Figure 2 in the main paper. Further details on the employed training and inference procedures are provided in Appendix C for the object detection experiments, and in Appendix D for the visual tracking experiments. Lastly, Appendix E and Appendix F contain details and full results for the experiments on age estimation and head-pose estimation, respectively. Note that equations, tables, figures and algorithms in this supplementary document are numbered with the prefix “S”. Numbers without this prefix refer to the main paper.

### A Prediction Algorithm

Our prediction procedure (Section 3.3) is detailed in Algorithm S1, where  $\lambda$  denotes the gradient ascent step-length,  $\eta$  is a decay of the step-length and  $T$  is the number of iterations. In our experiments, we fix  $T$  (typically,  $T = 10$ ) and select  $\{\lambda, \eta\}$  using grid search on a validation set.

---

**Algorithm S1** Prediction via gradient-based refinement.

---

**Input:**  $x^*, \hat{y}, T, \lambda, \eta$ .

```
1:  $y \leftarrow \hat{y}$ .
2: for  $t = 1, \dots, T$  do
3:   PrevValue  $\leftarrow f_\theta(x^*, y)$ .
4:    $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ .
5:   NewValue  $\leftarrow f_\theta(x^*, \tilde{y})$ .
6:   if NewValue > PrevValue then
7:      $y \leftarrow \tilde{y}$ .
8:   else
9:      $\lambda \leftarrow \eta \lambda$ .
10:  Return  $y$ .
```

---

### B Illustrative Example

The ground truth conditional target density  $p(y|x)$  in Figure 2 is defined by a mixture of two Gaussian components (with weights 0.2 and 0.8) for  $x < 0$ , and a log-normal distribution (with  $\mu = 0.0$ ,  $\sigma = 0.25$ ) for  $x \geq 0$ . The

training data  $\{(x_i, y_i)\}_{i=1}^{2000}$  was generated by uniform random sampling of  $x$ ,  $x_i \sim U(-3, 3)$ . Both models were trained for 75 epochs with a batch size of 32 using the ADAM [65] optimizer.

The Gaussian model is defined using a DNN  $f_\theta(x)$  according to,

$$\begin{aligned} p(y|x; \theta) &= \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x)), \\ f_\theta(x) &= [\mu_\theta(x) \quad \log \sigma_\theta^2(x)]^\top \in \mathbb{R}^2. \end{aligned} \quad (\text{S1})$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \mu_\theta(x_i))^2}{\sigma_\theta^2(x_i)} + \log \sigma_\theta^2(x_i). \quad (\text{S2})$$

The DNN  $f_\theta$  is a simple feed-forward neural network, containing two shared fully-connected layers (dimensions:  $1 \rightarrow 10$ ,  $10 \rightarrow 10$ ) and two identical heads for  $\mu$  and  $\log \sigma^2$  of three fully-connected layers ( $10 \rightarrow 10$ ,  $10 \rightarrow 10$ ,  $10 \rightarrow 1$ ).

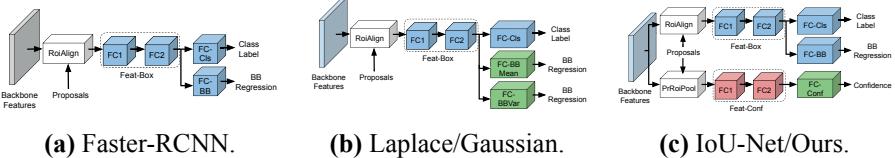
Our proposed model  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$  (Eq. 1 in the paper) is defined using a feed-forward neural network  $f_\theta(x, y)$  containing two fully-connected layers ( $1 \rightarrow 10$ ,  $10 \rightarrow 10$ ) for both  $x$  and  $y$ , and three fully-connected layers ( $20 \rightarrow 10$ ,  $10 \rightarrow 10$ ,  $10 \rightarrow 1$ ) processing the concatenated  $(x, y)$  feature vector. It is trained using  $M = 1024$  samples from a proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper) with  $L = 2$  and variances  $\sigma_1^2 = 0.1^2$ ,  $\sigma_2^2 = 0.8^2$ .

## C Object Detection

Here, we provide further details about the network architectures, training procedure, and hyperparameters used for our experiments on object detection (Section 4.1 in the paper).

### C.1 Network Architecture

We use the Faster-RCNN [1] detector with ResNet50-FPN [28] as our baseline. As visualized in Figure S1a, Faster-RCNN generates object proposals using a region proposal network (RPN). The features from the proposal regions are then pooled to a fixed-sized feature map using the RoiPool layer [66]. The pooled features are then passed through a feature extractor (denoted Feat-Box) consisting of two fully-connected (FC) layers. The output feature vector is then passed through two parallel FC layers, one which predicts the class label (denoted FC-Cls), and another which regresses the offsets between the proposal



**Figure S1:** Network architectures for the different object detection networks used in our experiments (Section 4.1 in the paper). The backbone feature extractor (ResNet50-FPN), and the region proposal network (RPN) is not shown for clarity. We do not train the blocks in blue color, using the pre-trained Faster-RCNN weights from [67] instead. The blocks in red are initialized with the pre-trained Faster-RCNN weights and fine-tuned. The blocks in green on the other hand are trained from scratch.

and the ground truth box (denoted FC-BB). We use the PyTorch implementation for Faster-RCNN from [67]. Note that we use the RoiAlign [68] layer instead of RoiPool in our experiments as it has been shown to achieve better performance [68].

For the Gaussian and Laplace probabilistic models (Gaussian and Laplace in Table 2 in the paper), we replace the FC-BB layer in Faster-RCNN with two parallel FC layers, denoted FC-BBMean and FC-BBVar, which predict the mean and the log-variance of the distribution modeling the offset between the proposal and the ground truth box for each coordinate. This architecture is shown in Figure S1b. For the mixtures of  $K = \{2, 4, 8\}$  Gaussians, we duplicate FC-BBMean and FC-BBVar  $K$  times, and add an FC layer for predicting the  $K$  component weights. For the cVAE, FC-BBMean and FC-BBVar instead outputs the mean and log-variance of a Gaussian distribution for the latent variable  $z \in \mathbb{R}^{10}$ . Duplicates of FC-BBMean and FC-BBVar, modified to also take sampled  $z$  values as input, then predicts the mean and log-variance of the distribution modeling the bounding box offset.

For our confidence-based IoU-Net [2] models (IoU-Net and IoU-Net\* in Table 2), we use the same network architecture as employed in the original paper, shown in Figure S1c. That is, we add an additional branch to predict the IoU overlap between the proposal box and the ground truth. This branch uses the PrRoiPool [2] layer to pool the features from the proposal regions. The pooled features are passed through a feature extractor (denoted Feat-Conf) consisting of two FC layers. The output feature vector is passed through another FC layer, FC-Conf, which predicts the IoU. We use an identical architecture for our approach, but train it to output  $f_\theta(x, y)$  in  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$  instead.

## C.2 Training

We use the pre-trained weights for Faster-RCNN from [67]. Note that the bounding box regression in Faster-RCNN is trained using a direct method, with an Huber loss [32]. We trained the other networks in Table 2 in the paper (Gaussian, Gaussian Mixt. 2, Gaussian Mixt. 4, Gaussian Mixt. 8, Gaussian cVAE, Laplace, IoU-Net, IoU-Net\* and Ours) on the MS-COCO [29] training split (*2017 train*) using stochastic gradient descent (SGD) with a batch size of 16 for 60k iterations. The base learning rate  $lr_{\text{base}}$  is reduced by a factor of 10 after 40k and 50k iterations, for all the networks. We also warm up the training by linearly increasing the learning rate from  $\frac{1}{3}lr_{\text{base}}$  to  $lr_{\text{base}}$  during the first 500 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. For all the networks, we only trained the newly added layers, while keeping the backbone and the region proposal network fixed.

For the Gaussian, mixture of Gaussians, cVAE and Laplace models, we only train the final predictors (FC-BBMean and FC-BBVar), while keeping the class predictor (FC-Cls) and the box feature extractor (Feat-Box) fixed. We also tried fine-tuning the FC-Cls and Feat-Box weights for the Gaussian model, with different learning rate settings, but obtained worse performance on the validation set. The weights for both FC-BBMean and FC-BBVar were initialized with zero mean Gaussian with standard deviation of 0.001. All these models were trained with a base learning rate  $lr_{\text{base}} = 0.005$  by minimizing the negative log-likelihood, except for the cVAE which is trained by maximizing the ELBO (using 128 sampled  $z$  values to approximate the expectation).

For the IoU-Net, IoU-Net\* and our proposed model, we only trained the newly added confidence branch. We found it beneficial to initialize the feature extractor block (Feat-Conf) with the corresponding weights from Faster-RCNN, i.e. the Feat-Box block. The weights for the predictor FC-Conf were initialized with zero mean Gaussian with standard deviation of 0.001. As in the original paper [2], we used a base learning rate  $lr_{\text{base}} = 0.01$  for the IoU-Net and IoU-Net\* networks. For our proposed model, we used  $lr_{\text{base}} = 0.001$  due to the different scaling of the loss. Note that we did not perform any parameter tuning for setting the learning rates. We generate 128 proposals for each ground truth box during training. For the IoU-Net, we use the proposal generation strategy mentioned in the original paper [2]. That is, for each ground truth box, we generate a large set of candidate boxes which have an IoU overlap of at least 0.5 with the ground truth, and uniformly sample 128 proposals from this candidate set w.r.t. the IoU. For IoU-Net\* and our model, we sample boxes from a proposal distribution (Eq. 5 in the paper) generated by  $L = 3$  Gaussians with standard deviations of 0.0375, 0.075 and 0.15. The IoU-Net and IoU-Net\* are trained by minimizing the Huber loss between the predicted IoU and the ground truth, while our model is trained by minimizing the negative log likelihood of the training data (Eq. 4 in the paper).

**Table S1:** Impact of  $L$  and  $\{\sigma_l\}_{l=1}^L$  in the proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper), for the object detection task on the 2017 *val* split of COCO [29].

$L$	$\{\sigma_l\}_{l=1}^L$	AP (%)
1	{0.01875}	38.07
	{0.0375}	38.47
	{0.075}	37.52
	{0.15}	35.05
2	{0.025, 0.1}	38.97
	{0.0375, 0.15}	39.05
	{0.04375, 0.175}	39.07
	{0.05, 0.2}	39.02
2	{0.0125, 0.025}	38.19
	{0.025, 0.05}	38.65
	{0.075, 0.15}	37.14
3	{0.0125, 0.025, 0.05}	38.61
	{0.025, 0.05, 0.1}	38.95
	{0.0375, 0.075, 0.15}	<b>39.11</b>
	{0.04375, 0.0875, 0.175}	39.00
	{0.05, 0.1, 0.2}	38.76
	{0.0625, 0.125, 0.25}	37.96
	{0.075, 0.15, 0.3}	37.42

### C.3 Analysis of Proposal Distribution

An extensive ablation study for the number of components  $L$  and standard deviations  $\{\sigma_l\}_{l=1}^L$  in the proposal distribution  $q(y|y_i) = \frac{1}{L} \sum_{l=1}^L \mathcal{N}(y; y_i, \sigma_l^2)$  (Eq. 5 in the paper) is provided in Table S1, which is an extended version of Table 1 in the paper. We find that  $L = 1$  downgrades performance, while there is no significant difference between  $L = 2$  and  $L = 3$ . For  $L \in \{2, 3\}$ , the results are not particularly sensitive to the specific choice of  $\{\sigma_l\}_{l=1}^L$ , but benefits from including *both* small and relatively large values in  $\{\sigma_l\}_{l=1}^L$ .

### C.4 Inference

The inference in both the Gaussian and Laplace models is identical to the one employed by Faster-RCNN, except the output mean is taken as the prediction. Thus, we do not utilize the output variances during inference. For the mixture of  $K = \{2, 4, 8\}$  Gaussians, we compute the mean of the distribution and take that as our prediction. Instead picking the component with the largest weight and taking its mean as the prediction resulted in somewhat worse validation performance. For cVAE, we approximately compute the mean (using 128 samples) of the distribution and take that as our prediction.

**Table S2:** Impact of  $L$  and  $\{\sigma_l\}_{l=1}^L$  in the proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper), for the visual tracking task on the combined OTB [56] and NFS [57] datasets.

$L$	$\{\sigma_l\}_{l=1}^L$	OP <sub>0.50</sub> (%)	OP <sub>0.75</sub> (%)	AUC (%)
1	{0.05}	75.77	45.72	63.37
2	{0.01, 0.1}	77.25	46.09	61.48
2	{0.03, 0.3}	79.27	48.59	63.65
2	{0.05, 0.5}	<b>79.90</b>	<b>48.71</b>	<b>64.10</b>
2	{0.07, 0.7}	78.41	47.72	62.75

For IoU-Net and IoU-Net\*, we perform IoU-Guided NMS as described in [2], followed by gradient-based refinement (Algorithm S1). For our proposed approach we adopt the same NMS technique, but guide it with the values  $f_\theta(x, y)$  predicted by our network instead. We use a step-length  $\lambda = 0.5$  and step-length decay  $\eta = 0.1$  for IoU-Net. For IoU-Net\* and our approach we perform the gradient-based refinement in the relative bounding box parametrization  $y = (c_x/w_0, c_y/h_0, \log w, \log h)$  (see Section 4.1 in the paper). Here, we employ different step-lengths for position and size. For IoU-Net\*, we use  $\lambda = 0.002$  and  $\lambda = 0.008$  respectively, with a decay of  $\eta = 0.2$ . For our proposed approach, we use  $\lambda = 0.0001$  and  $\lambda = 0.0004$  with  $\eta = 0.5$ . For all methods, these hyperparameters ( $\lambda$  and  $\eta$ ) were set using a grid search on the COCO validation split (*2017 val*). We used  $T = 10$  refinement iterations for each of the three models. Note that since a given image  $x$  can have multiple ground truth instances, multiple bounding boxes are usually refined. The gradient-based refinement then moves each individual box  $y$  towards the maximum of a *local* mode in  $f_\theta(x, y)$ . Thus, they will not converge to a single solution. Also note that  $f_\theta(x, y)$  is class-conditional (as in the IoU-Net baseline), eliminating the risk of confusing neighboring objects of different classes.

## D Visual Tracking

Here, we provide further details about the training procedure and hyperparameters used for our experiments on visual object tracking (Section 4.2 in the paper).

### D.1 Training

We adopt the ATOM [15] tracker as our baseline, and use the PyTorch implementation and pre-trained weights from [69]. ATOM trains an IoU-Net-based module to predict the IoU overlap between a candidate box and the ground truth, conditioned on the first-frame target appearance. The IoU predictor is

trained by generating 16 candidates for each ground truth box. The candidates are generated by adding a Gaussian noise for each ground truth box coordinate, while ensuring a minimum IoU overlap of 0.1 between the candidate box and the ground truth. The network is trained by minimizing the squared error ( $L^2$  loss) between the predicted and ground truth IoU.

Our proposed model is instead trained by sampling 128 candidate boxes from a proposal distribution (Eq. 5 in the paper) generated by  $L = 2$  Gaussians with standard deviations of 0.05 and 0.5, and minimizing the negative log likelihood of the training data. An ablation study for the proposal distribution is found in Table S2. We use the training splits of the TrackingNet [30], LaSOT [70], GOT10k [71], and COCO datasets for our training. Our network is trained for 50 epochs, using the ADAM optimizer with a base learning rate of 0.001 which is reduced by a factor of 5 after every 15 epochs. The rest of the training parameters are exactly the same as in ATOM. The ATOM\* model is trained by using the exact same proposal distribution, datasets and settings. It only differs by the loss, which is the same squared error between the predicted and ground truth IoU as in the original ATOM.

## D.2 Inference

During tracking, the ATOM tracker first applies the classification head network, which is trained online, to coarsely localize the target object. 10 random boxes are then sampled around this prediction, to be refined by the IoU prediction network. We only alter the final bounding box refinement step of the 10 given random initial boxes, and preserve all other settings as in the original ATOM tracker. The original version performs  $T = 5$  gradient ascent iterations with a step length of  $\lambda = 1.0$ . For our proposed model and the ATOM\* version, we use  $T = 10$  iterations, employing the bounding box parameterization described in Section 4.1. For our approach, we set the step length to  $\lambda = 2 \cdot 10^{-4}$  for position and  $\lambda = 10^{-3}$  for size dimensions. For ATOM\*, we use  $\lambda = 10^{-2}$  for position and  $\lambda = 5 \cdot 10^{-2}$  for size dimensions. These parameters were set on the separate validation set. For simplicity, we adopt the vanilla gradient ascent strategy employed in ATOM for the two other methods as well. That is, we have no decay ( $\eta = 1$ ) and do not perform checks whether the confidence score is increasing in each iteration.

## D.3 Qualitative Results

Illustrative examples of the target density  $p(y|x; \theta) \propto e^{f_\theta(x,y)}$  predicted by our approach during tracking are visualized in Figure S2.



**Figure S2:** Visualization of the conditional target density  $p(y|x; \theta) \propto e^{f_\theta(x, y)}$  predicted by our network for the task of bounding box estimation in visual tracking. Since the target space  $y \in \mathbb{R}^4$  is 4-dimensional, we visualize the density for different locations of the top-right corner as a heatmap, while the bottom-left is kept fixed at the tracker output (red box). Our network predicts flexible densities which qualitatively capture meaningful uncertainty in challenging cases.

**Table S3:** Impact of  $\{\sigma_l\}_{l=1}^2$  in the proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper), for the age estimation task on our validation split of the UTKFace [59] dataset.

$\{\sigma_l\}_{l=1}^2$	MAE
{0.1, 10}	7.62
{0.1, 20}	<b>5.12</b>
{0.01, 20}	5.36
{0.1, 40}	5.24

## E Age Estimation

In this appendix, further details on the age estimation experiments (Section 4.3 in the paper) are provided.

### E.1 Network Architecture

The DNN architecture  $f_\theta(x, y)$  of our proposed model first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The age  $y$  is processed by four fully-connected layers (dimensions:  $1 \rightarrow 16, 16 \rightarrow 32, 32 \rightarrow 64, 64 \rightarrow 128$ ), generating  $g_y \in \mathbb{R}^{128}$ . The two feature vectors  $g_x, g_y$  are then concatenated to form  $g_{x,y} \in \mathbb{R}^{2048+128}$ , which is processed by two fully-connected layers ( $2048 + 128 \rightarrow 2048, 2048 \rightarrow 1$ ), outputting  $f_\theta(x, y) \in \mathbb{R}$ .

### E.2 Training

Our model is trained using  $M = 1024$  samples from a proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper) with  $L = 2$  and variances  $\sigma_1^2 = 0.1^2$ ,  $\sigma_2^2 = 20^2$ . An ablation study for the variances is found in Table S3. The

model is trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. The images  $x$  are of size  $200 \times 200$ . For data augmentation, we use random flipping along the vertical axis and random scaling in the range  $[0.7, 1.4]$ . After random flipping and scaling, a random image crop of size  $200 \times 200$  is also selected. The ResNet50 is imported from `torchvision.models` in PyTorch with the pretrained option set to true, all other network parameters are randomly initialized using the default initializer in PyTorch.

### E.3 Prediction

For this experiment, we use a slight variation of Algorithm S1, which is found in Algorithm S2. There,  $T$  is the number of gradient ascent iterations,  $\lambda$  is the stepsize,  $\Omega_1$  is an early-stopping threshold and  $\Omega_2$  is a degeneration tolerance. Following IoU-Net, we set  $T = 5$ ,  $\Omega_1 = 0.001$  and  $\Omega_2 = -0.01$ . Based on the validation set, we select  $\lambda = 3$ . We refine a single estimate  $\hat{y}$ , predicted by each baseline model.

---

**Algorithm S2** Prediction via gradient-based refinement (variation).

---

**Input:**  $x^*$ ,  $\hat{y}$ ,  $T$ ,  $\lambda$ ,  $\Omega_1$ ,  $\Omega_2$ .

```

1:  $y \leftarrow \hat{y}$ .
2: for  $t = 1, \dots, T$  do
3:    $\text{PrevValue} \leftarrow f_\theta(x^*, y)$ .
4:    $y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ .
5:    $\text{NewValue} \leftarrow f_\theta(x^*, y)$ .
6:   if  $|\text{PrevValue} - \text{NewValue}| < \Omega_1$  or  $(\text{NewValue} - \text{PrevValue}) < \Omega_2$  then
7:     Return  $y$ .
8: Return  $y$ .

```

---

### E.4 Baselines

All baselines are trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. Identical data augmentation and parameter initialization as for our proposed model is used.

**Direct** The DNN architecture of *Direct* first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 1$ ), outputting the prediction  $\hat{y} \in \mathbb{R}$ . It is trained by minimizing either the Huber or  $L^2$  loss.

**Table S4:** Full results for the age estimation experiments. Gradient-based refinement using our proposed method consistently improves MAE (lower is better) for the age predictions outputted by a number of baselines.

Method	MAE
Niu et al. [60]	$5.74 \pm 0.05$
Cao et al. [11]	$5.47 \pm 0.01$
Direct - Huber	$4.80 \pm 0.06$
Direct - Huber + Refinement	$4.74 \pm 0.06$
Direct - L2	$4.81 \pm 0.02$
Direct - L2 + Refinement	$4.65 \pm 0.02$
Gaussian	$4.79 \pm 0.06$
Gaussian + Refinement	$4.66 \pm 0.04$
Laplace	$4.85 \pm 0.04$
Laplace + Refinement	$4.81 \pm 0.04$
Softmax - CE & L2	$4.78 \pm 0.05$
Softmax - CE & L2 + Refinement	$4.65 \pm 0.04$
Softmax - CE, L2 & Var	$4.81 \pm 0.03$
Softmax - CE, L2 & Var + Refinement	$4.69 \pm 0.03$

**Gaussian** The Gaussian model is defined using a DNN  $f_\theta(x)$  according to,

$$\begin{aligned} p(y|x; \theta) &= \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x)), \\ f_\theta(x) &= [\mu_\theta(x) \quad \log \sigma_\theta^2(x)]^\top \in \mathbb{R}^2. \end{aligned} \quad (\text{S3})$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{(y_i - \mu_\theta(x_i))^2}{\sigma_\theta^2(x_i)} + \log \sigma_\theta^2(x_i). \quad (\text{S4})$$

The DNN architecture of  $f_\theta(x)$  first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two heads of two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 1$ ) to output  $\mu_\theta(x)$  and  $\log \sigma_\theta^2(x)$ . The mean  $\mu_\theta(x)$  is taken as the prediction  $\hat{y}$ .

**Laplace** The Laplace model is defined using a DNN  $f_\theta(x)$  according to,

$$\begin{aligned} p(y|x; \theta) &= \frac{1}{2\beta_\theta(x)} \exp \left\{ -\frac{|y - \mu_\theta(x)|}{\beta_\theta(x)} \right\}, \\ f_\theta(x) &= [\mu_\theta(x) \quad \log \beta_\theta(x)]^\top \in \mathbb{R}^2. \end{aligned} \quad (\text{S5})$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \mu_\theta(x_i)|}{\beta_\theta(x_i)} + \log \beta_\theta(x_i). \quad (\text{S6})$$

The DNN architecture of  $f_\theta(x)$  first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two heads of two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 1$ ) to output  $\mu_\theta(x)$  and  $\log \beta_\theta(x)$ . The mean  $\mu_\theta(x)$  is taken as the prediction  $\hat{y}$ .

**Softmax** The DNN architecture of *Softmax* first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow C$ ), outputting logits for  $C = 101$  discretized classes  $\{0, 1, \dots, 100\}$ . It is trained by minimizing either the cross-entropy (CE) and  $L^2$  losses,  $J = J_{CE} + 0.1J_{L^2}$ , or the CE,  $L^2$  and variance [10] losses,  $J = J_{CE} + 0.1J_{L^2} + 0.05J_{Var}$ . The prediction  $\hat{y}$  is computed as the softmax expected value.

## E.5 Full Results

Full experiment results, extending the results found in Table 4 (Section 4.3 in the paper), are provided in Table S4.

## F Head-Pose Estimation

In this appendix, further details on the head-pose estimation experiments (Section 4.4 in the paper) are provided.

### F.1 Network Architecture

The DNN architecture  $f_\theta(x, y)$  of our proposed model first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The pose  $y \in \mathbb{R}^3$  is processed by four fully-connected layers (dimensions:  $3 \rightarrow 16$ ,  $16 \rightarrow 32$ ,  $32 \rightarrow 64$ ,  $64 \rightarrow 128$ ), generating  $g_y \in \mathbb{R}^{128}$ . The two feature vectors  $g_x, g_y$  are then concatenated to form  $g_{x,y} \in \mathbb{R}^{2048+128}$ , which is processed by two fully-connected layers ( $2048 + 128 \rightarrow 2048$ ,  $2048 \rightarrow 1$ ), outputting  $f_\theta(x, y) \in \mathbb{R}$ .

### F.2 Training

Our model is trained using  $M = 1024$  samples from a proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper) with  $L = 2$  and variances  $\sigma_1^2 = 1^2$ ,  $\sigma_2^2 = 20^2$  for yaw, pitch and roll. An ablation study for the variances is found in Table S5. The model is trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. The images  $x$  are of size  $64 \times 64$ . For

**Table S5:** Impact of  $\{\sigma_l\}_{l=1}^2$  in the proposal distribution  $q(y|y_i)$  (Eq. 5 in the paper), for the head-pose estimation task on our validation split of the BIWI [62] dataset.

$\{\sigma_l\}_{l=1}^2$	Average MAE
{0.1, 20}	6.96
{1, 20}	<b>5.08</b>
{1, 30}	5.24
{2, 20}	7.02
{1, 10}	7.56

data augmentation, we use random flipping along the vertical axis and random scaling in the range [0.7, 1.4]. After random flipping and scaling, a random image crop of size  $64 \times 64$  is also selected. The ResNet50 is imported from `torchvision.models` in PyTorch with the pretrained option set to true, all other network parameters are randomly initialized using the default initializer in PyTorch.

### F.3 Prediction

For this experiment, we also use the prediction procedure detailed in Algorithm S2. Again following IoU-Net, we set  $T = 5$ ,  $\Omega_1 = 0.001$  and  $\Omega_2 = -0.01$ . Based on the validation set, we select  $\lambda = 0.1$ . We refine a single estimate  $\hat{y}$ , predicted by each baseline model.

### F.4 Baselines

All baselines are trained for 75 epochs with a batch size of 32, using the ADAM optimizer with weight decay of 0.001. Identical data augmentation and parameter initialization as for our proposed model is used.

**Direct** The DNN architecture of *Direct* first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 3$ ), outputting the prediction  $\hat{y} \in \mathbb{R}^3$ . It is trained by minimizing either the Huber or  $L^2$  loss.

**Table S6:** Full results for the head-pose estimation experiments. Gradient-based refinement using our proposed method consistently improves the average MAE for yaw, pitch, roll (lower is better) for the predicted poses outputted by a number of baselines.

Method	Yaw MAE	Pitch MAE	Roll MAE	Avg. MAE
Yang et al. [42]	4.24	4.35	4.19	4.26
Gu et al. [63]	3.91	4.03	3.03	3.66
Yang et al. [8]	2.89	4.29	3.60	3.60
Direct - Huber	$2.78 \pm 0.09$	$3.73 \pm 0.13$	$2.90 \pm 0.09$	$3.14 \pm 0.07$
Direct - Huber + Refine.	$2.75 \pm 0.08$	$3.70 \pm 0.11$	$2.87 \pm 0.09$	$3.11 \pm 0.06$
Direct - L2	$2.81 \pm 0.08$	$\textbf{3.60} \pm 0.14$	$2.85 \pm 0.08$	$3.09 \pm 0.07$
Direct - L2 + Refine.	$2.78 \pm 0.08$	$3.62 \pm 0.13$	$2.81 \pm 0.08$	$3.07 \pm 0.07$
Gaussian	$2.89 \pm 0.09$	$3.64 \pm 0.13$	$2.83 \pm 0.09$	$3.12 \pm 0.08$
Gaussian + Refine.	$2.84 \pm 0.08$	$3.67 \pm 0.12$	$2.81 \pm 0.08$	$3.11 \pm 0.07$
Laplace	$2.93 \pm 0.08$	$3.80 \pm 0.15$	$2.90 \pm 0.07$	$3.21 \pm 0.06$
Laplace + Refine.	$2.89 \pm 0.07$	$3.81 \pm 0.13$	$2.88 \pm 0.06$	$3.19 \pm 0.06$
Softmax - CE & L2	$2.73 \pm 0.09$	$3.63 \pm 0.13$	$2.77 \pm 0.11$	$3.04 \pm 0.08$
Softmax - CE & L2 + Refine.	$\textbf{2.67} \pm 0.08$	$3.61 \pm 0.12$	$\textbf{2.75} \pm 0.10$	$\textbf{3.01} \pm 0.07$
Softmax - CE, L2 & Var	$2.83 \pm 0.12$	$3.79 \pm 0.10$	$2.84 \pm 0.11$	$3.15 \pm 0.07$
Softmax - CE, L2 & Var + Refine.	$2.76 \pm 0.10$	$3.74 \pm 0.09$	$2.83 \pm 0.10$	$3.11 \pm 0.06$

**Gaussian** The Gaussian model is defined using a DNN  $f_\theta(x)$  according to,

$$\begin{aligned}
p(y|x; \theta) &= \mathcal{N}(y; \mu_\theta(x), \Sigma_\theta(x)), \quad \Sigma_\theta(x) = \text{diag}(\sigma_\theta^2(x)), \\
y &= [y_1 \quad y_2 \quad y_3]^\top \in \mathbb{R}^3, \\
\mu_\theta(x) &= [\mu_{1,\theta}(x) \quad \mu_{2,\theta}(x) \quad \mu_{3,\theta}(x)]^\top \in \mathbb{R}^3, \\
\sigma_\theta^2(x) &= [\sigma_{1,\theta}^2(x) \quad \sigma_{2,\theta}^2(x) \quad \sigma_{3,\theta}^2(x)]^\top \in \mathbb{R}^3, \\
f_\theta(x) &= [\mu_\theta(x)^\top \quad \log \sigma_\theta^2(x)^\top]^\top \in \mathbb{R}^6.
\end{aligned} \tag{S7}$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \left( \sum_{k=1}^3 \frac{(y_{k,i} - \mu_{k,\theta}(x_i))^2}{\sigma_{k,\theta}^2(x_i)} + \log \sigma_{k,\theta}^2(x_i) \right). \tag{S8}$$

The DNN architecture of  $f_\theta(x)$  first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two heads of two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 3$ ) to output  $\mu_\theta(x) \in \mathbb{R}^3$  and  $\log \sigma_\theta^2(x) \in \mathbb{R}^3$ . The mean  $\mu_\theta(x)$  is taken as the prediction  $\hat{y}$ .

**Laplace** Following [22], the Laplace model is defined using a DNN  $f_\theta(x)$  according to,

$$\begin{aligned} p(y|x;\theta) &= \prod_{k=1}^3 \beta_{k,\theta}(x)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\sum_{k=1}^3 \frac{(y_k - \mu_{k,\theta}(x))^2}{\beta_{k,\theta}(x)}\right)^{\frac{1}{2}}\right\}, \\ y &= [y_1 \quad y_2 \quad y_3]^\top \in \mathbb{R}^3, \\ \mu_\theta(x) &= [\mu_{1,\theta}(x) \quad \mu_{2,\theta}(x) \quad \mu_{3,\theta}(x)]^\top \in \mathbb{R}^3, \\ \beta_\theta(x) &= [\beta_{1,\theta}(x) \quad \beta_{2,\theta}(x) \quad \beta_{3,\theta}(x)]^\top \in \mathbb{R}^3, \\ f_\theta(x) &= [\mu_\theta(x)^\top \quad \log \beta_\theta(x)^\top]^\top \in \mathbb{R}^6. \end{aligned} \tag{S9}$$

It is trained by minimizing the negative log-likelihood, corresponding to the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \left\{ \left( \sum_{k=1}^3 \frac{(y_{k,i} - \mu_{k,\theta}(x_i))^2}{\beta_{k,\theta}(x_i)} \right)^{\frac{1}{2}} + \sum_{k=1}^3 \log \beta_{k,\theta}(x_i) \right\}. \tag{S10}$$

The DNN architecture of  $f_\theta(x)$  first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by two heads of two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow 3$ ) to output  $\mu_\theta(x) \in \mathbb{R}^3$  and  $\log \beta_\theta(x) \in \mathbb{R}^3$ . The mean  $\mu_\theta(x)$  is taken as the prediction  $\hat{y}$ .

**Softmax** The DNN architecture of *Softmax* first extracts ResNet50 features  $g_x \in \mathbb{R}^{2048}$  from the input image  $x$ . The feature vector  $g_x$  is then processed by three heads of two fully-connected layers ( $2048 \rightarrow 2048$ ,  $2048 \rightarrow C$ ), outputting logits for  $C = 151$  discretized classes  $\{-75, -74, \dots, 75\}$  for the yaw, pitch and roll angles (in degrees). It is trained by minimizing either the cross-entropy (CE) and  $L^2$  losses,  $J = J_{CE} + 0.1J_{L^2}$ , or the CE,  $L^2$  and variance [10] losses,  $J = J_{CE} + 0.1J_{L^2} + 0.05J_{Var}$ . The prediction  $\hat{y}$  is obtained by computing the softmax expected value for yaw, pitch and roll.

## F.5 Full Results

Full experiment results, extending the results found in Table 5 (Section 4.4 in the paper), are provided in Table S6.

# Paper II

## Title

How to Train Your Energy-Based Model for Regression

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Radu Timofte, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. “How to Train Your Energy-Based Model for Regression.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2020





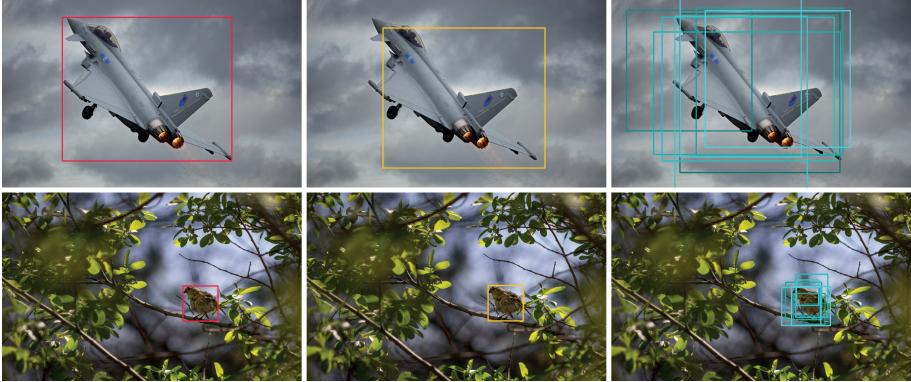
# How to Train Your Energy-Based Model for Regression

## Abstract

Energy-based models (EBMs) have become increasingly popular within computer vision in recent years. While they are commonly employed for generative image modeling, recent work has applied EBMs also for regression tasks, achieving state-of-the-art performance on object detection and visual tracking. Training EBMs is however known to be challenging. While a variety of different techniques have been explored for generative modeling, the application of EBMs to regression is not a well-studied problem. How EBMs should be trained for best possible regression performance is thus currently unclear. We therefore accept the task of providing the first detailed study of this problem. To that end, we propose a simple yet highly effective extension of noise contrastive estimation, and carefully compare its performance to six popular methods from literature on the tasks of 1D regression and object detection. The results of this comparison suggest that our training method should be considered the go-to approach. We also apply our method to the visual tracking task, achieving state-of-the-art performance on five datasets. Notably, our tracker achieves 63.7% AUC on LaSOT and 78.7% Success on TrackingNet. Code is available at [https://github.com/fregu856/ebms\\_regression](https://github.com/fregu856/ebms_regression).

## 1 Introduction

Energy-based models (EBMs) [1] have a rich history in machine learning [2, 3, 4, 5, 6]. An EBM specifies a probability density  $p(x; \theta) = e^{f_\theta(x)} / \int e^{f_\theta(x)} dx$  directly via a parameterized scalar function  $f_\theta(x)$ . By defining  $f_\theta(x)$  using a deep neural network (DNN),  $p(x; \theta)$  becomes expressive enough to learn practically any density from observed data. EBMs have therefore become increasingly popular within computer vision in recent years, commonly being applied for various generative image modeling tasks [7, 8, 9, 10, 11, 12, 13].



**Figure 1:** We propose NCE+ to train EBMs  $p(y|x; \theta)$  for tasks such as bounding box regression. NCE+ is a highly effective extension of NCE, accounting for noise in the annotation process of real-world datasets. Given a label  $y_i$  (red box), the EBM is trained by having to discriminate between  $y_i + \nu_i$  (yellow box) and noise samples  $\{y^{(i,m)}\}_{m=1}^M$  (blue boxes).

Recent work [14, 15] has also explored conditional EBMs as a general formulation for regression, demonstrating particularly impressive performance on the tasks of object detection [16, 17, 18] and visual tracking [19, 20, 21]. Regression entails predicting a continuous target  $y$  from an input  $x$ , given a training set of observed input-target pairs. This was addressed in [14, 15] by learning a conditional EBM  $p(y|x; \theta)$ , capturing the distribution of the target value  $y$  given the input  $x$ . At test time, gradient ascent was then used to maximize  $p(y|x; \theta)$  w.r.t.  $y$ , producing highly accurate predictions. Regression is a fundamental problem within computer vision with many additional applications [22, 23, 24, 25, 26], which all would benefit from such accurate predictions. In this work, we therefore study the use of EBMs for regression in detail, aiming to further improve its performance and applicability.

While the modeling capacity of EBMs makes them highly attractive for many applications, training EBMs is known to be challenging. This is because the EBM  $p(x; \theta) = e^{f_\theta(x)} / \int e^{f_\theta(x)} dx$  involves an intractable integral, complicating the use of standard maximum likelihood (ML) learning. A variety of different techniques have therefore been explored in the generative modeling literature, including alternative estimation methods [27, 13, 28, 29, 30] and approximations based on Markov chain Monte Carlo (MCMC) [31, 10, 9, 12]. The application of EBMs for regression is however not a particularly well-studied problem. [14, 15] both applied importance sampling to approximate intractable integrals, an approach known to scale poorly with the data dimensionality, and considered no alternative techniques. How EBMs  $p(y|x; \theta)$  should be trained for best possible performance on computer vision regression tasks is thus an open question, which we set out to investigate in this work.

**Contributions** We propose a simple yet highly effective extension of noise contrastive estimation (NCE) [27] to train EBMs  $p(y|x; \theta)$  for regression tasks. Our proposed method, termed *NCE+*, can be understood as a direct generalization of NCE, accounting for noise in the annotation process. We evaluate NCE+ on illustrative 1D regression problems and on the task of bounding box regression in object detection. We also provide a detailed comparison of NCE+ and *six* popular methods from previous work, the results of which suggest that NCE+ should be considered the go-to training method. Lastly, we apply our proposed NCE+ to the task of visual tracking, achieving state-of-the-art results on *five* common datasets.

## 2 Energy-Based Models for Regression

We study the application of EBMs to important regression tasks in computer vision, using energy-based models of the conditional density  $p(y|x)$ . Here, we first define the general regression problem and our employed EBM in Section 2.1. Our prediction strategy based on gradient ascent is then described in Section 2.2. Lastly, we discuss the challenges associated with training EBMs, and describe six popular methods from the literature, in Section 2.3.

### 2.1 Problem & Model Definition

In a supervised regression problem, we are given a training set  $\mathcal{D}$  of i.i.d. input-target pairs,  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ . The task is then to learn how to predict a target  $y^* \in \mathcal{Y}$  given a new input  $x^* \in \mathcal{X}$ . The target space  $\mathcal{Y}$  is continuous,  $\mathcal{Y} = \mathbb{R}^K$  for some  $K \geq 1$ , and the input space  $\mathcal{X}$  usually corresponds to the space of images.

As in [14, 15], we address this problem by creating an energy-based model  $p(y|x; \theta)$  of the conditional target density  $p(y|x)$ . To that end, we specify a DNN  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  with model parameters  $\theta \in \mathbb{R}^P$ . This DNN directly maps any input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  to a scalar  $f_\theta(x, y) \in \mathbb{R}$ . The model  $p(y|x; \theta)$  of the conditional target density is then defined as,

$$p(y|x; \theta) = \frac{e^{f_\theta(x, y)}}{Z(x, \theta)}, \quad Z(x, \theta) = \int e^{f_\theta(x, \tilde{y})} d\tilde{y}, \quad (1)$$

where the DNN output  $f_\theta(x, y) \in \mathbb{R}$  is interpreted as the negative energy of the density, and  $Z(x, \theta)$  is the input-dependent normalizing partition function. Since  $p(y|x; \theta)$  in (1) is directly defined by the DNN  $f_\theta$ , minimal restricting assumptions are put on the true  $p(y|x)$ . The predictive power of the DNN can

thus be fully exploited, enabling learning of, e.g., multi-modal and asymmetric densities directly from data. This expressivity however comes at the cost of  $Z(x, \theta)$  being intractable, which complicates evaluating or sampling from  $p(y|x; \theta)$ .

## 2.2 Prediction

At test time, the problem of predicting a target value  $y^*$  from an input  $x^*$  corresponds to finding a point estimate of the predicted conditional density  $p(y|x^*; \theta)$ . The most natural choice is to select the most likely target under the model,  $y^* = \arg \max_y p(y|x^*; \theta) = \arg \max_y f_\theta(x^*, y)$ . The prediction  $y^*$  is thus obtained by directly maximizing the DNN scalar output  $f_\theta(x^*, y)$  w.r.t.  $y$ , not requiring  $Z(x^*, \theta)$  to be evaluated nor any samples from  $p(y|x^*; \theta)$  to be generated. Following [14, 15], we estimate  $y^* = \arg \max_y f_\theta(x^*, y)$  by performing gradient ascent to refine an initial estimate  $\hat{y}$  and find a local maximum of  $f_\theta(x^*, y)$ . Starting at  $y = \hat{y}$ , we thus run  $T$  gradient ascent iterations,  $y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ , with step-length  $\lambda$ . An algorithm for this prediction procedure is found in the supplementary material.

## 2.3 Training

To train the DNN  $f_\theta(x, y)$  specifying the EBM (1), different techniques for fitting a density  $p(y|x; \theta)$  to observed data  $\{(x_i, y_i)\}_{i=1}^N$  can be used. In general, the most commonly applied such technique is ML learning, which entails minimizing the negative log-likelihood (NLL),

$$-\sum_{i=1}^N \log p(y_i|x_i; \theta) = \sum_{i=1}^N \log \left( \int e^{f_\theta(x_i, y)} dy \right) - f_\theta(x_i, y_i), \quad (2)$$

w.r.t. the parameters  $\theta$ . The integral in (2) is however intractable, and exact evaluation of the NLL is thus not possible. [14, 15] employed importance sampling to approximate such intractable integrals, obtaining state-of-the-art performance on object detection and visual tracking. Recent work [13, 32, 33, 10, 12, 11] on generative image modeling has however applied a variety of different training methods not considered in [14, 15], including the ML learning alternatives NCE [27] and score matching [28]. How we should train the DNN  $f_\theta$  to obtain best possible regression performance is thus unclear. In this work, we therefore carefully compare our proposed method to six popular training methods from the literature.

**ML with Importance Sampling (ML-IS)** A straightforward training method is proposed in [14], which we term *ML with Importance Sampling (ML-IS)*.

Using ML-IS, [14] successfully applied the EBM (1) to the regression tasks of object detection, visual tracking, age estimation and head-pose estimation. In ML-IS, the DNN  $f_\theta$  is trained by directly minimizing the NLL (2) w.r.t.  $\theta$ , using importance sampling to approximate the intractable integral,

$$-\log p(y_i|x_i; \theta) \approx \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y^{(i,m)})}}{q(y^{(i,m)}|y_i)} \right) - f_\theta(x_i, y_i). \quad (3)$$

Here,  $\{y^{(i,m)}\}_{m=1}^M$  are  $M$  samples drawn from a proposal distribution  $q(y|y_i)$  that depends on the ground truth target  $y_i$ . In [14],  $q(y|y_i)$  is set to a mixture of  $K$  Gaussians centered at  $y_i$ ,

$$q(y|y_i) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I). \quad (4)$$

The loss  $J(\theta)$  is obtained by averaging over all pairs  $\{(x_i, y_i)\}_{i=1}^n$  in the current mini-batch,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y^{(i,m)})}}{q(y^{(i,m)}|y_i)} \right) - f_\theta(x_i, y_i). \quad (5)$$

**KL Divergence with Importance Sampling (KLD-IS)** Instead of minimizing the NLL (2), [15] considers the Kullback-Leibler (KL) divergence  $D_{\text{KL}}(p(y|y_i) \| p(y|x_i; \theta))$  between the EBM  $p(y|x_i; \theta)$  and an assumed density  $p(y|y_i)$  of the true target  $y$  given the label  $y_i$ . The density  $p(y|y_i)$  models noise in the annotation process of our given training set  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ . In [15],  $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$ , where  $\sigma$  is a hyperparameter. As shown in [15],

$$D_{\text{KL}}(p(y|y_i) \| p(y|x_i; \theta)) = \log \left( \int e^{f_\theta(x_i, y)} dy \right) - \int f_\theta(x_i, y) p(y|y_i) dy + C, \quad (6)$$

where  $C$  is a constant that does not depend on  $\theta$ . [15] approximates the integrals in (6) using importance sampling, employing the ML-IS proposal  $q(y|y_i)$  in (4). By then averaging over all pairs  $\{(x_i, y_i)\}_{i=1}^n$  in the current mini-batch, the loss  $J(\theta)$  used to train  $f_\theta$  is obtained as,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y^{(i,m)})}}{q(y^{(i,m)}|y_i)} \right) - \frac{1}{M} \sum_{m=1}^M f_\theta(x_i, y^{(i,m)}) \frac{p(y^{(i,m)}|y_i)}{q(y^{(i,m)}|y_i)}, \quad (7)$$

where  $\{y^{(i,m)}\}_{m=1}^M$  are  $M$  samples drawn from the proposal  $q(y|y_i)$ . We term this training method *KL Divergence with Importance Sampling (KLD-IS)*. When applied to visual tracking in [15], KLD-IS outperformed ML-IS and set a new state-of-the-art.

**ML with MCMC (ML-MCMC)** To minimize the negative log-likelihood (2) w.r.t. the parameters  $\theta$ , the following identity for the expression of the gradient  $\nabla_\theta - \log p(y_i|x_i; \theta)$  can be utilized [1],

$$\nabla_\theta - \log p(y_i|x_i; \theta) = \mathbb{E}_{p(y|x_i; \theta)} \left[ \nabla_\theta f_\theta(x_i, y) \right] - \nabla_\theta f_\theta(x_i, y_i). \quad (8)$$

The expectation in (8) is then approximated using samples  $\{y^{(i,m)}\}_{m=1}^M$  drawn from  $p(y|x_i; \theta)$ , i.e. from the EBM itself. To obtain each sample  $y^{(i,m)} \sim p(y|x_i; \theta)$ , MCMC is used. Specifically, we follow recent work [7, 8, 10, 9, 12, 11] on generative image modeling and run  $L \geq 1$  steps of Langevin dynamics [34]. Starting at  $y_{(0)}$ , we thus update  $y_{(l)}$  according to,

$$y_{(l+1)} = y_{(l)} + \frac{\alpha^2}{2} \nabla_y f_\theta(x_i, y_{(l)}) + \alpha \epsilon_l, \quad \epsilon_l \sim \mathcal{N}(0, I), \quad (9)$$

and set  $y^{(i,m)} = y_{(L)}$ . Here,  $\alpha > 0$  is a small constant step-length. Following the principle of contrastive divergence [1, 31, 2], we start the Markov chain (9) at the ground truth target,  $y_{(0)} = y_i$ . By approximating (8) with the samples  $\{y^{(i,m)}\}_{m=1}^M$ , and by averaging over all pairs  $\{(x_i, y_i)\}_{i=1}^n$  in the current mini-batch, the loss  $J(\theta)$  used to train the DNN  $f_\theta$  is obtained as,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \left( \frac{1}{M} \sum_{m=1}^M f_\theta(x_i, y^{(i,m)}) \right) - f_\theta(x_i, y_i). \quad (10)$$

We term this specific training method *ML with MCMC (ML-MCMC)*.

**Noise Contrastive Estimation (NCE)** As an alternative to ML learning, Gutmann and Hyvärinen proposed NCE [27] for estimating unnormalized parametric models. NCE entails generating samples from some noise distribution  $p_N$ , and learning to discriminate between these noise samples and observed data examples. It has recently been applied to generative image modeling with EBMs [13], and the NCE loss is also utilized in various frameworks for self-supervised learning [35, 36, 37]. Moreover, NCE has been applied to train EBMs for supervised *classification* tasks within language modeling [38, 39, 40, 41], where the target space  $\mathcal{Y}$  is a large but finite set of possible labels. We adopt NCE for regression by using a noise distribution  $p_N(y|y_i)$  of the same form as the ML-IS proposal in (4),

$$p_N(y|y_i) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I), \quad (11)$$

and by employing the ranking NCE objective [40], as described in [41]. We choose ranking NCE over the binary objective since it is consistent under a

weaker assumption [41]. We thus define  $y^{(i,0)} \triangleq y_i$ , and train the DNN  $f_\theta$  by minimizing the following loss,

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \log \frac{\exp \{f_\theta(x_i, y^{(i,0)}) - \log p_N(y^{(i,0)}|y_i)\}}{\sum_{m=0}^M \exp \{f_\theta(x_i, y^{(i,m)}) - \log p_N(y^{(i,m)}|y_i)\}}, \quad (12)$$

where  $\{y^{(i,m)}\}_{m=1}^M$  are  $M$  noise samples drawn from  $p_N(y|y_i)$  in (11).

**Score Matching (SM)** Another alternative estimation method is score matching (SM), as proposed by Hyvärinen [28] and further studied for supervised problems in [42]. The method focuses on the *score* of  $p(y|x; \theta)$ , defined as  $\nabla_y \log p(y|x; \theta) = \nabla_y f_\theta(x, y)$ , aiming for it to approximate the score of the true target density  $p(y|x)$ . Note that the EBM score  $\nabla_y f_\theta(x, y)$  does not depend on the intractable  $Z(x, \theta)$ . SM was applied to simple conditional density estimation problems in [42], using a combination of feed-forward networks and reproducing kernels to specify the EBM. Following [42], we train the DNN  $f_\theta$  by minimizing the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \text{tr}(\nabla_y^2 f_\theta(x_i, y_i)) + \frac{1}{2} \|\nabla_y f_\theta(x_i, y_i)\|_2^2, \quad (13)$$

where only the diagonal of  $\nabla_y^2 f_\theta(x_i, y_i)$  actually is needed to compute the first term.

**Denoising Score Matching (DSM)** By modifying the SM objective, denoising score matching (DSM) was proposed by Vincent [29]. DSM does not require computation of any second derivatives, improving its scalability to high-dimensional data. The method entails employing SM on noise-corrupted data points. Recently, DSM has been successfully applied to generative image modeling [32, 30, 33]. DSM was also extended to train EBMs of conditional densities in [43], where it was applied to a transfer learning problem. Following [43], we use a Gaussian noise distribution and train the DNN  $f_\theta$  by minimizing the loss,

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{1}{M} \sum_{m=1}^M \left\| \nabla_y f_\theta(x_i, \tilde{y}^{(i,m)}) + \frac{\tilde{y}^{(i,m)} - y_i}{\sigma^2} \right\|_2^2, \quad (14)$$

where  $\{\tilde{y}^{(i,m)}\}_{m=1}^M$  are  $M$  samples drawn from the noise distribution  $p_\sigma(\tilde{y}|y_i) = \mathcal{N}(\tilde{y}; y_i, \sigma^2 I)$ .

### 3 Proposed Training Method

To train the DNN  $f_\theta$  specifying our EBM  $p(y|x; \theta)$  in (1), we propose a *simple yet highly effective* extension of NCE [27]. Motivated by the improved performance of KLD-IS compared to ML-IS on visual tracking [15], we extend

NCE with the capability to model annotation noise. To that end, we adopt the standard NCE noise distribution  $p_N$  (11) and loss (12), but instead of defining  $y^{(i,0)} \triangleq y_i$ , we sample  $\nu_i \sim p_\beta(y)$  and define  $y^{(i,0)} \triangleq y_i + \nu_i$ . The distribution  $p_\beta$  is a zero-centered version of  $p_N$  in which  $\{\sigma_k\}_{k=1}^K$  are scaled with  $\beta > 0$ ,

$$p_N(y|y_i) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I), \quad p_\beta(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; 0, \beta \sigma_k^2 I). \quad (15)$$

Instead of training the DNN  $f_\theta$  by learning to discriminate between noise samples  $\{y^{(i,m)}\}_{m=1}^M$  and the label  $y_i$ , it thus has to discriminate between the samples  $\{y^{(i,m)}\}_{m=1}^M$  and  $y_i + \nu_i$ . Examples of  $y_i + \nu_i$  and  $\{y^{(i,m)}\}_{m=1}^M$  in the task of bounding box regression are visualized in Figure 1. Similar to KLD-IS, in which an assumed density of the true target value  $y$  given  $y_i$  is employed, our approach thus accounts for possible noise and inaccuracies in the provided label  $y_i$ . Specifically, our proposed training method entails sampling  $\{y^{(i,m)}\}_{m=1}^M \sim p_N(y|y_i)$  and  $\nu_i \sim p_\beta(y)$ , setting  $y^{(i,0)} \triangleq y_i + \nu_i$ , and minimizing the following loss,

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^n \log \frac{\exp \{f_\theta(x_i, y^{(i,0)}) - \log p_N(y^{(i,0)}|y_i)\}}{\sum_{m=0}^M \exp \{f_\theta(x_i, y^{(i,m)}) - \log p_N(y^{(i,m)}|y_i)\}}. \quad (16)$$

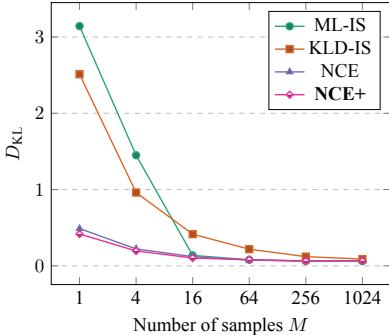
As  $\beta \rightarrow 0$ , samples  $\nu_i \sim p_\beta(y)$  will concentrate increasingly close to zero, and the standard NCE method is in practice recovered. Our proposed training method can thus be understood as a direct generalization of NCE. Compared to NCE, our method adds no significant training cost and requires tuning of a single additional hyperparameter  $\beta$ . A value for  $\beta$  is selected in a simple two-step procedure. First, we fix  $y^{(i,0)} = y_i$  and select the standard deviations  $\{\sigma_k\}_{k=1}^K$  based on validation set performance, just as in NCE. We then fix  $\{\sigma_k\}_{k=1}^K$  and vary  $\beta$  to find the value corresponding to maximum validation performance. Typically, we start this ablation with  $\beta = 0.1$ . We term our proposed training method *NCE+*.

## 4 Comparison of Training Methods

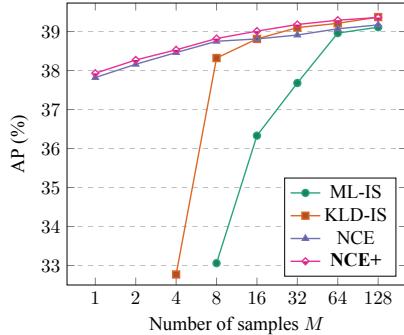
We provide a detailed comparison of the six training methods from Section 2.3 and our proposed NCE+. To that end, we perform extensive experiments on 1D regression (Section 4.1) and object detection (Section 4.2). Our findings are summarized in Section 4.3. All experiments are implemented in PyTorch [44] and the code is publically available. For both tasks, further details and results are also provided in the supplementary material.

**Table 1:** Comparison of training methods for the 1D regression experiments.

	ML-IS	ML-MCMC-1	ML-MCMC-16	ML-MCMC-256	KLD-IS	NCE	SM	DSM	NCE+
$D_{KL} \downarrow$	<b>0.062</b>	0.865	0.449	0.106	0.088	0.068	0.781	0.395	0.066
Training Cost $\downarrow$	<b>0.44</b>	0.54	2.41	30.8	<b>0.44</b>	0.45	0.60	0.47	0.46



**Figure 2:** Detailed comparison of the top-performing methods for the illustrative 1D regression experiments. NCE and NCE+ here demonstrate clear superior performance for small number of samples  $M$ .



**Figure 3:** Detailed comparison of the top-performing methods for object detection, on the 2017 val split of COCO [45]. Missing values for ML-IS and KLD-IS correspond to failed training due to numerical issues.

## 4.1 1D Regression Experiments

We first perform experiments on illustrative 1D regression problems. The DNN  $f_\theta(x, y)$  is here a simple feed-forward network, taking  $x \in \mathbb{R}$  and  $y \in \mathbb{R}$  as inputs. We employ two synthetic datasets, and evaluate the training methods by how well the learned model  $p(y|x; \theta)$  (1) approximates the known ground truth  $p(y|x)$ , as measured by the KL divergence  $D_{KL}$ .

**Results** A comparison of all seven training methods in terms of  $D_{KL}$  and training cost (seconds per epoch) is found in Table 1. For ML-MCMC, we include results for  $L \in \{1, 16, 256\}$  Langevin steps (9). We observe that ML-IS, KLD-IS, NCE and NCE+ clearly have the best performance. While ML-MCMC is relatively close in terms of  $D_{KL}$  for  $L = 256$ , this comes at the expense of a massive increase in training cost. DSM outperforms SM in terms of both metrics, but is not close to the top-performing methods. The four best methods are further compared in Figure 2, showing  $D_{KL}$  as a function of  $M$ . Here, we observe that NCE and NCE+ significantly outperform ML-IS and KLD-IS for small number of samples  $M$ .

**Table 2:** Comparison of training methods for the object detection experiments, on the *2017 test-dev* split of COCO [45]. Our proposed NCE+ achieves the best performance.

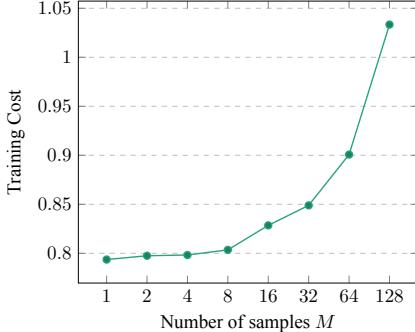
	ML-IS	ML-MCMC-1	ML-MCMC-4	ML-MCMC-8	KLD-IS	NCE	DSM	NCE+
AP (%) $\uparrow$	39.4	36.4	36.4	36.4	39.6	39.5	36.3	<b>39.7</b>
AP <sub>50</sub> (%) $\uparrow$	58.6	57.9	57.9	58.0	58.6	58.6	57.9	<b>58.7</b>
AP <sub>75</sub> (%) $\uparrow$	42.1	38.8	39.0	39.0	42.6	42.4	38.9	<b>42.7</b>
Training Cost $\downarrow$	1.03	2.47	7.05	13.3	<b>1.02</b>	1.04	3.84	1.09

## 4.2 Object Detection Experiments

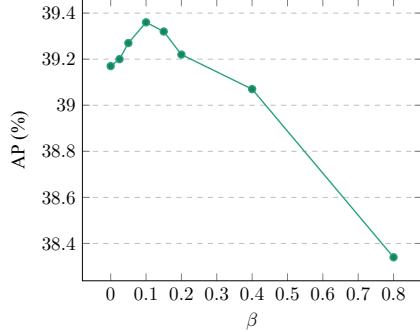
Next, we evaluate the methods on the task of bounding box regression in object detection. We employ an identical network architecture for  $f_\theta(x, y)$  as in [14]. An extra network branch, consisting of three fully-connected layers with parameters  $\theta$ , is thus added onto a pre-trained and fixed FPN Faster-RCNN detector [46]. Given an image  $x$  and bounding box  $y \in \mathbb{R}^4$ , the image is first processed by the detector backbone network (ResNet50-FPN), outputting image features  $h_1(x)$ . Using a differentiable PrRoiPool [47] layer,  $h_1(x)$  is then pooled to extract features  $h_2(x, y)$ . Finally,  $h_2(x, y)$  is processed by the added network branch, outputting  $f_\theta(x, y) \in \mathbb{R}$ . As in [14], predictions  $y^*$  are produced by performing guided NMS [47] followed by gradient-based refinement (Section 2.2), taking the Faster-RCNN detections as initial estimates  $\hat{y}$ . Experiments are performed on the large-scale COCO dataset [45]. We use the *2017 train* split ( $\approx 118\,000$  images) for training, the *2017 val* split ( $\approx 5\,000$  images) for setting hyperparameters, and report results on the *2017 test-dev* split ( $\approx 20\,000$  images). The standard COCO metrics AP, AP<sub>50</sub> and AP<sub>75</sub> are used, where AP is the primary metric.

**Results** A comparison of the training methods in terms of the COCO metrics and training cost (seconds per iteration) is found in Table 2. Since DSM clearly outperformed SM in the 1D regression experiments, we here only include DSM. For ML-MCMC, results for  $L \in \{1, 4, 8\}$  are included. We observe that ML-IS, KLD-IS, NCE and NCE+ clearly have the best performance. In terms of the COCO metrics, NCE+ outperforms NCE and all other methods. ML-IS is also outperformed by KLD-IS. The four top-performing methods are further compared in Figure 3, in terms of AP as a function of the number of samples  $M$ . NCE and NCE+ here demonstrate clear superior performance for small values of  $M$ , and do not experience numerical issues even for  $M = 1$ . KLD-IS improves this robustness compared ML-IS, but is not close to matching NCE or NCE+. In terms of training cost, the four top-performing methods are virtually identical. For ML-IS, e.g., we observe in Figure 4 that setting  $M = 1$  decreases the training cost with 23% compared to the standard case of  $M = 128$ .

**Analysis of NCE+ Hyperparameters** How the value of  $\beta > 0$  in  $p_\beta$  (15) affects validation performance is studied in Figure 5. Here, we observe that



**Figure 4:** Effect of the number of samples  $M$  on training cost (seconds per iteration), for ML-IS on object detection.



**Figure 5:** Effect of the NCE+ hyperparameter  $\beta$  on detection performance ( $\uparrow$ ), on the 2017 val split of COCO [45].

**Table 3:** Ablation study for NCE, on the 2017 val split of COCO [45].

$\{\sigma_k\}_{k=1}^3$	{0.0125, 0.025, 0.05}	{0.025, 0.05, 0.1}	{0.05, 0.1, 0.2}	{0.075, 0.15, 0.3}	{0.1, 0.2, 0.4}
AP (%) $\uparrow$	38.58	38.95	39.12	<b>39.17</b>	39.05

quite a large range of values improve the performance compared to the NCE baseline ( $\beta \rightarrow 0$ ), before it eventually degrades for  $\beta \gtrsim 0.3$ . We also observe that the performance is optimized for  $\beta = 0.1$ . In Figure 5, the standard deviations  $\{\sigma_k\}_{k=1}^K$  in  $p_N, p_\beta$  (15) are set to  $\{0.075, 0.15, 0.3\}$ . These values are selected in an initial step based on an ablation study for NCE, which is found in Table S4.

### 4.3 Discussion

The results on both set of experiments are highly consistent. First of all, ML-IS, KLD-IS, NCE and NCE+ are by far the top-performing training methods. ML-MCMC, the method commonly employed for generative image modeling in recent years, does not come close to matching these top-performing methods, especially not given similar computational budgets. When studying the performance as a function of the number of samples  $M$ , NCE and NCE+ are the superior methods by a significant margin. In particular, this study demonstrates that the NCE and NCE+ losses are numerically more stable than those of ML-IS and KLD-IS. In the 1D regression problems, which employ synthetic datasets without any annotation noise, NCE and NCE+ have virtually identical performance. In the object detection experiments however, where we employ real-world datasets, NCE+ consistently improves the NCE performance. On object detection, NCE+ also improves or matches the performance of KLD-IS, which explicitly models annotation noise and outperforms ML-IS. Overall, the

results of the comparison suggest that our proposed NCE+ should be considered the go-to training method.

## 5 Visual Tracking Experiments

Lastly, we apply our proposed NCE+ to the task of visual tracking. Specifically, we consider *generic visual object tracking*, which entails estimating the bounding box  $y \in \mathbb{R}^4$  of a target object in every frame of a video. The target object does not belong to any pre-specified class, but is instead defined by a given bounding box in the initial video frame. We compare the performance both to NCE and KLD-IS, and to state-of-the-art trackers. Code and trained models are available at [48]. Further details are also found in the supplementary material.

**Tracking Approach** We base our tracker on the recent DiMP [21] and PrDiMP [15]. The target object is thus first coarsely localized in the current video frame via 2D image-coordinate regression of its center point, emphasizing robustness over accuracy. Then, the full bounding box  $y \in \mathbb{R}^4$  of the target is accurately regressed by gradient-based refinement (Section 2.2). The two stages employ separate network branches which are trained jointly end-to-end. As a strong baseline, we combine the DiMP method for center point regression with the PrDiMP bounding box regression approach. We term this resulting tracker *DiMP-KLD-IS*. By also modifying common training parameters (batch size, data augmentation etc.), DiMP-KLD-IS significantly outperforms both DiMP and PrDiMP. Our proposed tracker, termed *DiMP-NCE+*, is then obtained simply by using NCE+ instead of KLD-IS to train the bounding box regression branch. In both cases, the number of samples  $M = 128$ . As in [21, 15], the training splits of TrackingNet [49], LaSOT [50], GOT-10k [51] and COCO [45] are used for training. Similar to PrDiMP, our DiMP-NCE+ tracker runs at about 30 FPS on a single GPU.

**Results** We evaluate DiMP-NCE+ on five commonly used tracking datasets. Tracking-Net [49] is a large-scale dataset containing videos sampled from YouTube. Results are reported on its test set of 511 videos. We also evaluate on the LaSOT [50] test set, containing 280 long videos (2 500 frames on average). Moreover, we report results on the UAV123 [52] dataset, consisting of 123 videos which feature small targets and distractor objects. Results are also reported on the 30 FPS version of the need for speed (NFS) [53] dataset, containing 100 videos with fast motions. Finally, we evaluate on the 100 videos of OTB-100 [54]. Our tracker is evaluated in terms of overlap precision (OP). For a threshold  $T \in [0, 1]$ ,  $OP_T$  is the percentage of frames in which the IoU overlap between the estimated and ground truth target bounding box is larger than  $T$ . By averaging  $OP_T$  over  $T \in [0, 1]$ , the *AUC* score is then obtained.

**Table 4:** Results for the visual tracking experiments. The AUC (Success) metric is reported on five common datasets. Our DiMP-NCE+ tracker significantly outperforms strong baselines and achieves state-of-the-art performance on all datasets. For SiamRCNN [55], results for the ResNet50 version are given in parentheses when available.

	MDNet [56]	UPDT [57]	DaSiamRPN [58]	ATOM [20]	SiamRPN++ [19]	DiMP [21]	SiamRCNN [55]	PrDiMP [15]	DiMP- KLD-IS	DiMP- NCE	DiMP- NCE+
TrackingNet	60.6	61.1	63.8	70.3	73.3	74.0	<b>81.2</b>	75.8	78.1	77.1	78.7
LaSOT	39.7	-	-	51.5	49.6	56.9	<b>64.8</b> (62.3)	59.8	63.1	62.8	63.7
UAV123	52.8	54.5	57.7	63.2	61.3	64.3	64.9	66.7	66.6	65.2	<b>67.2</b>
NFS	42.2	53.7	-	58.4	-	62.0	63.9	63.5	64.7	64.3	<b>65.0</b>
OTB-100	67.8	70.2	65.8	66.9	69.6	68.4	70.1 (68.0)	69.6	70.1	69.3	<b>70.7</b>

For TrackingNet, the term *Success* is used in place of AUC. Results in terms of AUC on all five datasets are found in Table 4. To ensure significance, the average AUC over 5 runs is reported for our trackers. We observe that DiMP-NCE+ consistently outperforms both our DiMP-KLD-IS baseline, and a variant employing NCE instead of NCE+. Compared to previous approaches, only the very recent SiamRCNN [55] achieves results competitive with our DiMP-NCE+. SiamRCNN is however slower than DiMP-NCE+ (5 FPS vs 30 FPS) and employs a larger backbone network (ResNet101 vs ResNet50). Results for the ResNet50 version of SiamRCNN are only available on two of the datasets, on which it is outperformed by our DiMP-NCE+. More detailed results are provided in the supplementary material.

## 6 Conclusion

We proposed a simple yet highly effective extension of NCE to train EBMs  $p(y|x; \theta)$  for computer vision regression tasks. Our proposed method NCE+ can be understood as a direct generalization of NCE, accounting for noise in the annotation process of real-world datasets. We also provided a detailed comparison of NCE+ and six popular methods from literature, the results of which suggest that NCE+ should be considered the go-to training method. This comparison is the first comprehensive study of how EBMs should be trained for best possible regression performance. Finally, we applied our proposed NCE+ to the task of visual tracking, achieving state-of-the-art performance on five commonly used datasets. We hope that our simple training method and promising results will encourage the research community to further explore the application of EBMs to various regression tasks.

**Acknowledgments** This research was financially supported by the Swedish Foundation for Strategic Research via the project *ASSEMBLE*, the Swedish Research Council via the project *Learning flexible models for nonlinear dynamics*, the ETH Zürich Fund (OK), a Huawei Technologies Oy (Finland) project, an Amazon AWS grant, and Nvidia.

## References

- [1] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).
- [2] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. “Energy-based models for sparse overcomplete representations.” In: *Journal of Machine Learning Research* (2003).
- [3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. “A neural probabilistic language model.” In: *Journal of machine learning research* (2003).
- [4] A. Mnih and G. Hinton. “Learning nonlinear constraints with contrastive backpropagation.” In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE. 2005.
- [5] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. “Unsupervised discovery of nonlinear structure using contrastive backpropagation.” In: *Cognitive science* (2006).
- [6] M. Osadchy, M. L. Miller, and Y. L. Cun. “Synergistic face detection and pose estimation with energy-based models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.
- [7] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. “A theory of generative convnet.” In: *International Conference on Machine Learning (ICML)*. 2016.
- [8] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. “Learning generative convnets via multi-grid modeling and sampling.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [9] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. “Learning non-convergent non-persistent short-run MCMC toward energy-based model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [10] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [11] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. “Your classifier is secretly an energy based model and you should treat it like one.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [12] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu. “On the Anatomy of MCMC-based Maximum Likelihood Learning of Energy-Based Models.” In: *Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2020.
- [13] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. “Flow Contrastive Estimation of Energy-Based Models.” In: *arXiv preprint arXiv:1912.00589* (2019).

- [14] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [15] M. Danelljan, L. Van Gool, and R. Timofte. “Probabilistic Regression for Visual Tracking.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [16] S. Ren, K. He, R. B. Girshick, and J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2015).
- [17] H. Law and J. Deng. “CornerNet: Detecting objects as paired keypoints.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [18] X. Zhou, J. Zhuo, and P. Krahenbuhl. “Bottom-up object detection by grouping extreme and center points.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [19] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. “SiamRPN++: Evolution of siamese visual tracking with very deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [20] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg. “ATOM: Accurate tracking by overlap maximization.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [21] G. Bhat, M. Danelljan, L. V. Gool, and R. Timofte. “Learning discriminative model prediction for tracking.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.
- [22] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [23] B. Xiao, H. Wu, and Y. Wei. “Simple baselines for human pose estimation and tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [24] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. “FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation from a Single Image.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [25] R. Rothe, R. Timofte, and L. Van Gool. “Deep expectation of real and apparent age from a single image without facial landmarks.” In: *International Journal of Computer Vision (IJCV)* (2016).
- [26] H. Pan, H. Han, S. Shan, and X. Chen. “Mean-variance loss for deep age estimation from a face.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

- [27] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010.
- [28] A. Hyvärinen. “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research* (2005).
- [29] P. Vincent. “A connection between score matching and denoising autoencoders.” In: *Neural computation* (2011).
- [30] Y. Song and S. Ermon. “Generative modeling by estimating gradients of the data distribution.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [31] G. E. Hinton. “Training products of experts by minimizing contrastive divergence.” In: *Neural computation* (2002).
- [32] S. Saremi, A. Mehrjou, B. Schölkopf, and A. Hyvärinen. “Deep energy estimator networks.” In: *arXiv preprint arXiv:1805.08306* (2018).
- [33] Z. Li, Y. Chen, and F. T. Sommer. “Learning Energy-Based Models in High-Dimensional Spaces with Multi-scale Denoising Score Matching.” In: *arXiv preprint arXiv:1910.07762* (2019).
- [34] M. Welling and Y. W. Teh. “Bayesian learning via stochastic gradient Langevin dynamics.” In: *International Conference on Machine Learning (ICML)*. 2011.
- [35] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning deep representations by mutual information estimation and maximization.” In: *International Conference on Learning Representations (ICLR)*. 2019.
- [36] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning representations by maximizing mutual information across views.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [37] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations.” In: *arXiv preprint arXiv:2002.05709* (2020).
- [38] A. Mnih and Y. W. Teh. “A fast and simple algorithm for training neural probabilistic language models.” In: *International Conference on Machine Learning (ICML)*. 2012.
- [39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013.
- [40] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the limits of language modeling.” In: *arXiv preprint arXiv:1602.02410* (2016).

- [41] Z. Ma and M. Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [42] H. Sasaki and A. Hyvärinen. “Neural-kernelized conditional density estimation.” In: *arXiv preprint arXiv:1806.01754* (2018).
- [43] I. Khemakhem, R. P. Monti, D. P. Kingma, and A. Hyvärinen. “ICE-BeeM: Identifiable Conditional Energy-Based Deep Models.” In: *arXiv preprint arXiv:2002.11537* (2020).
- [44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [45] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. “Microsoft COCO: Common objects in context.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014.
- [46] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature pyramid networks for object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [47] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. “Acquisition of localization confidence for accurate object detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [48] M. Danelljan and G. Bhat. *PyTracking: Visual tracking library based on PyTorch*. <https://github.com/visionml/pytracking>. Accessed: 30/04/2020. 2019.
- [49] M. Muller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem. “TrackingNet: A large-scale dataset and benchmark for object tracking in the wild.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [50] H. Fan, L. Lin, F. Yang, P. Chu, G. Deng, S. Yu, H. Bai, Y. Xu, C. Liao, and H. Ling. “LaSOT: A high-quality benchmark for large-scale single object tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [51] L. Huang, X. Zhao, and K. Huang. “GOT-10k: A large high-diversity benchmark for generic object tracking in the wild.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [52] M. Mueller, N. Smith, and B. Ghanem. “A benchmark and simulator for UAV tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016.

- [53] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey. “Need for speed: A benchmark for higher frame rate object tracking.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
- [54] Y. Wu, J. Lim, and M.-H. Yang. “Object tracking benchmark.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2015).
- [55] P. Voigtlaender, J. Luiten, P. H. Torr, and B. Leibe. “Siam R-CNN: Visual tracking by re-detection.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [56] H. Nam and B. Han. “Learning multi-domain convolutional neural networks for visual tracking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [57] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg. “Unveiling the power of deep tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [58] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. “Distractor-aware siamese networks for visual object tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [59] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [60] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. “Fully-Convolutional Siamese Networks for Object Tracking.” In: *European Conference on Computer Vision (ECCV) Workshops*. 2016.

## Supplementary Material

In this supplementary material, we provide additional details and results. It consists of Appendix A - Appendix D. Appendix A contains a detailed algorithm for our employed prediction strategy. Further experimental details are provided in Appendix B for 1D regression, and in Appendix C for object detection. Lastly, Appendix D contains details and further results for the visual tracking experiments. Note that equations, tables, figures and algorithms in this supplementary document are numbered with the prefix “S”. Numbers without this prefix refer to the main paper.

### A Prediction Algorithm

Our prediction procedure (Section 2.2) is detailed in Algorithm S1, where  $\lambda$  denotes the gradient ascent step-length,  $\eta$  is a decay of the step-length and  $T$  is the number of iterations.

---

**Algorithm S1** Prediction via gradient-based refinement.

---

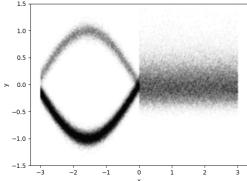
**Input:**  $x^*$ ,  $\hat{y}$ ,  $T$ ,  $\lambda$ ,  $\eta$ .

```
1:  $y \leftarrow \hat{y}$ .  
2: for  $t = 1, \dots, T$  do  
3:   PrevValue  $\leftarrow f_\theta(x^*, y)$ .  
4:    $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ .  
5:   NewValue  $\leftarrow f_\theta(x^*, \tilde{y})$ .  
6:   if NewValue > PrevValue then  
7:      $y \leftarrow \tilde{y}$ .  
8:   else  
9:      $\lambda \leftarrow \eta \lambda$ .  
10:  Return  $y$ .
```

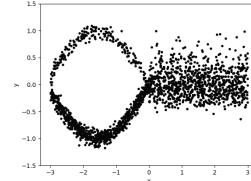
---

### B 1D Regression

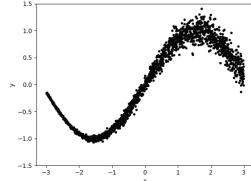
Here, we provide details on the two synthetic datasets, the network architecture, the evaluation procedure, and hyperparameters used for our 1D regression experiments (Section 4.1). For all seven training methods, the DNN  $f_\theta(x, y)$  was trained (by minimizing the associated loss  $J(\theta)$ ) for 75 epochs with a batch size of 32 using the ADAM [59] optimizer.



**Figure S1:** Visualization of the true  $p(y|x)$  for the first 1D regression dataset.



**Figure S2:** Training data  $\{(x_i, y_i)\}_{i=1}^{2000}$  for the first 1D regression dataset.



**Figure S3:** Training data  $\{(x_i, y_i)\}_{i=1}^{2000}$  for the second 1D regression dataset.

## B.1 Datasets

The ground truth  $p(y|x)$  for the first dataset is visualized in Figure S1. It is defined by a mixture of two Gaussian components (with weights 0.2 and 0.8) for  $x < 0$ , and a log-normal distribution (with  $\mu = 0.0$ ,  $\sigma = 0.25$ ) for  $x \geq 0$ . The training data  $\mathcal{D}_1 = \{(x_i, y_i)\}_{i=1}^{2000}$  was generated by uniform random sampling of  $x$  in the interval  $[-3, 3]$ , and is visualized in Figure S2. The ground truth  $p(y|x)$  for the second dataset is defined according to,

$$\begin{aligned} p(y|x) &= \mathcal{N}(y; \mu(x), \sigma^2(x)), \\ \mu(x) &= \sin(x), \quad \sigma(x) = 0.15(1 + e^{-x})^{-1}. \end{aligned} \tag{S1}$$

The training data  $\mathcal{D}_2 = \{(x_i, y_i)\}_{i=1}^{2000}$  was generated by uniform random sampling of  $x$  in the interval  $[-3, 3]$ , and is visualized in Figure S3.

## B.2 Network Architecture

The DNN  $f_\theta(x, y)$  is a feed-forward network taking  $x \in \mathbb{R}$  and  $y \in \mathbb{R}$  as inputs. It consists of two fully-connected layers (dimensions:  $1 \rightarrow 10$ ,  $10 \rightarrow 10$ ) for  $x$ , one fully-connected layer ( $1 \rightarrow 10$ ) for  $y$ , and four fully-connected layers ( $20 \rightarrow 10$ ,  $10 \rightarrow 10$ ,  $10 \rightarrow 10$ ,  $10 \rightarrow 1$ ) processing the concatenated  $(x, y)$  feature vector.

## B.3 Evaluation

The training methods are evaluated in terms of the KL divergence  $D_{\text{KL}}(p(y|x) \parallel p(y|x; \theta))$  between the learned EBM  $p(y|x; \theta) = e^{f_\theta(x, y)} / \int e^{f_\theta(x, \tilde{y})} d\tilde{y}$  and the true conditional density  $p(y|x)$ . To approximate  $D_{\text{KL}}(p(y|x) \parallel p(y|x; \theta))$ , we compute  $e^{f_\theta(x, y)}$  and  $p(y|x)$  for all  $(x, y)$  pairs in a  $2048 \times 2048$  uniform grid in the region  $\{(x, y) \in \mathbb{R}^2 : x \in [-3, 3], y \in$

$[-3, 3]\}$ . We then normalize across all values associated with each  $x$ , employ the formula for KL divergence between two discrete distributions  $q_1(y)$  and  $q_2(y)$ ,

$$D_{\text{KL}}(q_1 \parallel q_2) = \sum_{y \in \mathcal{Y}} q_1(y) \log \frac{q_1(y)}{q_2(y)}, \quad (\text{S2})$$

and finally average over all 2048 values of  $x$ . For each dataset and training method, we independently train the DNN  $f_\theta(x, y)$  and compute  $D_{\text{KL}}(p(y|x) \parallel p(y|x; \theta))$  20 times. We then take the mean of the 5 best runs, and finally average this value for the two datasets.

## B.4 Hyperparameters

The number of samples  $M = 1024$  for all applicable training methods. All other hyperparameters were selected to optimize the performance, evaluated according to Section B.3.

**ML-IS** Following [14], we set  $K = 2$  in the proposal distribution  $q(y|y_i)$  in (4). After ablation, we set  $\sigma_1 = 0.2$ ,  $\sigma_2 = 1.6$ .

**KLD-IS** We use the same proposal distribution  $q(y|y_i)$  as for ML-IS. After ablation, we set  $\sigma = 0.025$  in  $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$ .

**ML-MCMC** After ablation, we set the Langevin dynamics step-length  $\alpha = 0.05$ .

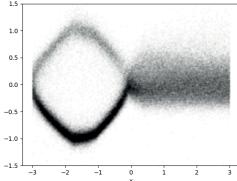
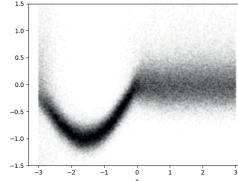
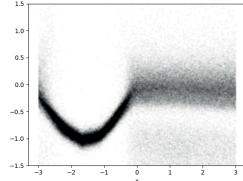
**NCE** To match ML-IS, we set  $K = 2$  in the noise distribution  $p_N(y|y_i)$  in (11). After ablation, we set  $\sigma_1 = 0.1$ ,  $\sigma_2 = 0.8$ .

**DSM** After ablation, we set  $\sigma = 0.2$  in  $p_\sigma(\tilde{y}|y_i) = \mathcal{N}(\tilde{y}; y_i, \sigma^2 I)$ .

**NCE+** We use the same noise distribution  $p_N(y|y_i)$  as for NCE. After ablation, we set  $\beta = 0.025$ .

## B.5 Qualitative Results

An example of  $p(y|x; \theta)$  trained using NCE on the first dataset is visualized in Figure S4. As can be observed, this is quite close to the true  $p(y|x)$  visualized in Figure S1. Similar results are obtained with all four top-performing training methods. Examples of  $p(y|x; \theta)$  instead trained using DSM and SM are visualized in Figure S5 and Figure S6, respectively. These do not approximate the true  $p(y|x)$  quite as well, matching the worse performance in terms of  $D_{\text{KL}}$  reported in Table 1.

**Figure S4:** Example of  $p(y|x; \theta)$  trained with NCE.**Figure S5:** Example of  $p(y|x; \theta)$  trained with DSM.**Figure S6:** Example of  $p(y|x; \theta)$  trained with SM.**Table S1:** Used step-lengths  $\lambda_{\text{pos}}$  and  $\lambda_{\text{size}}$  for the object detection experiments.

	ML-IS	ML-MCMC-1	ML-MCMC-4	ML-MCMC-8	KLD-IS	NCE	DSM	NCE+
$\lambda_{\text{pos}}$	0.0004	0.000025	0.000025	0.000025	0.0004	0.0004	0.000025	0.0008
$\lambda_{\text{size}}$	0.0016	0.0001	0.0001	0.0001	0.0016	0.0016	0.0001	0.0032

## C Object Detection

Here, we provide details on the prediction procedure and hyperparameters used for our object detection experiments (Section 4.2). We employ an identical network architecture and training procedure as described in [14], only modifying the loss when using a different method than ML-IS to train  $f_\theta(x, y)$ .

### C.1 Prediction

Predictions  $y^*$  are produced by performing guided NMS [47] followed by gradient-based refinement (Algorithm S1), taking the Faster-RCNN detections as initial estimates  $\hat{y}$ . As in [14], we run  $T = 10$  gradient ascent iterations. We fix the step-length decay to  $\eta = 0.5$ , which is the value used in [14]. For each trained model, we select the gradient ascent step-length  $\lambda$  to optimize performance in terms of AP on the *2017 val* split of COCO [45]. Like [14], we use different step-lengths for the bounding box position ( $\lambda_{\text{pos}}$ ) and size ( $\lambda_{\text{size}}$ ). We start this ablation with  $\lambda_{\text{pos}} = 0.0001$ ,  $\lambda_{\text{size}} = 0.0004$ . The used step-lengths for all training methods are given in Table S1.

### C.2 Hyperparameters

The number of samples  $M = 128$  for all applicable training methods. All other hyperparameters were selected to optimize performance in terms of AP on the *2017 val* split of COCO [45].

**ML-IS** Following [14], we set  $K = 3$  in the proposal distribution  $q(y|y_i)$  in (4) with  $\sigma_1 = 0.0375$ ,  $\sigma_2 = 0.075$ ,  $\sigma_3 = 0.15$ .

**Table S2:** Ablation study for KLD-IS, on the *2017 val* split of COCO [45].

$\sigma$	0.0075	0.015	0.0225	0.03	0.0375
AP (%) $\uparrow$	38.32	39.19	<b>39.38</b>	39.33	39.23

**Table S3:** Ablation study for ML-MCMC-1, on the *val* split of COCO [45].

$\alpha$	0.000001	0.00001	0.0001
AP (%) $\uparrow$	36.14	<b>36.19</b>	36.04

**Table S4:** Ablation study for NCE, on the *2017 val* split of COCO [45].

$\{\sigma_k\}_{k=1}^3$	{0.0125, 0.025, 0.05}	{0.025, 0.05, 0.1}	{0.05, 0.1, 0.2}	{0.075, 0.15, 0.3}	{0.1, 0.2, 0.4}
AP (%) $\uparrow$	38.58	38.95	39.12	<b>39.17</b>	39.05

**KLD-IS** We use the same proposal distribution  $q(y|y_i)$  as for ML-IS. Based on the ablation study in Table S2, we set  $\sigma = 0.0225$  in  $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$ .

**ML-MCMC** Based on the ablation study in Table S3, we set the Langevin dynamics step-length  $\alpha = 0.00001$ .

**NCE** To match ML-IS, we set  $K = 3$  in the noise distribution  $p_N(y|y_i)$  in (11). Based on the ablation study in Table S4, we set  $\sigma_1 = 0.075$ ,  $\sigma_2 = 0.15$ ,  $\sigma_3 = 0.3$ .

**DSM** Based on the ablation study in Table S5, we set  $\sigma = 0.075$  in  $p_\sigma(\tilde{y}|y_i) = \mathcal{N}(\tilde{y}; y_i, \sigma^2 I)$ .

**NCE+** We use the same noise distribution  $p_N(y|y_i)$  as for NCE. Based on the ablation study in Table S6, we set  $\beta = 0.1$ .

### C.3 Detailed Results

A comparison of the training methods on the *2017 val* split of COCO [45] is provided in Table S7.

## D Visual Tracking

Here, we provide detailed results and hyperparameters for our visual tracking experiments (Section 5). We employ an identical network architecture, train-

**Table S5:** Ablation study for DSM, on the *2017 val* split of COCO [45].

$\sigma$	0.0375	0.075	0.15
AP (%) $\uparrow$	36.11	<b>36.12</b>	36.05

**Table S6:** Ablation study for NCE+, on the *2017 val* split of COCO [45].

$\beta$	0.05	0.1	0.15
AP (%) $\uparrow$	39.27	<b>39.36</b>	39.32

**Table S7:** Comparison of training methods for the object detection experiments, on the *2017 val* split of COCO [45]. NCE+ and KLD-IS achieve the best performance.

	ML-IS	ML-MCMC-1	ML-MCMC-4	ML-MCMC-8	KLD-IS	NCE	DSM	NCE+
AP (%) ↑	39.11	36.19	36.24	36.25	<b>39.38</b>	39.17	36.12	39.36
AP <sub>50</sub> (%) ↑	57.95	57.34	57.45	57.28	<b>58.07</b>	57.96	57.29	57.99
AP <sub>75</sub> (%) ↑	41.97	38.77	38.81	38.88	42.47	42.07	38.84	<b>42.63</b>
Training Cost ↓	1.03	2.47	7.05	13.3	<b>1.02</b>	1.04	3.84	1.09

ing procedure and prediction procedure for DiMP-KLD-IS, DiMP-NCE and DiMP-NCE+, only the loss is modified.

## D.1 Training Parameters

DiMP-KLD-IS is obtained by combining the DiMP [21] method for center point regression with the PrDiMP [15] bounding box regression approach, and modifying a few training parameters. Specifically, we change the batch size from 10 to 20, we change the LaSOT sampling weight from 0.25 to 1.0, we change the number of samples per epoch from 26 000 to 40 000, and we add random horizontal flipping with probability 0.5. Since we increase the batch size, we also freeze conv1, layer1 and layer2 of the ResNet backbone to save memory.

## D.2 Hyperparameters

The number of samples  $M = 128$  for all three training methods.

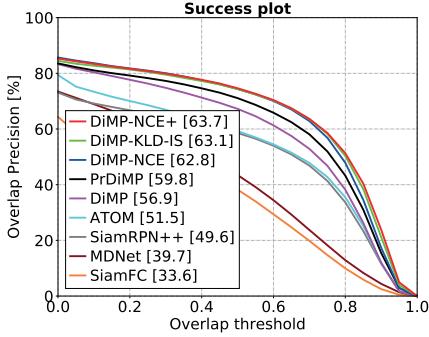
**DiMP-KLD-IS** Following PrDiMP, we set  $K = 2$  in the proposal distribution  $q(y|y_i)$  in (4) with  $\sigma_1 = 0.05$ ,  $\sigma_2 = 0.5$ , and we set  $\sigma = 0.05$  in  $p(y|y_i) = \mathcal{N}(y; y_i, \sigma^2 I)$ .

**DiMP-NCE** Matching DiMP-KLD-IS, we set  $K = 2$  in the noise distribution  $p_N(y|y_i)$  in (11) with  $\sigma_1 = 0.05$ ,  $\sigma_2 = 0.5$ . A quick ablation study on the vali-

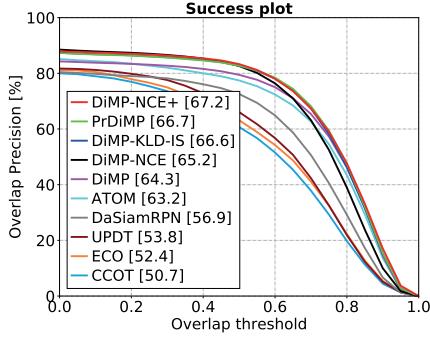
**Table S8:** Full results on the TrackingNet [49] test set, in terms of precision, normalized precision, and success (AUC). Our proposed DiMP-NCE+ is here only outperformed by the very recent SiamRCNN [55]. SiamRCNN is however slower than DiMP-NCE+ (5 FPS vs 30 FPS) and employs a larger backbone network (ResNet101 vs ResNet50).

	SiamFC [60]	MDNet [56]	UPDT [57]	DaSiamRPN [58]	ATOM [20]	SiamRPN++ [19]	DiMP [21]	SiamRCNN [55]	PrDiMP [15]	DiMP- KLD-IS	DiMP- NCE	DiMP- NCE+
Precision ↑	53.3	56.5	55.7	59.1	64.8	69.4	68.7	<b>80.0</b>	70.4	73.3	69.8	73.7
Norm. Prec. ↑	66.6	70.5	70.2	73.3	77.1	80.0	80.1	<b>85.4</b>	81.6	83.5	82.4	83.7
Success (AUC)	57.1	60.6	61.1	63.8	70.3	73.3	74.0	<b>81.2</b>	75.8	78.1	77.1	78.7

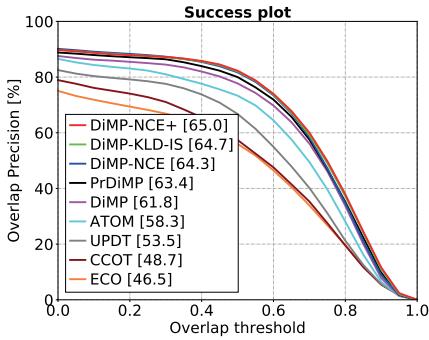
↑



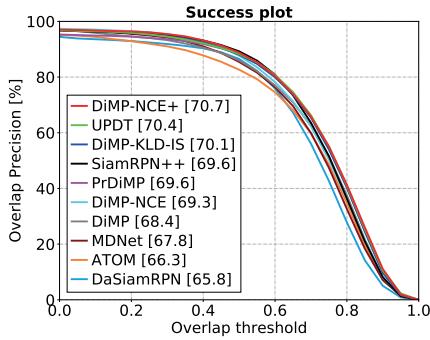
**Figure S7:** Success plot on LaSOT [50].



**Figure S8:** Success plot on UAV123 [52].



**Figure S9:** Success plot on NFS [53].



**Figure S10:** Success plot on OTB-100 [54].

dation set of GOT-10k [51] did not find values of  $\sigma_1, \sigma_2$  resulting in improved performance.

**DIMP-NCE+** We use the same noise distribution  $p_N(y|y_i)$  as for NCE. We set  $\beta = 0.1$ , as this corresponded to the best performance on the object detection experiments (Table S6).

### D.3 Detailed Results

Full results on the TrackingNet [49] test set, in terms of all three TrackingNet metrics, are found in Table S8. Success plots for LaSOT, UAV123, NFS and OTB-100 are found in Figure S7-S10, showing the overlap precision  $OP_T$  as a function of the overlap threshold  $T$ .



# Paper III

## Title

Learning Proposals for Practical Energy-Based Regression

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Learning Proposals for Practical Energy-Based Regression.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2022



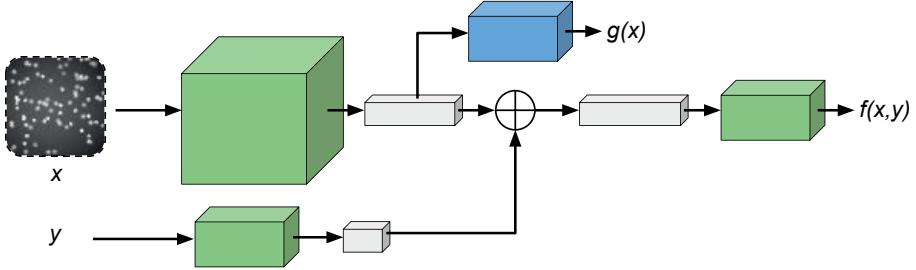
# Learning Proposals for Practical Energy-Based Regression

## Abstract

Energy-based models (EBMs) have experienced a resurgence within machine learning in recent years, including as a promising alternative for probabilistic regression. However, energy-based regression requires a proposal distribution to be manually designed for training, and an initial estimate has to be provided at test-time. We address both of these issues by introducing a conceptually simple method to automatically learn an effective proposal distribution, which is parameterized by a separate network head. To this end, we derive a surprising result, leading to a unified training objective that jointly minimizes the KL divergence from the proposal to the EBM, and the negative log-likelihood of the EBM. At test-time, we can then employ importance sampling with the trained proposal to efficiently evaluate the learned EBM and produce stand-alone predictions. Furthermore, we utilize our derived training objective to learn mixture density networks (MDNs) with a jointly trained energy-based teacher, consistently outperforming conventional MDN training on four real-world regression tasks within computer vision. Code is available at [https://github.com/fregu856/ebms\\_proposals](https://github.com/fregu856/ebms_proposals).

## 1 Introduction

Energy-based models (EBMs) [1] have been extensively studied within the field of machine learning in the past [2, 3, 4, 5, 6]. By using deep neural networks to parameterize the energy function [7], EBMs have recently also experienced a significant resurgence. Most widely, EBMs are now employed for generative modelling tasks [8, 10, 11, 12, 13, 14, 15, 16, 17, 9]. Recent work has further demonstrated the promise of EBMs for probabilistic regression, achieving impressive results for a variety of important low-dimensional regression tasks, including object detection, visual tracking, pose estimation, age estimation and robot policy learning [18, 19, 20, 21, 22, 23, 24].



**Figure 1:** We propose a method to automatically learn an effective MDN proposal  $q(y|x; \phi)$  (blue) during training of the EBM  $p(y|x; \theta)$  (green), thus addressing the main practical limitations of energy-based regression. The MDN  $q$  is trained by minimizing its KL divergence to the EBM  $p$ , i.e. by minimizing  $D_{\text{KL}}(p \parallel q)$ .

Probabilistic regression aims to estimate the predictive conditional distribution  $p(y|x)$  of the target  $y$  given the input  $x$  [25, 26, 27, 28, 29, 30, 31]. As its primary advantage, the EBM directly represents this distribution by a neural network through a learnable energy function  $f_\theta(x, y)$ , as  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$ . While this flexibility allows the EBM to learn highly complex and accurate distributions, it comes at a significant cost. Firstly, evaluating the resulting distribution  $p(y|x; \theta)$  is generally intractable, as it requires the computation of the partition function  $Z(x, \theta)$ . This particularly imposes challenges for training the EBM, which often leads to application of Monte Carlo approximations with hand-tuned proposal distributions in order to pursue maximum likelihood-based learning. Secondly, EBMs are known to be difficult to sample from, which complicates their practical use at test-time. To produce predictions, prior work [18, 19, 20, 21] resort to gradient-based refinement of an initial estimate generated by a separately trained network.

In this work, we address both aforementioned drawbacks of this energy-based regression approach by jointly learning a proposal distribution  $q$  during EBM training. Specifically, we parametrize the proposal using a mixture density network (MDN) [32]  $q(y|x; \phi)$  conditioned on the input  $x$ . In order to maximize its effectiveness during training, we learn  $q$  by minimizing its Kullback–Leibler (KL) divergence to the EBM  $p$ . To this end, we derive a surprising result, consisting of a unified objective that jointly minimizes the KL divergence from the proposal  $q$  to the EBM  $p$  *and* the negative log-likelihood (NLL) of the latter. As our result does not rely on the reparameterization trick, it is directly applicable to a wide class of proposal distributions, including mixture models. Compared to previous approaches for training EBMs for regression, our approach does not require tedious hand-tuning of the proposal distribution, instead providing a fully learnable alternative. Moreover, rather than conditioning on the ground-truth target  $y$ , our proposal distribution  $q$  is conditioned

on the input  $x$ . It can therefore be employed at test-time to efficiently evaluate and sample from the EBM.

Learning the MDN  $q$  according to our derived objective leads to another interesting observation: MDNs trained to mimic the EBM via this objective tend to learn more accurate predictive distributions compared to an MDN trained with the standard NLL loss. Inspired by this finding, we apply our derived result for a second purpose, namely to find a better learning formulation for MDNs. When jointly trained with an energy-based teacher network according to our objective, the resulting MDN is shown to consistently outperform the NLL baseline on challenging real-world regression tasks. In contrast to a single ground-truth sample, the EBM provides comprehensive supervision for the predictive distribution  $q$ , leading to a more accurate model of the underlying true distribution.

In summary, our main contributions are as follows:

- We derive an efficient and convenient objective that can be employed to train a parameterized distribution  $q(y|x; \phi)$  by directly minimizing its KL divergence to a conditional EBM.
- We employ the proposed objective to jointly learn an effective MDN proposal distribution during EBM training, thus addressing the main practical limitations of energy-based regression.
- We further utilize the proposed objective to improve training of stand-alone MDNs, learning more accurate predictive distributions compared to MDNs trained by minimizing the NLL.
- We perform comprehensive experiments on four challenging computer vision regression tasks.

## 2 Background

Regression entails learning to predict targets  $y^* \in \mathcal{Y}$  from inputs  $x^* \in \mathcal{X}$ , given a training set of  $N$  i.i.d. input-target pairs  $\{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ . The target space  $\mathcal{Y}$  is continuous,  $\mathcal{Y} = \mathbb{R}^K$  for some  $K \geq 1$ . We focus on probabilistic regression, which aims to not only produce a prediction  $y^*$ , but also estimate the full predictive conditional distribution  $p(y|x)$ . This probabilistic formulation provides a more general view of the regression problem, allowing for the encapsulation of uncertainty, generation of multiple hypotheses, and handling of ill-posed settings [25, 26, 27, 28, 29, 30, 31].

## 2.1 Energy-Based Regression

In energy-based regression [18, 19, 20], the task is addressed by learning to model the distribution  $p(y|x)$  with a conditional EBM  $p(y|x; \theta)$ , defined according to,

$$p(y|x; \theta) = \frac{e^{f_\theta(x, y)}}{Z(x, \theta)}, \quad Z(x, \theta) = \int e^{f_\theta(x, \tilde{y})} d\tilde{y}. \quad (1)$$

The EBM  $p(y|x; \theta)$  is directly specified via  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ , a deep neural network (DNN) mapping any input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  to a scalar  $f_\theta(x, y) \in \mathbb{R}$ . The EBM in (1) is therefore highly flexible and capable of learning complex distributions directly from data. However, the resulting distribution  $p(y|x; \theta)$  is also challenging to evaluate or sample from, since its partition function  $Z(x, \theta)$  generally is intractable. The EBM  $p(y|x; \theta)$  is therefore quite challenging to train, and a variety of different approaches have recently been explored [20, 33]. The most straightforward approach would be to directly minimize the NLL  $\mathcal{L}(\theta) = \sum_{i=1}^N \log Z(x_i, \theta) - f_\theta(x_i, y_i)$ . While exact computation of  $\mathcal{L}(\theta)$  is intractable, importance sampling can be utilized to approximate the  $\log Z(x_i, \theta)$  term. The DNN  $f_\theta(x, y)$  can therefore be trained by minimizing the resulting loss,

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{q(y_i^{(m)})} \right) - f_\theta(x_i, y_i), \quad (2)$$

where  $\{y_i^{(m)}\}_{m=1}^M \sim q(y)$  are  $M$  samples drawn from a proposal distribution  $q(y)$ . The aforementioned approach is relatively simple, yet it has been shown effective for various regression tasks within computer vision [18, 19, 20]. In these works, the proposal  $q(y)$  is set to a mixture of  $K$  Gaussian components centered at the true target  $y_i$ , i.e.  $q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I)$ . Training thus requires the task-dependent hyperparameters  $K$  and  $\{\sigma_k^2\}_{k=1}^K$  to be carefully tuned, limiting general applicability. Moreover, this proposal  $q(y)$  depends on  $y_i$  and can therefore only be utilized during training. To produce a prediction  $y^*$  at test-time, previous energy-based regression methods [18, 19, 20, 21, 22, 23] employ gradient ascent to refine an initial estimate  $\hat{y}$ . This prediction strategy therefore requires access to a good initial estimate. Hence, most previous works [18, 19, 20, 21] even rely on a separately trained DNN to provide  $\hat{y}$ , further limiting general applicability.

## 2.2 Mixture Density Networks

Alternatively, the regression task can be addressed by learning to model the conditional distribution  $p(y|x)$  with an MDN  $q(y|x; \phi)$  [32, 30, 34, 31]. An

MDN is a mixture of  $K$  components of a certain base distribution. Specifically for a Gaussian MDN, the distribution  $q(y|x; \phi)$  is defined according to,

$$q(y|x; \phi) = \sum_{k=1}^K \pi_\phi^{(k)}(x) \mathcal{N}(y; \mu_\phi^{(k)}(x), \sigma_\phi^{(k)}(x)I), \quad (3)$$

where the set of Gaussian mixture parameters  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$  is outputted by a DNN  $g_\phi(x)$ . In contrast to EBMs, the MDN distribution  $q(y|x; \phi)$  is by design simple to both evaluate and sample from. The DNN  $g_\phi(x)$  can thus be trained by directly minimizing the NLL  $\mathcal{L}(\phi) = \sum_{i=1}^N -\log q(y_i|x_i; \phi)$ . While MDNs generally are less flexible models than EBMs, they are still capable of capturing multi-modality and other more complex features of the true distribution  $p(y|x)$ . MDNs thus offer a convenient yet quite flexible alternative to EBMs. Training an MDN  $q(y|x; \phi)$  via the NLL is however known to occasionally suffer from certain inefficiencies such as mode-collapse, and various more sophisticated training methods have therefore been explored [35, 36, 30, 37, 38].

### 3 Method

We first address the main practical limitations of energy-based regression by proposing a method to automatically learn an effective proposal  $q(y; \phi)$  during training of the EBM  $p(y|x; \theta)$  in (1). To enable  $q(y; \phi)$  to be utilized also at test-time, we condition it on the input  $x$  instead of on the true target  $y_i$ . We further require the resulting proposal distribution  $q(y|x; \phi)$  to be flexible, yet efficient and convenient to evaluate and sample from. In this work, we therefore parametrize the proposal  $q(y|x; \phi)$  using an MDN.

When training the EBM  $p(y|x; \theta)$  by minimizing the approximated NLL in (2), we wish to use the proposal  $q(y|x; \phi)$  that yields the best possible NLL approximation. In general, this is achieved when the proposal equals the EBM, i.e. when  $q(y|x; \phi) = p(y|x; \theta)$ <sup>1</sup>. We therefore aim to learn the proposal parameters  $\phi$  by directly minimizing the KL divergence to the EBM,  $D_{\text{KL}}(p \parallel q)$ . While this approach is conceptually simple and attractive, exact computation of  $D_{\text{KL}}(p \parallel q)$  is intractable. This calls for an effective and efficient approximation, which can easily be employed during training. In Section 3.1 we show that such an approximation, interestingly enough, is achieved by simply minimizing the objective (2) w.r.t. the proposal  $q(y|x; \phi)$ .

In Section 3.2, we further employ this result to design a method for jointly learning the EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$ . There, we also detail how  $q(y|x; \phi)$  can be utilized with importance sampling to approximately

---

<sup>1</sup>Details are provided in the supplementary material.

evaluate and sample from the EBM at test-time. Lastly, in Section 3.3 we propose to utilize our derived approximation of  $D_{\text{KL}}(p \parallel q)$  as an additional loss for training MDNs. We argue that guiding an MDN towards a more flexible and accurate distribution learned by the EBM provides more extensive supervision for the MDN in a regression setting, leading to improved results.

### 3.1 Learning the Proposal to Match an EBM

We have a parameterized distribution  $q(y|x; \phi)$  that we want to be a close approximation of the EBM  $p(y|x; \theta)$ . Specifically, we want to find the parameters  $\phi$  that minimize the KL divergence between  $q(y|x; \phi)$  and the EBM  $p(y|x; \theta)$ . Therefore, we seek to compute  $\nabla_{\phi} D_{\text{KL}}(p(y|x; \theta) \parallel q(y|x; \phi))$ , i.e. the gradient of the KL divergence w.r.t.  $\phi$ . The gradient  $\nabla_{\phi} D_{\text{KL}}$  is generally intractable, but can be conveniently approximated by the following result.

**Result 1.** *For a conditional EBM  $p(y|x; \theta) = e^{f_{\theta}(x,y)} / \int e^{f_{\theta}(x,\tilde{y})} d\tilde{y}$  and distribution  $q(y|x; \phi)$ ,*

$$\nabla_{\phi} D_{\text{KL}}(p \parallel q) \approx \nabla_{\phi} \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_{\theta}(x,y^{(m)})}}{q(y^{(m)}|x; \phi)} \right), \quad (4)$$

where  $\{y^{(m)}\}_{m=1}^M$  are  $M$  independent samples drawn from  $q(y|x; \phi)$ .

A complete derivation of Result 1 is provided in the supplementary material. Note that the samples  $\{y^{(m)}\}_{m=1}^M$  in (4) are drawn from  $q(y|x; \phi)$  but *not* considered functions of  $\phi$ , making this approximation particularly simple to compute in practice. Importantly, the approximation (4) does *not* rely on the reparameterization trick and is therefore directly applicable for a wide class of distributions  $q(y|x; \phi)$ , including mixture models. Given data  $\{x_i\}_{i=1}^N$ , Result 1 implies that  $q(y|x; \phi)$  can be trained to approximate the EBM  $p(y|x; \theta)$  by minimizing the loss,

$$J_{\text{KL}}(\phi) = \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_{\theta}(x_i,y_i^{(m)})}}{q(y_i^{(m)}|x_i; \phi)} \right), \quad (5)$$

where  $\{y_i^{(m)}\}_{m=1}^M \sim q(y|x_i; \phi)$ . Note that  $J_{\text{KL}}(\phi)$  is identical to the first term of the EBM loss  $J(\theta)$  in (2). In fact, since the second term  $f_{\theta}(x_i, y_i)$  in (2) does not depend on  $\phi$ , (2) can be used as a joint objective for training both  $q(y|x; \phi)$  and the EBM  $p(y|x; \theta)$ .

## 3.2 Practical Energy-Based Regression

We first employ our Result 1 to jointly train the EBM  $p(y|x; \theta) = e^{f_\theta(x,y)} / \int e^{f_\theta(x,\tilde{y})} d\tilde{y}$  and the MDN proposal  $q(y|x; \phi)$ . We define the MDN  $q(y|x; \phi)$  by adding a second network head  $g_\phi$  onto the same backbone feature extractor shared with the EBM DNN  $f_\theta$ . The MDN head  $g_\phi$  outputs the Gaussian mixture model parameters  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$ , as defined in (3). The resulting overall network architecture is illustrated in Figure 1.

### Training

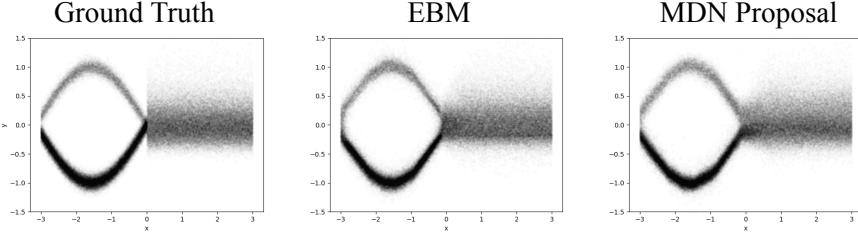
We train the EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$  jointly using standard techniques based on stochastic gradient descent. At each iteration, we first predict the MDN mixture parameters  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$  and draw  $M$  samples  $\{y_i^{(m)}\}_{m=1}^M \sim q(y|x_i; \phi)$  from the resulting distribution. The MDN parameters  $\phi$  are then updated via  $J_{\text{KL}}(\phi)$  in (5), while the EBM parameters  $\theta$  are updated via  $J(\theta)$  in (2). In fact, this can be implemented by jointly minimizing (2) w.r.t. both  $\theta$  and  $\phi$ .

EBMs can however be trained also via various alternative approaches, including noise contrastive estimation (NCE) [39, 40]. How EBMs should be trained specifically for regression tasks was extensively studied in [20], concluding that NCE should be considered the go-to method. NCE entails training the EBM DNN  $f_\theta$  by minimizing the loss,

$$J_{\text{NCE}}(\theta) = -\frac{1}{N} \sum_{i=1}^N J_{\text{NCE}}^{(i)}(\theta),$$

$$J_{\text{NCE}}^{(i)}(\theta) = \log \frac{\exp\{f_\theta(x_i, y_i^{(0)}) - \log q(y_i^{(0)})\}}{\sum_{m=0}^M \exp\{f_\theta(x_i, y_i^{(m)}) - \log q(y_i^{(m)})\}}, \quad (6)$$

where  $y_i^{(0)} \triangleq y_i$ , and  $\{y_i^{(m)}\}_{m=1}^M$  are  $M$  samples drawn from a noise distribution  $q(y)$ . The NCE loss  $J_{\text{NCE}}(\theta)$  in (6) can be interpreted as the softmax cross-entropy loss for a classification problem, distinguishing the true target  $y_i$  from the  $M$  noise samples  $\{y_i^{(m)}\}_{m=1}^M \sim q(y)$ . Moreover,  $J_{\text{NCE}}(\theta)$  has much similarity with the importance sampling-based loss  $J(\theta)$  in (2) [40, 41]. In particular, the noise distribution  $q(y)$  in NCE directly corresponds to the proposal  $q$  in (2). In fact, all prior work [20, 21, 22] on energy-based regression using NCE has employed the same manually designed distribution  $q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I)$ . Due to the close relationship between NCE and importance sampling, our approach for learning the proposal distribution  $q$  is also applicable for NCE-based training of the EBM. In this work, we there-



**Figure 2:** An illustrative 1D regression problem [20], demonstrating the effectiveness of our proposed method to jointly train an EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$ . In this example, the MDN has  $K = 4$  components. The EBM is trained using NCE with  $q(y|x; \phi)$  acting as the noise distribution, whereas the MDN is trained by minimizing its KL divergence to  $p(y|x; \theta)$ , i.e. by minimizing  $D_{\text{KL}}(p \parallel q)$ .

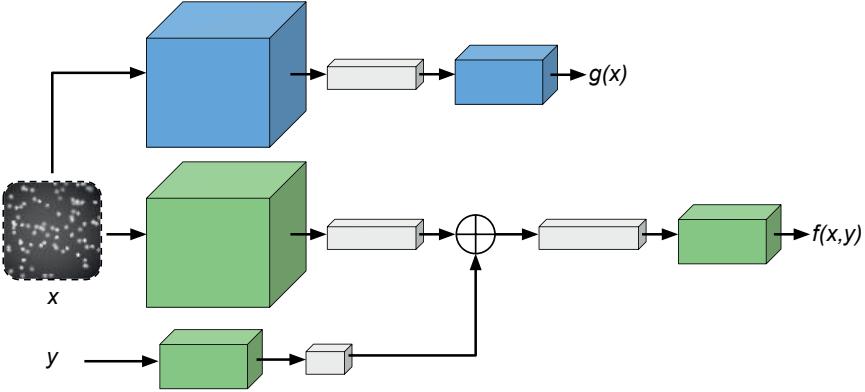
fore adopt the NCE loss to train the EBM  $p(y|x; \theta)$ , since it has been shown to achieve favorable results [20].

Our approach still entails jointly training the EBM  $p(y|x; \theta)$  and MDN  $q(y|x; \phi)$ , but employs NCE with  $q(y|x; \phi)$  acting as a noise distribution for training the EBM. At each iteration we thus draw samples  $\{y_i^{(m)}\}_{m=1}^M \sim q(y|x_i; \phi)$ , update  $\phi$  via the loss  $J_{\text{KL}}(\phi)$  in (5), and update  $\theta$  via  $J_{\text{NCE}}(\theta)$  in (6). Note that the update of the MDN parameters  $\phi$  only affects the added network head in Figure 1, not the feature extractor. The effectiveness of this proposed joint training method is demonstrated on an illustrative 1D regression problem in Figure 2. In the supplementary material (Figure S3), we also show an example of how both the EBM and the MDN proposal iteratively converge towards the ground truth during joint training.

Training an EBM using our joint training method is somewhat slower than using standard NCE with the manually designed  $q(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I)$ , since we now also have to update the added network head  $g_\phi$  of the MDN proposal at each iteration. For both methods, the main computational bottleneck is however the backbone feature extractor. In fact, our proposed method usually requires less total training in practice, since the task-dependent hyperparameters  $K$  and  $\{\sigma_k^2\}_{k=1}^K$  have to be tuned for the NCE baseline.

## Prediction

To avoid evaluating the intractable  $Z(x^*, \theta)$  at test-time, previous work on energy-based regression [18, 19, 20, 21] approximately compute  $\arg \max_y p(y|x^*; \theta) = \arg \max_y f_\theta(x^*, y)$  to produce a prediction  $y^*$ . Specifically,  $T$  steps of gradient ascent,  $y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ , is used to refine an initial estimate  $\hat{y}$ , moving it towards a local maximum of  $f_\theta(x^*, y)$ . While shown to produce highly accurate predictions, this approach requires a good initial estimate  $\hat{y}$  to be provided at test-time, limiting its general applicability.



**Figure 3:** We extend our method of jointly training an EBM  $p(y|x; \theta)$  (green) and MDN  $q(y|x; \phi)$  (blue), improving MDN training. Instead of defining the MDN by adding a network head onto the EBM (Figure 1), the MDN is now defined in terms of a full DNN  $g_\phi$ .

In contrast to previous work [18, 19, 20, 21], we have access to a proposal  $q(y|x; \phi)$  that is conditioned only on the input  $x$  and thus can be utilized also at test-time. Since our MDN proposal  $q(y|x; \phi)$  has been trained to approximate the EBM  $p(y|x; \theta)$ , it can be utilized with self-normalized importance sampling [42] to efficiently approximate expectations  $\mathbb{E}_p$  w.r.t. the EBM  $p(y|x; \theta)$ ,

$$\begin{aligned} \mathbb{E}_p[\xi(y)] &= \int \xi(y)p(y|x; \theta)dy \approx \sum_{m=1}^M w^{(m)}\xi(y^{(m)}), \\ w^{(m)} &= \frac{e^{f_\theta(x, y^{(m)})}/q(y^{(m)}|x; \phi)}{\sum_{l=1}^M e^{f_\theta(x, y^{(l)})}/q(y^{(l)}|x; \phi)}. \end{aligned} \quad (7)$$

Here,  $\{y^{(m)}\}_{m=1}^M \sim q(y|x; \phi)$  are samples drawn from the MDN proposal, and  $\xi(y)$  is the quantity over which we are taking the expectation. For example, setting  $\xi(y) = y$  in (7) enables us to approximately compute the EBM mean. In this manner, we can thus directly produce a stand-alone prediction  $y^*$  for the EBM  $p(y|x; \theta)$ . Using the same technique, we can also estimate the variance of the EBM as a measure of its uncertainty.

Note that we can also draw approximate samples from the EBM  $p(y|x; \theta)$  by re-sampling with replacement from the set  $\{y^{(m)}\}_{m=1}^M \sim q(y|x; \phi)$  of proposal samples, drawing each  $y^{(m)}$  with probability  $w^{(m)}$  [43]. We demonstrate this sampling technique in Figure S4 in the supplementary material. There, we observe that the technique produces accurate EBM samples even when the proposal is unimodal and thus not a particularly close approximation of the EBM.

### 3.3 Improved MDN Training

Lastly, we employ Result 1 to improve the training of MDNs  $q(y|x; \phi)$ . We simply extend our proposed approach for jointly training an EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$  from Section 3.2. Instead of defining the MDN by adding a network head onto the EBM (Figure 1), we now define  $q(y|x; \phi)$  in terms of a full DNN  $g_\phi$ , as illustrated in Figure 3. Now, we thus train two separate DNNs  $f_\theta$  and  $g_\phi$ . As in Section 3.2, we train the EBM  $p(y|x; \theta)$  and MDN  $q(y|x; \phi)$  jointly. At each iteration, we draw samples  $\{y_i^{(m)}\}_{m=1}^M \sim q(y|x_i; \phi)$  and update  $\theta$  via the loss  $J_{\text{NCE}}(\theta)$  in (6). The EBM is thus trained using NCE with the MDN acting as a noise distribution. At each iteration, we also update the MDN parameters  $\phi$  via the loss,

$$J_{\text{MDN}}(\phi) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{q(y_i^{(m)}|x_i; \phi)} \right) - \frac{1}{2} \log q(y_i|x_i; \phi). \quad (8)$$

The MDN  $q(y|x; \phi)$  is thus trained by minimizing a sum of its NLL  $-\log q(y_i|x_i; \phi)$  and the  $J_{\text{KL}}(\theta)$  loss in (5). Compared to conventional MDN training, we thus employ our approximation of  $D_{\text{KL}}(p \parallel q)$  as an additional loss, guiding  $q(y|x; \phi)$  towards the EBM  $p(y|x; \theta)$ .

In contrast to MDNs, EBMs are not restricted to distributions which are convenient to evaluate and sample. The EBM  $p(y|x; \theta)$  is thus generally a more flexible model than  $q(y|x; \phi)$  and therefore able to better approximate the underlying true distribution  $p(y|x)$ . Compared to MDNs, which define a distribution  $q(y|x; \phi)$  by mapping  $x$  to the set  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$ , the EBM  $p(y|x; \theta) = e^{f_\theta(x, y)} / \int e^{f_\theta(x, \tilde{y})} d\tilde{y}$  also offers a more direct representation of the distribution via its scalar function  $f_\theta(x, y)$ , potentially leading to a more straightforward learning problem. Therefore, we argue that guiding the MDN  $q(y|x; \phi)$  towards the EBM  $p(y|x; \theta)$  during training via the loss  $J_{\text{MDN}}(\phi)$  in (8) should help mitigate some of the known inefficiencies of MDN training. We note that our proposed joint training approach is twice as slow compared to conventional MDN training, as two separate DNNs  $f_\theta$  and  $g_\phi$  are updated at each iteration. After training, the EBM can however be discarded and does therefore not affect the computational cost of the MDN at test-time.

## 4 Related Work

Our proposed approach to automatically learn a proposal during EBM training is related to the work of [44, 45, 46, 47], training EBMs for generative modelling tasks by jointly learning an auxiliary sampler via adversarial training. We instead train conditional EBMs for regression and are able to derive a particularly convenient KL divergence approximation (Result 1).

**Table 1:** Results for the EBM 1D regression experiments. Results are in terms of approximate KL divergence for the first dataset [20], and in terms of approximate NLL for the second [58].

Dataset	NCE					<b>Ours</b>		
	$\sigma_1=0.05$	$\sigma_1=0.1$	$\sigma_1=0.2$	$\sigma_1=0.4$	$\sigma_1=0.8$	$K=1$	$K=4$	$K=16$
[20]	0.042	0.036	0.040	0.042	0.042	0.038	<b>0.032</b>	0.035
[58]	2.30	1.98	1.72	1.67	1.70	1.69	1.67	<b>1.65</b>

Our approach is also inspired by the concept of cooperative learning [48, 49, 50, 51], which entails jointly training an EBM and a generator network via Markov chain Monte Carlo (MCMC) teaching. Specifically, the generator serves as a proposal and provides initial samples which are refined via MCMC to approximately sample from the EBM, training the EBM via contrastive divergence. Then, the generator network is trained to match these refined MCMC samples using a standard regression loss. Cooperative learning has recently also been extended to train EBMs for conditional generative modelling tasks [52, 53]. While our proposed method also entails jointly training conditional EBMs and proposals, we specifically study the important application of low-dimensional regression. In this setting, MCMC-based training of EBMs has been shown highly inefficient [20]. By deriving Result 1, we can instead employ the more effective training method of NCE, and train the proposal by directly minimizing its KL divergence to the EBM. Since MCMC is not employed, our proposed method is also computationally efficient, and very simple to implement, compared to cooperative learning.

Our method to improve the training of an MDN by guiding it towards an EBM is related to [15], who train a generative flow-based model jointly with an EBM through a minimax game. In contrast, our joint training method is non-adversarial and can even be implemented by directly minimizing one unified objective. On a conceptual level, our MDN training approach is also related to work on teacher-student networks and knowledge distillation [54, 55, 56, 57]. In a knowledge distillation problem, a teacher network is utilized to improve the performance of a more lightweight student network. While knowledge distillation for regression is not a particularly well-studied topic, it has been studied for image-based regression tasks in very recent work [57]. A student network is there enhanced by augmenting its training set with images and pseudo targets generated by a conditional GAN and a pre-trained teacher network, respectively. In contrast, our approach entails distilling the conditional EBM distribution  $p(y|x; \theta)$  into a student MDN for each example in the original training set. Furthermore, our approach trains the teacher EBM and student MDN jointly, where the student MDN generates proposal samples used for training the EBM teacher.

**Table 2:** Results in terms of approximate NLL for the EBM steering angle prediction experiments.

NCE					<b>Ours</b>
$\sigma_1=0.1, \sigma_2=20$	$\sigma_1=1, \sigma_2=20$	$\sigma_1=2, \sigma_2=20$	$\sigma_1=1, \sigma_2=10$	$\sigma_1=1, \sigma_2=40$	$K=4$
1.59±0.08	1.51±0.05	1.56±0.04	2.03±0.14	<b>1.39±0.02</b>	1.58±0.13

## 5 Experiments

We perform comprehensive experiments on illustrative 1D regression problems and four image-based regression tasks, which are all detailed below. We first evaluate our proposed method for automatically learning an effective proposal during EBM training in Section 5.1. There, we compare our EBM training method with NCE, achieving highly competitive performance across all five tasks without having to tune any task-dependent hyperparameters. In Section 5.2, we then evaluate our proposed approach for training MDNs. Compared to conventional MDN training, we consistently obtain improved test log-likelihoods. All experiments are implemented in PyTorch [59]. Example model and training code is found in the supplementary material, and our complete implementation is also made publicly available. All models were trained on individual NVIDIA TITAN Xp GPUs.

**1D Regression** We study two illustrative 1D regression problems with  $x \in \mathbb{R}$  and  $y \in \mathbb{R}$ . The first dataset is specified in [20] and contains 2 000 training examples. It is visualized in Figure 2. The second dataset is specified in [58], containing 1 900 test examples and 1 700 examples for training.

**Steering Angle Prediction** Here, we are given an image  $x$  from a forward-facing camera mounted inside of a car. The task is to predict the corresponding steering angle  $y \in \mathbb{R}$  of the car at that moment. We utilize the dataset from [60, 61], containing 12 271 examples. We randomly split the dataset into training (80%) and test (20%) sets. All images  $x$  are of size  $64 \times 64$ .

**Cell-Count Prediction** Given a synthetic fluorescence microscopy image  $x$ , the task is here to predict the number of cells  $y \in \mathbb{R}_+$  in the image. We utilize the dataset from [60, 61], which consists of 200 000 grayscale images of size  $64 \times 64$ . From this dataset, we randomly draw 10 000 images each to construct training and test sets. An example image  $x$  is visualized in Figure 1.

**Age Estimation** In age estimation, we are given an image  $x$  of a person’s face and are tasked with predicting the age  $y \in \mathbb{R}_+$  of this person. We utilize the UTKFace [62] dataset, specifically the processed version provided by [60, 61]. This dataset contains 14 760 examples, which we randomly split into training (80%) and test (20%) sets. All images  $x$  are of size  $64 \times 64$ .

**Table 3:** Results in terms of approximate NLL for the EBM cell-count prediction experiments.

NCE					<b>Ours</b> $K=4$
$\sigma_1=0.1, \sigma_2=40$	$\sigma_1=1, \sigma_2=40$	$\sigma_1=2, \sigma_2=40$	$\sigma_1=1, \sigma_2=20$	$\sigma_1=1, \sigma_2=80$	
$2.71 \pm 0.07$	<b><math>2.64 \pm 0.05</math></b>	$2.65 \pm 0.05$	$3.12 \pm 0.37$	$2.70 \pm 0.05$	$2.66 \pm 0.03$

**Table 4:** Results in terms of approximate NLL for the EBM age estimation experiments.

NCE					<b>Ours</b> $K=4$
$\sigma_1=0.01, \sigma_2=20$	$\sigma_1=0.1, \sigma_2=20$	$\sigma_1=1, \sigma_2=20$	$\sigma_1=0.1, \sigma_2=10$	$\sigma_1=0.1, \sigma_2=40$	
$4.18 \pm 0.30$	<b><math>3.81 \pm 0.18</math></b>	$4.13 \pm 0.48$	$3.97 \pm 0.21$	$4.47 \pm 0.25$	$4.30 \pm 0.30$

**Head-Pose Estimation** In this case, we are given an image  $x$  of a person, and the task is to predict the orientation  $y \in \mathbb{R}^3$  of this person’s head. Here,  $y$  is the yaw, pitch and roll angles of the head. We utilize the BIWI [63] dataset, specifically the processed version provided by [64]. We employ protocol 2 as defined in [64], giving 5 065 test images and 10 613 images for training. All images  $x$  are of size  $64 \times 64$ .

## 5.1 EBM Experiments

We first evaluate our proposed method for automatically learning an effective proposal during EBM training, by performing extensive experiments on all five regression tasks.

**1D Regression** The EBM DNN  $f_\theta(x, y)$  is here a simple feed-forward network, taking  $x \in \mathbb{R}$  and  $y \in \mathbb{R}$  as inputs. Separate sets of fully-connected layers extract features  $h_x \in \mathbb{R}^{10}$  from  $x$  and  $h_y \in \mathbb{R}^{10}$  from  $y$ . The two feature vectors are then concatenated and processed to output  $f_\theta(x, y) \in \mathbb{R}$ . The MDN network head  $g_\phi(x)$  takes the feature  $h_x \in \mathbb{R}^{10}$  as input and outputs  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$ . We use the ADAM [65] optimizer to jointly train  $f_\theta$  and  $g_\phi$ . For the first dataset, we follow [20] and evaluate the training methods in terms of how close the EBM  $p(y|x; \theta)$  is to the known ground truth  $p(y|x)$ , as measured by the (approximately computed) KL divergence. For the second dataset [58] we approximately compute the test set NLL of the EBM  $p(y|x; \theta)$ , by evaluating  $f_\theta(x, y)$  at densely sampled  $y$  values in an interval  $[y_{\min}, y_{\max}]$ . We compare our proposed approach with training the EBM using NCE, employing the noise distribution  $q(y) = \frac{1}{2} \sum_{k=1}^2 \mathcal{N}(y; y_i, \sigma_k^2 I)$ . Following [20], we set  $\sigma_1 = 0.1, \sigma_2 = 8\sigma_1$ . We also report results for the values  $\sigma_1 \in \{0.05, 0.2, 0.4, 0.8\}$ . For our proposed approach, we report results for using  $K \in \{1, 4, 16\}$  components in the MDN proposal. We train 20 networks for each setting and dataset, and report the mean of the 5 best runs. The

**Table 5:** Results in terms of approximate NLL for the EBM head-pose estimation experiments.

$\sigma_1=0.1, \sigma_2=20$	$\sigma_1=1, \sigma_2=20$	$\sigma_1=2, \sigma_2=20$	$\sigma_1=1, \sigma_2=10$	$\sigma_1=1, \sigma_2=40$	<b>Ours</b> $K=4$
$13.68 \pm 0.10$	$10.99 \pm 0.29$	$10.85 \pm 0.11$	$10.73 \pm 0.19$	$11.20 \pm 0.15$	<b>9.51</b> $\pm 0.07$

results are found in Table 1. We observe that our proposed training method achieves highly competitive performance for all values of  $K$ . For NCE, the performance varies quite significantly with  $\sigma_1$ , which would have to be tuned for each dataset.

**Image-Based Regression** We employ a virtually identical network architecture for all four image-based regression tasks, only making minor modifications for the head-pose estimation task to accommodate the higher target dimension  $y \in \mathbb{R}^3$ . The EBM DNN  $f_\theta(x, y)$  is composed of a ResNet18 [66] that extracts features  $h_x \in \mathbb{R}^{512}$  from the input image  $x$ . From the target  $y$ , fully-connected layers extract features  $h_y \in \mathbb{R}^{128}$ . After concatenation of  $h_x$  and  $h_y$ , fully-connected layers then output  $f_\theta(x, y) \in \mathbb{R}$ . The MDN network head  $g_\phi(x)$  takes the image features  $h_x \in \mathbb{R}^{512}$  as input and outputs  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$ . Again, we use ADAM to jointly train  $f_\theta$  and  $g_\phi$ . We evaluate the training methods by approximately computing the test set NLL of the EBM  $p(y|x; \theta)$ . We compare our proposed approach with training the EBM using NCE, employing the noise distribution  $q(y) = \frac{1}{2} \sum_{k=1}^2 \mathcal{N}(y; y_i, \sigma_k^2 I)$ . For each of the four tasks, we initially set  $\{\sigma_1, \sigma_2\}$  to what was used for age estimation and head-pose estimation in [19] and then carefully tune them further. Based on the 1D regression results in Table 1, we use  $K = 4$  components in the MDN proposal for our proposed approach. We train 20 networks for each setting and dataset, and report the mean of the 5 best runs. The results are found in Table 2 to Table 5. We observe that our proposed training method achieves highly competitive performance. In particular, our method significantly outperforms the NCE baseline on the more challenging head-pose estimation task (Table 5), which has a multi-dimensional target space. Note that we use an identical architecture for the MDN proposal in our training method across all four tasks, while the task-dependent NCE hyperparameters  $\{\sigma_1, \sigma_2\}$  are tuned directly on each of the corresponding *test* sets. Thus, NCE is here a very strong baseline.

## 5.2 MDN Experiments

Lastly, we perform experiments on the four image-based regression tasks to evaluate our proposed approach for training MDNs  $q(y|x; \phi)$ . For the EBM DNN  $f_\theta(x, y)$ , an identical network architecture is used as in the EBM

**Table 6:** Results in terms of NLL for the MDN experiments on four image-based regression tasks.

Task	NLL			<b>Ours</b>		
	$K=4$	$K=8$	$K=16$	$K=4$	$K=8$	$K=16$
Steering angle	1.45±0.13	1.25±0.05	-	<b>1.00</b> ±0.03	1.01±0.04	-
Cell-count	2.80±0.09	2.90±0.06	-	2.80±0.06	<b>2.75</b> ±0.06	-
Age	4.88±0.21	4.71±0.35	-	<b>3.57</b> ±0.28	3.65±0.18	-
Head-pose	11.02±0.16	10.68±0.39	10.71±0.17	<b>8.69</b> ±0.10	8.79±0.06	8.77±0.09

experiments (Section 5.1). The MDN network  $g_\phi(x)$  is now a full DNN. It consists of a ResNet18 that extracts image features  $h_x \in \mathbb{R}^{512}$ , and a head of fully-connected layers that takes  $h_x \in \mathbb{R}^{512}$  as input and outputs  $\{\pi_\phi^{(k)}, \mu_\phi^{(k)}, \sigma_\phi^{(k)}\}_{k=1}^K$ . As described in Section 3.3, the MDN DNN  $g_\phi$  is trained by minimizing the loss  $J_{\text{MDN}}(\phi)$  in (8), whereas  $f_\theta$  is trained via  $J_{\text{NCE}}(\theta)$  in (6). As in the previous experiments, ADAM is used to jointly train  $f_\theta$  and  $g_\phi$ . We compare our proposed approach with the conventional MDN training method, i.e. minimizing the NLL  $\sum_{i=1}^N -\log q(y_i|x_i; \phi)$ . We evaluate the training methods in terms of test set NLL, for MDNs with  $K \in \{4, 8, 16\}$  components. We train 20 networks for each setting and dataset, and report the mean of the 5 best runs. The results are found in Table 6. We observe that our proposed training method consistently outperforms the baseline of pure NLL training. For the steering angle prediction and age estimation tasks, our approach achieves substantial improvements. Moreover, in the particularly challenging head-pose estimation task, our approach outperforms the standard MDN by a significant margin.

## 6 Conclusion

We derived an efficient and convenient objective that can be employed to train a parameterized distribution  $q(y|x; \phi)$  by minimizing its KL divergence to a conditional EBM  $p(y|x; \theta)$ . We then applied the derived objective to jointly learn an effective MDN proposal distribution during EBM training, thus addressing the main practical limitations of energy-based regression. We evaluated our proposed EBM training method on illustrative 1D regression problems and real-world regression tasks within computer vision, achieving highly competitive performance without having to tune any task-dependent hyperparameters. Lastly, we employed the derived objective to improve training of stand-alone MDNs, consistently obtaining more accurate predictive distributions compared to conventional MDN training. Future directions include estimating the EBM uncertainty via test-time use of the trained MDN proposal, and applying our MDN training approach to additional tasks.

**Acknowledgments** This research was supported by the Swedish Foundation for Strategic Research via the project *ASSEMBLE* (contract number: RIT15-0012), by the Swedish Research Council via the project *NewLEADS - New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079), and by the *Kjell & Märta Beijer Foundation*.

## References

- [1] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).
- [2] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. “Energy-based models for sparse overcomplete representations.” In: *Journal of Machine Learning Research* (2003).
- [3] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. “A neural probabilistic language model.” In: *Journal of machine learning research* (2003).
- [4] A. Mnih and G. Hinton. “Learning nonlinear constraints with contrastive backpropagation.” In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE. 2005.
- [5] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. “Unsupervised discovery of nonlinear structure using contrastive backpropagation.” In: *Cognitive science* (2006).
- [6] M. Osadchy, M. L. Miller, and Y. L. Cun. “Synergistic face detection and pose estimation with energy-based models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.
- [7] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. “A theory of generative convnet.” In: *International Conference on Machine Learning (ICML)*. 2016.
- [8] J. Xie, S.-C. Zhu, and Y. Nian Wu. “Synthesizing dynamic patterns by spatial-temporal generative convnet.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [9] Y. Du, S. Li, J. Tenenbaum, and I. Mordatch. “Improved contrastive divergence training of energy based models.” In: *International Conference on Machine Learning (ICML)*. 2021.
- [10] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. “Learning generative convnets via multi-grid modeling and sampling.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [11] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu. “Learning descriptor networks for 3d shape synthesis and analysis.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

- [12] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. “Learning non-convergent non-persistent short-run MCMC toward energy-based model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [13] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [14] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. “Your classifier is secretly an energy based model and you should treat it like one.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [15] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. “Flow contrastive estimation of energy-based models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [16] B. Pang, T. Han, E. Nijkamp, S.-C. Zhu, and Y. N. Wu. “Learning latent space energy-based prior model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [17] F. Bao, C. Li, K. Xu, H. Su, J. Zhu, and B. Zhang. “Bi-level Score Matching for Learning Energy-based Latent Variable Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [18] M. Danelljan, L. Van Gool, and R. Timofte. “Probabilistic Regression for Visual Tracking.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [19] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [20] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. “How to Train Your Energy-Based Model for Regression.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2020.
- [21] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Accurate 3D Object Detection using Energy-Based Models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2021.
- [22] J. N. Hendriks, F. K. Gustafsson, A. H. Ribeiro, A. G. Wills, and T. B. Schön. “Deep Energy-Based NARX Models.” In: *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*. 2021.
- [23] K. Murphy, C. Esteves, V. Jampani, S. Ramalingam, and A. Makadia. “Implicit-PDF: Non-Parametric Representation of Probability Distributions on the Rotation Manifold.” In: *International Conference on Machine Learning (ICML)*. 2021.

- [24] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. “Implicit Behavioral Cloning.” In: *The 5th Annual Conference on Robot Learning (CoRL)*. 2021.
- [25] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [26] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [27] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [28] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [29] E. Ilg, O. Cicek, S. Galessio, A. Klein, O. Makansi, F. Hutter, and T. Brox. “Uncertainty estimates and multi-hypotheses networks for optical flow.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [30] O. Makansi, E. Ilg, O. Cicek, and T. Brox. “Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [31] A. Varamesh and T. Tuytelaars. “Mixture Dense Regression for Object Detection and Human Pose Estimation.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [32] C. M. Bishop. *Mixture density networks*. 1994.
- [33] Y. Song and D. P. Kingma. “How to Train Your Energy-Based Models.” In: *arXiv preprint arXiv:2101.03288* (2021).
- [34] C. Li and G. H. Lee. “Generating multiple hypotheses for 3d human pose estimation with mixture density network.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [35] L. U. Hjorth and I. T. Nabney. “Regularisation of mixture density networks.” In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*. IET. 1999.

- [36] C. Rupprecht, I. Laina, R. DiPietro, M. Baust, F. Tombari, N. Navab, and G. D. Hager. “Learning in an uncertain world: Representing ambiguity through multiple hypotheses.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
- [37] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric. “Multimodal trajectory predictions for autonomous driving using deep convolutional networks.” In: *International Conference on Robotics and Automation (ICRA)*. IEEE. 2019.
- [38] Y. Zhou, J. Gao, and T. Asfour. “Movement primitive learning and generalization: Using mixture density networks.” In: *IEEE Robotics & Automation Magazine* (2020).
- [39] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010.
- [40] Z. Ma and M. Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [41] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the limits of language modeling.” In: *arXiv preprint arXiv:1602.02410* (2016).
- [42] A. B. Owen. *Monte Carlo theory, methods and examples*. 2013.
- [43] D. B. Rubin. “The calculation of posterior distributions by data augmentation: Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The SIR algorithm.” In: *Journal of the American Statistical Association* (1987).
- [44] T. Kim and Y. Bengio. “Deep directed generative models with energy-based probability estimation.” In: *arXiv preprint arXiv:1606.03439* (2016).
- [45] S. Zhai, Y. Cheng, R. Feris, and Z. Zhang. “Generative adversarial networks as variational training of energy based models.” In: *arXiv preprint arXiv:1611.01799* (2016).
- [46] R. Kumar, S. Ozair, A. Goyal, A. Courville, and Y. Bengio. “Maximum entropy generators for energy-based models.” In: *arXiv preprint arXiv:1901.08508* (2019).
- [47] W. S. Grathwohl, J. J. Kelly, M. Hashemi, M. Norouzi, K. Swersky, and D. Duvenaud. “No MCMC for me: Amortized sampling for fast and stable training of energy-based models.” In: *International Conference on Learning Representations (ICLR)*. 2021.

- [48] J. Xie, Y. Lu, R. Gao, and Y. N. Wu. “Cooperative learning of energy-based model and latent variable model via MCMC teaching.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.
- [49] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. “Cooperative training of descriptor and generator networks.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2018).
- [50] J. Xie, Z. Zheng, and P. Li. “Learning energy-based model with variational auto-encoder as amortized sampler.” In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*. 2021.
- [51] J. Xie, Z. Zheng, X. Fang, S.-C. Zhu, and Y. N. Wu. “Learning Cycle-Consistent Cooperative Networks via Alternating MCMC Teaching for Unsupervised Cross-Domain Translation.” In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*. 2021.
- [52] J. Xie, Z. Zheng, X. Fang, S.-C. Zhu, and Y. N. Wu. “Cooperative training of fast thinking initializer and slow thinking solver for conditional learning.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2021).
- [53] J. Zhang, J. Xie, Z. Zheng, and N. Barnes. “Energy-Based Generative Cooperative Saliency Prediction.” In: *arXiv preprint arXiv:2106.13389* (2021).
- [54] G. Hinton, O. Vinyals, and J. Dean. “Distilling the knowledge in a neural network.” In: *arXiv preprint arXiv:1503.02531* (2015).
- [55] S. I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh. “Improved knowledge distillation via teacher assistant.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.
- [56] G. Xu, Z. Liu, X. Li, and C. C. Loy. “Knowledge distillation meets self-supervision.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [57] X. Ding, Y. Wang, Z. Xu, Z. J. Wang, and W. J. Welch. “Distilling and Transferring Knowledge via cGAN-generated Samples for Image Classification and Regression.” In: *arXiv preprint arXiv:2104.03164* (2021).
- [58] A. Brando, J. A. Rodríguez-Serrano, J. Vitria, and A. Rubio. “Modelling heterogeneous distributions with an Uncountable Mixture of Asymmetric Laplacians.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

- [60] X. Ding, Y. Wang, Z. Xu, W. J. Welch, and Z. J. Wang. “CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation.” In: *International Conference on Learning Representations (ICLR)*. 2021.
- [61] X. Ding, Y. Wang, Z. Xu, W. J. Welch, and Z. J. Wang. “Continuous Conditional Generative Adversarial Networks for Image Generation: Novel Losses and Label Input Mechanisms.” In: *arXiv preprint arXiv:2011.07466* (2020).
- [62] Z. Zhang, Y. Song, and H. Qi. “Age progression/regression by conditional adversarial autoencoder.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [63] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. “Random forests for real time 3d face analysis.” In: *International Journal of Computer Vision (IJCV)* (2013).
- [64] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. “FSA-Net: Learning Fine-Grained Structure Aggregation for Head Pose Estimation from a Single Image.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [65] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [66] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

## Supplementary Material

In this supplementary material, we provide additional details and results. It consists of Appendix A - Appendix G. After discussing limitations and societal impacts in Appendix A, we provide implementation details in Appendix B. Then, we describe all utilized datasets more closely in Appendix C. We then provide a complete derivation of Result 1 in Appendix D. Additional results for the 1D regression task is then provided in Appendix E. Lastly, Appendix F and Appendix G contain example model and training code. Note that figures in this supplementary material are numbered with the prefix “S”. Numbers without this prefix refer to the main paper.

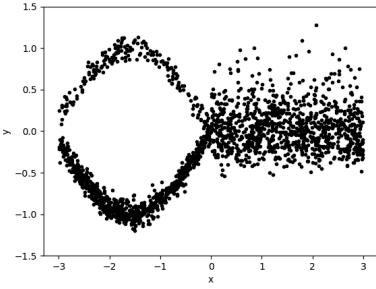
## A Limitations & Societal Impacts

Our approach is primarily intended for regression tasks, where the target space has a limited number of dimensions. For each training sample, several target values are sampled from the proposal distribution. Our approach is therefore not intended to scale to very high-dimensional generative modeling tasks, such as image generation.

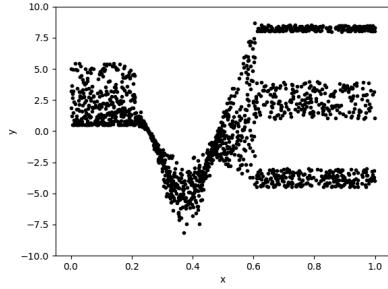
Training an EBM using our proposed method in Section 3.2 is somewhat slower than using the NCE baseline method, since we also have to update an MDN proposal at each iteration. The NCE baseline however requires hyperparameters to be tuned specifically for each task at hand. The *total* environmental impact due to training is therefore likely smaller for our proposed method. Our proposed approach for training MDNs in Section 3.3 does however not offer similar benefits compared to conventional MDN training, and is twice as slow to train. This issue would be mitigated to a certain extent by sharing parts of the network among the EBM and MDN, which could be explored in future work.

## B Implementation Details

We train all networks for 75 epochs with a batch size of 32. The number of samples  $M$  is always set to  $M = 1024$ . All networks are trained on individual NVIDIA TITAN Xp GPUs. Training 20 networks for a specific setting and dataset on one such GPU takes at most 24 – 48 hours. Producing the results in Table 1 to Table 6 thus required approximately 50 GPU days of training. We utilized an internal GPU cluster.



**Figure S1:** Training data  $\{(x_i, y_i)\}_{i=1}^{2000}$  for the first 1D regression dataset [20].



**Figure S2:** Training data  $\{(x_i, y_i)\}_{i=1}^{1700}$  for the second 1D regression dataset [58].

PyTorch code defining the network architecture used for the head-pose estimation task in Section 5.1 is found in Appendix F below. PyTorch code for the corresponding main training loop is found in Appendix G.

In Section 5.1, the EBM  $p(y|x; \theta) = e^{f_\theta(x, y)} / \int e^{f_\theta(x, \tilde{y})} d\tilde{y}$  is evaluated by approximately computing its test set negative log-likelihood (NLL). We do so by evaluating  $f_\theta(x, y)$  at densely sampled  $y$  values in an interval  $[y_{\min}, y_{\max}]$ . For the second 1D regression dataset, we evaluate at 8 192 values in  $[-12.5, 12.5]$ . For steering angle prediction, 20 000 values in  $[-100, 100]$ . For cell-count prediction, 19 900 values in  $[1, 200]$ . For age estimation, 5 900 values in  $[1, 60]$ . For head-pose estimation, 27 000 values in  $\{x \in \mathbb{R}^3 : x_i \in [-80, 80], i = 1, 2, 3\}$ .

## C Dataset Details

The training data for the two 1D regression problems is visualized in Figure S1 and Figure S2.

For steering angle prediction, cell-count prediction and age estimation, our utilized datasets from [60, 61] are all available at [https://github.com/UBCDingXin/improved\\_CcGAN](https://github.com/UBCDingXin/improved_CcGAN).

The original age estimation dataset UTKFace [62] is available at <https://susanzq.github.io/UTKFace/>, for non-commercial research purposes only. The dataset consists of images collected from the internet, i.e. images collected without explicitly obtained consent to be used specifically for training age estimation models. Thus, we choose to not display any dataset examples.

For head-pose estimation, the BIWI [63] dataset is available for research purposes only. The dataset was created by recording 20 people (research subjects)

while they freely turned their heads around. The processed version provided by [64] that we utilize is available at <https://github.com/shamangary/FSA-Net>. Since the dataset images could potentially contain personally identifiable information, we choose to not display any dataset examples.

## D Derivation of Result 1

To derive Result 1 in Section 3.1 of the main paper, we first rewrite the KL divergence  $\nabla_\phi D_{\text{KL}}(p(y|x; \theta) \| q(y|x; \phi))$  according to,

$$\begin{aligned}
 \nabla_\phi D_{\text{KL}}(p(y|x; \theta) \| q(y|x; \phi)) &= \nabla_\phi \int p(y|x; \theta) \log \frac{p(y|x; \theta)}{q(y|x; \phi)} dy \\
 &= \int p(y|x; \theta) \nabla_\phi \log \frac{p(y|x; \theta)}{q(y|x; \phi)} dy \\
 &= \int p(y|x; \theta) \nabla_\phi \left( \log p(y|x; \theta) - \log q(y|x; \phi) \right) dy \\
 &= - \int p(y|x; \theta) \nabla_\phi \log q(y|x; \phi) dy \\
 &= - \int p(y|x; \theta) \frac{1}{q(y|x; \phi)} \nabla_\phi q(y|x; \phi) dy \\
 &= - \int \frac{e^{f_\theta(x,y)}}{\int e^{f_\theta(x,\tilde{y})} d\tilde{y}} \frac{1}{q(y|x; \phi)} \nabla_\phi q(y|x; \phi) dy \\
 &= - \frac{1}{\int e^{f_\theta(x,\tilde{y})} d\tilde{y}} \int e^{f_\theta(x,y)} \frac{1}{q(y|x; \phi)} \nabla_\phi q(y|x; \phi) dy.
 \end{aligned}$$

Then, we approximate the two integrals using Monte Carlo importance sampling,

$$\begin{aligned}
 &- \frac{1}{\int e^{f_\theta(x,\tilde{y})} d\tilde{y}} \int e^{f_\theta(x,y)} \frac{1}{q(y|x; \phi)} \nabla_\phi q(y|x; \phi) dy \\
 &= - \frac{1}{\int e^{f_\theta(x,y)} dy} \int \frac{e^{f_\theta(x,y)}}{q(y|x; \phi)^2} (\nabla_\phi q(y|x; \phi)) q(y|x; \phi) dy \\
 &= - \frac{1}{\int \frac{e^{f_\theta(x,y)}}{q(y|x; \phi)} q(y|x; \phi) dy} \int \frac{e^{f_\theta(x,y)}}{q(y|x; \phi)^2} (\nabla_\phi q(y|x; \phi)) q(y|x; \phi) dy \\
 &\approx - \frac{1}{\frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x,y^{(m)})}}{q(y^{(m)}|x; \phi)}} \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x,y^{(m)})}}{q(y^{(m)}|x; \phi)^2} \nabla_\phi q(y^{(m)}|x; \phi) \right),
 \end{aligned}$$

where  $\{y^{(m)}\}_{m=1}^M$  are  $M$  independent samples drawn from  $q(y|x; \phi)$ . Finally, we further rewrite the resulting expression according to,

$$\begin{aligned}
& -\frac{1}{\frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)}} \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)^2} \nabla_\phi q(y^{(m)}|x; \phi) \right) \\
& = \frac{1}{\frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)}} \left( \frac{1}{M} \sum_{m=1}^M e^{f_\theta(x, y^{(m)})} \nabla_\phi \frac{1}{q(y^{(m)}|x; \phi)} \right) \\
& = \frac{1}{\frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)}} \left( \frac{1}{M} \sum_{m=1}^M \nabla_\phi \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)} \right) \\
& = \frac{1}{\frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)}} \nabla_\phi \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)} \right) \\
& = \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)} \right)^{-1} \nabla_\phi \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)} \right) \\
& = \nabla_\phi \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x, y^{(m)})}}{q(y^{(m)}|x; \phi)} \right).
\end{aligned}$$

## D.1 Best Possible Proposal

We here expand on the footnote on page 5 of the main paper. When training the EBM  $p(y|x; \theta) = e^{f_\theta(x, y)} / Z(x, \theta)$  by minimizing the approximated NLL in (2), we wish to use the proposal  $q(y|x; \phi)$  that yields the best possible NLL approximation. In general, this is achieved when the proposal equals the EBM,

i.e. when  $q(y|x; \phi) = p(y|x; \theta)$ . To see why this is true, we set  $q = p$  in (2),

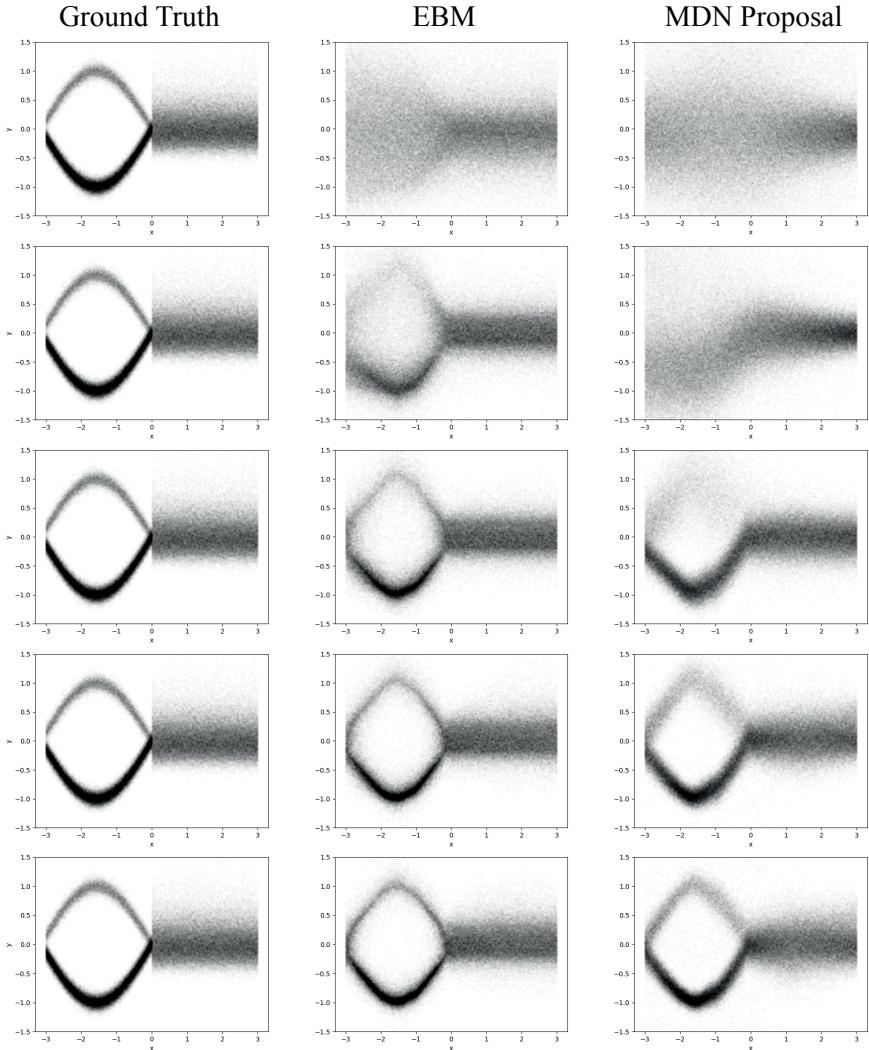
$$\begin{aligned}
J(\theta) &= \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{q(y_i^{(m)})} \right) - f_\theta(x_i, y_i) \\
&= \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{p(y_i^{(m)}|x_i; \theta)} \right) - f_\theta(x_i, y_i) \\
&= \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M \frac{e^{f_\theta(x_i, y_i^{(m)})}}{e^{f_\theta(x_i, y_i^{(m)})}/Z(x_i, \theta)} \right) - f_\theta(x_i, y_i) \\
&= \frac{1}{N} \sum_{i=1}^N \log \left( \frac{1}{M} \sum_{m=1}^M Z(x_i, \theta) \right) - f_\theta(x_i, y_i) \\
&= \frac{1}{N} \sum_{i=1}^N \log Z(x_i, \theta) - f_\theta(x_i, y_i) \\
&= \frac{1}{N} \sum_{i=1}^N -\log \left( \frac{e^{f_\theta(x_i, y_i)}}{Z(x_i, \theta)} \right) \\
&= \frac{1}{N} \sum_{i=1}^N -\log p(y_i|x_i; \theta),
\end{aligned}$$

which corresponds to the exact NLL objective.

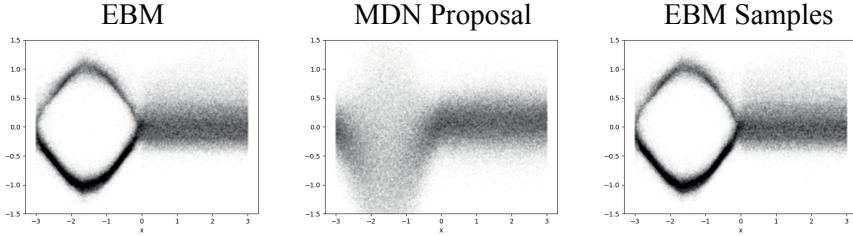
## E Additional Results

Figure 2 in the main paper visualizes the fully trained EBM and MDN proposal, i.e. after 75 epochs of training. In Figure S3, we instead visualize the EBM and MDN after 5 (top row), 10, 15, 20 and 25 (bottom row) epochs of training. We observe that the EBM is closer to the ground truth early on during training, guiding the MDN via the  $J_{KL}(\phi)$  loss in (5).

In Figure S4, we visualize the fully trained EBM and MDN proposal when instead using just  $K = 1$  component in the MDN. We observe that the EBM still is close to the ground truth. Apart from visualizing the EBM using the technique from [20] (evaluating  $f_\theta(x, y)$  at densely sampled  $y$  values in the interval  $[-3, 3]$  for each  $x$ ), we here also demonstrate that we can draw approximate samples from the EBM using the method described in Section 3.2.2. For each  $x$ , we draw samples  $\{y_i^{(m)}\}_{m=1}^{1024} \sim q(y|x; \phi)$  from the proposal, compute weights  $\{w_i^{(m)}\}_{m=1}^{1024}$  according to (7), and then re-sample one value from this set  $\{y_i^{(m)}\}_{m=1}^{1024}$  (drawing each  $y_i^{(m)}$  with probability  $w_i^{(m)}$ ). We observe in Figure S4 that this method produces accurate EBM samples, even when the



**Figure S3:** An illustrative 1D regression problem [20], demonstrating the effectiveness of our proposed method to jointly train an EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$ . In this example, the MDN has  $K = 4$  components. The EBM is trained using NCE with  $q(y|x; \phi)$  acting as the noise distribution, whereas the MDN is trained by minimizing its KL divergence to  $p(y|x; \theta)$ . The EBM and MDN are here visualized after 5 (top row), 10, 15, 20 and 25 (bottom row) epochs of training.



**Figure S4:** An illustrative 1D regression problem [20], demonstrating the effectiveness of our proposed method to jointly train an EBM  $p(y|x; \theta)$  and MDN proposal  $q(y|x; \phi)$ . In this example, the MDN has  $K = 1$  component. We here also demonstrate that we can draw approximate samples from the EBM using the method described in Section 3.2.2.

proposal is unimodal and thus not a particularly close approximation of the EBM.

## F PyTorch Code - Network Architecture

```

class NoiseNet(nn.Module):
    def __init__(self, hidden_dim):
        super().__init__()

        self.K = 4

        self.fc1_mean = nn.Linear(hidden_dim, hidden_dim)
        self.fc2_mean = nn.Linear(hidden_dim, 3*self.K)

        self.fc1_sigma = nn.Linear(hidden_dim, hidden_dim)
        self.fc2_sigma = nn.Linear(hidden_dim, 3*self.K)

        self.fc1_weight = nn.Linear(hidden_dim, hidden_dim)
        self.fc2_weight = nn.Linear(hidden_dim, self.K)

    def forward(self, x_feature):
        means = F.relu(self.fc1_mean(x_feature))
        means = self.fc2_mean(means)

        log_sigma2s = F.relu(self.fc1_sigma(x_feature))
        log_sigma2s = self.fc2_sigma(log_sigma2s)

        weight_logits = F.relu(self.fc1_weight(x_feature))
        weight_logits = self.fc2_weight(weight_logits)
        weights = torch.softmax(weight_logits, dim=1)

        return means, log_sigma2s, weights

class PredictorNet(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super().__init__()

        self.fc1_y = nn.Linear(input_dim, 16)
        self.fc2_y = nn.Linear(16, 32)
        self.fc3_y = nn.Linear(32, 64)
        self.fc4_y = nn.Linear(64, 128)

        self.fc1_xy = nn.Linear(hidden_dim+128, hidden_dim)

```

```

        self.fc2_xy = nn.Linear(hidden_dim, 1)

    def forward(self, x_feature, y):
        batch_size, num_samples, _ = y.shape

        x_feature = x_feature.view(batch_size, 1, -1).expand(-1, num_samples, -1)
        x_feature = x_feature.reshape(batch_size*num_samples, -1)

        y = y.reshape(batch_size*num_samples, -1)

        y_feature = F.relu(self.fc1_y(y))
        y_feature = F.relu(self.fc2_y(y_feature))
        y_feature = F.relu(self.fc3_y(y_feature))
        y_feature = F.relu(self.fc4_y(y_feature))

        xy_feature = torch.cat([x_feature, y_feature], 1)

        xy_feature = F.relu(self.fc1_xy(xy_feature))
        score = self.fc2_xy(xy_feature)

        score = score.view(batch_size, num_samples)

    return score

class FeatureNet(nn.Module):
    def __init__(self):
        super().__init__()

        resnet18 = models.resnet18(pretrained=True)
        self.resnet18 = nn.Sequential(*list(resnet18.children())[:-2])

        self.avg_pool = nn.AdaptiveAvgPool2d((1, 1))

    def forward(self, x):
        x_feature = self.resnet18(x)
        x_feature = self.avg_pool(x_feature)
        x_feature = x_feature.squeeze(2).squeeze(2)

    return x_feature

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        hidden_dim = 512

        self.feature_net = FeatureNet()
        self.noise_net = NoiseNet(hidden_dim)
        self.predictor_net = PredictorNet(3, hidden_dim)

    def forward(self, x, y):
        x_feature = self.feature_net(x)
        return self.noise_net(x_feature)

```

## G PyTorch Code - Training Loop

```

for step, (xs, ys) in enumerate(train_loader):
    xs = xs.cuda() # (shape: (batch_size, 3, img_size, img_size))
    ys = ys.cuda() # (shape: (batch_size, 3))

    x_features = network.feature_net(xs) # (shape: (batch_size, hidden_dim))

    means, log_sigma2s, weights = network.noise_net(x_features.detach())
    # (means has shape: (batch_size, 3K))
    # (log_sigma2s has shape: (batch_size, 3K))
    # (weights has shape: (batch_size, K))

```

```

sigmas = torch.exp(log_sigma2s/2.0) # (shape: (batch_size, 3K))
means = means.view(-1, 3, K) # (shape: (batch_size, 3, K))
sigmas = sigmas.view(-1, 3, K) # (shape: (batch_size, 3, K))

q_distr = torch.distributions.normal.Normal(loc=means, scale=sigmas)
q_ys_K = torch.exp(q_distr.log_prob(ys.unsqueeze(2)).sum(1)) # (shape: (batch_size, K)
q_ys = torch.sum(weights*q_ys_K, dim=1) # (shape: (batch_size))

y_samples_K = q_distr.sample(sample_shape=torch.Size([num_samples]))
# (shape: (num_samples, batch_size, 3, K))
inds = torch.multinomial(weights, num_samples=num_samples,
                        replacement=True).unsqueeze(2).unsqueeze(2)
# (shape: (batch_size, num_samples, 1, 1))
inds = inds.expand(-1, -1, 3, 1) # (shape: (batch_size, num_samples, 3, 1))
inds = torch.transpose(inds, 1, 0) # (shape: (num_samples, batch_size, 3, 1))
y_samples = y_samples_K.gather(3, inds).squeeze(3) # (shape: (num_samples, batch_size, 3))
y_samples = y_samples.detach()
q_y_samples_K = torch.exp(q_distr.log_prob(y_samples.unsqueeze(3)).sum(2))
# (shape: (num_samples, batch_size, K))
q_y_samples = torch.sum(weights.unsqueeze(0)*q_y_samples_K, dim=2)
# (shape: (num_samples, batch_size))
y_samples = torch.transpose(y_samples, 1, 0) # (shape: (batch_size, num_samples, 3))
q_y_samples = torch.transpose(q_y_samples, 1, 0) # (shape: (batch_size, num_samples))

scores_gt = network.predictor_net(x_features, ys.unsqueeze(1)) # (shape: (batch_size, 1))
scores_gt = scores_gt.squeeze(1) # (shape: (batch_size))

scores_samples = network.predictor_net(x_features, y_samples)
# (shape: (batch_size, num_samples))

#####
# compute loss:
#####
f_samples = scores_samples
p_N_samples = q_y_samples.detach()
f_0 = scores_gt
p_N_0 = q_ys.detach()
exp_vals_0 = f_0-torch.log(p_N_0)
exp_vals_samples = f_samples-torch.log(p_N_samples)
exp_vals = torch.cat([exp_vals_0.unsqueeze(1), exp_vals_samples], dim=1)
loss_ebm_nce = -torch.mean(exp_vals_0 - torch.logsumexp(exp_vals, dim=1))

log_Z = torch.logsumexp(scores_samples.detach()
                        - torch.log(q_y_samples), dim=1) - math.log(num_samples)
loss_mdn_kl = torch.mean(log_Z)

loss = loss_ebm_nce + loss_mdn_kl

optimizer.zero_grad()
loss.backward()
optimizer.step()

```

# Paper IV

## Title

Accurate 3D Object Detection using Energy-Based Models

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Accurate 3D Object Detection using Energy-Based Models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2021



# Accurate 3D Object Detection using Energy-Based Models

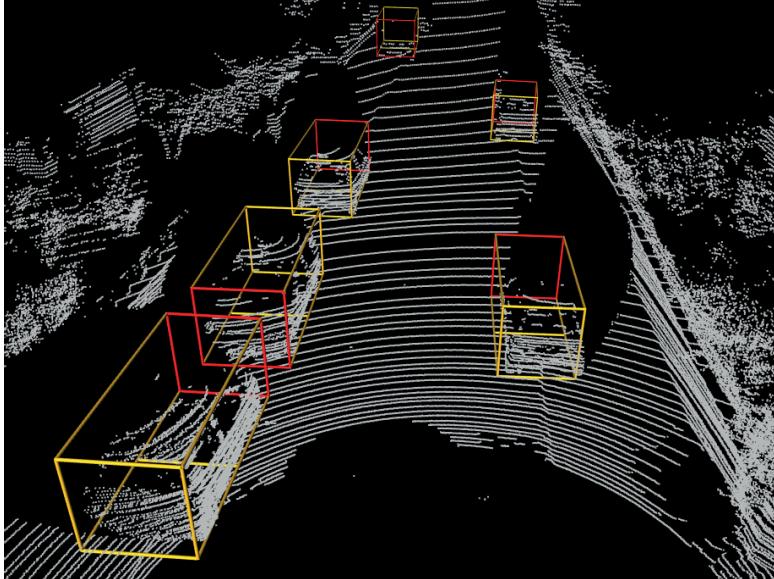
## Abstract

Accurate 3D object detection (3DOD) is crucial for safe navigation of complex environments by autonomous robots. Regressing accurate 3D bounding boxes in cluttered environments based on sparse LiDAR data is however a highly challenging problem. We address this task by exploring recent advances in conditional energy-based models (EBMs) for probabilistic regression. While methods employing EBMs for regression have demonstrated impressive performance on 2D object detection in images, these techniques are not directly applicable to 3D bounding boxes. In this work, we therefore design a differentiable pooling operator for 3D bounding boxes, serving as the core module of our EBM network. We further integrate this general approach into the state-of-the-art 3D object detector SA-SSD. On the KITTI dataset, our proposed approach consistently outperforms the SA-SSD baseline across all 3DOD metrics, demonstrating the potential of EBM-based regression for highly accurate 3DOD. Code is available at [https://github.com/fregu856/ebms\\_3dod](https://github.com/fregu856/ebms_3dod).

## 1 Introduction

3D object detection (3DOD) is a key perception task for self-driving vehicles and other autonomous robots. 3DOD entails detecting various objects from sensor data, and estimating their size and position in the 3D world. Specifically, the goal of 3DOD is to place oriented 3D bounding boxes which tightly contain all surrounding objects of interest. See Figure 1 for an example. These 3D bounding boxes then serve as input to important high-level tasks such as planning and collision avoidance. Accurate 3DOD is thus crucial for safe autonomous navigation of different complex environments.

In the automotive domain, 3DOD is usually performed from LiDAR point clouds [1, 2, 3], images captured by vehicle-mounted cameras [4, 5, 6], or

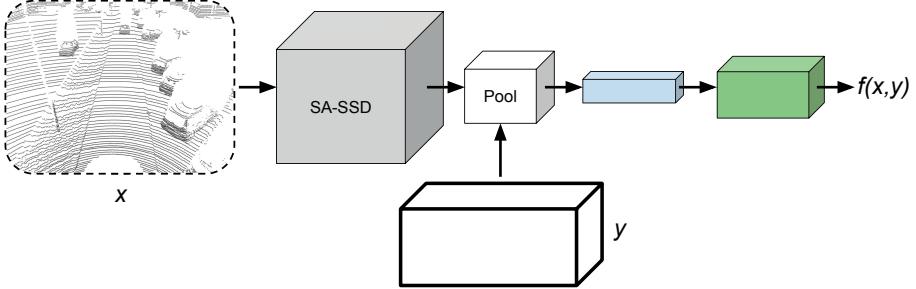


**Figure 1:** We study how energy-based models (EBMs) can be applied to accurately regress 3D bounding boxes in 3DOD from LiDAR point clouds. Here, we visualize the output of our detector on a validation example from the KITTI [18] dataset.

from a combination of both data modalities [7, 8, 9]. Radar sensors are sometimes also utilized [10, 11, 12]. State-of-the-art 3D object detectors employ deep neural networks (DNNs) to learn powerful feature representations directly from this data [3, 13, 14]. The 3DOD task is then commonly divided into two sub-tasks, in which anchor or proposal 3D bounding boxes are classified as either background or a specific class of object, and then regressed toward ground truth boxes [15, 16, 17].

In general, regression entails predicting a continuous target  $y$  from an input  $x$ . This is a fundamental machine learning problem that can be addressed using a variety of different techniques [20, 21, 22, 23, 24]. Specifically in 3DOD, the 3D bounding box regression problem is usually addressed by letting a DNN directly predict a target bounding box  $y$  for a given input  $x$ , and training the DNN by minimizing the  $L^2$  or Huber loss [25, 15, 1, 3, 19]. Alternatively, a probabilistic regression approach has also been employed. The conditional target density  $p(y|x)$ , i.e. the distribution for the target 3D bounding box  $y$  given the input  $x$ , is then explicitly modelled using a DNN, which is trained by minimizing the associated negative log-likelihood. Previous work on 3DOD has mainly explored Gaussian models of  $p(y|x)$  [26, 27, 28, 29].

A Gaussian model is however fairly restrictive, limiting  $p(y|x)$  to unimodal and symmetric distributions. Instead, recent work [30, 31, 32] has demonstrated that improved regression accuracy can be obtained on various tasks by em-



**Figure 2:** An overview of our proposed approach, applying EBM-based regression to the task of 3D object detection. We integrate a conditional EBM  $p(y|x; \theta) = e^{f_\theta(x,y)} / \int e^{f_\theta(x,\tilde{y})} d\tilde{y}$  into the state-of-the-art 3D object detector SA-SSD [19]. We achieve this by designing a differentiable pooling operator that, given a 3D bounding box  $y$ , extracts a feature vector from the SA-SSD output. This feature vector is then processed by three fully-connected layers, outputting the scalar energy  $f_\theta(x, y) \in \mathbb{R}$ .

ploying energy-based models (EBMs) [33] to represent the conditional target density  $p(y|x)$ . Specifically, this approach entails modeling  $p(y|x)$  with the conditional EBM  $p(y|x; \theta) = e^{f_\theta(x,y)} / \int e^{f_\theta(x,\tilde{y})} d\tilde{y}$ , and then using gradient ascent to maximize  $p(y|x; \theta)$  w.r.t.  $y$  at test-time. Since the EBM  $p(y|x; \theta)$  is directly specified via the scalar function  $f_\theta(x, y)$ , which is defined using a DNN, it is a highly expressive model that puts minimal restricting assumptions on  $p(y|x)$ . Even potential multi-modality in the distribution  $p(y|x)$  can therefore be learned directly from data. This EBM-based regression approach is thus an attractive alternative also for 3D bounding box regression, especially considering the impressive performance demonstrated on conventional 2D bounding box regression in images [30, 31, 32].

Extending the approach from 2D to 3D is however challenging. In particular, using gradient ascent to maximize the EBM  $p(y|x; \theta)$  at test-time requires the scalar DNN output  $f_\theta(x, y)$  to be differentiable w.r.t. the bounding box  $y$ . For 2D bounding boxes in images, this was achieved by applying a differentiable pooling operator [34] on image features [30, 31, 32], but this technique is not directly applicable to 3D bounding boxes. How EBM-based regression should be applied to 3DOD is thus currently an open question, which we set out to investigate in this work.

**Contributions** We apply conditional EBMs  $p(y|x; \theta)$  to the task of 3D bounding box regression, extending the recent EBM-based regression approach [30, 31, 32] from 2D to 3D object detection. This is achieved by adding an extra network branch to the state-of-the-art 3D object detector SA-SSD [19], and designing a differentiable pooling operator for 3D bounding boxes  $y$ . We evaluate our proposed detector on the KITTI [18] dataset and consistently outperform

the SA-SSD baseline detector across all 3DOD metrics. Our work thus demonstrates the potential of EBM-based regression for highly accurate 3DOD.

## 2 Energy-Based Models for Regression

EBMs were extensively studied by the machine learning community in the past [33, 35, 36, 37, 38, 39]. In recent years they have also had a resurgence within the field of computer vision, frequently being employed for generative image modeling [40, 41, 42, 43, 44, 45, 46, 47]. In comparison, the application of EBMs to regression problems has not been a particularly well-studied topic. Very recent work [30, 31, 32] has however demonstrated their efficacy on diverse computer vision regression tasks such as visual object tracking, head-pose estimation and age estimation.

In regression, the task is to learn to predict targets  $y^* \in \mathcal{Y}$  from inputs  $x^* \in \mathcal{X}$ , given a training set  $\mathcal{D}$  of i.i.d. input-target pairs,  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, (x_i, y_i) \sim p(x, y)$ . The input space  $\mathcal{X}$  depends on the specific problem, but can e.g. correspond to the space of images or point clouds. The target space  $\mathcal{Y}$  is continuous,  $\mathcal{Y} = \mathbb{R}^K$  for some  $K \geq 1$ .

In EBM-based regression [30, 31, 32], this task is addressed by modelling the distribution  $p(y|x)$  of  $y$  given  $x$  with a conditional EBM  $p(y|x; \theta)$ , defined according to,

$$p(y|x; \theta) = \frac{e^{f_\theta(x, y)}}{Z(x, \theta)}, \quad Z(x, \theta) = \int e^{f_\theta(x, \tilde{y})} d\tilde{y}. \quad (1)$$

Here,  $f_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$  is a DNN that maps any input-target pair  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  directly to a scalar  $f_\theta(x, y) \in \mathbb{R}$ , and  $Z(x, \theta)$  is the input-dependent normalizing partition function. The DNN output  $f_\theta(x, y)$  is interpreted as the (negative) energy of the distribution  $p(y|x; \theta)$ .

### 2.1 Prediction

At test-time, EBM-based regression entails predicting the most likely target under the model given an input  $x^*$ , i.e.  $y^* = \arg \max_y p(y|x^*; \theta) = \arg \max_y f_\theta(x^*, y)$ . In practice,  $y^* = \arg \max_y f_\theta(x^*, y)$  is approximated by refining an initial estimate  $\hat{y}$  via  $T$  steps of gradient ascent,

$$y \leftarrow y + \lambda \nabla_y f_\theta(x^*, y), \quad (2)$$

thus finding a local maximum of  $f_\theta(x^*, y)$ . Evaluation of the partition function  $Z(x^*, \theta)$  is therefore not required.

## 2.2 Training

The DNN  $f_\theta(x, y)$  that specifies the conditional EBM (1) can be trained using various methods for fitting a density  $p(y|x; \theta)$  to observed data  $\{(x_i, y_i)\}_{i=1}^N$ . Generally, the most straightforward such method is probably to minimize the negative log-likelihood  $\mathcal{L}(\theta) = -\sum_{i=1}^N \log p(y_i|x_i; \theta)$ , which for the EBM  $p(y|x; \theta)$  is given by,

$$\mathcal{L}(\theta) = \sum_{i=1}^N \log \left( \int e^{f_\theta(x_i, y)} dy \right) - f_\theta(x_i, y_i). \quad (3)$$

The integral in (3) is however intractable, preventing exact evaluation of  $\mathcal{L}(\theta)$ . One possible solution to this problem is to approximate the intractable integral using importance sampling, as employed in [30]. However, numerous alternative approaches also exist, including noise contrastive estimation (NCE) [48] and score matching [49]. The problem of how EBMs should be trained specifically for regression was studied in detail in [32], comparing six methods on the task of 2D bounding box regression in images. From this comparison, [32] concluded that a simple extension of NCE should be considered the go-to training method.

NCE entails learning to discriminate between observed data examples and samples drawn from a noise distribution. NCE was adopted for EBM-based regression only recently in [32], but has often been used to train EBMs for classification tasks in the past [50, 51, 52, 53]. Recently, it has also become highly utilized within self-supervised representation learning [54, 55, 56, 57]. Applying NCE to regression means training the DNN  $f_\theta(x, y)$  by minimizing the loss,

$$J(\theta) = -\frac{1}{N} \sum_{i=1}^N J_i(\theta),$$

$$J_i(\theta) = \log \frac{\exp\{f_\theta(x_i, y_i^{(0)}) - \log q(y_i^{(0)}|y_i)\}}{\sum_{m=0}^M \exp\{f_\theta(x_i, y_i^{(m)}) - \log q(y_i^{(m)}|y_i)\}}, \quad (4)$$

where  $y_i^{(0)} \triangleq y_i$ , and  $\{y_i^{(m)}\}_{m=1}^M$  are  $M$  samples drawn from a noise distribution  $q(y|y_i)$  that depends on the true target  $y_i$ . Effectively,  $J(\theta)$  in (4) is the softmax cross-entropy loss for a classification problem with  $M + 1$  classes. A simple choice for  $q(y|y_i)$  that was shown effective in [32] is setting  $q$  to a mixture of  $K$  Gaussians centered at  $y_i$ ,

$$q(y|y_i) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; y_i, \sigma_k^2 I), \quad (5)$$

where  $K$  and the variances  $\{\sigma_k^2\}_{k=1}^K$  are hyperparameters.

A simple extension to NCE, termed NCE+, was proposed and demonstrated to further improve the regression accuracy on certain tasks in [32]. The DNN  $f_\theta$  is still trained by minimizing  $J(\theta)$  in (4), but  $y_i^{(0)}$  is now defined as  $y_i^{(0)} \triangleq y_i + \nu_i$ . The true target  $y_i$  is thus perturbed with  $\nu_i \sim q_\beta(y)$ , where  $q_\beta$  is a zero-centered and scaled version of  $q(y|y_i)$  in (5), i.e.  $q_\beta(y) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y; 0, \beta\sigma_k^2 I)$ . NCE+ accounts for possible inaccuracies in the annotation process producing  $y_i$ , and can be understood as a direct generalization of NCE. In fact, NCE is recovered as a special case when  $\beta \rightarrow 0$  in  $q_\beta(y)$ .

### 3 Method

We apply EBM-based regression to 3DOD by extending the state-of-the-art 3D object detector SA-SSD [19] with a conditional EBM  $p(y|x; \theta)$  (1). In Sec. 3.1, we first provide necessary background on SA-SSD, including a description of its input and output data format. We then detail how the EBM  $p(y|x; \theta)$  is defined, employing differentiable pooling of 3D bounding boxes  $y$  and an added network branch, in Sec. 3.2. Our approach for training  $p(y|x; \theta)$  is based on NCE and further described in Sec. 3.3. Lastly, our prediction strategy using gradient ascent is detailed in Sec. 3.4.

#### 3.1 The SA-SSD 3D Object Detector

SA-SSD [19] takes a LiDAR point cloud of the scene as input  $x$  and produces a set  $\{d_i\}_{i=1}^D$  of  $D$  detections. Each detection  $d$  consists of a predicted 3D bounding box  $y$ ,

$$y = [c_x \ c_y \ c_z \ h \ w \ l \ \phi]^T \in \mathbb{R}^7, \quad (6)$$

and an associated classification confidence score  $s \in (0, 1)$ . In (6),  $(c_x, c_y, c_z)$  is the 3D coordinate of the bounding box center,  $(h, w, l)$  is the 3D bounding box size, and  $\phi$  is the heading angle of the bounding box.

The input LiDAR point cloud  $x = \{(p_x^{(i)}, p_y^{(i)}, p_z^{(i)})\}_{i=1}^n$  of  $n$  points is encoded into a sparse 3D tensor by means of voxelization. This tensor is then processed by a backbone network utilizing submanifold sparse 3D convolutional layers [58, 59], producing a 3D feature tensor  $h_1(x)$  of shape  $W \times L \times H \times C$ . A bird’s eye view (BEV) feature representation of the scene is then created by flattening  $h_1(x)$  into the 2D feature map  $h_2(x)$  of shape  $W \times L \times HC$ . Then,  $h_2(x)$  is further processed by six standard 2D convolutional layers, outputting the feature map  $h_3(x)$  of shape  $W \times L \times C'$ . Finally,  $h_3(x)$  is fed to a detection network, in which two  $1 \times 1$  convolutions are applied. The first outputs

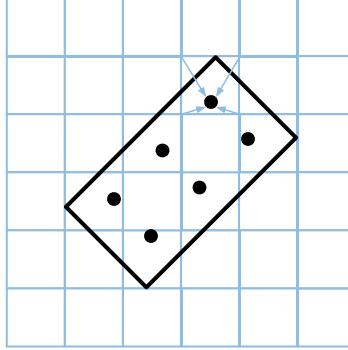
classification confidence scores and the second outputs offsets for a  $W \times L$  grid of anchor 3D bounding boxes.

The SA-SSD backbone and detection networks are trained by minimizing a weighted sum of multiple losses. The focal loss [60] is employed for the classification sub-task, and the Huber loss [25] is used for the regression of anchor bounding box offsets. Additionally, SA-SSD employs two losses stemming from auxiliary tasks. By inverting the voxelization via interpolation, 3D feature tensors in the backbone network are represented as point-wise feature vectors. These are then utilized for point-wise foreground segmentation, i.e. predicting whether or not a point lies within any ground truth 3D bounding box, and point-wise center offset regression, i.e. predicting the offset from a foreground point to the center of its 3D bounding box.

## 3.2 Conditional EBM Definition

In this work, we extend the SA-SSD 3D object detector with a conditional EBM  $p(y|x; \theta) = e^{f_\theta(x,y)} / \int e^{f_\theta(x,\tilde{y})} d\tilde{y}$ , which is fully specified by the DNN  $f_\theta$ . To enable the use of gradient ascent at test time (Sec. 2.1), the DNN must be designed such that its scalar output  $f_\theta(x, y)$  is differentiable w.r.t. the 3D bounding box  $y$  (6). To achieve this, we take inspiration from the recent work [30, 31, 32] applying EBM-based regression to 2D bounding box regression in images. Thus, we design a differentiable pooling operator that, for a given 3D bounding box  $y$ , extracts a feature vector from the SA-SSD backbone network output. This feature vector is then processed by an added network branch of fully-connected layers, outputting the energy value  $f_\theta(x, y) \in \mathbb{R}$ .

**Differentiable Pooling of 3D Bounding Boxes** Various pooling operators for 3D bounding boxes  $y$  (6) have been utilized for refining proposal bounding boxes in previous work [17, 2, 1, 3], none of which are however differentiable w.r.t. the bounding box  $y$ . [17] extracts all points in the point cloud  $x = \{(p_x^{(i)}, p_y^{(i)}, p_z^{(i)})\}_{i=1}^n$  which lie within a given box  $y$ , and then processes the associated point-wise features to extract a feature vector for  $y$ . This operator is however not differentiable w.r.t.  $y$ , due to the required discrete assessment of whether a point  $(p_x^{(i)}, p_y^{(i)}, p_z^{(i)})$  lies within the 3D bounding box  $y$  or not. [2] instead divides the box  $y$  into a 3D grid and extracts all points which lie within each grid cell. By also encoding which grid cells are empty, this pooling operator better captures geometric information. Because of the discrete extraction of points for each grid cell, it is however still not differentiable w.r.t. the 3D bounding box  $y$ . For similar reasons, the pooling operators utilized in [1, 3], which capture even richer contextual information, are not differentiable w.r.t.  $y$  either.



**Figure 3:** Illustration of our modified variant of RoIAlign [61] for oriented 2D bounding boxes. In this example, the regular  $W' \times L'$  grid is  $2 \times 3$ . Bilinear interpolation is used to extract a feature vector for each of the  $W'L'$  grid points.

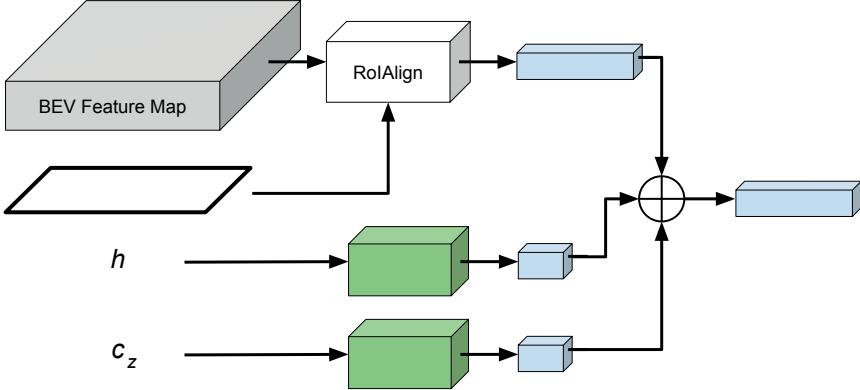
Instead, we utilize the 2D feature map  $h_3(x)$  of shape  $W \times L \times C'$  that is produced by the SA-SSD backbone network. This is a compact yet powerful BEV feature representation of the scene. Specifically, we extract a feature vector  $h_4(x, y^{\text{BEV}})$  by pooling  $h_3(x)$  with  $y^{\text{BEV}}$ ,

$$y^{\text{BEV}} = [c_x \ c_y \ w \ l \ \phi]^T \in \mathbb{R}^5, \quad (7)$$

which is the BEV version of the 3D bounding box  $y$  (6). Since  $y^{\text{BEV}}$  is an oriented 2D bounding box and not necessarily axis-aligned, we can not directly apply standard 2D bounding box pooling operators [62, 61, 34]. Instead we employ a modified variant of RoIAlign [61], which entails dividing  $y^{\text{BEV}}$  into a regular  $W' \times L'$  grid, and extracting a feature vector  $g \in \mathbb{R}^{C'}$  in each grid point via bilinear interpolation of  $h_3(x)$ . See Figure 3 for an illustration. This operation results in a 2D feature map of shape  $W' \times L' \times C'$ , which we then flatten to obtain the feature vector  $h_4(x, y^{\text{BEV}}) \in \mathbb{R}^{W'L'C'}$ . By flattening the feature map instead of e.g. averaging over it, more information is preserved in  $h_4(x, y^{\text{BEV}})$ . It can thus be used to discriminate between a given box and the same box rotated  $\pi$  rad.

This pooling operation is differentiable w.r.t.  $y^{\text{BEV}}$ , but the extracted feature vector  $h_4(x, y^{\text{BEV}}) \in \mathbb{R}^{W'L'C'}$  is of course only a function of  $y^{\text{BEV}}$  (7), not of the full 3D bounding box  $y$  (6). Using gradient ascent at test-time would thus not update the  $z$  coordinate  $c_z$  or height  $h$  of the bounding box  $y$ . To resolve this, we take inspiration from the architecture used for EBM-based age estimation [30]. We thus process  $c_z \in \mathbb{R}$  and  $h \in \mathbb{R}$  by two small fully-connected layers, generating feature vectors  $g_{c_z} \in \mathbb{R}^{C''}$  and  $g_h \in \mathbb{R}^{C''}$ . Finally, we concatenate the three vectors to obtain  $h_5(x, y)$ ,

$$h_5(x, y) = h_4(x, y^{\text{BEV}}) \oplus g_{c_z} \oplus g_h \in \mathbb{R}^{W'L'C'+2C''}, \quad (8)$$



**Figure 4:** Detailed illustration of the proposed differentiable pooling operation from 3D bounding box  $y$  (6) to feature vector  $h_5(x, y)$  (8). The BEV version of  $y$  is pooled with the BEV feature map produced by SA-SSD. The  $z$  coordinate  $c_z$  and height  $h$  of the box  $y$  are processed by two fully-connected layers.

where  $\oplus$  indicates vector concatenation. The complete pooling operation from 3D bounding box  $y$  to feature vector  $h_5(x, y)$  is illustrated in Figure 4.

**Energy Prediction Branch** Following [30, 31, 32], we add an extra network branch onto SA-SSD for processing the extracted feature vector. The network branch consists of three fully-connected layers. It takes the feature vector  $h_5(x, y) \in \mathbb{R}^{W'L'C'+2C''}$  as input and outputs the scalar energy  $f_\theta(x, y) \in \mathbb{R}$ , thus fully specifying the conditional EBM  $p(y|x; \theta)$  (1). The complete architecture of  $p(y|x; \theta)$  is illustrated in Figure 2.

### 3.3 Detector Training

Following the work on EBM-based 2D object detection [30, 32], the extra fully-connected layers described in Sec 3.2 are added onto a pre-trained and fixed SA-SSD detector. The parameters  $\theta$  in  $f_\theta(x, y)$  thus only stem from these added fully-connected layers, and the SA-SSD backbone and detection networks are kept fixed during training of the DNN  $f_\theta$ . To train  $f_\theta$ , we use NCE as described in Sec 2.2. We employ the same training parameters (batch size, data augmentation etc.) as for SA-SSD [19], only replacing the original detector loss with the NCE loss (4).

### 3.4 Detector Inference

At test-time, the input LiDAR point cloud  $x^*$  is first processed by the SA-SSD detector. SA-SSD outputs the 2D feature map  $h_3(x^*)$  and a set  $\{(\hat{y}_i, s_i)\}_{i=1}^D$

**Algorithm 1** Gradient-based refinement.

---

**Input:**  $x^*, \{\hat{y}_i\}_{i=1}^D, T, \lambda, \eta$ .

```

1: for  $i = 1, \dots, D$  do
2:    $y \leftarrow \hat{y}_i$ .
3:   for  $t = 1, \dots, T$  do
4:      $\text{PrevValue} \leftarrow f_\theta(x^*, y)$ .
5:      $\tilde{y} \leftarrow y + \lambda \nabla_y f_\theta(x^*, y)$ .
6:      $\text{NewValue} \leftarrow f_\theta(x^*, \tilde{y})$ .
7:     if  $\text{NewValue} > \text{PrevValue}$  then
8:        $y \leftarrow \tilde{y}$ .
9:     else
10:       $\lambda \leftarrow \eta \lambda$ .
11:    $y_i \leftarrow y$ .
12: Return  $\{y_i\}_{i=1}^D$ .
```

---

of  $D$  detections, where  $\hat{y}_i$  is a 3D bounding box (6) and  $s_i$  is the associated classification confidence score. We then take all bounding boxes  $\{\hat{y}_i\}_{i=1}^D$  as initial estimates and refine these via  $T$  steps of gradient ascent (Sec 2.1), producing  $\{y_i\}_{i=1}^D$ . The initial 3D bounding boxes  $\{\hat{y}_i\}_{i=1}^D$  are thus refined by being moved toward different local maxima of  $f_\theta(x^*, y)$ . The refined boxes  $\{y_i\}_{i=1}^D$  are finally combined with the original confidence scores, returning the detections  $\{(y_i, s_i)\}_{i=1}^D$ .

This gradient-based refinement of the detections produced by SA-SSD of course lowers the detector inference speed somewhat. The point cloud  $x^*$  is however still processed by SA-SSD only once, and the scalar  $f_\theta(x^*, y)$  is extracted from  $h_3(x^*)$  using an efficient pooling operator and just a few fully-connected layers. Moreover, the gradient  $\nabla_y f_\theta(x^*, y)$  can be efficiently evaluated using automatic differentiation. The complete refinement procedure is detailed in Algorithm 1, where  $\lambda$  denotes the gradient ascent step-length,  $\eta$  is a decay of the step-length, and the  $\text{NewValue} > \text{PrevValue}$  check ensures that  $f_\theta(x^*, y)$  is never decreased.

## 4 Experiments

We evaluate our EBM-based 3DOD approach on the KITTI 3DOD dataset [18] and compare it with the SA-SSD [19] baseline and other state-of-the-art methods. Our detector is implemented in PyTorch [63]. Training and inference code is publicly available.

## 4.1 Dataset

KITTI [18] is the most commonly used dataset for automotive 3DOD. It contains 7481 examples for training, and 7518 *test* examples without publicly available ground truth annotations. Following common practice [19, 3], the training examples are further divided into *train* (3712 examples) and *val* (3769 examples) splits. We train models exclusively on the *train* split and set hyper-parameters using the *val* split. We report results both on *val*, and on the *test* split by submitting detections to the KITTI benchmark server. Following SA-SSD, we conduct experiments only on the car object class.

**Evaluation Metrics** On the KITTI benchmark server, models are evaluated in terms of average precision (AP) in both 3D and BEV. It considers three different difficulty levels (easy, moderate and hard), based on how far away and occluded objects are. AP is the area under the precision-recall curve, where a predicted bounding box is considered a true positive if its 3D/BEV IoU with a ground truth box exceeds a certain threshold. For cars, the threshold is set to 0.7 on the KITTI benchmark. Two predicted boxes with IoU of, e.g., 0.71 and 0.99 thus have identical effect on this metric. Since our main goal is to improve the accuracy of all predicted bounding boxes, we also report the AP for higher thresholds  $\{0.75, 0.8, 0.85, 0.9\}$  on the *val* split. All reported AP values are computed using 40 recall positions.

## 4.2 Implementation Details

We utilize the open-source implementation and pre-trained model provided<sup>1</sup> by the SA-SSD authors. The feature map  $h_3(x)$  that is produced by the backbone network is of shape  $200 \times 176 \times 256$ . We divide each  $y^{\text{BEV}}$  (7) into a regular  $4 \times 7$  grid, meaning that the feature vector  $h_4(x, y^{\text{BEV}}) \in \mathbb{R}^{7168}$ . We process  $c_z \in \mathbb{R}$  and  $h \in \mathbb{R}$  with separate fully-connected layers (dimensions:  $1 \rightarrow 16$ ,  $16 \rightarrow 16$ ), generating  $g_{c_z} \in \mathbb{R}^{16}$  and  $g_h \in \mathbb{R}^{16}$ . After concatenation, we thus obtain  $h_5(x, y) \in \mathbb{R}^{7200}$ . Finally,  $h_5(x, y)$  is processed by three fully-connected layers of dimensions  $7200 \rightarrow 1024$ ,  $1024 \rightarrow 1024$ ,  $1024 \rightarrow 1$ . To train the DNN  $f_\theta(x, y)$ , i.e. the added fully-connected layers, we just replace the original detector loss with the NCE loss (Sec. 3.3). We also considered NCE+ with  $\beta > 0$ , but saw no clear improvements over NCE. We hypothesize this is because there is less inherent ambiguity in the annotation process of 3D bounding boxes than of 2D bounding boxes in images. As in [32], we set  $K = 3$  with  $\sigma_1 = \sigma_3/4$ ,  $\sigma_2 = \sigma_3/2$  for the noise distribution  $q(y|y_i)$  (5). After ablation, optimizing 3D AP (moderate difficulty) on the *val* split, we set  $\sigma_3$  differently for different components of the 3D box  $y$  (6). Specifically,  $\sigma_3 = 0.25$  for  $(c_x, c_y)$ ,  $\sigma_3 = 0.125$  for  $(c_z, h, w, l)$  and  $\sigma_3 = 0.0625$  for

---

<sup>1</sup><https://github.com/skyhehe123/SA-SSD>

**Table 1:** Results on KITTI *test* in terms of 3D and BEV AP. Our SA-SSD+EBM detector consistently outperforms the SA-SSD baseline, and achieves highly competitive performance also compared to other state-of-the-art methods.

	3D @ 0.7			BEV @ 0.7		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Part-A <sup>2</sup> [2]	87.81	78.49	73.51	91.70	87.79	84.61
SERCNN [64]	87.74	78.96	74.30	94.11	88.10	83.43
EPNet [65]	89.81	79.28	74.59	94.22	88.47	83.69
Point-GNN [66]	88.33	79.47	72.29	93.11	89.17	83.90
3DSSD [67]	88.36	79.57	74.55	92.66	89.02	85.86
STD [1]	87.95	79.71	75.09	94.74	89.19	86.42
SA-SSD [19]	88.75	79.79	74.16	95.03	<b>91.03</b>	85.96
3D-CVF [14]	89.20	80.05	73.11	93.52	89.56	82.45
CLOCs-PVCas [13]	88.94	80.67	<b>77.15</b>	93.05	89.80	<b>86.57</b>
PV-RCNN [3]	90.25	<b>81.43</b>	76.82	94.98	90.65	86.14
SA-SSD	88.80	79.52	72.30	95.44	89.63	84.34
<b>SA-SSD+EBM</b>	<b>91.05</b>	80.12	72.78	<b>95.64</b>	89.86	84.56
<i>Rel. Improvement</i>	+2.53%	+0.75%	+0.66%	+0.21%	+0.26%	+0.26%

$\phi$ . Following [30, 32], we also set  $T = 10$  and  $\eta = 0.5$  for gradient-based refinement (Algorithm 1). The step-length  $\lambda = 0.0002$  was selected based on ablation.

### 4.3 3DOD Results on KITTI

Results on KITTI *test* in terms of 3D and BEV AP (0.7 threshold) are found in Table 1. We mainly compare the performance of our EBM-based 3D object detector (SA-SSD+EBM) to the pre-trained SA-SSD it extends, and include other state-of-the-art detectors for reference. We also include the results for SA-SSD reported in the original paper [19], as these differ somewhat from those obtained with the provided pre-trained model. In Table 1, we observe that the added EBM and gradient-based refinement consistently improves the SA-SSD baseline across all metrics. We also observe that our SA-SSD+EBM detector achieves very competitive performance compared to previous methods.

Results on KITTI *val* in terms of 3D and BEV AP (0.7 threshold) are found in Table 2. There, we only include detectors for which AP values computed using 40 recall positions are available. In Table 2, we again observe that our EBM-based detector consistently outperforms the SA-SSD baseline. On KITTI *val*, our SA-SSD+EBM also sets a new state-of-the-art in terms of all but one of the metrics.

A further comparison of SA-SSD+EBM and the SA-SSD baseline is provided in Table 3. There, we report AP for higher thresholds  $\{0.75, 0.8, 0.85, 0.9\}$

**Table 2:** Results on KITTI *val* in terms of 3D and BEV AP. Our proposed detector consistently outperforms the SA-SSD baseline, and sets a new state-of-the-art for all but one of the metrics.

	3D @ 0.7			BEV @ 0.7		
	Easy	Moderate	Hard	Easy	Moderate	Hard
SA-SSD [19]	93.23	84.30	81.36	-	-	-
CLOCs-PVCas [13]	92.78	85.94	<b>83.25</b>	93.48	91.98	89.48
PV-RCNN [3]	92.57	84.83	82.69	95.76	91.11	88.93
SA-SSD	93.14	84.65	81.86	96.56	92.84	90.36
<b>SA-SSD+EBM</b>	<b>95.45</b>	<b>86.83</b>	82.23	<b>96.60</b>	<b>92.92</b>	<b>90.43</b>
<i>Rel. Improvement</i>	+2.48%	+2.58%	+0.45%	+0.04%	+0.09%	+0.08%

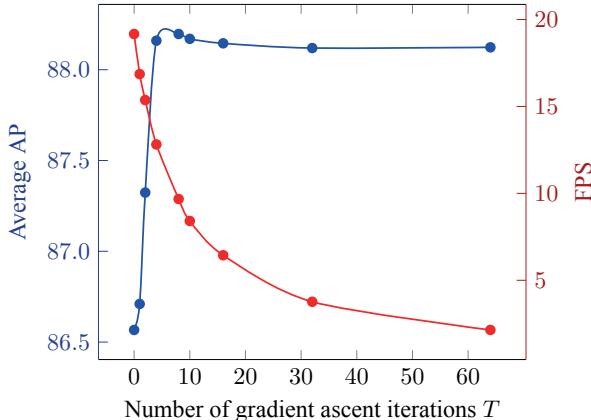
**Table 3:** Results on KITTI *val* in terms of 3D and BEV AP for higher thresholds  $\{0.75, 0.8, 0.85, 0.9\}$ . Our SA-SSD+EBM detector consistently outperforms the SA-SSD baseline across all metrics, and the relative improvement increases with the AP threshold.

	3D @ 0.75			3D @ 0.8			3D @ 0.85			3D @ 0.9		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SA-SSD	84.48	73.91	70.99	60.89	50.08	47.37	24.29	19.58	18.05	2.06	1.58	1.33
<b>SA-SSD+EBM</b>	87.85	74.96	71.95	66.70	54.32	51.36	31.02	23.91	21.95	3.45	2.74	2.26
<i>Rel. Improvement</i>	+3.99%	+1.42%	+1.35%	+9.54%	+8.47%	+8.42%	+27.7%	+22.1%	+21.6%	+67.5%	+73.4%	+69.9%
	BEV @ 0.75			BEV @ 0.8			BEV @ 0.85			BEV @ 0.9		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SA-SSD	95.41	87.47	84.79	87.12	79.07	74.65	61.53	54.15	50.39	17.48	15.71	14.58
<b>SA-SSD+EBM</b>	95.47	87.54	84.88	88.31	80.06	77.25	68.40	58.62	54.48	26.60	22.03	19.48
<i>Rel. Improvement</i>	+0.06%	+0.08%	+0.11%	+1.37%	+1.25%	+3.48%	+11.2%	+8.25%	+8.12%	+52.2%	+40.2%	+33.6%

on KITTI *val*. We observe that the gradient-based refinement consistently improves detector performance across all metrics, and that the relative gain is larger for higher thresholds. Our approach thus also refines accurate bounding boxes even further, an effect not captured by the standard AP metrics.

#### 4.4 Analysis of Inference Speed

The improved detection performance compared to SA-SSD comes with a decreased inference speed. On a single NVIDIA TITAN Xp GPU, SA-SSD runs at 19.2 FPS, while SA-SSD+EBM runs at 8.4 FPS for  $T = 10$  gradient ascent iterations. We further analyze how the choice of  $T$  affects detector inference speed and performance in Figure 5. The performance is here given in terms of 3D AP (0.7 threshold) averaged over the three difficulty levels (easy, moderate, hard), on KITTI *val*. We observe that the choice  $T = 4$  provides approximately equal performance compared to  $T = 10$ , while only decreasing the inference speed to 12.8 FPS. This trade-off between detector performance and inference speed could potentially be further improved by using fewer grid points in our RoIAlign variant, or by using a more lightweight energy prediction network



**Figure 5:** Impact of the number of gradient ascent iterations  $T$  in Algorithm 1 on detector performance (3D AP with 0.7 threshold, averaged over easy, moderate and hard) and detector inference speed (FPS), on KITTI *val*. Refinement with  $T = 4$  iterations significantly improves the detector performance, while only decreasing the inference speed from 19.2 to 12.8 FPS.

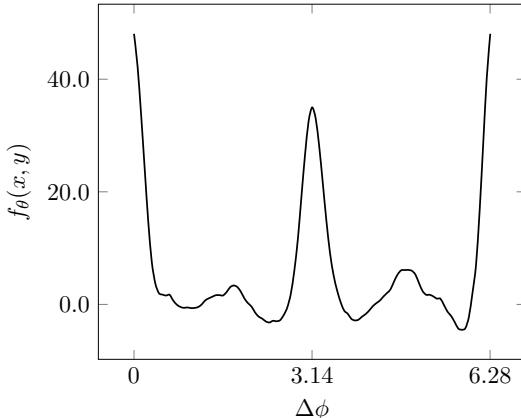
branch. Our approach could also be very well-suited for offboard 3DOD [68], where inference speed is less of a concern. Exploring these directions is left for future work.

#### 4.5 Analysis of Learned Distribution

For 3DOD from LiDAR point clouds, it can be inherently difficult to correctly predict the heading angle  $\phi$  of a 3D bounding box  $y$  (6). This is because it is often difficult, when only given a point cloud, to distinguish between two otherwise identical cars which are heading in opposite directions. The true distribution  $p(y|x)$  will thus often have two distinct modes, one at the true heading angle  $\phi$  and one at  $\phi + \pi$ . In Figure 6, we visualize  $f_\theta(x, y) \in \mathbb{R}$  as a function of  $\Delta\phi$  when a predicted 3D bounding box  $y$  is rotated  $\Delta\phi$  rad, demonstrating that our trained EBM  $p(y|x; \theta)$  does indeed capture this inherent multi-modality in the true  $p(y|x)$ . Future directions include investigating if the trained EBM  $p(y|x; \theta)$  could be used to construct accurate estimates of prediction uncertainty, or provide other useful insights.

### 5 Conclusion

We applied conditional EBMs  $p(y|x; \theta)$  to the task of 3D bounding box regression, thus extending the recent EBM-based regression approach from 2D to



**Figure 6:** Visualization of the DNN scalar output  $f_\theta(x, y)$  when a predicted 3D bounding box  $y$  (6) is rotated  $\Delta\phi$  rad. The two distinct modes at  $\Delta\phi = 0$  and  $\Delta\phi = \pi$  demonstrate that the trained EBM  $p(y|x; \theta)$  captures the inherent multi-modality in  $p(y|x)$ .

3D object detection. By designing a differentiable pooling operator for 3D bounding boxes, we could integrate a conditional EBM  $p(y|x; \theta)$  into the state-of-the-art 3D object detector SA-SSD. On the KITTI dataset, our approach consistently outperformed the SA-SSD baseline across all 3DOD metrics, and achieved highly competitive performance also compared to other state-of-the-art methods. By demonstrating the potential of EBM-based regression for highly accurate 3DOD, we hope that our work will encourage the research community to further explore the application of EBMs to 3DOD and other important regression tasks.

**Acknowledgments** This research was supported by the Swedish Foundation for Strategic Research via the project *ASSEMBLE*, the Knut and Alice Wallenberg Foundation via the *Wallenberg AI, Autonomous Systems and Software Program (WASP)*, and the *Kjell & Märta Beijer Foundation*.

## References

- [1] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. “STD: Sparse-to-dense 3d object detector for point cloud.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.
- [2] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li. “From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2020).

- [3] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. “PV-RCNN: Point-voxel feature set abstraction for 3D object detection.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [4] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kortschieder. “Disentangling monocular 3d object detection.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019.
- [5] Y. Chen, L. Tai, K. Sun, and M. Li. “MonoPair: Monocular 3D Object Detection Using Pairwise Spatial Relationships.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [6] X. Shi, Z. Chen, and T.-K. Kim. “Distance-Normalized Unified Representation for Monocular 3D Object Detection.” In: *European Conference on Computer Vision (ECCV)*. Springer. 2020.
- [7] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. “Multi-task multi-sensor fusion for 3d object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [8] M. Liang, B. Yang, S. Wang, and R. Urtasun. “Deep continuous fusion for multi-sensor 3d object detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [9] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. “Joint 3d proposal generation and object detection from view aggregation.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.
- [10] M. Meyer and G. Kuschk. “Automotive radar dataset for deep learning based 3d object detection.” In: *2019 16th European Radar Conference (EuRAD)*. IEEE. 2019.
- [11] M. Meyer and G. Kuschk. “Deep learning based 3d object detection for automotive radar and camera.” In: *2019 16th European Radar Conference (EuRAD)*. IEEE. 2019.
- [12] B. Yang, R. Guo, M. Liang, S. Casas, and R. Urtasun. “RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [13] S. Pang, D. Morris, and H. Radha. “CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.
- [14] J. H. Yoo, Y. Kim, J. S. Kim, and J. W. Choi. “3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-View Spatial Feature Fusion for 3D Object Detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

- [15] Y. Zhou and O. Tuzel. “VoxelNet: End-to-end learning for point cloud based 3D object detection.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [16] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. “Pointpillars: Fast encoders for object detection from point clouds.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [17] S. Shi, X. Wang, and H. Li. “Pointrcnn: 3d object proposal generation and detection from point cloud.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [18] A. Geiger, P. Lenz, and R. Urtasun. “Are we ready for autonomous driving? The KITTI vision benchmark suite.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
- [19] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang. “Structure Aware Single-stage 3D Object Detection from Point Cloud.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [20] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [21] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [22] O. Makansi, E. Ilg, O. Cicek, and T. Brox. “Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [23] H. Pan, H. Han, S. Shan, and X. Chen. “Mean-variance loss for deep age estimation from a face.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [24] R. Diaz and A. Marathe. “Soft Labels for Ordinal Regression.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [25] P. J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* (1964).
- [26] D. Feng, L. Rosenbaum, and K. Dietmayer. “Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3D vehicle detection.” In: *IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

- [27] D. Feng, L. Rosenbaum, F. Timm, and K. Dietmayer. “Leveraging heteroscedastic aleatoric uncertainties for robust real-time lidar 3D object detection.” In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019.
- [28] D. Feng, L. Rosenbaum, C. Glaeser, F. Timm, and K. Dietmayer. “Can we trust you? On calibration of a probabilistic object detector for autonomous driving.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop* (2019).
- [29] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington. “Lasernet: An efficient probabilistic 3d object detector for autonomous driving.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [30] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [31] M. Danelljan, L. Van Gool, and R. Timofte. “Probabilistic Regression for Visual Tracking.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [32] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. “How to Train Your Energy-Based Model for Regression.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2020.
- [33] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).
- [34] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. “Acquisition of localization confidence for accurate object detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [35] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. “Energy-based models for sparse overcomplete representations.” In: *Journal of Machine Learning Research* (2003).
- [36] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. “A neural probabilistic language model.” In: *Journal of machine learning research* (2003).
- [37] A. Mnih and G. Hinton. “Learning nonlinear constraints with contrastive backpropagation.” In: *Proceedings of the IEEE International Joint Conference on Neural Networks*. IEEE. 2005.
- [38] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. “Unsupervised discovery of nonlinear structure using contrastive backpropagation.” In: *Cognitive science* (2006).
- [39] M. Osadchy, M. L. Miller, and Y. L. Cun. “Synergistic face detection and pose estimation with energy-based models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.

- [40] J. Xie, Y. Lu, S.-C. Zhu, and Y. Wu. “A theory of generative convnet.” In: *International Conference on Machine Learning (ICML)*. 2016.
- [41] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu. “Learning generative convnets via multi-grid modeling and sampling.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [42] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. “Learning non-convergent non-persistent short-run MCMC toward energy-based model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [43] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [44] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. “Your classifier is secretly an energy based model and you should treat it like one.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [45] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu. “Flow contrastive estimation of energy-based models.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [46] B. Pang, T. Han, E. Nijkamp, S.-C. Zhu, and Y. N. Wu. “Learning latent space energy-based prior model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [47] F. Bao, C. Li, K. Xu, H. Su, J. Zhu, and B. Zhang. “Bi-level Score Matching for Learning Energy-based Latent Variable Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [48] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010.
- [49] A. Hyvärinen. “Estimation of non-normalized statistical models by score matching.” In: *Journal of Machine Learning Research* (2005).
- [50] A. Mnih and Y. W. Teh. “A fast and simple algorithm for training neural probabilistic language models.” In: *International Conference on Machine Learning (ICML)*. 2012.
- [51] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2013.

- [52] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. “Exploring the limits of language modeling.” In: *arXiv preprint arXiv:1602.02410* (2016).
- [53] Z. Ma and M. Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [54] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. “Learning deep representations by mutual information estimation and maximization.” In: *International Conference on Learning Representations (ICLR)*. 2019.
- [55] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning representations by maximizing mutual information across views.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [56] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A simple framework for contrastive learning of visual representations.” In: *arXiv preprint arXiv:2002.05709* (2020).
- [57] T. Han, W. Xie, and A. Zisserman. “Self-supervised Co-training for Video Representation Learning.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.
- [58] Y. Yan, Y. Mao, and B. Li. “Second: Sparsely embedded convolutional detection.” In: *Sensors* (2018).
- [59] B. Graham, M. Engelcke, and L. Van Der Maaten. “3d semantic segmentation with submanifold sparse convolutional networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [60] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal loss for dense object detection.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
- [61] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. “Mask R-CNN.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2017).
- [62] R. B. Girshick. “Fast R-CNN.” In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2015).
- [63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

- [64] D. Zhou, J. Fang, X. Song, L. Liu, J. Yin, Y. Dai, H. Li, and R. Yang. “Joint 3D Instance Segmentation and Object Detection for Autonomous Driving.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [65] T. Huang, Z. Liu, X. Chen, and X. Bai. “EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [66] W. Shi and R. Rajkumar. “Point-GNN: Graph neural network for 3d object detection in a point cloud.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [67] Z. Yang, Y. Sun, S. Liu, and J. Jia. “3DSSD: Point-based 3d single stage object detector.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [68] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov. “Offboard 3D Object Detection from Point Cloud Sequences.” In: *arXiv preprint arXiv:2103.05073* (2021).



# Paper V

## Title

Deep Energy-Based NARX Models

## Authors

Johannes Hendriks, Fredrik K. Gustafsson, Antônio Ribeiro, Adrian Wills, Thomas B. Schön

## Edited version of

J. N. Hendriks, F. K. Gustafsson, A. H. Ribeiro, A. G. Wills, and T. B. Schön. “Deep Energy-Based NARX Models.” In: *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*. 2021



# Deep Energy-Based NARX Models

## Abstract

This paper is directed towards the problem of learning nonlinear ARX models based on observed input–output data. In particular, our interest is in learning a conditional distribution of the current output based on a finite window of past inputs and outputs. To achieve this, we consider the use of so-called energy-based models, which have been developed in allied fields for learning unknown distributions based on data. This energy-based model relies on a general function to describe the distribution, and here we consider a deep neural network for this purpose. The primary benefit of this approach is that it is capable of learning both simple and highly complex noise models, which we demonstrate on simulated and experimental data.

## 1 Introduction

This paper considers the problem of learning a model for dynamic systems based on observed input–output data. This problem has a long and fruitful history within the system identification, statistics and machine learning communities and there are many different ways to approach it. For example, a regularly employed approach is to first define a suitable parameterized model structure based on knowledge of the system. Then we learn, adapt, infer or estimate the parameters based on the available evidence in the data. To decide between different parameters, and ultimately provide the best values, the user is required to choose a performance criterion such as the maximum-likelihood (ML) or prediction-error criteria.

It is important to note that both the model structure and estimation method involve assumptions about uncertainty, be they explicit or implicit. That is, the probability distribution that represents this uncertainty is assumed. For example, it is not uncommon that users explicitly assume additive white Gaussian noise as a way of modelling measured output uncertainty. Further, it can be argued that this same assumption is implicit in mean-squared-error estimation.

More generally, in many practical situations, it is difficult to simply justify these assumptions from the available prior system knowledge or even from the data.

This paper details a means for addressing this difficulty by allowing the distribution itself to be modelled using a highly flexible function that is learned from the available data. The primary benefit of this approach is that it can easily adapt to both highly complex distributions and also less complicated ones such as a unimodal Gaussian. The inspiration for this approach comes from the allied field of machine learning where so-called energy-based models (EBMs), typically combined with deep neural networks (DNNs), are employed for modelling unknown distributions with great success [1, 2, 3].

To make these ideas concrete, this paper will concentrate on the class of nonlinear-autoregressive-exogenous-input (NARX) dynamic models [4]. In particular, it will be assumed that the current system output  $y_t$  is related to past outputs  $y_{t-1}, \dots, y_{t-D_y}$ , and past inputs  $u_{t-1}, \dots, u_{t-D_u}$ ; where  $D_y$  is the maximum output delay and  $D_u$  is the maximum input delay. Our particular interest here is in providing a conditional distribution of  $y_t$  given the past data window. That is, we are concerned with describing

$$y_t|x_t \sim p(y_t|x_t), \quad (1)$$

where  $x_t$  contains the past data window:

$$x_t = \{y_{t-1}, \dots, y_{t-D_y}, u_{t-1}, \dots, u_{t-D_u}\}. \quad (2)$$

Unfortunately, it is not immediately obvious how to choose this distribution so that it explains measured system data. One way to address this difficulty is to assume a functional form for this distribution that relies on some unknown parameters  $\theta$ , which we denote as  $p_\theta(y_t|x_t)$ . The idea then is to estimate these parameter values based on the available evidence in the data. This raises at least two questions; how should we parameterise this distribution, and, how should we learn from the data?

Regarding the first problem of parameterisation, a traditional approach for NARX models is to first formulate an output equation form

$$y_t = f_\theta(x_t) + e_t, \quad (3)$$

where  $f_\theta$  is a function that is traditionally linear in the parameters  $\theta$ , but is otherwise quite a general function of the past data  $x_t$ . The added term  $e_t$  is a random variable that characterises the error between the function output  $f_\theta(x_t)$  and the measured output  $y_t$ , and, its distribution may also depend on  $\theta$ . Therefore, by construction, the conditional distribution of interest,  $p_\theta(y_t|x_t)$ , will depend on the assumed choice of distribution for  $e_t$ .

Regarding the second problem of learning from the data, again a traditional approach is to formulate and solve the associated ML problem [4]. By way of

a concrete example, assuming that  $y_t \in \mathbb{R}$  and the noise  $e_t$  is i.i.d. Gaussian with zero mean and variance  $\sigma^2$ , then the ML solution for  $\theta$  coincides with

$$\hat{\theta} = \arg \min_{\theta} \sum_{t=1}^T \|y_t - f_{\theta}(x_t)\|^2. \quad (4)$$

Therefore, an estimate of the desired conditional distribution  $p(y_t|x_t)$  is given by

$$y_t|x_t \sim \mathcal{N}(f_{\hat{\theta}}(x_t), \sigma^2). \quad (5)$$

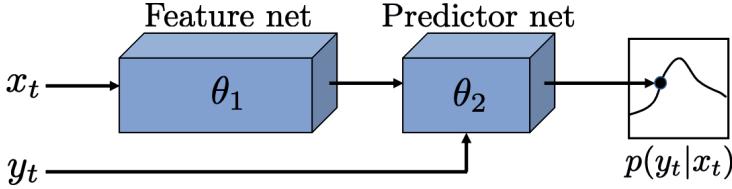
More complex distributions for  $e_t$  can also be accommodated within the ML framework, but this requires the user to choose a suitable distributional family. In many practical situations, it is not obvious how to select this family based on prior system knowledge.

This paper aims to address this difficulty by providing a highly flexible class of distributions that is adapted to each new problem based on the available system data. In particular,  $p(y_t|x_t)$  will be modelled with the conditional EBM  $p_{\theta}(y_t|x_t) = e^{g_{\theta}(y_t, x_t)} / \int e^{g_{\theta}(\gamma, x_t)} d\gamma$ , where the scalar function  $g_{\theta}$  is represented by a DNN with associated parameters  $\theta$ . This energy-based approach puts very few restricting assumptions on the true distribution  $p(y_t|x_t)$ , enabling it to be learned directly from data.

**Contributions** The main contribution of this paper is an energy-based model capable of learning  $p(y_t|x_t)$  for dynamic systems. We evaluate the new construction on both simulated and experimental data, demonstrating its benefits compared to more traditional NARX models. This paper thus illustrates the utility of EBMs and their potential within system identification.

## 2 Related Work

During the last decade, there has been a surge of interest in DNN models and these models have been used to obtain state-of-the-art solutions for many applications, including computer vision, speech recognition and natural language processing [5]. While the use of neural networks in system identification problems has a long history [6, 7], the success of the method in neighbouring areas has brought a new wave of interest within the system identification community [8], with recent papers leveraging acquired knowledge and being inspired by successful ideas from recent DNN applications. Examples of deep-learning-inspired ideas applied in system identification include; convolutional network layers [9], encoder-decoder structure [10] and recurrent neural networks and its extensions [10, 8].



**Figure 1:** Structure of the deep EB-NARX model used. Here,  $x_t$  is the known input data and  $y_t$  is a possible output value that we wish to estimate the conditional probability of. During training, the measured  $y_t$  will be used to train the predictor net.

EBMs have been extensively studied by the machine learning community [11, 12, 13]. They are usually employed for unsupervised learning applications, and have in recent years become particularly popular for generative modelling within computer vision [14, 1, 2]. In comparison, the application of EBMs to supervised learning problems is not a very well-studied topic, but their effectiveness has been demonstrated for both classification [15] and regression [3]. Most closely related to our proposed approach is the very recent work on employing conditional EBM's for regression [3, 16, 17], achieving state-of-the-art performance on tasks such as object detection and tracking.

### 3 Energy-Based NARX Models

Inspired by [3], we model the distribution  $p(y_t|x_t)$  with the conditional EBM

$$p_\theta(y_t|x_t) = \frac{e^{g_\theta(y_t, x_t)}}{\int e^{g_\theta(\gamma, x_t)} d\gamma}, \quad (6)$$

where  $g_\theta$  is a DNN that maps any pair  $(y_t, x_t)$  directly to a scalar output  $g_\theta(y_t, x_t) \in \mathbb{R}$ .

Here,  $p_\theta(y_t|x_t)$  is directly specified via the DNN  $g_\theta$ , which provides a highly flexible class of functions. This enables  $p_\theta(y_t|x_t)$  to model a wide range of distributions, including heavy-tailed, asymmetric or multimodal ones. Related to this, we note that the DNN output value  $g_\theta(y_t, x_t) \in \mathbb{R}$  is proportional to the logarithm of the distribution  $p_\theta(y_t|x_t)$ , not to the output  $y_t$  itself. This has implications for how the model may be used, which will be discussed in Section 3.3 below.

Evaluating the denominator  $Z(x_t) = \int g_\theta(y_t|x_t) dy_t$  in (6) presents a challenge since this integral is analytically intractable in general. For the case when  $y_t$  is low-dimensional, the integral may be evaluated using standard quadrature methods. In the more general case, we advocate the use of Monte Carlo methods for solving this integral (see [3] for details on this approach).

Since the EBM (6) relies on a nonlinear combination of previous data  $x_t$ , we will refer to this as an energy-based NARX (EB-NARX) model. Next, we first provide more details on the structure of the DNN  $g_\theta$  in Section 3.1. We then describe how to learn the unknown DNN parameters  $\theta$  based on a set of training data  $\mathcal{D} = \{y_t, x_t\}_{t=1}^T$ , in Section 3.2. Finally, we discuss how the model can be used for prediction, in Section 3.3.

### 3.1 Neural Network Structure

The DNN  $g_\theta$  is composed of two smaller neural networks; a feature net and a predictor net parametrised by  $\theta_1$  and  $\theta_2$ , respectively. The feature net takes  $x_t$  as input and produces a feature vector. This feature vector is then combined with  $y_t$  and fed as input to the predictor net, which finally outputs the unnormalised log density  $g_\theta(y_t, x_t) \in \mathbb{R}$  of (6). See Figure 1 for an illustration. This structure has the benefit that when making predictions the feature net only needs to be evaluated once, after which the predictor net can be evaluated for a range of  $y_t$  values.

### 3.2 Training the Neural Network

Presented with the data  $\mathcal{D} = \{y_t, x_t\}_{t=1}^T$  and the DNN  $g_\theta(y_t, x_t)$ , it is tempting to consider the ML problem as a means for learning the parameters  $\theta$ . Towards this, we can express the joint likelihood, under the assumption of independence, as

$$p_\theta(y_{1:T} | u_{1:T}) = p_\theta(y_T | y_{1:T-1}, u_{1:T}) p_\theta(y_{1:T-1} | u_{1:T}), \quad (7)$$

where we have used conditional probability to arrive at the expression on the right. Noting the assumed temporal and causal nature of the NARX model, then repeated application of conditional probability delivers

$$p_\theta(y_{1:T} | u_{1:T}) = \prod_{t=1}^T p_\theta(y_t | x_t) = \prod_{t=1}^T \frac{e^{g_\theta(y_t, x_t)}}{\int e^{g_\theta(\gamma, x_t)} d\gamma}. \quad (8)$$

Therefore, the ML estimate for  $\theta$  coincides with

$$\hat{\theta} = \arg \max_{\theta} p_\theta(y_{1:T} | u_{1:T}), \quad (9)$$

$$= \arg \min_{\theta} -\ln p_\theta(y_{1:T} | u_{1:T}), \quad (10)$$

$$= \arg \min_{\theta} \sum_{t=1}^T \left( -g_\theta(y_t, x_t) + \ln \int e^{g_\theta(\gamma, x_t)} d\gamma \right), \quad (11)$$

where the second equality relies on logarithm being a monotonic operator, which implies that the solutions coincide. The third equality is simply the negative logarithm applied to (8). This ML problem is not immediately soluble, due to the analytically intractable integral. Numerical integration can however be employed to obtain an approximate solution, as shown in [3].

Alternative cost functions for fitting the parameters,  $\theta$ , of the distribution  $p_\theta(y_t|x_t)$  given the observed data  $\{y_t, x_t\}_{t=1}^T$  exist. These alternatives were studied in detail for conditional EBMs by [17], recommending noise contrastive estimation (NCE) [18] over ML. We thus employ NCE and learn  $\theta$  by minimizing the cost function  $L(\theta) = -\frac{1}{T} \sum_{t=1}^T L_t(\theta)$ ,

$$L_t(\theta) = \ln \frac{\exp(g_\theta(y_t^{(0)}, x_t) - \ln q(y_t^{(0)}|y_t))}{\sum_{m=0}^M \exp(g_\theta(y_t^{(m)}, x_t) - \ln q(y_t^{(m)}|y_t))}, \quad (12)$$

where  $y_t^{(0)} \triangleq y_t$ , and  $\{y_t^{(m)}\}_{m=1}^M$  are  $M$  noise samples drawn from  $q(y|y_t)$ . This noise distribution is a mixture of  $K$  Gaussians centered at  $y_t$ ,

$$q(y|y_t) = \frac{1}{K} \sum_{k=1}^K \mathcal{N}(y|y_t, \sigma_k^2 I). \quad (13)$$

Since (12) can be interpreted as the cross-entropy loss for a classification problem with  $M + 1$  classes, NCE intuitively entails learning to discriminate between the output  $y_t$  and sampled noise  $\{y_t^{(m)}\}_{m=1}^M$ .

### 3.3 Prediction using the Deep EBM

Rather than giving a point prediction, the proposed deep EB-NARX model predicts  $g_\theta(y_t, x_t) \propto \ln p_\theta(y_t|x_t)$ . There are two ways in which this prediction could be used: if the uncertainty of the prediction is important then we can evaluate  $p_\theta(y_t|x_t)$ ; alternatively, if we only require a point estimate then we could choose the maximum a posterior (MAP) estimate.

The MAP estimate,  $\hat{y}_t$ , can be found by solving

$$\hat{y}_t = \arg \max_{y_t} p_\theta(y_t|x_t) = \arg \max_{y_t} g_\theta(y_t, x_t). \quad (14)$$

Since there is no guarantee that  $p_\theta(y_t|x_t)$  is unimodal, it was found practical to evaluate  $g_\theta(y_t, x_t)$  for a spread of values and then refine the best of these using gradient ascent,  $y_t \leftarrow y_t + \lambda \nabla_{y_t} g_\theta(y_t, x_t)$ .

An estimate of  $p_\theta(y_t|x_t)$  can be determined by evaluating (6) for a range of feasible values of  $y_t$ , where the denominator can be determined by numerical integration, such as Monte Carlo integration.

## 4 Examples

This section provides several examples which illustrate the utility of the EB-NARX model when applied to data from dynamic systems. These examples include both simulated linear and non-linear data, as well as real data from the CE8 coupled electric drives nonlinear data set [19]. For the linear examples, qualitative comparisons are made between the estimated and true distributions. For the non-linear examples, qualitative comparisons are made between a fully connected network (FCN) and EB-NARX estimates of the conditional distributions.

While simple, FCN's obtain highly competitive results in nonlinear system identification benchmarks, even when compared with more sophisticated approaches, such as convolutional and recurrent neural networks, see the benchmarks in [9]. The FCN models are estimated in the functional form (3), nonetheless the conversion to a probabilistic form (1) is straightforward: we use the implicit assumption of Gaussian noise (which is made when minimizing the least square cost function), where the mean is the output of the model and the variance is the sample variance.

Quantitative comparison between the EB-NARX model estimates and the true values are given using the mean squared error (MSE) based on the MAP value from the predicted conditional distribution. Python code for these examples is available at: [github.com/jnh277/ebm\\_arx](https://github.com/jnh277/ebm_arx).

### 4.1 Pedagogical Example

First, the ability of the EB-NARX model to learn different distributions is illustrated. To do this, the method is applied to data generated using a simple autoregressive (AR) model with different distributions for the noise;

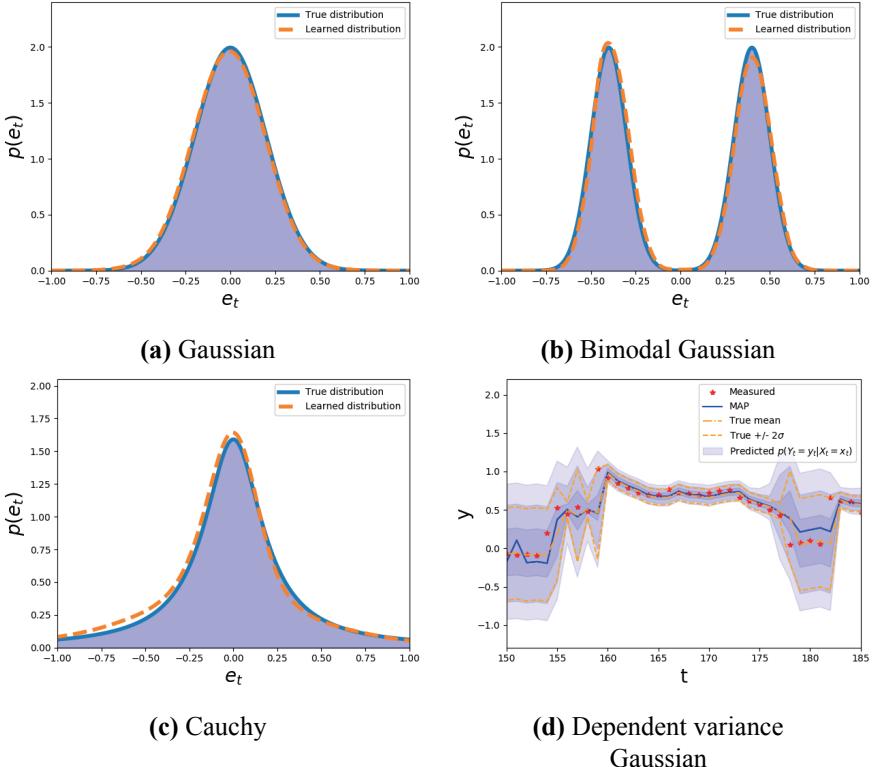
$$y_t = 0.95y_{t-1} + e_t. \quad (15)$$

Four different distributions for the noise  $e_t$  are considered:

- a) zero-mean Gaussian,  $e_t \sim \mathcal{N}(0, 0.2^2)$ ,
- b) bimodal Gaussian,  $e_t \sim 0.5\mathcal{N}(0.4, 0.1^2) + 0.5\mathcal{N}(-0.4, 0.1^2)$ ,
- c) zero-mean Cauchy,  $e_t \sim \mathcal{C}(0, 0.2^2)$ ,
- d) Gaussian with variance dependent on the systems state,

$$e_t \sim \begin{cases} \mathcal{N}(0, 0.3^2) & \text{if } |y_{t-1}| < 0.5 \\ \mathcal{N}(0, 0.05^2) & \text{otherwise.} \end{cases} \quad (16)$$

The learned distributions are shown in Figure 2. While Gaussian noise is often a fair assumption, the utility of a more flexible noise model is made apparent by considering that measurement outliers can be modelled by Student's T



**Figure 2:** Pedagogical example of learning different distributions using a deep EB-NARX model from data generated using a simple AR model (15).

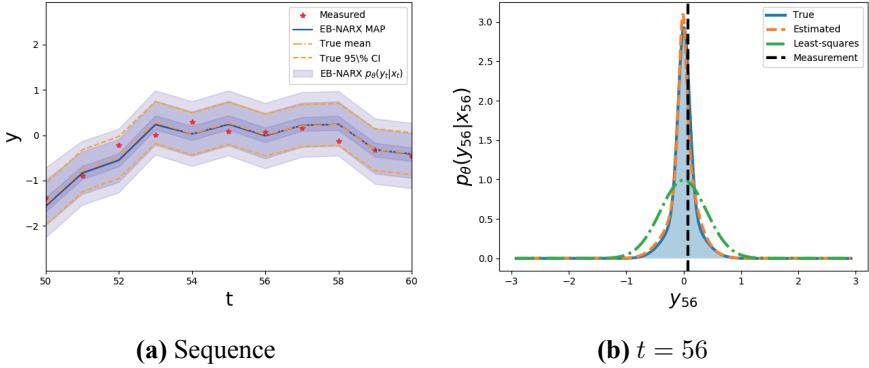
or Cauchy distributions. Moreover, in Section 4.4 the real data gives rise to distribution that is conditional on  $x_t$  and in some cases bimodal.

## 4.2 Linear ARX

To further build confidence in the method’s ability to learn the distribution  $p_\theta(y_t|x_t)$ , it is demonstrated on data generated using a second-order linear autoregressive eXogenous (ARX) model;

$$y_t = 1.5y_{t-1} - 0.7y_{t-2} + u_{t-1} + 0.5u_{t-2} + e_t, \quad (17)$$

where  $e_t \sim 0.6\mathcal{N}(0, 0.1^2) + 0.4\mathcal{N}(0, 0.3^2)$ . An EB-NARX model is trained on 1000 data points and then used to predict the distribution for 200 validation data points. Figure 3a shows part of the predicted sequence along with the true mean and 95% confidence interval (CI). Figure 3b shows the prediction  $p_\theta(y_t|x_t)$  for  $t = 56$  given by the EB-NARX model and an ML estimate given by least-



**Figure 3:** (a) Estimates of  $p_\theta(y_t|x_t)$  for a validation data sequence. The blue shading indicates the 65%, 95%, 99% confidence regions. (b) The EB-NARX and least-squares estimates and true distribution for  $t = 56$ .

squares<sup>1</sup>, compared to the true Gaussian mixture distribution. This illustrates that the EB-NARX model is able to accurately learn the mixture distribution and provide significantly more accurate quantification of the uncertainty than a standard ML approach.

### 4.3 Simulated Nonlinear Problem

So far, the method has been demonstrated on linear problems for which the learned distributions could be easily compared to the true distributions. The method is now applied to data simulated using the nonlinear model [20];

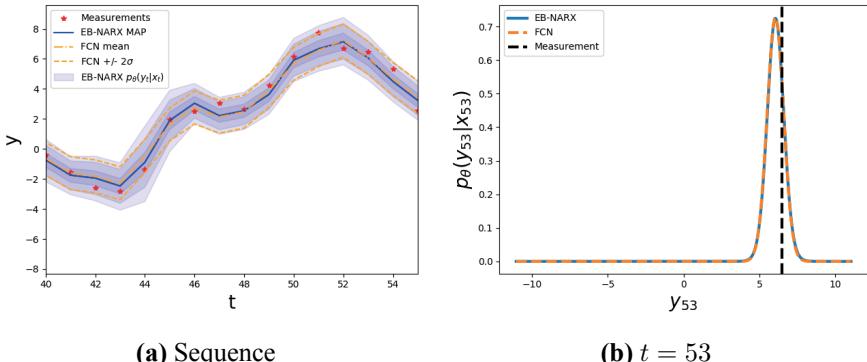
$$\begin{aligned} y_t^* &= \left(0.8 - 0.5e^{-y_{t-1}^{*2}}\right) y_{t-1}^* - \left(0.3 + 0.9e^{-y_{t-1}^{*2}}\right) y_{t-2}^* \\ &\quad + u_{t-1} + 0.2u_{t-2} + 0.1u_{t-1}u_{t-2} + v_t, \\ y_t &= y_t^* + w_t, \end{aligned} \tag{18}$$

where  $v_t \sim \mathcal{N}(0, \sigma_v^2)$  and  $w_t \sim \mathcal{N}(0, \sigma_w^2)$ . Using  $D_u = D_y = 2$ , the performance of the EB-NARX model is compared to that of an FCN for a range of noise standard deviations and training sequence lengths in Table 1. These results indicate that the EB-NARX model performs competitively with the FCN for this data despite making no assumptions about the form of the distribution. An example of the predicted distributions for data generated using  $\sigma_v = \sigma_w = 0.3$  and  $N = 1000$  is shown in Figure 4. Since training the FCN using a squared-error loss function implicitly assumes Gaussian noise, it is, therefore, possible to determine the Gaussian distribution for the estimates

<sup>1</sup>This ML estimate makes an implicit Gaussian assumption.

**Table 1:** Simulated nonlinear MSE on the validation set for the FCN and EB-NARX model trained on datasets generated with different noise levels ( $\sigma_v = \sigma_w = \sigma$ ) and lengths (N). Only the best results are reported from among the different hyperparameters and architectures considered (see Appendix A for details).

	$N = 100$		$N = 250$		$N = 500$	
	FCN	EB-NARX	FCN	EB-NARX	FCN	EB-NARX
$\sigma = 0.1$	0.122	0.099	0.069	0.070	0.057	0.054
$\sigma = 0.3$	0.398	0.390	0.353	0.354	0.289	0.308
$\sigma = 0.5$	0.860	0.869	0.809	0.822	0.754	0.779



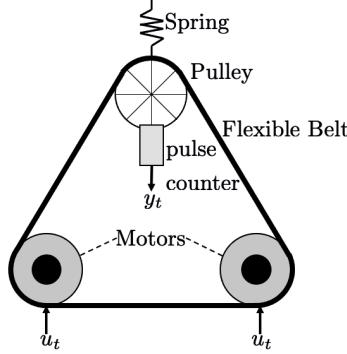
**Figure 4:** Estimates of  $p_\theta(y_t|x_t)$  for a validation data set generated using the nonlinear ARX model presented by [20]. The blue shading indicates the 65%, 95%, 99% confidence regions.

and compare this to the distribution learned using the EB-NARX model. The variance of the FCN distribution has been calculated as the sample variance.

#### 4.4 Real Data: Coupled Electric Drives

We now demonstrate the practical utility of the presented method by application to the CE8 coupled electric drives benchmark data set [19]. The coupled electric drives system, illustrated in Figure 5, consists of two electric motors that drive a pulley using a flexible belt. The pulley is held by a spring and its angular speed is measured by a pulse counter, which is insensitive to the sign of the angular velocity. This creates an ambiguity in the measurements. The input to the system is the signal sent to both motors.

The first three data sets described in [19], which use a random binary input signal, were combined into one set, that was then randomly split 50/50 between training and validation, giving 750 data points in each set. This data was used



**Figure 5:** Illustration of the CE8 coupled electric drives system [19].

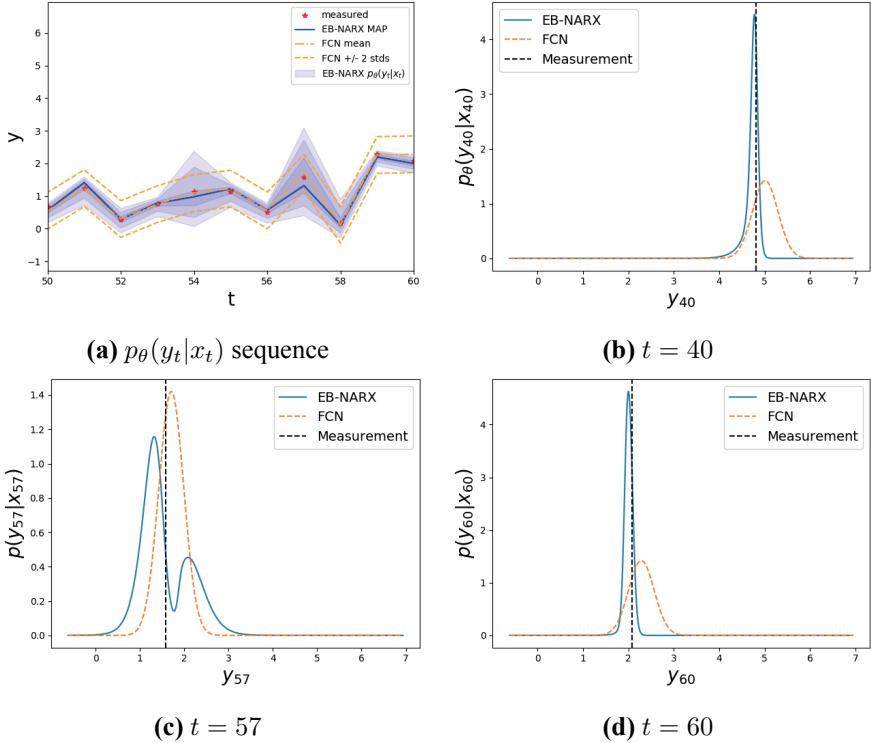
to train an FCN and an EB-NARX model, with the delays  $D_u = D_y = 3$  and the selection of hyperparameters and structure detailed in Appendix A.

The best result for the FCN was an MSE of 0.0521, and for the EB-NARX model an MSE of 0.0503. Figure 6 shows examples of estimates produced using the FCN and EB-NARX models. As in Section 4.3, the sample variance has been used for the Gaussian distribution of the FCN prediction. This variance is constant for all time steps, whereas the EB-NARX model predicts distinctly different and even non-Gaussian distributions at some time steps.

This example demonstrates the flexibility of the EB-NARX model since the magnitude of the angular velocity is measured rather than the angular velocity itself. This produces a sign ambiguity, which has an impact when the velocity crosses zero (there is a reflection in the speed). Intuitively, we expect the measurement distribution to be multi-modal around these points and indeed this intuition is supported by the estimates from the EB-NARX model. In contrast, the sample variance for the FCN predictions does not capture the dependence of the distribution on  $x_t$  and therefore over-estimates the variance away from zero and under-estimates it close to zero.

## 5 Conclusion & Discussion

The salient feature of the EB-NARX model is that it has a highly flexible functional form, which is capable of adapting both to simple and more complex distributions. By contrast, more traditional approaches typically assume a noise distribution that is convenient for learning purposes. While the examples demonstrate that this flexibility is quite useful, it should be noted that the comparisons presented in this work only considered a relatively limited number of data sets, model types, and model structures. As such, a more thorough comparison should be undertaken as future work.



**Figure 6:** Estimates of  $p_\theta(y_t|x_t)$  for a sequence of validation data from the CE8 coupled electric drives benchmark data set [19]. The blue shading indicates the 65%, 95%, 99% confidence regions. The sample variance was used to determine the variance of the FCN assumed Gaussian distribution.

Given that the EB-NARX model is learning the full conditional distribution rather than the point estimate, it might be expected that the performance of the point predictions would suffer when compared to the standard application of an FCN. However, for the particular data sets studied in the nonlinear simulation example, the results in Table 1 indicate that the EB-NARX model approach gives competitive point estimates. Further, when applied to a real data set from the CE8 coupled electric drives system, the EB-NARX model gave point estimates with a lower MSE than the estimates from a standard FCN. This suggests that the EB-NARX model may be a better choice when the conditional distribution depends on the current state of the system.

In this work, the EB-NARX model was composed of two networks; a predictor net and a feature net. This structure is suggested by [3] in the context of regression tasks with high dimensional input spaces, such as images. Hence, it may be less beneficial in the current setting where  $x_t$  is typically of relatively low dimension. The exploration of other structures that may be more suitable in the system identification context is another avenue for future research.

A limitation of the presented work is that it only considers one-step-ahead predictions and not multi-step-ahead predictions or even free-run simulations. Since the EB-NARX model predicts the full conditional distribution yet it takes as inputs point data, it is not clear how these predictions could be propagated forward in time. Whilst it would be possible to propagate the MAP estimate this does remove the main benefit over the standard FCN approach and further has questionable validity if the distribution is multimodal.

Finally, the presented work has only considered NARX systems and an interesting area of future research would be to consider deep EBM's for other types of system identification problems.

**Acknowledgments** This research was supported via the projects *Learning flexible models for nonlinear dynamics* (contract number: 2017-03807) and *NewLEADS – New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) by the Swedish Research Council, by the Brazilian research agency CAPES and by the *Kjell & Märta Beijer Foundation*.

## References

- [1] Y. Du and I. Mordatch. “Implicit Generation and Modeling with Energy Based Models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [2] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, M. Norouzi, and K. Swersky. “Your classifier is secretly an energy based model and you should treat it like one.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [3] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [4] L. Ljung. “System identification.” In: *Wiley encyclopedia of electrical and electronics engineering* (1999).
- [5] Y. LeCun, Y. Bengio, and G. Hinton. “Deep Learning.” In: *Nature* (2015).
- [6] K. S. Narendra and K. Parthasarathy. “Identification and Control of Dynamical Systems Using Neural Networks.” In: *IEEE Transactions on Neural Networks* (1990).
- [7] S. Chen, S. A. Billings, and P. M. Grant. “Non-Linear System Identification Using Neural Networks.” In: *International Journal of Control* (1990).

- [8] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön. “Deep Learning and System Identification.” In: *Proceedings of the IFAC Congress, Berlin*. 2020.
- [9] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön. “Deep Convolutional Networks in System Identification.” In: *Proceedings of the 58th IEEE Conference on Decision and Control (CDC)* (2019).
- [10] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung. “Deep state space models for nonlinear system identification.” In: *Proceedings of the 19th IFAC Symposium on System Identification (SYSID)*. 2021.
- [11] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. “A tutorial on energy-based learning.” In: *Predicting structured data* (2006).
- [12] Y. W. Teh, M. Welling, S. Osindero, and G. E. Hinton. “Energy-based models for sparse overcomplete representations.” In: *Journal of Machine Learning Research* (2003).
- [13] M. Osadchy, M. L. Miller, and Y. L. Cun. “Synergistic face detection and pose estimation with energy-based models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2005.
- [14] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu. “Learning non-convergent non-persistent short-run MCMC toward energy-based model.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [15] Z. Ma and M. Collins. “Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency.” In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.
- [16] M. Danelljan, L. Van Gool, and R. Timofte. “Probabilistic Regression for Visual Tracking.” In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [17] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön. “How to Train Your Energy-Based Model for Regression.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2020.
- [18] M. Gutmann and A. Hyvärinen. “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.” In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010.
- [19] T. Wigren and M. Schoukens. “Coupled electric drives data set and reference models.” In: *Technical Report Uppsala Universitet* (2017).
- [20] S. Chen, S. Billings, and P. Grant. “Non-linear system identification using neural networks.” In: *International journal of control* (1990).

## Appendix

### A Hyper-Parameter and Structure Selection

For each data set, 500 FCN and EBM models were trained covering a range of structures and hyper-parameters. For the FCN, the number of layers ranged from 2 to 4. The dimension of each layer was varied from 50 to 300, and both  $\tanh$  and ReLU activation functions were considered. For the EBM, the feature net was composed of two fully connected layers with ReLU nonlinearities and for the predictor net a neural network with four layers,  $\tanh$  nonlinearities and skip connections. The hidden dimension of both the feature and predictor net was varied from 50 to 300.

For the training of both networks, batch sizes of 32, 64 and 128 were considered and training was carried out until the cost had plateaued. An initial learning rate of 0.001 with a decay rate of 0.99 was used in all cases. A different random seed was used to initialise the parameters each time.



# Paper VI

## Title

Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2020



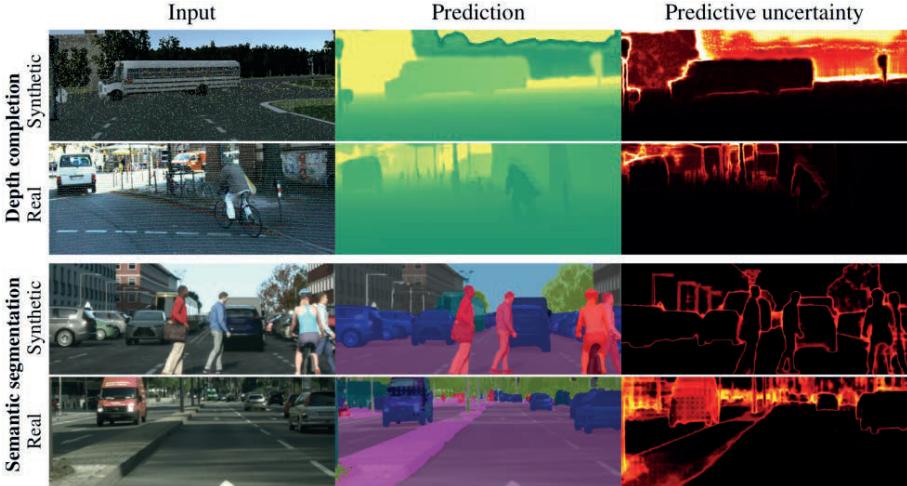
# Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision

## Abstract

While deep neural networks have become the go-to approach in computer vision, the vast majority of these models fail to properly capture the uncertainty inherent in their predictions. Estimating this predictive uncertainty can be crucial, for example in automotive applications. In Bayesian deep learning, predictive uncertainty is commonly decomposed into the distinct types of aleatoric and epistemic uncertainty. The former can be estimated by letting a neural network output the parameters of a certain probability distribution. Epistemic uncertainty estimation is a more challenging problem, and while different scalable methods recently have emerged, no extensive comparison has been performed in a real-world setting. We therefore accept this task and propose a comprehensive evaluation framework for scalable epistemic uncertainty estimation methods in deep learning. Our proposed framework is specifically designed to test the robustness required in real-world computer vision applications. We also apply this framework to provide the first properly extensive and conclusive comparison of the two current state-of-the-art scalable methods: ensembling and MC-dropout. Our comparison demonstrates that ensembling consistently provides more reliable and practically useful uncertainty estimates. Code is available at [https://github.com/fregu856/evaluating\\_bdl](https://github.com/fregu856/evaluating_bdl).

## 1 Introduction

Deep Neural Networks (DNNs) have become the standard paradigm within most computer vision problems due to their astonishing predictive power compared to previous alternatives. Current applications include many safety-critical tasks, such as street-scene semantic segmentation [5, 6, 7], 3D object detection [8, 9] and depth completion [4, 10]. Since erroneous predictions can have



**Figure 1:** We propose a comprehensive evaluation framework for *scalable* epistemic uncertainty estimation methods in deep learning. The proposed framework employs state-of-the-art DNN models on the tasks of depth completion and street-scene semantic segmentation. All models are trained exclusively on synthetic data (the Virtual KITTI [1] and Synscapes [2] datasets). We here show the input (left), prediction (center) and estimated predictive uncertainty (right) for ensembling with  $M = 8$  ensemble members, on both synthetic and real (the KITTI [3, 4] and Cityscapes [5] datasets) example validation images. Black pixels correspond to minimum predictive uncertainty, white pixels to maximum uncertainty.

disastrous consequences, such applications require an accurate measure of the predictive uncertainty. The vast majority of these DNN models do however fail to properly capture the uncertainty inherent in their predictions. They are thus not fully capable of the type of *uncertainty-aware* reasoning that is highly desired e.g. in automotive applications.

The approach of *Bayesian deep learning* aims to address this issue in a principled manner. Here, predictive uncertainty is commonly decomposed into two distinct types, which both should be captured by the learned DNN [11, 12]. *Epistemic* uncertainty accounts for uncertainty in the DNN model parameters, while *aleatoric* uncertainty captures inherent and irreducible data noise. Input-dependent *aleatoric* uncertainty about the target  $y$  arises due to e.g. noise and ambiguities inherent in the input  $x$ . This is present for instance in street-scene semantic segmentation, where image pixels at object boundaries are inherently ambiguous, and in 3D object detection where the location of a distant object is less certain due to noise and limited sensor resolution. In many computer vision applications, this aleatoric uncertainty can be effectively estimated by letting a DNN directly output the parameters of a certain probability distribution, modeling the conditional distribution  $p(y|x)$  of the target given the input. For

classification tasks, a predictive categorical distribution is commonly realized by a softmax output layer, although recent work has also explored Dirichlet models [13, 14, 15]. For regression, Laplace and Gaussian models have been employed [16, 17, 12, 18].

Directly predicting the conditional distribution  $p(y|x)$  with a DNN does however not capture *epistemic* uncertainty, as information about the uncertainty in the model parameters is disregarded. This often leads to highly confident predictions that are incorrect, especially for inputs  $x$  that are not well-represented by the training distribution [19, 18]. For instance, a DNN can fail to generalize to unfamiliar weather conditions or environments in automotive applications, but still generate confident predictions. Reliable estimation of epistemic uncertainty is thus of great importance. However, this task has proven to be highly challenging, largely due to the vast dimensionality of the parameter space, which renders standard Bayesian inference approaches intractable. To tackle this problem, a wide variety of approximations have been explored [20, 21, 22, 23, 24, 25, 26], but only a small number have been demonstrated to be applicable even to the large-scale DNN models commonly employed in *real-world* computer vision tasks. Among such *scalable* methods, MC-dropout [11, 12, 27, 28] and ensembling [18, 17, 16] are clearly the most widely employed, due to their demonstrated effectiveness and simplicity. While *scalable* techniques for epistemic uncertainty estimation recently have emerged, the research community however lacks a common and comprehensive evaluation framework for such methods. Consequently, both researchers and practitioners are currently unable to properly assess and compare newly proposed methods. In this work, we therefore accept this task and set out to design exactly such an evaluation framework, aiming to benefit and inspire future research in the field.

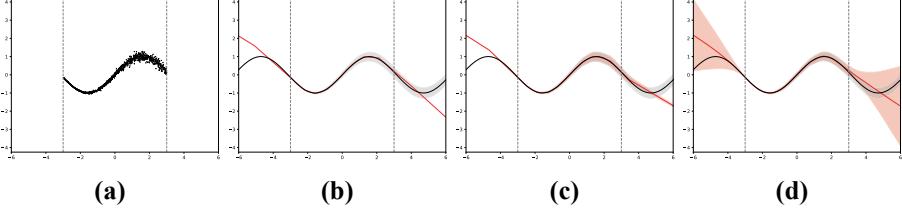
Previous studies have provided only partial insight into the performance of different *scalable* methods for epistemic uncertainty estimation. Kendall and Gal [12] evaluated MC-dropout alone on the tasks of semantic segmentation and monocular depth regression, providing mainly qualitative results. Lakshminarayanan et al. [18] introduced ensembling as a non-Bayesian alternative and found it to generally outperform MC-dropout. Their experiments were however based on relatively small-scale models and datasets, limiting the real-world applicability. Ilg et al. [16] compared ensembling and MC-dropout on the task of optical-flow estimation, but only in terms of the AUSE metric which is a *relative* measure of the uncertainty estimation quality. While finding ensembling to be advantageous, their experiments were also limited to a fixed number ( $M = 8$ ) of ensemble members and MC-dropout forward passes, not allowing a completely fair comparison. Ovadia et al. [29] also fixed the number of ensemble members, and moreover only considered classification tasks. We improve upon this previous work and propose an evaluation framework that actually enables a conclusive ranking of the compared methods.

**Contributions** We propose a comprehensive evaluation framework for *scalable* epistemic uncertainty estimation methods in deep learning. The proposed framework is specifically designed to test the robustness required in *real-world* computer vision applications, and employs state-of-the-art DNN models on the tasks of depth completion (regression) and street-scene semantic segmentation (classification). It also employs a novel combination of quantitative evaluation metrics which explicitly measures the reliability and practical usefulness of estimated predictive uncertainties. We apply our proposed framework to provide the first properly extensive and conclusive comparison of the two current state-of-the-art *scalable* methods: ensembling and MC-dropout. This comparison demonstrates that ensembling consistently outperforms the highly popular MC-dropout method. Our work thus suggests that ensembling should be considered the new go-to approach, and encourages future research to understand and further improve its efficacy. Figure 1 shows example predictive uncertainty estimates generated by ensembling. Our framework can also directly be applied to compare other *scalable* methods, and we encourage external usage with publicly available code.

In our proposed framework, we predict the conditional distribution  $p(y|x)$  in order to estimate input-dependent aleatoric uncertainty. The methods for epistemic uncertainty estimation are then compared by quantitatively evaluating the estimated predictive uncertainty in terms of the relative AUSE metric and the *absolute* measure of uncertainty calibration. Our evaluation is the first to include *both* these metrics, and furthermore we apply them to both regression and classification tasks. To provide a deeper and more fair analysis, we also study all metrics as functions of the number of samples  $M$ , enabling a highly informative comparison of the rate of improvement. Moreover, we simulate challenging real-world conditions found e.g. in automotive applications, where robustness to out-of-domain inputs is required to ensure safety, by training our models exclusively on synthetic data and evaluating the predictive uncertainty on real-world data. By analyzing this important domain shift problem, we significantly increase the practical applicability of our evaluation. We also complement our real-world analysis with experiments on illustrative toy regression and classification problems. Lastly, to demonstrate the evaluation rigor necessary to achieve a conclusive comparison, we repeat each experiment multiple times and report results together with the observed variation.

## 2 Predictive Uncertainty Estimation using Bayesian Deep Learning

DNNs have been shown to excel at a wide variety of supervised machine learning problems, where the task is to predict a target value  $y \in \mathcal{Y}$  given an input



**Figure 2:** Toy regression problem illustrating the task of predictive uncertainty estimation with DNNs. The true data generator  $p(y|x)$  is a Gaussian, where the mean is given by the solid black line and the variance is represented in shaded gray. The predictive mean and variance are given by the solid red line and the shaded red area, respectively. **(a)** Training dataset with  $N = 1\,000$  examples. **(b)** A DNN trained to directly predict the target  $y$  captures no notion of uncertainty. **(c)** A corresponding Gaussian DNN model (2) trained via maximum-likelihood captures *aleatoric* but not epistemic uncertainty. **(d)** The Gaussian model instead trained via approximate Bayesian inference (4) captures both *aleatoric* and *epistemic* uncertainty.

$x \in \mathcal{X}$ . In computer vision, the input space  $\mathcal{X}$  often corresponds to the space of images. For classification problems, the target space  $\mathcal{Y}$  consists of a finite set of  $C$  classes, while a regression problem is characterized by a continuous target space, e.g.  $\mathcal{Y} = \mathbb{R}^K$ . For our purpose, a DNN is defined as a function  $f_\theta : \mathcal{X} \rightarrow \mathcal{U}$ , parameterized by  $\theta \in \mathbb{R}^P$ , that maps an input  $x \in \mathcal{X}$  to an output  $f_\theta(x) \in \mathcal{U}$ . Next, we cover alternatives for estimating both the aleatoric and epistemic uncertainty in the predictions of DNN models.

**Aleatoric Uncertainty** In classification problems, aleatoric uncertainty is commonly captured by predicting a categorical distribution  $p(y|x, \theta)$ . This is implemented by letting the DNN predict logit scores  $f_\theta(x) \in \mathbb{R}^C$ , which are then normalized by a Softmax function,

$$\begin{aligned} p(y|x, \theta) &= \text{Cat}(y; s_\theta(x)), \\ s_\theta(x) &= \text{Softmax}(f_\theta(x)). \end{aligned} \quad (1)$$

Given a training set of  $N$  i.i.d. sample pairs  $\mathcal{D} = \{X, Y\} = \{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ , the data likelihood is obtained as  $p(Y|X, \theta) = \prod_{i=1}^N p(y_i|x_i, \theta)$ . The maximum-likelihood estimate of the model parameters,  $\hat{\theta}_{\text{MLE}}$ , is obtained by minimizing the negative log-likelihood  $-\sum_i \log p(y_i|x_i, \theta)$ . For the Categorical model (1), this is equivalent to minimizing the well-known cross-entropy loss. At test time, the trained model predicts the distribution  $p(y^*|x^*, \hat{\theta}_{\text{MLE}})$  over the target class variable  $y^*$ , given a test input  $x^*$ . These DNN models are thus able to capture input-dependent aleatoric uncertainty, by outputting less confident predictions for inherently ambiguous cases.

In regression, the most common approach is to let the DNN directly predict targets,  $y^* = f_\theta(x^*)$ . The parameters  $\theta$  are learned by minimizing e.g. the  $L^2$

or  $L^1$  loss over the training dataset [8, 9]. However, such direct regression does not model aleatoric uncertainty. Instead, recent work [16, 12, 18] has explored predicting the distribution  $p(y|x, \theta)$ , similar to the classification case above. For instance,  $p(y|x, \theta)$  can be parameterized by a Gaussian distribution [17, 18], giving the following model in the 1D case,

$$\begin{aligned} p(y|x, \theta) &= \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x)), \\ f_\theta(x) &= [\mu_\theta(x) \quad \log \sigma_\theta^2(x)]^\top \in \mathbb{R}^2. \end{aligned} \quad (2)$$

Here, the DNN predicts the mean  $\mu_\theta(x)$  and variance  $\sigma_\theta^2(x)$  of the target  $y$ . The variance is naturally interpreted as a measure of input-dependent aleatoric uncertainty. As in classification, the model parameters  $\theta$  are learned by minimizing the negative log-likelihood  $-\sum_i \log p(y_i|x_i, \theta)$ .

**Epistemic Uncertainty** While the above models can capture aleatoric uncertainty, stemming from the data, they are agnostic to the uncertainty in the model parameters  $\theta$ . A principled means to estimate this epistemic uncertainty is to perform Bayesian inference. The aim is to utilize the posterior distribution  $p(\theta|\mathcal{D})$ , which is obtained from the data likelihood and a chosen prior  $p(\theta)$  by applying Bayes’ theorem. The uncertainty in the parameters  $\theta$  is then marginalized out to obtain the predictive posterior distribution,

$$\begin{aligned} p(y^*|x^*, \mathcal{D}) &= \int p(y^*|x^*, \theta)p(\theta|\mathcal{D})d\theta \\ &\approx \frac{1}{M} \sum_{i=1}^M p(y^*|x^*, \theta^{(i)}), \quad \theta^{(i)} \sim p(\theta|\mathcal{D}). \end{aligned} \quad (3)$$

Here, the generally intractable integral in (3) is approximated using  $M$  Monte Carlo samples  $\theta^{(i)}$ , ideally drawn from the posterior. In practice however, obtaining samples from the true posterior  $p(\theta|\mathcal{D})$  is virtually impossible, requiring an approximate posterior  $q(\theta) \approx p(\theta|\mathcal{D})$  to be used. We thus obtain the approximate predictive posterior as,

$$\hat{p}(y^*|x^*, \mathcal{D}) \triangleq \frac{1}{M} \sum_{i=1}^M p(y^*|x^*, \theta^{(i)}), \quad \theta^{(i)} \sim q(\theta), \quad (4)$$

which enables us to estimate both aleatoric and epistemic uncertainty of the prediction. The quality of the approximation (4) depends on the number of samples  $M$  and the method employed for generating  $q(\theta)$ . Prior work on such approximate Bayesian inference methods is discussed in Section 3. For the Categorical model (1),  $\hat{p}(y^*|x^*, \mathcal{D}) = \text{Cat}(y^*; \hat{s}(x^*))$ ,  $\hat{s}(x^*) = \frac{1}{M} \sum_{i=1}^M s_{\theta^{(i)}}(x^*)$ . For the Gaussian model (2),  $\hat{p}(y^*|x^*, \mathcal{D})$  is a uniformly weighted mixture of Gaussian distributions. We approximate this mixture with a single Gaussian, see Appendix A for details.

**Illustrative Example** To visualize and provide intuition for the problem of predictive uncertainty estimation with DNNs, we consider the problem of regressing a sinusoid corrupted by *input-dependent* Gaussian noise,

$$\begin{aligned} y &\sim \mathcal{N}(\mu(x), \sigma^2(x)), \\ \mu(x) &= \sin(x), \quad \sigma(x) = 0.15(1 + e^{-x})^{-1}. \end{aligned} \tag{5}$$

Training data  $\{(x_i, y_i)\}_{i=1}^{1000}$  is only given in the interval  $[-3, 3]$ , see Figure 2a. A DNN trained to directly predict the target  $y$  is able to accurately regress the mean for  $x^* \in [-3, 3]$ , see Figure 2b. However, this model does not capture any notion of uncertainty. A corresponding Gaussian DNN model (2) trained via maximum-likelihood obtains a predictive distribution that closely matches the ground truth for  $x^* \in [-3, 3]$ , see Figure 2c. While correctly accounting for aleatoric uncertainty, this model generates overly confident predictions for inputs  $|x^*| > 3$  not seen during training. Finally, the Gaussian DNN model trained via approximate Bayesian inference (4), with a prior distribution  $p(\theta) = \mathcal{N}(0, I_P)$  and  $M = 1\,000$  samples obtained via Hamiltonian Monte Carlo [30], is additionally able to predict more reasonable uncertainties in the region with no available training data, see Figure 2d.

### 3 Related Work

Here, we discuss prior work on approximate Bayesian inference. We also note that ensembling, which is often considered a non-Bayesian alternative, in fact can naturally be viewed as an approximate Bayesian inference method.

**Approximate Bayesian Inference** The method employed for approximating the posterior  $q(\theta) \approx p(\theta|\mathcal{D}) = p(Y|X, \theta)p(\theta)/p(Y|X)$  is a crucial choice, determining the quality of the approximate predictive posterior  $\hat{p}(y^*|x^*, \mathcal{D})$  in (4). There exists two main paradigms for constructing  $q(\theta)$ , the first one being *Markov chain Monte Carlo (MCMC)* methods. Here, samples  $\theta^{(i)}$  approximately distributed according to the posterior are obtained by simulating a Markov chain with  $p(\theta|\mathcal{D})$  as its stationary distribution. For DNNs, this approach was pioneered by Neal [20], who employed Hamiltonian Monte Carlo (HMC) on small feed-forward neural networks. HMC entails performing Metropolis-Hastings [31, 32] updates using Hamiltonian dynamics based on the potential energy  $U(\theta) \triangleq -\log p(Y|X, \theta)p(\theta)$ . To date, it is considered a “gold standard” method for approximate Bayesian inference, but does not scale to large DNNs or large-scale datasets. Therefore, *Stochastic Gradient MCMC (SG-MCMC)* [33] methods have been explored, in which stochastic gradients are utilized in place of their full-data counterparts. SG-MCMC variants include Stochastic Gradient Langevin Dynamics (SGLD) [23], where samples  $\theta^{(i)}$  are collected from the parameter trajectory given by the update

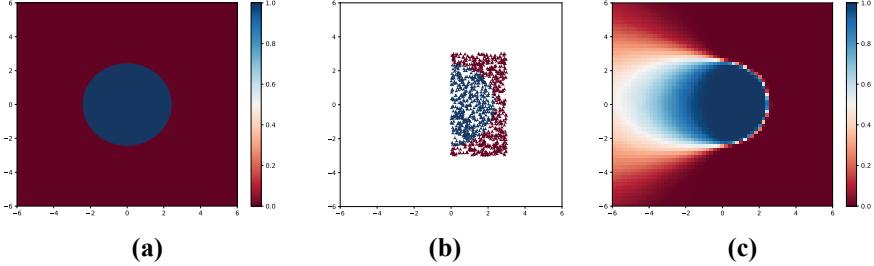
equation  $\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \tilde{U}(\theta_t) + \sqrt{2\alpha_t} \epsilon_t$ , where  $\epsilon_t \sim \mathcal{N}(0, 1)$  and  $\nabla_\theta \tilde{U}(\theta)$  is the stochastic gradient of  $U(\theta)$ . Save for the noise term  $\sqrt{2\alpha_t} \epsilon_t$ , this update is identical to the conventional SGD update when minimizing the maximum-a-posteriori (MAP) objective  $-\log p(Y|X, \theta)p(\theta)$ . Similarly, Stochastic Gradient HMC (SGHMC) [24] corresponds to SGD with momentum injected with properly scaled noise. Given a limited computational budget, SG-MCMC methods can however struggle to explore the high-dimensional and highly multi-modal posteriors of large DNNs. To mitigate this problem, Zhang et al. [34] proposed to use a cyclical stepsize schedule to help escaping local modes in  $p(\theta|\mathcal{D})$ .

The second paradigm is that of *Variational Inference (VI)* [21, 35, 36, 22]. Here, a distribution  $q_\phi(\theta)$  parameterized by variational parameters  $\phi$  is explicitly chosen, and the best possible approximation is found by minimizing the Kullback-Leibler (KL) divergence with respect to the true posterior  $p(\theta|\mathcal{D})$ . While principled, VI methods generally require sophisticated implementations, especially for more expressive variational distributions  $q_\phi(\theta)$  [37, 38, 39]. A particularly simple and scalable method is MC-dropout [40]. It entails using dropout [41] also at test time, which can be interpreted as performing VI with a Bernoulli variational distribution [40, 27, 28]. The approximate predictive posterior  $\hat{p}(y^*|x^*, \mathcal{D})$  in (4) is obtained by performing  $M$  stochastic forward passes on the same input.

**Ensembling** Lakshminarayanan et al. [18] created a parametric model  $p(y|x, \theta)$  of the conditional distribution using a DNN  $f_\theta$ , and learned multiple point estimates  $\{\hat{\theta}^{(m)}\}_{m=1}^M$  by repeatedly minimizing the MLE objective  $-\log p(Y|X, \theta)$  with random initialization. They then averaged over the corresponding parametric models to obtain the following predictive distribution,

$$\hat{p}(y^*|x^*) \triangleq \frac{1}{M} \sum_{m=1}^M p(y^*|x^*, \hat{\theta}^{(m)}). \quad (6)$$

The authors considered this a non-Bayesian alternative to predictive uncertainty estimation. However, since the point estimates  $\{\hat{\theta}^{(m)}\}_{m=1}^M$  always can be seen as samples from some distribution  $\hat{q}(\theta)$ , we note that (6) is virtually identical to the approximate predictive posterior in (4). Ensembling can thus naturally be viewed as approximate Bayesian inference, where the level of approximation is determined by how well the implicit sampling distribution  $\hat{q}(\theta)$  approximates the posterior  $p(\theta|\mathcal{D})$ . Ideally,  $\{\hat{\theta}^{(m)}\}_{m=1}^M$  should be distributed exactly according to  $p(\theta|\mathcal{D}) \propto p(Y|X, \theta)p(\theta)$ . Since  $p(Y|X, \theta)$  is highly *multi-modal* in the parameter space for DNNs [42, 43], so is  $p(\theta|\mathcal{D})$ . By minimizing  $-\log p(Y|X, \theta)$  multiple times, starting from *randomly chosen* initial points, we are likely to find different local optima. Ensembling can thus generate a compact set of samples  $\{\hat{\theta}^{(m)}\}_{m=1}^M$  that, even for small values of  $M$ , captures this important aspect of multi-modality in  $p(\theta|\mathcal{D})$ .



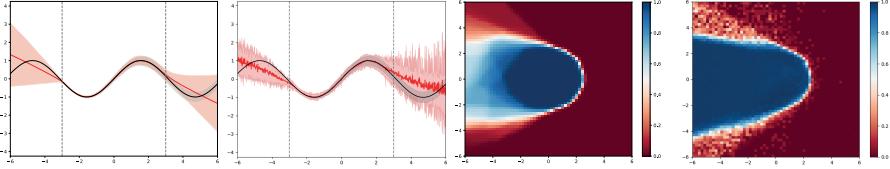
**Figure 3:** Toy binary classification problem. **(a)** True data generator, red and blue represents the two classes. **(b)** Training dataset with  $N = 1\,040$  examples. **(c)** ‘Ground truth’ predictive distribution, obtained using HMC [30].

## 4 Experiments

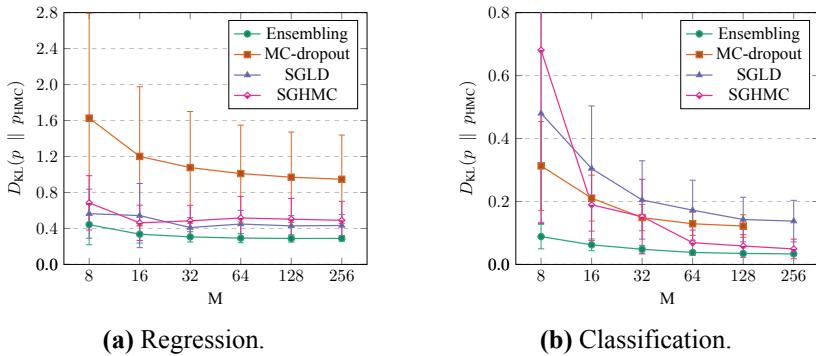
We conduct experiments both on illustrative toy regression and classification problems (Section 4.1), and on the real-world computer vision tasks of depth completion (Section 4.2) and street-scene semantic segmentation (Section 4.3). Our evaluation is motivated by real-world conditions found e.g. in automotive applications, where robustness to varying environments and weather conditions is required to ensure safety. Since images captured in these different circumstances could all represent distinctly different regions of the vast input image space, it is infeasible to ensure that all encountered inputs will be well-represented by the training data. Thus, we argue that robustness to out-of-domain inputs is crucial in such applications. To simulate these challenging conditions and test the robustness required for such real-world scenarios, we train all models on synthetic data and evaluate them on real-world data. To improve rigour of our evaluation, we repeat each experiment multiple times and report results together with the observed variation. A more detailed description of all results are found in the Appendix (Appendix B.3, C.2, D.2). All experiments are implemented in PyTorch [44].

### 4.1 Illustrative Toy Problems

We first present results on illustrative toy problems to gain insights into how ensembling and MC-dropout fare against other approximate Bayesian inference methods. For regression, we conduct experiments on the 1D problem defined in (5) and visualized in Figure 2. We use the Gaussian model (2) with two separate feed-forward neural networks outputting  $\mu_\theta(x)$  and  $\log \sigma_\theta^2(x)$ . We evaluate the methods by quantitatively measuring how well the obtained predictive distributions approximate that of the “gold standard” HMC [30] with  $M = 1\,000$  samples and prior  $p(\theta) = \mathcal{N}(0, I_P)$ . We thus consider the predictive distribution visualized in Figure 2d ground truth, and take as our metric



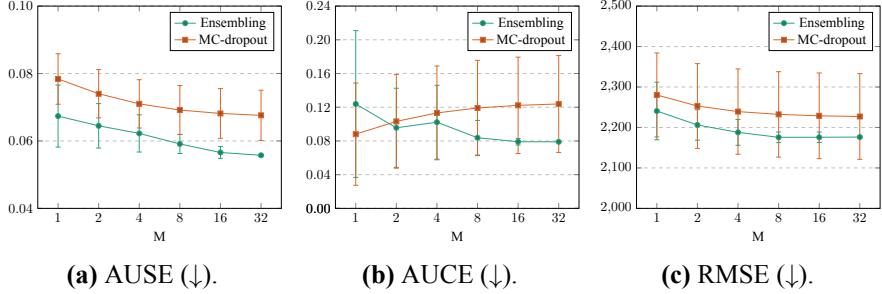
**Figure 4:** Illustrative toy problems - example predictive distributions for ensembling and MC-dropout with  $M = 16$  samples.



**Figure 5:** Illustrative toy problems - quantitative results. The plots show the KL divergence ( $\downarrow$ ) between the predictive distribution estimated by each method and the HMC “ground truth”, for different number of samples  $M$ .

the KL divergence  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  with respect to this target distribution  $p_{\text{HMC}}$ . For classification, we conduct experiments on the binary classification problem in Figure 3. The true data generator is visualized in Figure 3a, where red and blue represents the two classes. The training dataset contains 520 examples of each class, and is visualized in Figure 3b. We use the Categorical model (1) with a feed-forward neural network. As for regression, we quantitatively measure how well the obtained predictive distributions approximate that of HMC, which is visualized in Figure 3c. Further details are provided in Appendix B.

**Results** A comparison of ensembling, MC-dropout, SGLD and SGHMC in terms of  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  is found in Figure 5. The Adam optimizer [45] is here used for both ensembling and MC-dropout. We observe that ensembling consistently outperforms the compared methods, and MC-dropout in particular. Even compared to SG-MCMC variants such as SGLD and SGHMC, ensembling thus provides a better approximation to the MCMC method HMC. This result is qualitatively supported by visualized predictive distributions found in Appendix B.5. Example predictive distributions for ensembling and MC-dropout with  $M = 16$  are shown in Figure 4. We observe that ensembling

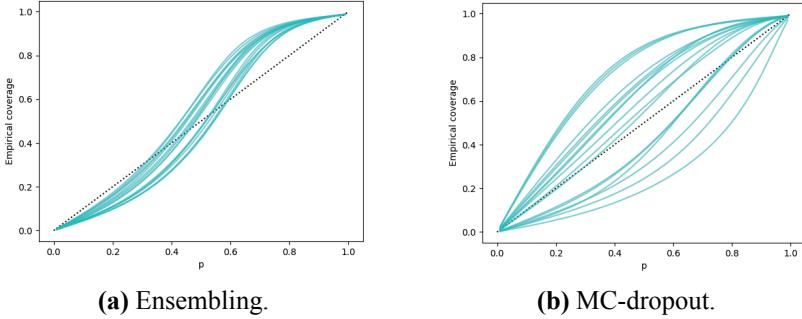


**Figure 6:** Depth completion - quantitative results. The plots show a comparison of ensembling and MC-dropout in terms of AUSE, AUCE and RMSE on the KITTI depth completion validation dataset, for different number of samples  $M$ .

provides reasonable approximations to HMC even for quite small values of  $M$ , especially compared to MC-dropout.

## 4.2 Depth Completion

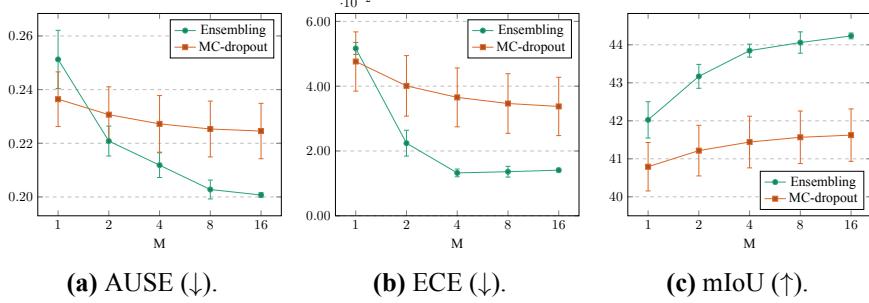
In depth completion, we are given an image  $x_{\text{img}} \in \mathbb{R}^{h \times w \times 3}$  from a forward-facing camera and an associated *sparse* depth map  $x_{\text{sparse}} \in \mathbb{R}^{h \times w}$ . Only non-zero pixels of  $x_{\text{sparse}}$  correspond to LiDAR depth measurements, projected onto the image plane. The goal is to predict a dense depth map  $y \in \mathbb{R}^{h \times w}$  of the scene. We utilize the KITTI depth completion [3, 4] and Virtual KITTI [1] datasets. KITTI depth completion contains more than 80 000 images  $x_{\text{img}}$ , sparse depth maps  $x_{\text{sparse}}$  and semi-dense target maps  $y$ . There are 1 000 selected validation examples, which we use for evaluation. Only about 4% of the pixels in  $x_{\text{sparse}}$  are non-zero and thus correspond to depth measurements. The semi-dense target maps are created by merging the LiDAR scans from 11 consecutive frames into one, producing  $y$  in which roughly 30% of the pixels are non-zero. Virtual KITTI contains synthetic images  $x_{\text{img}}$  and dense depth maps  $x_{\text{dense}}$  extracted from 5 driving sequences in a virtual world. It contains 2 126 unique frames, of which there are 10 different versions corresponding to various simulated weather and lighting conditions. We take sequence 0002 as our validation set, leaving a total of 18 930 training examples. We create targets  $y$  for training by setting all pixels in  $x_{\text{dense}}$  corresponding to a depth  $> 80\text{m}$  to 0, and then also randomly sample 5% of the remaining non-zero pixels uniformly to create  $x_{\text{sparse}}$ . We use the DNN model presented by Ma et al. [10]. The inputs  $x_{\text{img}}, x_{\text{sparse}}$  are separately processed by initial convolutional layers, concatenated and fed to an encoder-decoder architecture based on ResNet34 [46]. We employ the Gaussian model (2) by duplicating the final layer, outputting both  $\mu \in \mathbb{R}^{h \times w}$  and  $\log \sigma^2 \in \mathbb{R}^{h \times w}$  instead of only the predicted depth map  $\hat{y} \in \mathbb{R}^{h \times w}$ . We also employ the same basic training pro-



**Figure 7:** Depth completion - condensed calibration plots for ensembling and MC-dropout with  $M = 16$ .

cedure as Ma et al. [10] to train all our models, see Appendix C.1 for details. For the MC-dropout comparison, we take inspiration from Kendall et al. [27] and place a dropout layer with drop probability  $p = 0.5$  after the three last encoder blocks and the four first decoder blocks.

**Evaluation Metrics** We evaluate the methods in terms of the *Area Under the Sparsification Error curve (AUSE)* metric, as introduced by Ilg et al. [16]. AUSE is a *relative* measure of the uncertainty estimation quality, comparing the ordering of predictions induced by the estimated predictive uncertainty (sorted from least to most uncertain) with the “oracle” ordering in terms of the true prediction error. The metric thus reveals how well the estimated uncertainty can be used to sort predictions from worst (large true prediction error) to best (small prediction error). We compute AUSE in terms of *Root Mean Squared Error (RMSE)* and based on all pixels in the entire evaluation dataset. A perfect AUSE score can however be achieved even if the true predictive uncertainty is consistently underestimated. As an *absolute* measure of uncertainty estimation quality, we therefore also evaluate the methods in terms of calibration [47, 48]. In classification, the *Expected Calibration Error (ECE)* [19, 49] is a standard metric used to evaluate calibration. A well-calibrated model should then produce classification confidences which match the observed prediction accuracy, meaning that the model is not over-confident (outputting highly confident predictions which are incorrect), nor over-conservative. We here employ a metric that can be considered a natural generalization of ECE to the regression setting. Since our models output the mean  $\mu \in \mathbb{R}$  and variance  $\sigma^2 \in \mathbb{R}$  of a Gaussian distribution for each pixel, we can construct pixel-wise prediction intervals  $\mu \pm \Phi^{-1}(\frac{p+1}{2})\sigma$  of confidence level  $p \in ]0, 1[$ , where  $\Phi$  is the CDF of the standard normal distribution. When computing the proportion of pixels for which the prediction interval covers the true target  $y \in \mathbb{R}$ , we expect this value, denoted  $\hat{p}$ , to equal  $p \in ]0, 1[$  for a perfectly calibrated model. We compute the absolute error with respect to perfect calibration,  $|p - \hat{p}|$ , for 100 values of  $p \in ]0, 1[$  and use the area under this curve as our metric, which



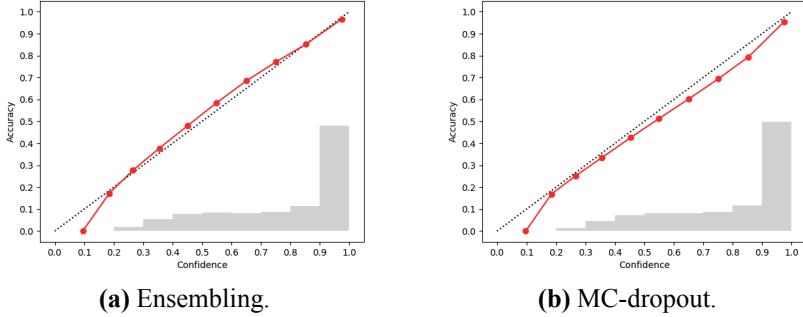
**Figure 8:** Street-scene semantic segmentation - quantitative results. The plots show a comparison of ensembling and MC-dropout in terms of AUSE, ECE and mIoU on the Cityscapes validation dataset, for different number of samples  $M$ .

we call *Area Under the Calibration Error curve (AUCE)*. Lastly, we also evaluate in terms of the standard RMSE metric.

**Results** A comparison of ensembling and MC-dropout in terms of AUSE, AUCE and RMSE on the KITTI depth completion validation dataset is found in Figure 6. We observe in Figure 6a that ensembling consistently outperforms MC-dropout in terms of AUSE. However, the curves decrease as a function of  $M$  in a similar manner. Sparsification plots and sparsification error curves are found in Appendix C.3. A ranking of the methods can be more readily conducted based on Figure 6b, where we observe a clearly improving trend as  $M$  increases for ensembling, whereas MC-dropout gets progressively worse. This result is qualitatively supported by the calibration plots found in Appendix C.3 and Figure 7. Note that  $M = 1$  corresponds to the baseline of only estimating aleatoric uncertainty.

### 4.3 Street-Scene Semantic Segmentation

In this task, we are given an image  $x \in \mathbb{R}^{h \times w \times 3}$  from a forward-facing camera. The goal is to predict  $y$  of size  $h \times w$ , in which each pixel is assigned to one of  $C$  different class labels (road, sidewalk, car, etc.). We utilize the popular Cityscapes [5] and recent Synscapes [2] datasets. Cityscapes contains 5 000 finely annotated images, mainly collected in various German cities. The annotations includes 30 class labels, but only  $C = 19$  are used in the training of models. Its validation set contains 500 examples, which we use for evaluation. Synscapes contains 25 000 synthetic images, all captured in virtual urban environments. To match the size of Cityscapes, we randomly select 2 975 of these for training and 500 for validation. The images are annotated with the same class labels as Cityscapes. We use the DeepLabv3 DNN model presented by Chen et al. [6]. The input image  $x$  is processed by a ResNet101 [46], out-



**Figure 9:** Street-scene semantic segmentation - example reliability diagrams for the two methods with  $M = 16$ .

putting a feature map of stride 8. The feature map is further processed by an ASPP module and a  $1 \times 1$  convolutional layer, outputting logits at 1/8 of the original resolution. These are then upsampled to image resolution using bilinear interpolation. The conventional Categorical model (1) is thus used for each pixel. We base our implementation on the one by Yuan and Wang [7], and also follow the same basic training procedure, see Appendix D.1 for details. For reference, the model obtains an mIoU [50] of 76.04% when trained on Cityscapes and evaluated on its validation set. For the MC-dropout comparison, we take inspiration from Mukhoti and Gal [28] and place a dropout layer with  $p = 0.5$  after the four last ResNet blocks.

**Evaluation Metrics** As for depth completion, we evaluate the methods in terms of the AUSE metric. In this classification setting, we compare the “oracle” ordering of predictions with the one induced by the predictive entropy. We compute AUSE in terms of Brier score and based on all pixels in the evaluation dataset. We also evaluate in terms of calibration by the ECE metric [19, 49]. All predictions are here partitioned into  $L$  bins based on the maximum assigned confidence. For each bin, the difference between the average predicted confidence and the actual accuracy is then computed, and ECE is obtained as the weighted average of these differences. We use  $L = 10$  bins of equal size.

**Results** A comparison of ensembling and MC-dropout in terms of AUSE, ECE and mIoU on the Cityscapes validation dataset is found in Figure 8. We observe that the metrics clearly improve as functions of  $M$  for both ensembling and MC-dropout, demonstrating the importance of epistemic uncertainty estimation. The rate of improvement is generally greater for ensembling. For ECE, we observe in Figure 8b a drastic improvement for ensembling as  $M$  is increased, followed by a distinct plateau. According to the condensed reliability diagrams in Appendix D.3, this corresponds to a transition from clear model over-confidence to slight over-conservatism. For MC-dropout, the corresponding diagrams suggest a stagnation while the model still is somewhat

over-confident. Example reliability diagrams for  $M = 16$  are shown in Figure 9, in which this over-confidence for MC-dropout can be observed. Note that the relatively low mIoU scores reported in Figure 8c, obtained by models trained exclusively on Synscapes, are expected [2] and caused by the intentionally challenging domain gap between synthetic and real-world data.

## 5 Discussion & Conclusion

We proposed a comprehensive evaluation framework for *scalable* epistemic uncertainty estimation methods in deep learning. The proposed framework is specifically designed to test the robustness required in real-world computer vision applications. We applied our proposed framework and provided the first properly extensive and conclusive comparison of ensembling and MC-dropout, the results of which demonstrates that ensembling consistently provides more reliable and practically useful uncertainty estimates. We attribute the success of ensembling to its ability, due to the random initialization, to capture the important aspect of multi-modality present in the posterior distribution  $p(\theta|\mathcal{D})$  of DNNs. MC-dropout has a large design-space compared to ensembling, and while careful tuning of MC-dropout potentially could close the performance gap on individual tasks, the simplicity and general applicability of ensembling must be considered key strengths. The main drawback of both methods is the computational cost at test time that grows linearly with  $M$ , limiting real-time applicability. Here, future work includes exploring the effect of model pruning techniques [51, 52] on predictive uncertainty quality. For ensembling, sharing early stages of the DNN among ensemble members is also an interesting future direction. A weakness of ensembling is the additional training required, which also scales linearly with  $M$ . The training of different ensemble members can however be performed in parallel, making it less of an issue in practice given appropriate computing infrastructure. In conclusion, our work suggests that ensembling should be considered the new go-to method for *scalable* epistemic uncertainty estimation.

**Acknowledgments** This research was supported by the Swedish Foundation for Strategic Research via the project *ASSEMBLE* and by the Swedish Research Council via the project *Learning flexible models for nonlinear dynamics*.

## References

- [1] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. “Virtual Worlds as Proxy for Multi-Object Tracking Analysis.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

- [2] M. Wrenninge and J. Unger. “Synscapes: A photorealistic synthetic dataset for street scene parsing.” In: *arXiv preprint arXiv:1810.08705* (2018).
- [3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. “Vision meets Robotics: The KITTI Dataset.” In: *International Journal of Robotics Research (IJRR)* (2013).
- [4] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. “Sparsity Invariant CNNs.” In: *International Conference on 3D Vision (3DV)*. 2017.
- [5] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. “The cityscapes dataset for semantic urban scene understanding.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [6] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. “Rethinking atrous convolution for semantic image segmentation.” In: *arXiv preprint arXiv:1706.05587* (2017).
- [7] Y. Yuan and J. Wang. “Ocnet: Object context network for scene parsing.” In: *arXiv preprint arXiv:1809.00916* (2018).
- [8] S. Shi, X. Wang, and H. Li. “PointRCNN: 3D object proposal generation and detection from point cloud.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [9] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. “Point-Pillars: Fast encoders for object detection from point clouds.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [10] F. Ma, G. V. Cavalheiro, and S. Karaman. “Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera.” In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2019.
- [11] Y. Gal. “Uncertainty in Deep Learning.” PhD thesis. University of Cambridge, 2016.
- [12] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [13] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [14] M. Sensoy, L. Kaplan, and M. Kandemir. “Evidential deep learning to quantify classification uncertainty.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

- [15] A. Malinin and M. Gales. “Predictive uncertainty estimation via prior networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [16] E. Ilg, O. Cicek, S. Galessio, A. Klein, O. Makansi, F. Hutter, and T. Bro. “Uncertainty estimates and multi-hypotheses networks for optical flow.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [17] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [18] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [19] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks.” In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.
- [20] R. M. Neal. “Bayesian learning for neural networks.” PhD thesis. University of Toronto, 1995.
- [21] G. Hinton and D. Van Camp. “Keeping neural networks simple by minimizing the description length of the weights.” In: *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory (COLT)*. 1993.
- [22] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight Uncertainty in Neural Network.” In: *International Conference on Machine Learning (ICML)*. 2015.
- [23] M. Welling and Y. W. Teh. “Bayesian learning via stochastic gradient Langevin dynamics.” In: *International Conference on Machine Learning (ICML)*. 2011.
- [24] T. Chen, E. Fox, and C. Guestrin. “Stochastic gradient Hamiltonian Monte Carlo.” In: *International Conference on Machine Learning (ICML)*. 2014.
- [25] J. M. Hernández-Lobato and R. Adams. “Probabilistic backpropagation for scalable learning of bayesian neural networks.” In: *International Conference on Machine Learning (ICML)*. 2015.
- [26] H. Wang, S. Xingjian, and D.-Y. Yeung. “Natural-parameter networks: A class of probabilistic neural networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

- [27] A. Kendall, V. Badrinarayanan, and R. Cipolla. “Bayesian SegNet: Model Uncertainty in Deep Convolutional Encoder-Decoder Architectures for Scene Understanding.” In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2017.
- [28] J. Mukhoti and Y. Gal. “Evaluating Bayesian Deep Learning Methods for Semantic Segmentation.” In: *arXiv preprint arXiv:1811.12709* (2018).
- [29] J. Snoek, Y. Ovadia, E. Fertig, B. Lakshminarayanan, S. Nowozin, D. Sculley, J. Dillon, J. Ren, and Z. Nado. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [30] R. M. Neal. “MCMC using Hamiltonian dynamics.” In: *Handbook of Markov chain Monte Carlo* (2011).
- [31] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. “Equation of state calculations by fast computing machines.” In: *The journal of chemical physics* (1953).
- [32] W. K. Hastings. “Monte Carlo sampling methods using Markov chains and their applications.” In: *Biometrika* (1970).
- [33] Y.-A. Ma, T. Chen, and E. Fox. “A complete recipe for stochastic gradient MCMC.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.
- [34] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. “Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning.” In: *arXiv preprint arXiv:1902.03932* (2019).
- [35] D. Barber and C. M. Bishop. “Ensemble learning in Bayesian neural networks.” In: *Nato ASI Series F Computer and Systems Sciences* (1998).
- [36] A. Graves. “Practical variational inference for neural networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2011.
- [37] C. Louizos and M. Welling. “Structured and efficient variational deep learning with matrix Gaussian posteriors.” In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016.
- [38] C. Louizos and M. Welling. “Multiplicative normalizing flows for variational Bayesian neural networks.” In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.
- [39] G. Zhang, S. Sun, D. Duvenaud, and R. Grosse. “Noisy Natural Gradient as Variational Inference.” In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018.
- [40] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning.” In: *International Conference on Machine Learning (ICML)*. 2016.

- [41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” In: *The Journal of Machine Learning Research* (2014).
- [42] P. Auer, M. Herbster, and M. K. Warmuth. “Exponentially many local minima for single neurons.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 1996.
- [43] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. “The loss surfaces of multilayer networks.” In: *Artificial Intelligence and Statistics*. 2015.
- [44] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. “Automatic differentiation in PyTorch.” In: *NeurIPS - Autodiff Workshop*. 2017.
- [45] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [46] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [47] J. Bröcker. “Reliability, sufficiency, and the decomposition of proper scores.” In: *Quarterly Journal of the Royal Meteorological Society* (2009).
- [48] J. Vaicenavicius, D. Widmann, C. Andersson, F. Lindsten, J. Roll, and T. B. Schön. “Evaluating model calibration in classification.” In: *arXiv preprint arXiv:1902.06977* (2019).
- [49] M. P. Naeini, G. Cooper, and M. Hauskrecht. “Obtaining well calibrated probabilities using Bayesian binning.” In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.
- [50] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.
- [51] T.-J. Yang, Y.-H. Chen, and V. Sze. “Designing energy-efficient convolutional neural networks using energy-aware pruning.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [52] S. Han, H. Mao, and W. J. Dally. “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.” In: *International Conference on Learning Representations (ICLR)*. 2016.
- [53] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman. “Pyro: Deep Universal Probabilistic Programming.” In: *Journal of Machine Learning Research* (2018).

## Supplementary Material

In this supplementary material, we provide additional details and results. It consists of Appendix A-D. Note that figures in this supplementary material are numbered with the prefix “S”. Numbers without this prefix refer to the main paper.

## A Approximating a Mixture of Gaussian Distributions

For the Gaussian model (2),  $\hat{p}(y^*|x^*, \mathcal{D})$  in (4) is a uniformly weighted mixture of Gaussian distributions. We approximate this mixture with a single Gaussian parameterized by the mixture mean and variance:

$$\begin{aligned}\hat{p}(y^*|x^*, \mathcal{D}) &= \frac{1}{M} \sum_{i=1}^M p(y^*|x^*, \theta^{(i)}), \quad \theta^{(i)} \sim q(\theta), \\ \hat{p}(y^*|x^*, \mathcal{D}) &= \frac{1}{M} \sum_{i=1}^M \mathcal{N}(y^*; \mu_{\theta^{(i)}}(x^*), \sigma_{\theta^{(i)}}^2(x^*)), \quad \theta^{(i)} \sim q(\theta), \\ \hat{p}(y^*|x^*, \mathcal{D}) &\approx \mathcal{N}(y^*; \hat{\mu}(x^*), \hat{\sigma}^2(x^*)), \\ \hat{\mu}(x) &= \frac{1}{M} \sum_{i=1}^M \mu_{\theta^{(i)}}(x), \quad \hat{\sigma}^2(x) = \frac{1}{M} \sum_{i=1}^M \left( (\mu_{\theta^{(i)}}(x) - \hat{\mu}(x))^2 + \sigma_{\theta^{(i)}}^2(x) \right).\end{aligned}$$

## B Illustrative Toy Problems

In this appendix, further details on the illustrative toy problems experiments (Section 4.1) are provided.

### B.1 Experimental Setup

Figure 5a (regression) shows  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  computed on  $[-7, 7]$ . All training data was given in  $[-3, 3]$ .

Figure 5b (classification) shows  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  computed on the region  $-6 \leq x_1 \leq 6, -6 \leq x_2 \leq 6$ . All training data was given in the region  $0 \leq x_1 \leq 3, -3 \leq x_2 \leq 3$ .

For regression,  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  is computed using the formula for KL divergence between two Gaussian distributions  $p_1(x) = \mathcal{N}(x; \mu_1, \sigma_1^2)$ ,  $p_2(x) =$

$\mathcal{N}(x; \mu_2, \sigma_2^2)$ :

$$D_{\text{KL}}(p_1 \parallel p_2) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}.$$

For classification,  $D_{\text{KL}}(p \parallel p_{\text{HMC}})$  is computed using the formula for KL divergence between two discrete distributions  $q_1(x), q_2(x)$ :

$$D_{\text{KL}}(q_1 \parallel q_2) = \sum_{x \in \mathcal{X}} q_1(x) \log \frac{q_1(x)}{q_2(x)}.$$

For both regression and classification, HMC with prior  $p(\theta) = \mathcal{N}(0, I_P)$  and  $M = 1000$  samples is implemented using Pyro [53]. Specifically, we use pyro.infer.mcmc.MCMC with pyro.infer.mcmc.NUTS as kernel, num\_samples = 1000 and warmup\_steps = 1000.

## B.2 Implementation Details

For regression, we use the Gaussian model (2) with two separate feed-forward neural networks outputting  $\mu_\theta(x)$  and  $\log \sigma_\theta^2(x)$ . Both neural networks have 2 hidden layers of size 10.

For classification, we use the Categorical model (1) with a feed-forward neural network with 2 hidden layers of size 10.

For the MC-dropout comparison, we place a dropout layer after the first hidden layer of each neural network. For regression, we use a drop probability  $p = 0.2$ . For classification, we use  $p = 0.1$ .

For ensembling, we train all ensemble models for 150 epochs with the Adam optimizer, a batch size of 32 and a fixed learning rate of 0.001.

For MC-dropout, we train models for 300 epochs with the Adam optimizer, a batch size of 32 and a fixed learning rate of 0.001.

For ensembling and MC-dropout, we minimize the MAP objective  $-\log p(Y|X, \theta)p(\theta)$ . In our case where the model parameters  $\theta \in \mathbb{R}^P$  and  $p(\theta) = \mathcal{N}(0, I_P)$ , this corresponds to the following loss for regression:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{(y_i - \hat{\mu}(x_i))^2}{\hat{\sigma}^2(x_i)} + \log \hat{\sigma}^2(x_i) + \frac{1}{N} \theta^\top \theta.$$

For classification, where  $y_i = [y_{i,1} \dots y_{i,C}]^\top$  (one-hot encoded) and  $\hat{s}(x_i) = [\hat{s}(x_i)_1 \dots \hat{s}(x_i)_C]^\top$  is the Softmax output, it corresponds to the following

loss:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^C y_{i,k} \log \hat{s}(x_i)_k + \frac{1}{2N} \theta^\top \theta.$$

For SGLD, we extract samples from the parameter trajectory given by the update equation:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla_\theta \tilde{U}(\theta_t) + \sqrt{2\alpha_t} \epsilon_t,$$

where  $\epsilon_t \sim \mathcal{N}(0, 1)$ ,  $\nabla_\theta \tilde{U}(\theta)$  is the stochastic gradient of  $U(\theta) = -\log p(Y|X, \theta)p(\theta)$  and  $\alpha_t$  is the stepsize. We run it for a total number of steps corresponding to  $256 \cdot 150$  epochs with a batch size of 32. The stepsize  $\alpha_t$  is decayed according to:

$$\alpha_t = \alpha_0 \left(1 - \frac{t}{T}\right)^{0.9}, \quad t = 1, 2, \dots, T,$$

where  $T$  is the total number of steps,  $\alpha_0 = 0.01$  (the initial stepsize) for regression and  $\alpha_0 = 0.05$  for classification.  $M \in \{8, 16, 32, 64, 128, 256\}$  samples are extracted starting at step  $t = \text{int}(0.75T)$ , ending at step  $t = T$  and spread out evenly between.

For SGHMC, we extract samples from the parameter trajectory given by the update equation:

$$\begin{aligned} \theta_{t+1} &= \theta_t + r_t, \\ r_{t+1} &= (1 - \eta)r_t - \alpha_t \nabla_\theta \tilde{U}(\theta_t) + \sqrt{2\eta\alpha_t} \epsilon_t, \end{aligned}$$

where  $\epsilon_t \sim \mathcal{N}(0, 1)$ ,  $\nabla_\theta \tilde{U}(\theta)$  is the stochastic gradient of  $U(\theta) = -\log p(Y|X, \theta)p(\theta)$ ,  $\alpha_t$  is the stepsize and  $\eta = 0.1$ . We run it for a total number of steps corresponding to  $256 \cdot 150$  epochs with a batch size of 32. The stepsize  $\alpha_t$  is decayed according to:

$$\alpha_t = \alpha_0 \left(1 - \frac{t}{T}\right)^{0.9}, \quad t = 1, 2, \dots, T,$$

where  $T$  is the total number of steps,  $\alpha_0 = 0.001$  (the initial stepsize) for regression and  $\alpha_0 = 0.01$  for classification.  $M \in \{8, 16, 32, 64, 128, 256\}$  samples are extracted starting at step  $t = \text{int}(0.75T)$ , ending at step  $t = T$  and spread out evenly between.

For all models, we randomly initialize the parameters  $\theta$  using the default initializer in PyTorch.

### B.3 Description of Results

The results in Figure 5a, 5b were obtained in the following way:

- **Ensembling:** 1024 models were trained using the same training procedure, the mean and standard deviation was computed based on  $1024/M$  unique sets of models for  $M \in \{8, 16, 32, 64, 128, 256\}$ .
- **MC-dropout:** 10 models were trained using the same training procedure, based on which the mean and standard deviation was computed.
- **SGLD:** 6 models were trained using the same training procedure, based on which the mean and standard deviation was computed.
- **SGHMC:** 6 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

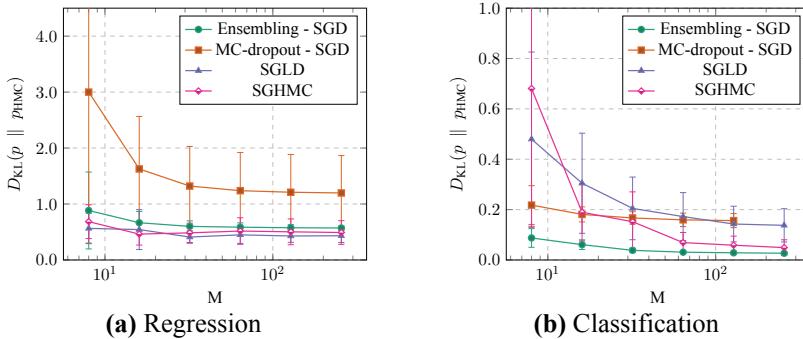
## B.4 Additional Results

Figure S1 and Figure S2 show the same comparison as Figure 5a, 5b, but using SGD and SGD with momentum for ensembling and MC-dropout, respectively. We observe that ensembling consistently outperforms the compared methods for classification, but that SGLD and SGHMC has better performance for regression in these cases. SGLD and SGHMC are however trained for 256 times longer than each ensemble model, complicating the comparison somewhat. If SGLD and SGHMC instead are trained for just 64 times longer than each ensemble model, we observe in Figure S3 that they are consistently outperformed by ensembling.

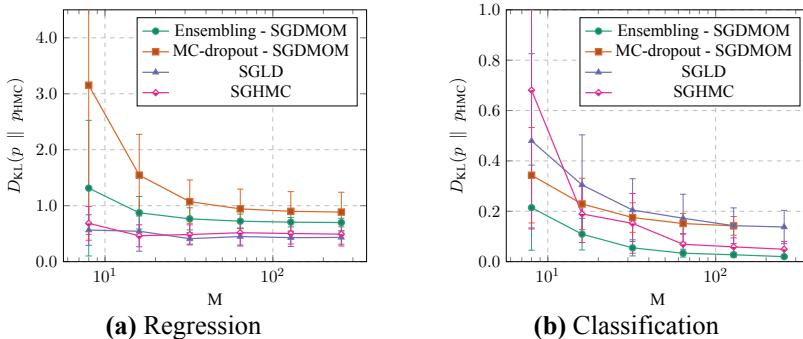
For MC-dropout using Adam, we also varied the drop probability  $p$  and chose the best performing variant. These results are found in Figure S4, in which \* marks the chosen variant.

## B.5 Qualitative Results

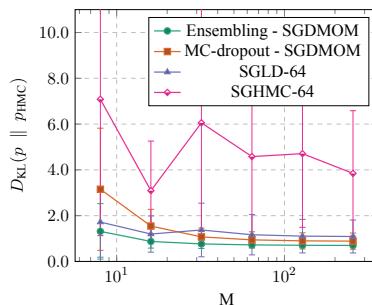
Here, we show visualizations of predictive distributions obtained by the different methods. Figure S5, S9 for ensembling, Figure S6, S10 for MC-dropout, Figure S7, S11 for SGLD, and Figure S8, S12 for SGHMC.



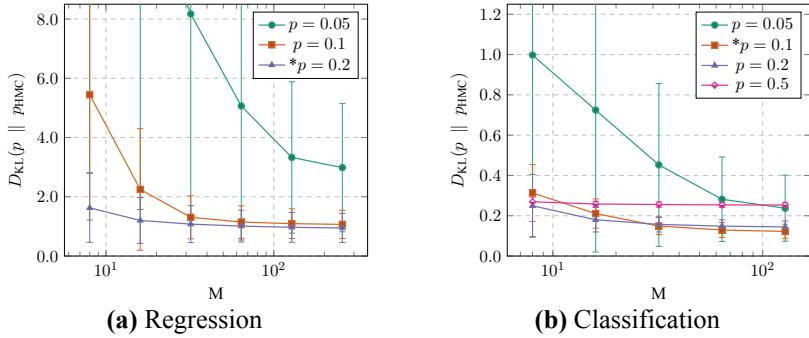
**Figure S1:** Illustrative toy problems, quantitative results. SGD is used for ensembling and MC-dropout instead of Adam.



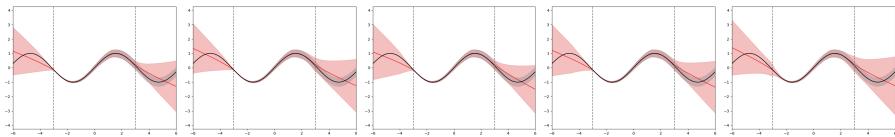
**Figure S2:** Illustrative toy problems, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam.



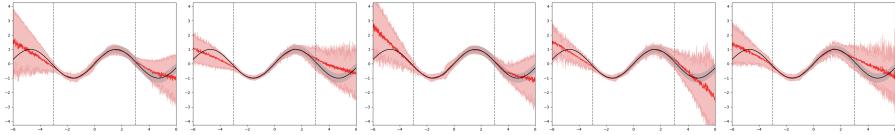
**Figure S3:** Illustrative toy regression problem, quantitative results. SGD with momentum is used for ensembling and MC-dropout instead of Adam. Less training for SGLD and SGHMC.



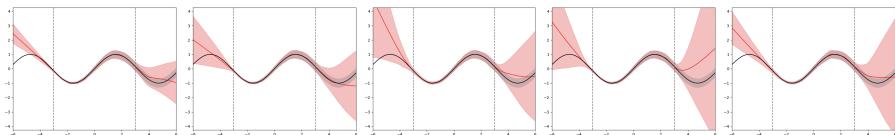
**Figure S4:** Illustrative toy problems, quantitative results. MC-dropout using Adam.



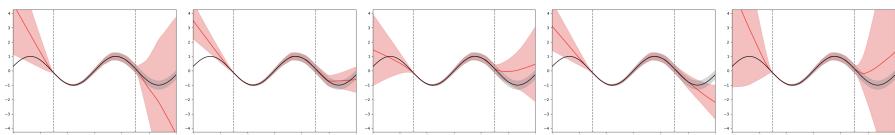
**Figure S5:** Toy regression problem, *ensembling*,  $M = 64$ . Examples of predictive distributions.



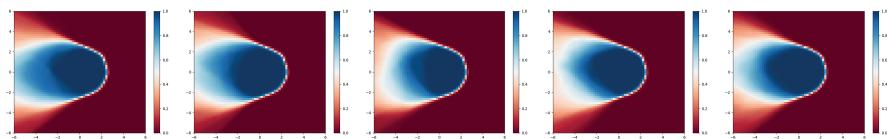
**Figure S6:** Toy regression problem, *MC-dropout*,  $M = 64$ . Examples of predictive distributions.



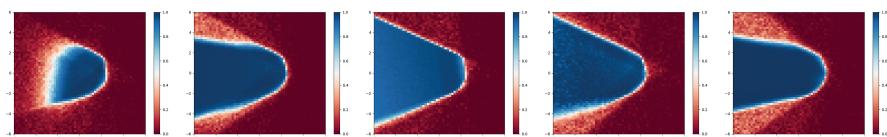
**Figure S7:** Toy regression problem, *SGLD*,  $M = 64$ . Examples of predictive distributions.



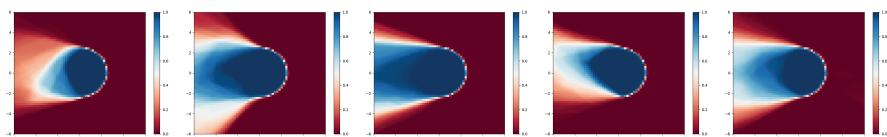
**Figure S8:** Toy regression problem, *SGHMC*,  $M = 64$ . Examples of predictive distributions.



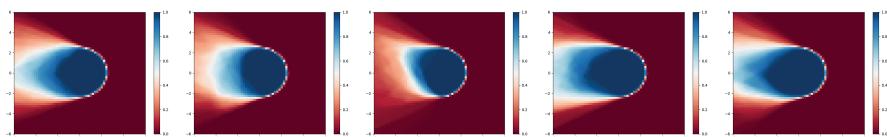
**Figure S9:** Toy classification problem, *ensembling*,  $M = 64$ . Examples of predictive distributions.



**Figure S10:** Toy classification problem, *MC-dropout*,  $M = 64$ . Examples of predictive distributions.



**Figure S11:** Toy classification problem, *SGLD*,  $M = 64$ . Examples of predictive distributions.



**Figure S12:** Toy classification problem, *SGHMC*,  $M = 64$ . Examples of predictive distributions.

## C Depth Completion

In this appendix, further details on the depth completion experiments (Section 4.2) are provided.

### C.1 Training Details

For both ensembling and MC-dropout, we train all models for 40 000 steps with the Adam optimizer, a batch size of 4, a fixed learning rate of  $10^{-5}$  and weight decay of 0.0005. We use a smaller batch size and train for fewer steps than Ma et al. [10] to enable an extensive evaluation with repeated experiments. For the same reason, we also train on randomly selected image crops of size  $352 \times 352$ . The only other data augmentation used is random flipping along the vertical axis. We follow Ma et al. [10] and randomly initialize all network weights from  $\mathcal{N}(0, 10^{-3})$  and all network biases with 0s. Models are trained on a single NVIDIA TITAN Xp GPU with 12GB of RAM.

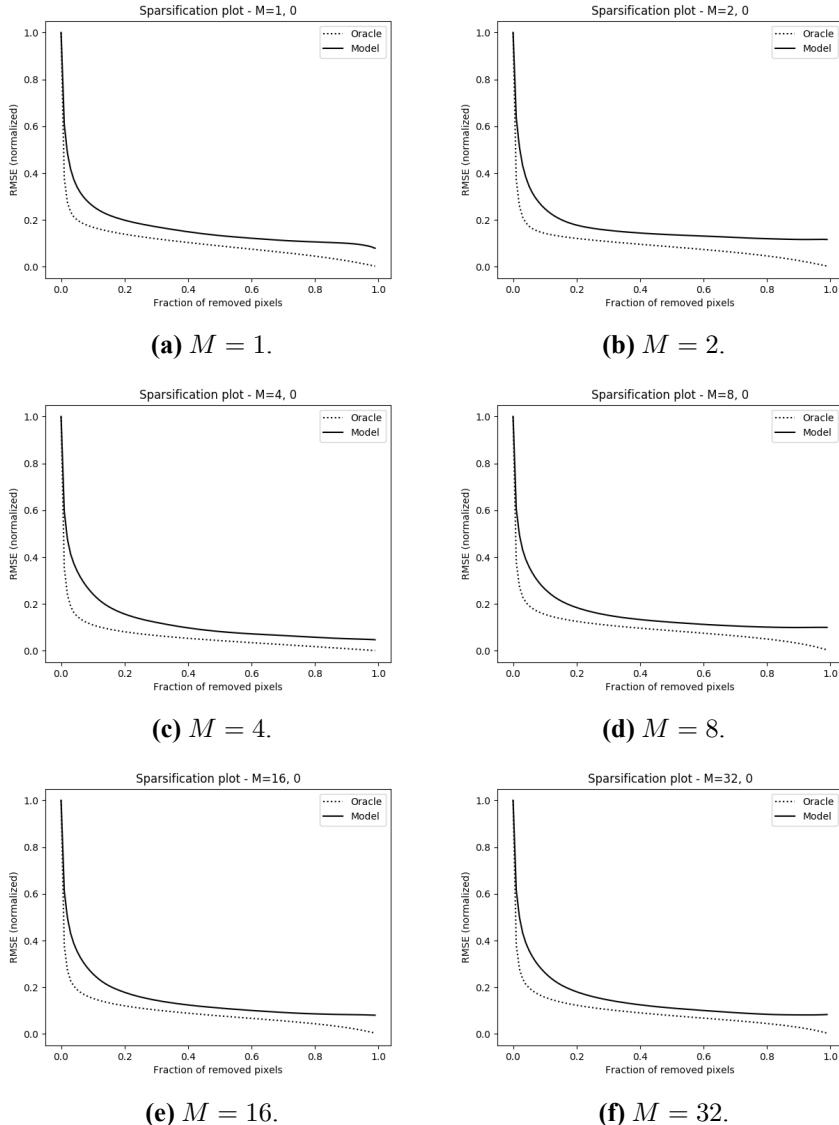
### C.2 Description of Results

The results in Figure 6 (Section 4.2) were obtained in the following way:

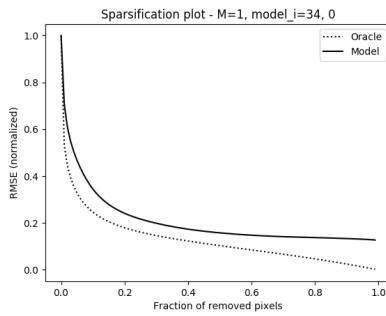
- **Ensembling:** 33 models were trained using the same training procedure, the mean and standard deviation was computed based on 32 ( $M = 1$ ), 16 ( $M = 2, 4, 8, 16$ ) or 4 ( $M = 32$ ) sets of randomly drawn models. The same set could not be drawn more than once.
- **MC-dropout:** 16 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

### C.3 Additional Results

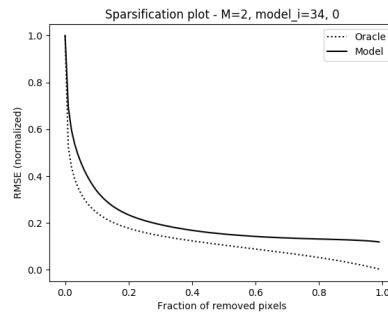
Here, we show sparsification plots, sparsification error curves and calibration plots. Examples of sparsification plots are found in Figure S13 for ensembling and Figure S14 for MC-dropout. Condensed sparsification error curves are found in Figure S15 for ensembling and Figure S16 for MC-dropout. Condensed calibration plots are found in Figure S17 for ensembling and Figure S18 for MC-dropout.



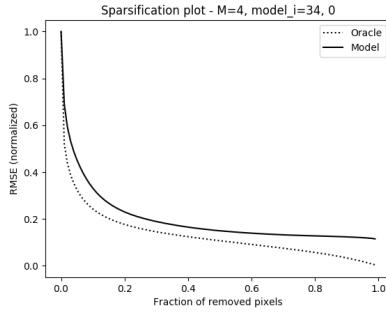
**Figure S13:** Results for *ensembling* on the KITTI depth completion validation dataset. Examples of sparsification plots.



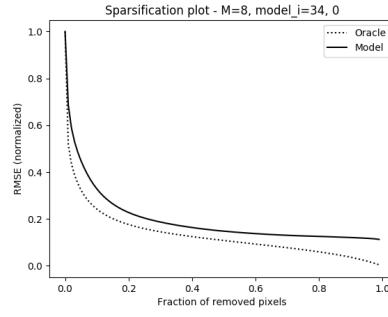
(a)  $M = 1$ .



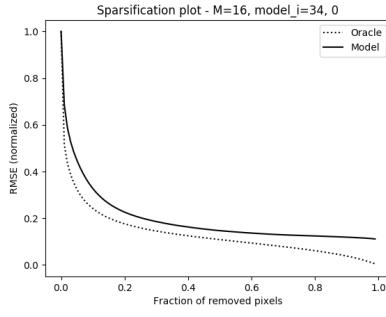
(b)  $M = 2$ .



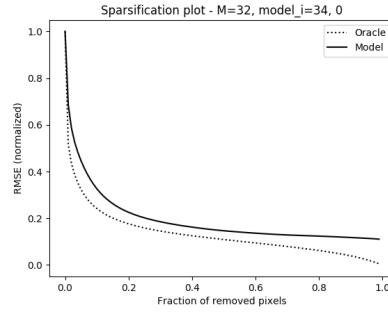
(c)  $M = 4$ .



(d)  $M = 8$ .

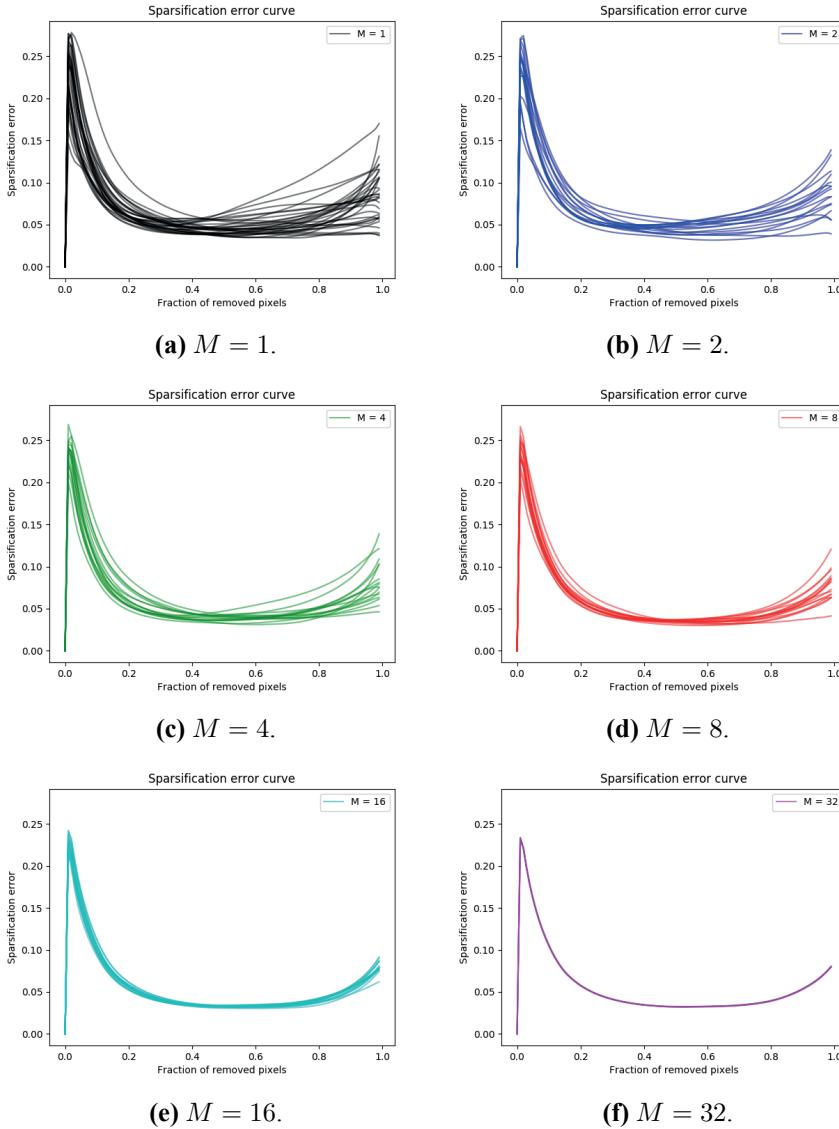


(e)  $M = 16$ .

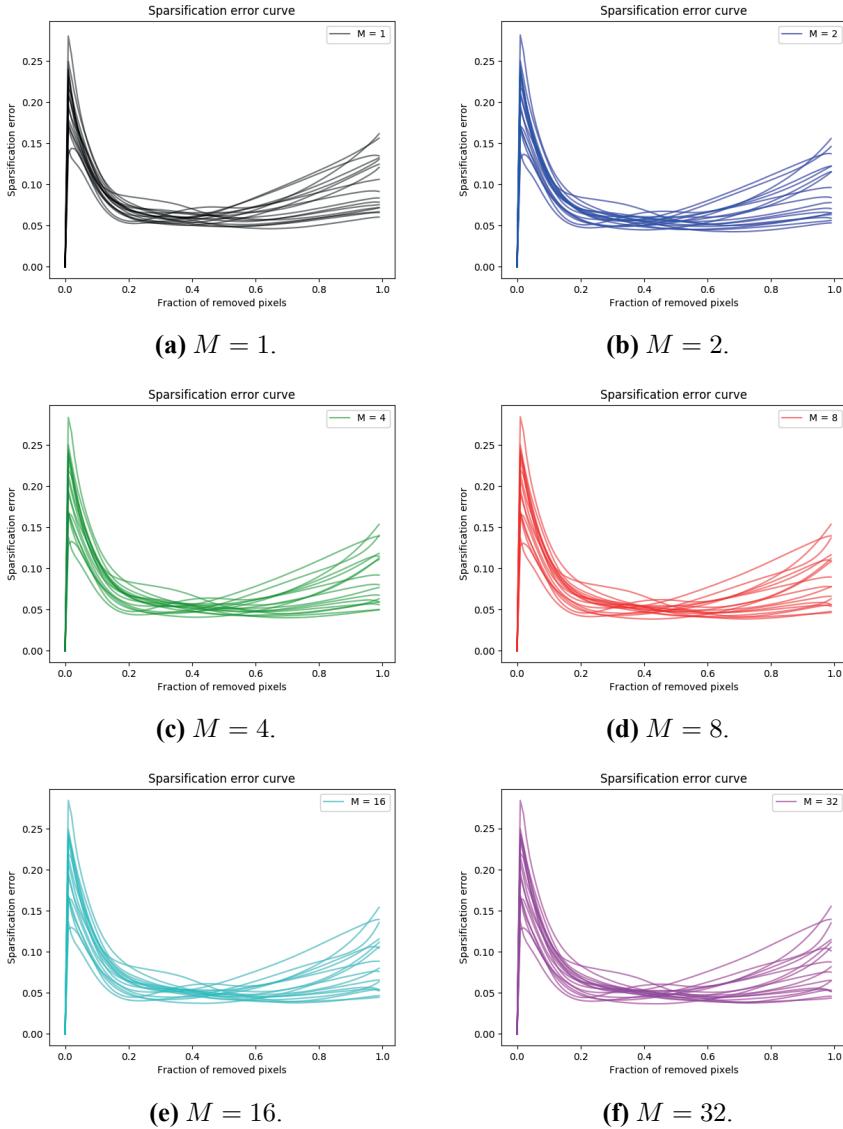


(f)  $M = 32$ .

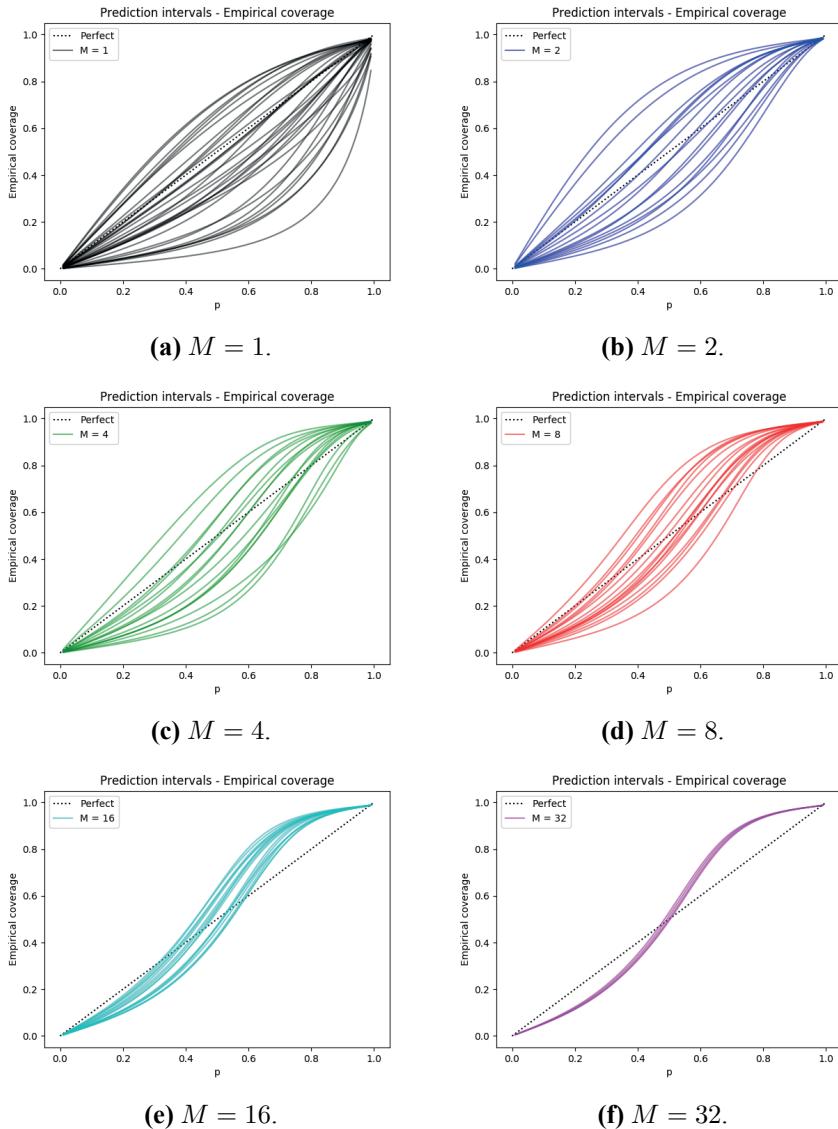
**Figure S14:** Results for *MC-dropout* on the KITTI depth completion validation dataset. Examples of sparsification plots.



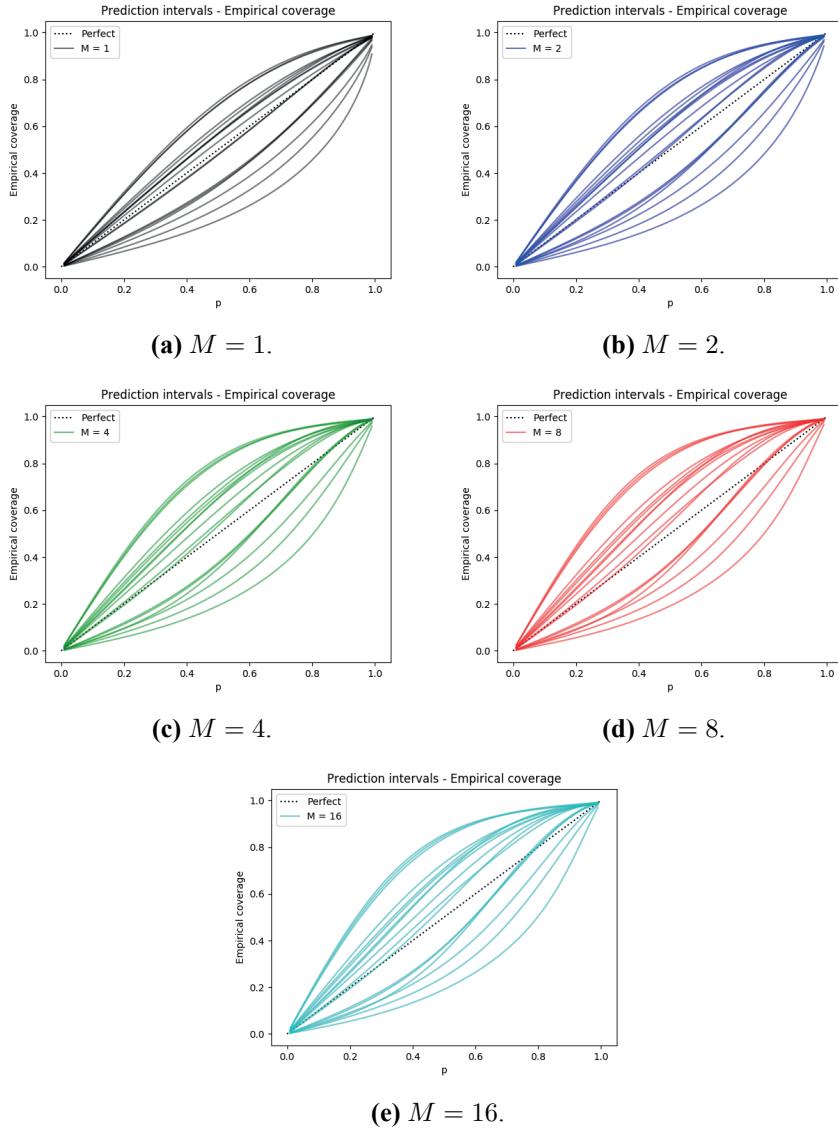
**Figure S15:** Results for *ensembling* on the KITTI depth completion validation dataset. Condensed sparsification error curves.



**Figure S16:** Results for *MC-dropout* on the KITTI depth completion validation dataset. Condensed sparsification error curves.



**Figure S17:** Results for *ensembling* on the KITTI depth completion validation dataset. Condensed calibration plots.



**Figure S18:** Results for *MC-dropout* on the KITTI depth completion validation dataset. Condensed calibration plots.

## D Street-Scene Semantic Segmentation

In this appendix, further details on the street-scene semantic segmentation experiments (Section 4.3) are provided.

### D.1 Training Details

For ensembling, we train all ensemble models for 40 000 steps with SGD + momentum (0.9), a batch size of 8 and weight decay of 0.0005. The learning rate  $\alpha_t$  is decayed according to:

$$\alpha_t = \alpha_0 \left(1 - \frac{t}{T}\right)^{0.9}, \quad t = 1, 2, \dots, T,$$

where  $T = 40\,000$  and  $\alpha_0 = 0.01$  (the initial learning rate). We train on randomly selected image crops of size  $512 \times 512$ . We choose a smaller crop size than Yuan and Wang [7] to enable an extensive evaluation with repeated experiments. The only other data augmentation used is random flipping along the vertical axis and random scaling in the range  $[0.5, 1.5]$ . The ResNet101 backbone is initialized with weights<sup>1</sup> from a model pretrained on the ImageNet dataset, all other model parameters are randomly initialized using the default initializer in PyTorch. Models are trained on two NVIDIA TITAN Xp GPUs with 12GB of RAM each. For MC-dropout, models are instead trained for 60 000 steps.

### D.2 Description of Results

The results in Figure 8 (Section 4.3) were obtained in the following way:

- **Ensembling:** 26 models were trained using the same training procedure, the mean and standard deviation was computed based on 8 sets of randomly drawn models for  $M \in \{1, 2, 4, 8, 16\}$ . The same set could not be drawn more than once.
- **MC-dropout:** 8 models were trained using the same training procedure, based on which the mean and standard deviation was computed.

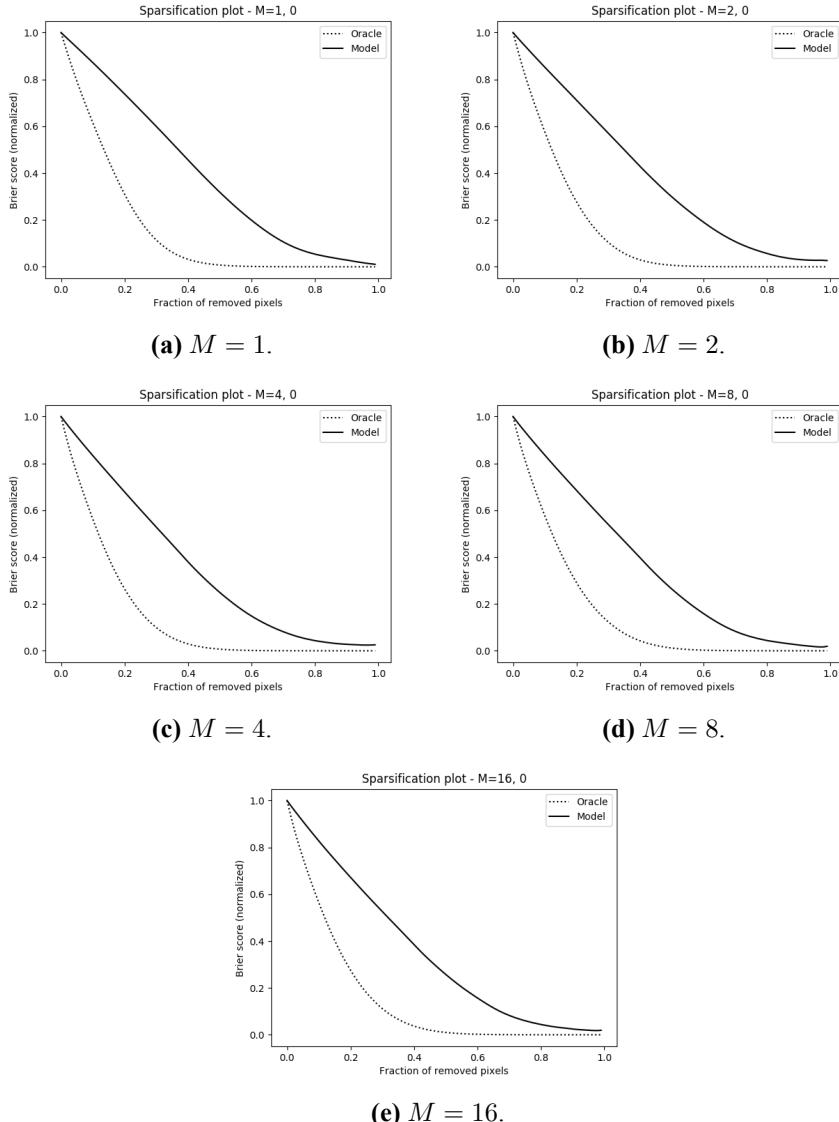
### D.3 Additional Results

Here, we show sparsification plots, sparsification error curves and reliability diagrams. Examples of sparsification plots are found in Figure S19 for ensem-

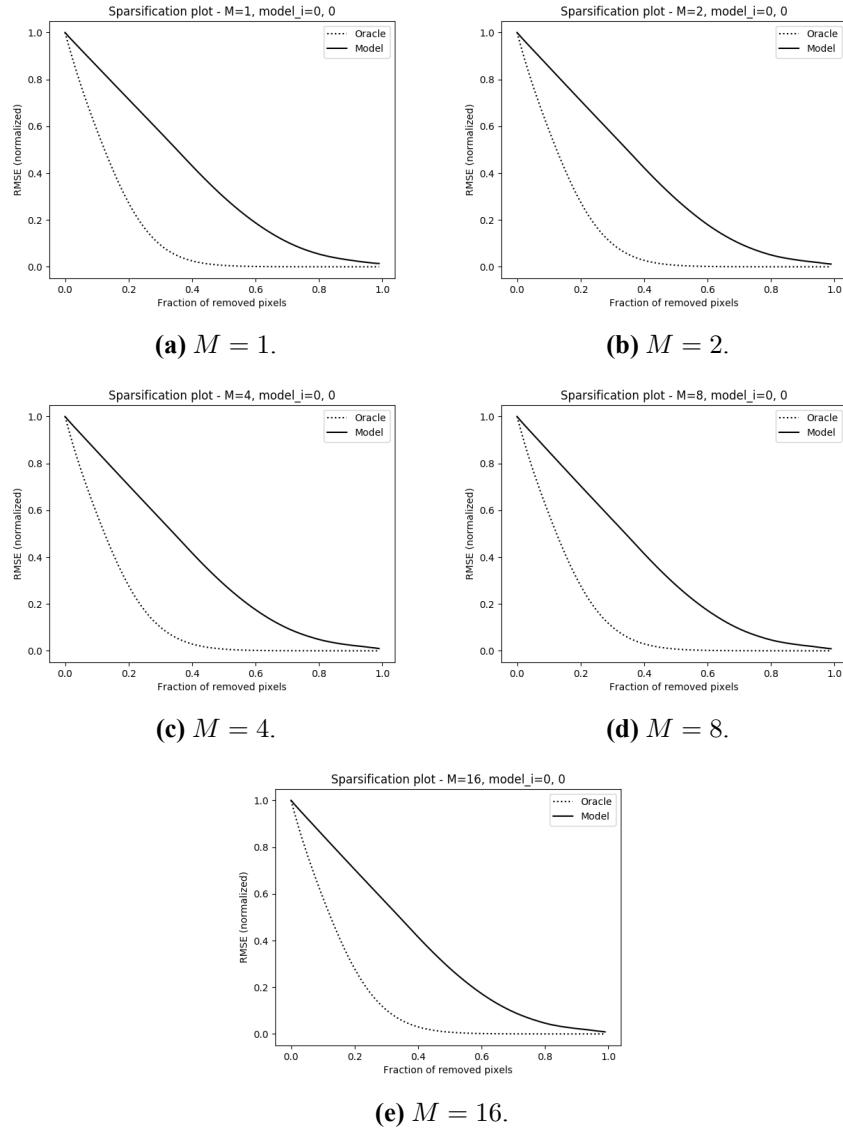
---

<sup>1</sup>[http://sceneparsing.csail.mit.edu/model/pretrained\\_resnet/resnet101-imagenet.pth](http://sceneparsing.csail.mit.edu/model/pretrained_resnet/resnet101-imagenet.pth).

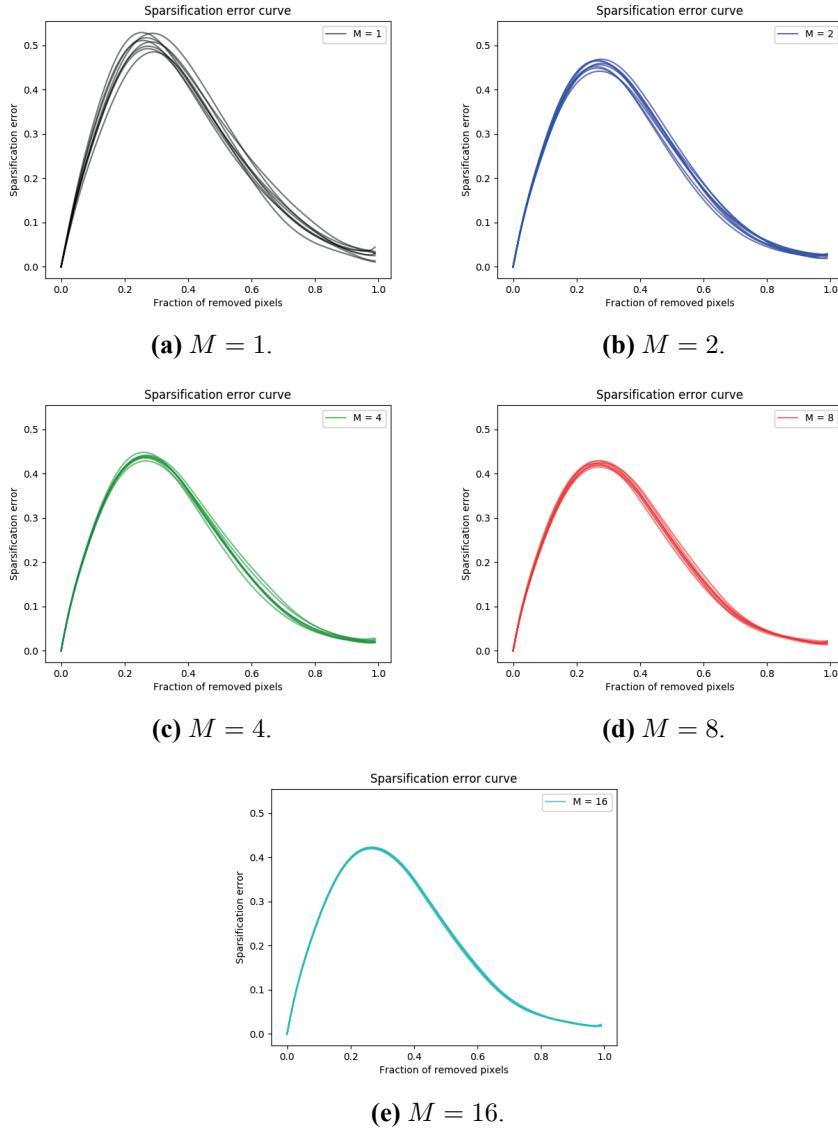
bling and Figure S20 for MC-dropout. Condensed sparsification error curves are found in Figure S21 for ensembling and Figure S22 for MC-dropout. Examples of reliability diagrams with histograms are found in Figure S23 for ensembling and Figure S24 for MC-dropout. Condensed reliability diagrams are found in Figure S25 for ensembling and Figure S26 for MC-dropout.



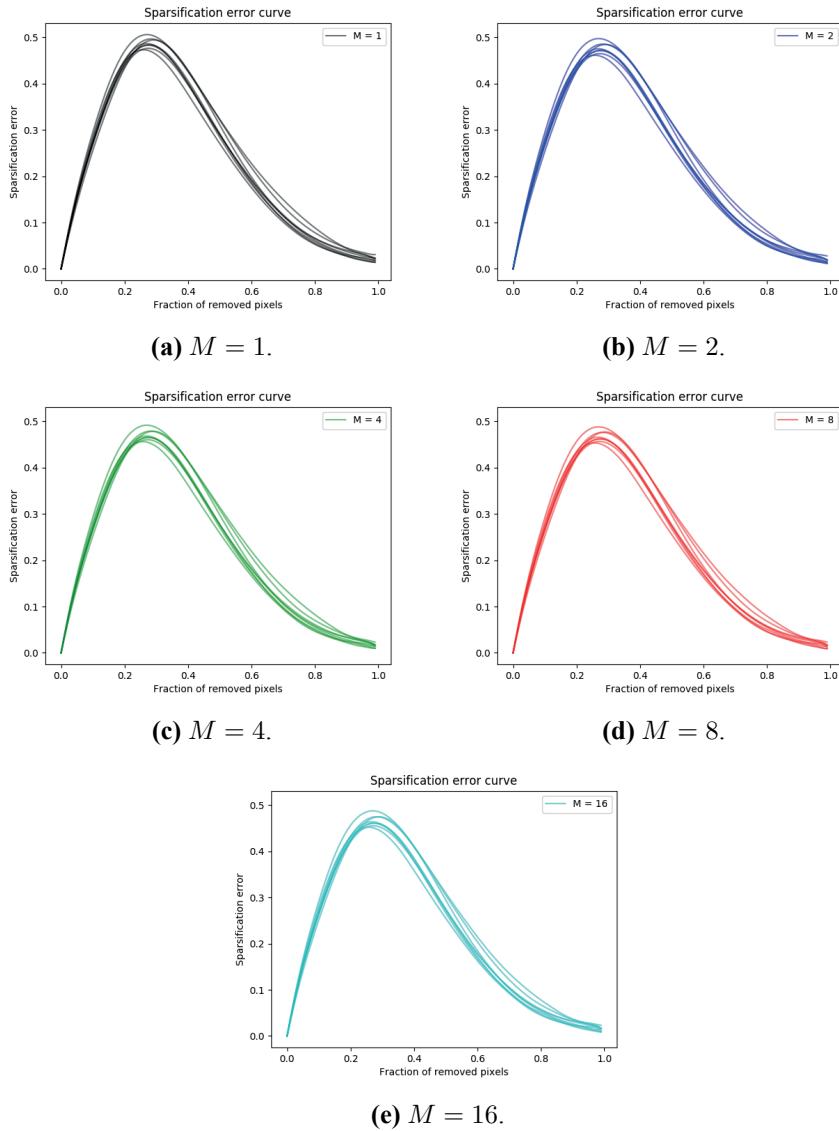
**Figure S19:** Results for *ensembling* on the Cityscapes validation dataset. Examples of sparsification plots.



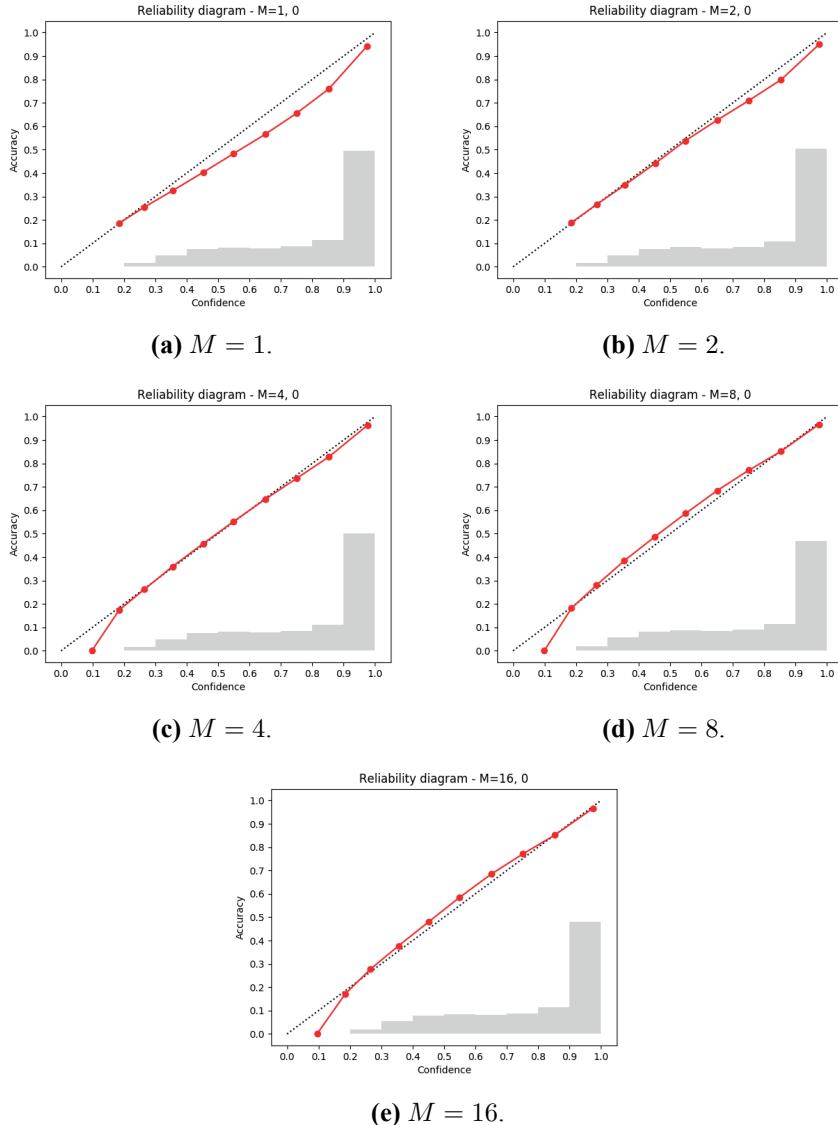
**Figure S20:** Results for *MC-dropout* on the Cityscapes validation dataset. Examples of sparsification plots.



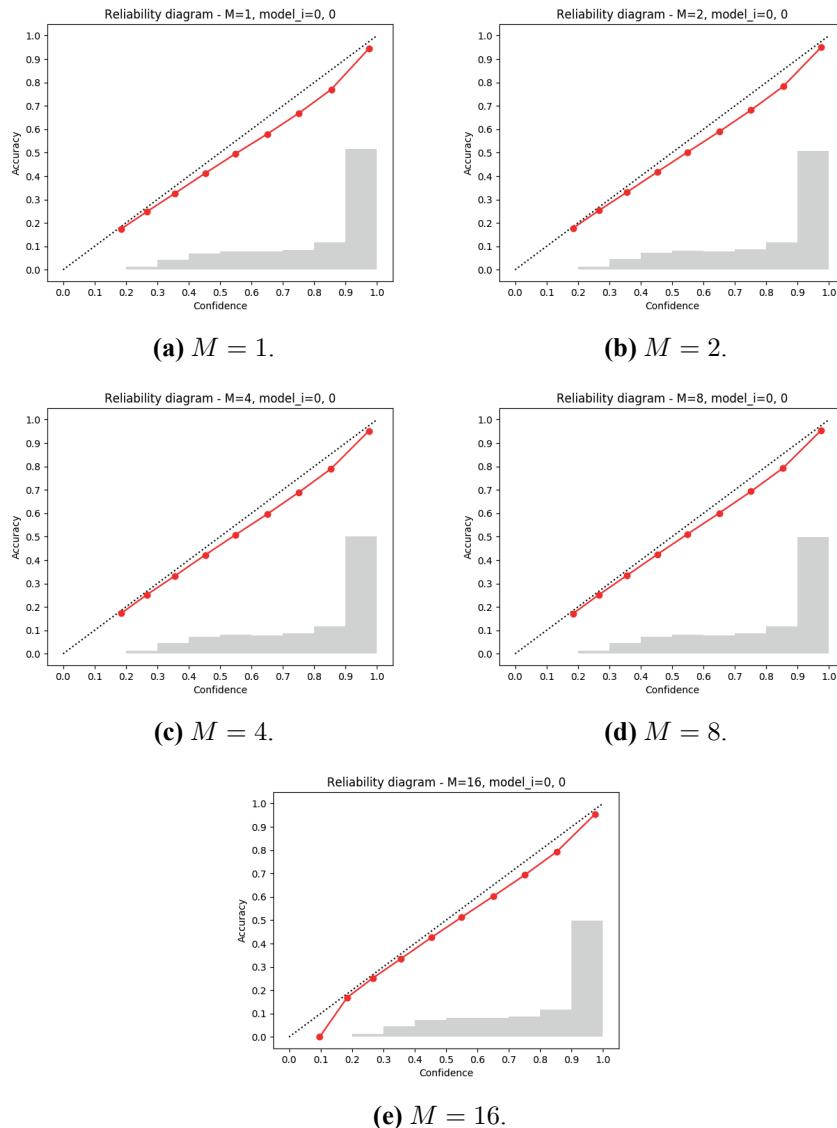
**Figure S21:** Results for *ensembling* on the Cityscapes validation dataset. Condensed sparsification error curves.



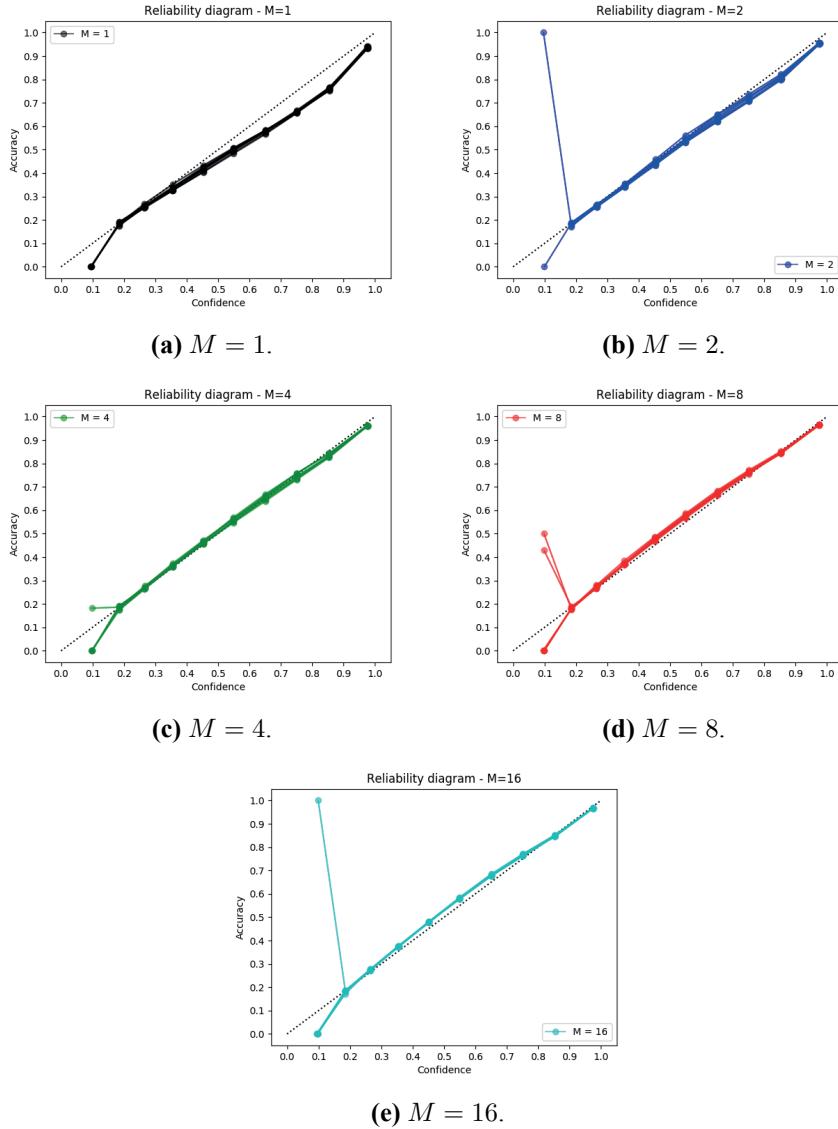
**Figure S22:** Results for *MC-dropout* on the Cityscapes validation dataset. Condensed sparsification error curves.



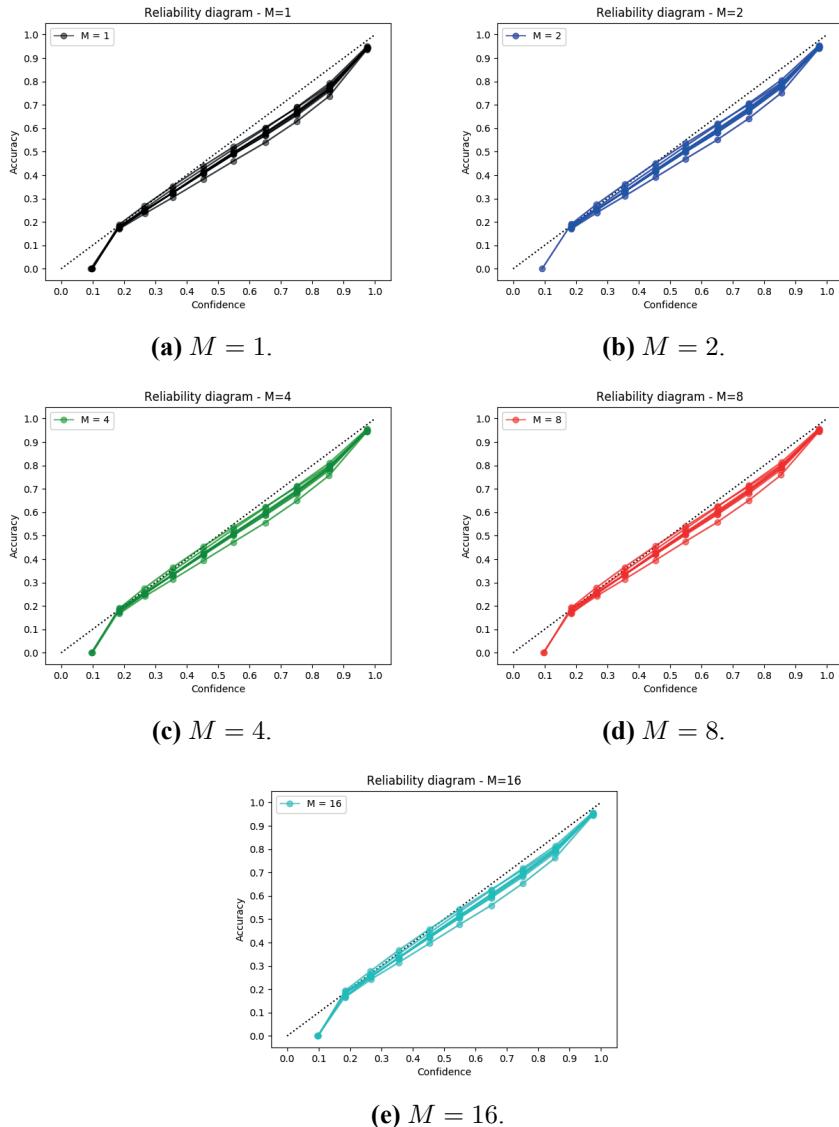
**Figure S23:** Results for *ensembling* on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



**Figure S24:** Results for *MC-dropout* on the Cityscapes validation dataset. Examples of reliability diagrams with histograms.



**Figure S25:** Results for *ensembling* on the Cityscapes validation dataset. Condensed reliability diagrams.



**Figure S26:** Results for *MC-dropout* on the Cityscapes validation dataset. Condensed reliability diagrams.

# Paper VII

## Title

How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?

## Authors

Fredrik K. Gustafsson, Martin Danelljan, Thomas B. Schön

## Edited version of

F. K. Gustafsson, M. Danelljan, and T. B. Schön. “How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?” In: *Transactions on Machine Learning Research (TMLR)*. 2023



# How Reliable is Your Regression Model’s Uncertainty Under Real-World Distribution Shifts?

## Abstract

Many important computer vision applications are naturally formulated as regression problems. Within medical imaging, accurate regression models have the potential to automate various tasks, helping to lower costs and improve patient outcomes. Such safety-critical deployment does however require reliable estimation of model uncertainty, also under the wide variety of distribution shifts that might be encountered in practice. Motivated by this, we set out to investigate the reliability of regression uncertainty estimation methods under various real-world distribution shifts. To that end, we propose an extensive benchmark of 8 image-based regression datasets with different types of challenging distribution shifts. We then employ our benchmark to evaluate many of the most common uncertainty estimation methods, as well as two state-of-the-art uncertainty scores from the task of out-of-distribution detection. We find that while methods are well calibrated when there is no distribution shift, they all become highly overconfident on many of the benchmark datasets. This uncovers important limitations of current uncertainty estimation methods, and the proposed benchmark therefore serves as a challenge to the research community. We hope that our benchmark will spur more work on how to develop truly reliable regression uncertainty estimation methods. Code is available at [https://github.com/fregu856/regression\\_uncertainty](https://github.com/fregu856/regression_uncertainty).

## 1 Introduction

Regression is a fundamental machine learning problem with many important computer vision applications [1, 2, 3, 4, 5, 6]. In general, it entails predicting continuous targets  $y$  from given inputs  $x$ . Within medical imaging, a number of tasks are naturally formulated as regression problems, including brain age estimation [7, 8, 9], prediction of cardiovascular volumes and risk factors [10,

	Cells-Tails	ChairAngle-Gap	AssetWealth	Ventricular Volume	BrainTumour Pixels	SkinLesion Pixels	Histology NucleiPixels	AerialBuilding Pixels
Train								
	$y = 100$	$y = 80.5$	$y = 1.594$	$y = 40.38$	$y = 252$	$y = 500$	$y = 1257$	$y = 1097$
Test								
	$y = 198$	$y = 43.8$	$y = 1.314$	$y = 85.93$	$y = 273$	$y = 516$	$y = 1156$	$y = 433$

**Figure 1:** We propose a benchmark consisting of 8 image-based regression datasets, testing the reliability of regression uncertainty estimation methods under real-world distribution shifts. Example train (top row) and test inputs  $x$ , along with the corresponding ground truth targets  $y$ , are here shown for each of the 8 datasets.

11] and body composition analysis [12, 13]. If machine learning models could be deployed to automatically regress various such properties within real-world clinical practice, this would ultimately help lower costs and improve patient outcomes across the medical system [14].

Real-world deployment in medical applications, and within other safety-critical domains, does however put very high requirements on such regression models. In particular, the common approach of training a deep neural network (DNN) to directly output a predicted regression target  $\hat{y} = f(x)$  is *not* sufficient, as it fails to capture any measure of uncertainty in the predictions  $\hat{y}$ . The model is thus unable to e.g. detect inputs  $x$  which are out-of-distribution (OOD) compared to its training data. Since the predictive accuracy of DNNs typically degrades significantly on OOD inputs [15, 16], this could have potentially catastrophic consequences. Much research effort has therefore been invested into various approaches for training uncertainty-aware DNN models [17, 18, 19, 20, 21], to explicitly estimate the uncertainty in the predictions.

These uncertainty estimates must however be accurate and reliable. Otherwise, if the model occasionally becomes overconfident and outputs highly confident yet incorrect predictions, providing uncertainty estimates might just instill a false sense of security – arguably making the model even less suitable for safety-critical deployment. Specifically, the uncertainty estimates must be *well calibrated* and properly align with the prediction errors [22, 23]. Moreover, the uncertainty must remain well calibrated also under the wide variety of *distribution shifts* that might be encountered during practical deployment [24, 25]. For example in medical applications, a model trained on data collected solely at a large urban hospital in the year 2020, for instance, should output well-calibrated predictions also in 2023, for patients both from urban and rural areas. While uncertainty calibration, as well as general DNN robustness [26],

has been evaluated under distribution shifts for classification tasks [27], this important problem is not well-studied for *regression*.

Motivated by this, we set out to investigate the reliability of regression uncertainty estimation methods under various real-world distribution shifts. To that end, we propose an extensive benchmark consisting of 8 image-based regression datasets (see Figure 1) with different types of distribution shifts. These are all publicly available and relatively large-scale datasets (6 592 - 20 614 training images), yet convenient to store and train models on ( $64 \times 64$  images with 1D regression targets). Four of the datasets are also taken from medical applications, with clinically relevant distribution shifts. We evaluate some of the most commonly used regression uncertainty estimation methods, including conformal prediction, quantile regression and what is often considered the state-of-the-art – ensembling [27, 28]. We also consider the approach of selective prediction [29], in which the regression model can abstain from outputting predictions for certain inputs. This enables us to evaluate uncertainty scores from the rich literature on OOD detection [30]. Specifically, we evaluate two recent scores based on feature-space density [31, 32, 33, 34].

In total, we evaluate 10 different methods. Among them, we find that not a single one is close to being perfectly calibrated across all datasets. While the methods are well calibrated on baseline variants with no distribution shifts, they all become highly overconfident on many of our benchmark datasets. Also the conformal prediction methods suffer from this issue, despite their commonly promoted theoretical guarantees. This highlights the importance of always being aware of underlying assumptions, assessing whether or not they are likely to hold in practice. Methods based on the state-of-the-art OOD uncertainty scores perform well *relative* to other methods, but are also overconfident in many cases – the *absolute* performance is arguably still not sufficient. Our proposed benchmark thus serves as a challenge to the research community, and we hope that it will spur more work on how to develop truly reliable regression uncertainty estimation methods.

**Summary of Contributions** We collect a set of 8 large-scale yet convenient image-based regression datasets with different types of challenging distribution shifts. Utilizing this, we propose a benchmark for testing the reliability of regression uncertainty estimation methods under real-world distribution shifts. We then employ our benchmark to evaluate many of the most common uncertainty estimation methods, as well as two state-of-the-art uncertainty scores from OOD detection. We find that all methods become highly overconfident on many of the benchmark datasets, thus uncovering limitations of current uncertainty estimation methods.

## 2 Background

In a regression problem, the task is to predict a target  $y^* \in \mathcal{Y}$  for any given input  $x^* \in \mathcal{X}$ . To solve this, we are also given a train set of i.i.d. input-target pairs,  $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^N$ ,  $(x_i, y_i) \sim p(x, y)$ . What separates regression from classification is that the target space  $\mathcal{Y}$  is continuous,  $\mathcal{Y} = \mathbb{R}^K$ . In this work, we only consider the 1D case, i.e. when  $\mathcal{Y} = \mathbb{R}$ . Moreover, the input space  $\mathcal{X}$  here always corresponds to the space of images.

**Prediction Intervals, Coverage & Calibration** Given a desired miscoverage rate  $\alpha$ , a *prediction interval*  $C_\alpha(x^*) = [L_\alpha(x^*), U_\alpha(x^*)] \subseteq \mathbb{R}$  is a function that maps the input  $x^*$  onto an interval that should cover the true regression target  $y^*$  with probability  $1 - \alpha$ . For any set  $\{(x_i^*, y_i^*)\}_{i=1}^{N^*}$  of  $N^*$  examples, the empirical interval *coverage* is the proportion of inputs for which the prediction interval covers the corresponding target,

$$\text{Coverage}(C_\alpha) = \frac{1}{N^*} \sum_{i=1}^{N^*} \mathbb{I}\{y_i^* \in C_\alpha(x_i^*)\}. \quad (1)$$

If the coverage equals  $1 - \alpha$ , we say that the prediction intervals are perfectly *calibrated*. Unless stated otherwise, we set  $\alpha = 0.1$  in this work. The prediction intervals should thus obtain a coverage of 90%.

### 2.1 Regression Uncertainty Estimation Methods

The most common approach to image-based regression is to train a DNN  $f_\theta : \mathcal{X} \rightarrow \mathbb{R}$  that outputs a predicted target  $\hat{y} = f_\theta(x)$  for any input  $x$ , using e.g. the L2 or L1 loss [6]. We are interested in methods which extend this standard direct regression approach to also provide uncertainty estimates for the predictions. Specifically, we consider methods which output a prediction interval  $C_\alpha(x)$  and a predicted target  $\hat{y}(x) \in C_\alpha(x)$  for each input  $x$ . The uncertainty in the prediction  $\hat{y}(x)$  is then quantified as the length of the interval  $C_\alpha(x)$  (larger interval - higher uncertainty). Some of the most commonly used regression uncertainty estimation methods fall under this category, as described in more detail below.

**Conformal Prediction** The standard regression approach can be extended by utilizing the framework of split conformal prediction [35, 36, 21]. This entails splitting the train set  $\{(x_i, y_i)\}_{i=1}^N$  into a proper train set  $\mathcal{I}_1$  and a calibration set  $\mathcal{I}_2$ . The DNN  $f_\theta$  is trained on  $\mathcal{I}_1$ , and absolute residuals  $R = \{|y_i - f_\theta(x_i)| : i \in \mathcal{I}_2\}$  are computed on the calibration set  $\mathcal{I}_2$ . Given a new input  $x^*$ , a prediction interval  $C_\alpha(x^*)$  is then constructed from the prediction  $f_\theta(x^*)$  as,

$$C_\alpha(x^*) = [f_\theta(x^*) - Q_{1-\alpha}(R, \mathcal{I}_2), f_\theta(x^*) + Q_{1-\alpha}(R, \mathcal{I}_2)], \quad (2)$$

where  $Q_{1-\alpha}(R, \mathcal{I}_2)$  is the  $(1 - \alpha)$ -th quantile of the absolute residuals  $R$ . Under the assumption of exchangeably drawn train and test data, this prediction interval is guaranteed to satisfy  $\mathbb{P}\{y^* \in C_\alpha(x^*)\} \geq 1 - \alpha$  (marginal coverage guarantee). The interval  $C_\alpha(x^*)$  has a fixed length of  $2Q_{1-\alpha}(R, \mathcal{I}_2)$  for all inputs  $x^*$ .

**Quantile Regression** A DNN can also be trained to directly output prediction intervals of input-dependent length, utilizing the quantile regression approach [37, 21, 38]. This entails estimating the conditional quantile function  $q^\alpha(x) = \inf\{y \in \mathbb{R} : F_{Y|X}(y|x) \geq \alpha\}$ , where  $F_{Y|X}$  is the conditional cumulative distribution function. Specifically, a DNN is trained to output estimates of the lower and upper quantiles  $q^{\alpha_{\text{lo}}}(x)$ ,  $q^{\alpha_{\text{up}}}(x)$  at  $\alpha_{\text{lo}} = \alpha/2$  and  $\alpha_{\text{up}} = 1 - \alpha/2$ . Given a new input  $x^*$ , a prediction interval  $C_\alpha(x^*)$  can then be directly formed,

$$C_\alpha(x^*) = [q_\theta^{\alpha_{\text{lo}}}(x^*), q_\theta^{\alpha_{\text{up}}}(x^*)]. \quad (3)$$

The estimated quantiles  $q_\theta^{\alpha_{\text{lo}}}(x^*)$ ,  $q_\theta^{\alpha_{\text{up}}}(x^*)$  can be output by a single DNN  $f_\theta$ , trained using the pinball loss [39]. A prediction  $\hat{y}(x^*)$  can also be extracted as the center point of  $C_\alpha(x^*)$ .

**Probabilistic Regression** Another approach is to explicitly model the conditional distribution  $p(y|x)$ , for example using a Gaussian model  $p_\theta(y|x) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$  [40, 41]. A single DNN  $f_\theta$  can be trained to output both the mean  $\mu_\theta(x)$  and variance  $\sigma_\theta^2(x)$  by minimizing the corresponding negative log-likelihood. For a given input  $x^*$ , a prediction interval can then be constructed as,

$$C_\alpha(x^*) = [\mu_\theta(x^*) - \sigma_\theta(x^*)\Phi^{-1}(1 - \alpha/2), \mu_\theta(x^*) + \sigma_\theta(x^*)\Phi^{-1}(1 - \alpha/2)], \quad (4)$$

where  $\Phi$  is the CDF of the standard normal distribution. The mean  $\mu_\theta(x^*)$  is also taken as a prediction  $\hat{y}$ .

**Epistemic Uncertainty** From the Bayesian perspective, quantile regression and Gaussian models capture aleatoric (inherent data noise) but not epistemic uncertainty, which accounts for uncertainty in the model parameters [18, 19]. This can be estimated in a principled manner via Bayesian inference, and various approximate methods have been explored [42, 43, 17, 44]. In practice, it has been shown difficult to beat the simple approach of ensembling [20, 27], which entails training  $M$  models  $\{f_{\theta_i}\}_{i=1}^M$  and combining their predictions. For Gaussian models, a single mean  $\hat{\mu}$  and variance  $\hat{\sigma}^2$  can be computed as,

$$\hat{\mu}(x^*) = \frac{1}{M} \sum_{i=1}^M \mu_{\theta_i}(x^*), \quad \hat{\sigma}^2(x^*) = \frac{1}{M} \sum_{i=1}^M \left( (\hat{\mu}(x^*) - \mu_{\theta_i}(x^*))^2 + \sigma_{\theta_i}^2(x^*) \right), \quad (5)$$

and then plugged into (4) to construct a prediction interval  $C_\alpha(x^*)$  for a given input  $x^*$ .

## 2.2 Selective Prediction

The framework of selective prediction has been applied both to classification [45, 29] and regression problems [46]. The general idea is to give a model the option to abstain from outputting predictions for some inputs. This is achieved by combining the prediction model  $f_\theta$  with an uncertainty function  $\kappa_f : \mathcal{X} \rightarrow \mathbb{R}$ . Given an input  $x^*$ , the prediction  $f_\theta(x^*)$  is output if the uncertainty  $\kappa_f(x^*) \leq \tau$  (for some user-specified threshold  $\tau$ ), otherwise  $x^*$  is rejected and no prediction is made. The *prediction rate* is the proportion of inputs for which a prediction is output,

$$\text{Prediction Rate} = \frac{1}{N^*} \sum_{i=1}^{N^*} \mathbb{I}\{\kappa_f(x_i^*) \leq \tau\}. \quad (6)$$

In principle, if high uncertainty  $\kappa_f(x^*)$  corresponds to a large prediction error  $|y^* - \hat{y}(x^*)|$  and vice versa, small errors will be achieved for all predictions which are actually output by the model. Specifically in this work, we combine selective prediction with the regression methods from Section 2.1. A prediction interval  $C_\alpha(x^*)$  and predicted target  $\hat{y}(x^*)$  are thus output if and only if (iff)  $\kappa_f(x^*) \leq \tau$ . Our aim is for this to improve the calibration (interval coverage closer to  $1 - \alpha$ ) of the output prediction intervals.

For the uncertainty function  $\kappa_f(x)$ , the variance  $\hat{\sigma}^2(x)$  of a Gaussian ensemble (5) could be used, for example. One could also use some of the various uncertainty scores employed in the rich OOD detection literature [30]. In OOD detection, the task is to distinguish in-distribution inputs  $x$ , inputs which are similar to those of the train set  $\{(x_i, y_i)\}_{i=1}^N$ , from out-of-distribution inputs. A principled approach to OOD detection would be to fit a model of  $p(x)$  on the train set. Inputs  $x$  for which  $p(x)$  is small are then deemed OOD [47]. In our considered case where inputs  $x$  are images, modelling  $p(x)$  can however be quite challenging. To mitigate this, a feature extractor  $g : \mathcal{X} \rightarrow \mathbb{R}^{D_g}$  can be utilized, modelling  $p(x)$  indirectly by fitting a simple model to the feature vectors  $g(x)$ . In the classification setting, [31] fit a Gaussian mixture model (GMM) to the feature vectors  $\{g(x_i)\}_{i=1}^N$  of the train set. Given an input  $x^*$ , it is then deemed OOD if the GMM density  $\text{GMM}(g(x^*))$  is small. [32] apply this approach also to regression problems. Instead of fitting a GMM to the feature vectors and evaluating its density, [33, 34] compute the distance  $\text{kNN}(g(x^*))$  between  $g(x^*)$  and its  $k$  nearest neighbors in the train set  $\{g(x_i)\}_{i=1}^N$ . The input  $x^*$  is then deemed OOD if this kNN distance is large.

### 3 Proposed Benchmark

We propose an extensive benchmark for testing the reliability of regression uncertainty estimation methods under real-world distribution shifts. The benchmark consists of 8 publicly available image-based regression datasets, which are described in detail in Section 3.1. Our complete evaluation procedure, evaluating uncertainty estimation methods mainly in terms of prediction interval coverage, is then described in Section 3.2.

#### 3.1 Datasets

In an attempt to create a standard benchmark for image-based regression under distribution shifts, we collect and modify 8 datasets from the literature. Two of them contain synthetic images while the remaining six are real-world datasets, four of which are taken from medical applications. Examples from each of the 8 datasets are shown in Figure 1. We create two additional variants of each of the synthetic datasets, thus resulting in 12 datasets in total. They are all relatively large-scale (6 592 - 20 614 train images) and contain input images  $x$  of size  $64 \times 64$  along with 1D regression targets  $y$ . Descriptions of all 12 datasets are given below (further details are also provided in Appendix A), starting with the two synthetic datasets and their variants.

**Cells** Given a synthetic fluorescence microscopy image  $x$ , the task is to predict the number of cells  $y$  in the image. We utilize the Cell-200 dataset from [48, 49], consisting of 200 000 grayscale images of size  $64 \times 64$ . We randomly draw 10 000 train images, 2 000 val images and 10 000 test images. Thus, there is no distribution shift between train/val and test. We therefore use this as a baseline dataset.

**Cells-Tails** A variant of *Cells* with a clear distribution shift between train/val and test. For train/val, the regression targets  $y$  are limited to  $]50, 150]$ . For test, the targets instead lie in the original range  $[1, 200]$ .

**Cells-Gap** Another variant of *Cells* with a clear distribution shift between train/val and test. For train/val, the regression targets  $y$  are limited to  $[1, 50] \cup ]150, 200]$ . For test, the targets instead lie in the original  $[1, 200]$ .

**ChairAngle** Given a synthetic image  $x$  of a chair, the task is to predict the yaw angle  $y$  of the chair. We utilize the RC-49 dataset [48, 49], which contains  $64 \times 64$  images of different chair models rendered at yaw angles ranging from  $0.1^\circ$  to  $89.9^\circ$ . We randomly split their training set and obtain 17 640 train images and 4 410 val images. By sub-sampling their test set we also get 11 225 test images. There is no clear distribution shift between train/val and test, and we therefore use this as a second baseline dataset.

**ChairAngle-Tails** A variant of *ChairAngle* with a clear distribution shift between train/val and test. For train/val, we limit the regression targets  $y$  to  $]15, 75[$ . For test, the targets instead lie in the original  $]0, 90[$ .

**ChairAngle-Gap** Another variant of *ChairAngle* with a clear distribution shift. For train/val, the regression targets  $y$  are limited to  $]0, 30[ \cup ]60, 90[$ . For test, the targets instead lie in the original  $]0, 90[$ .

**AssetWealth** Given a satellite image  $x$ , the task is to predict the asset wealth index  $y$  of the region. We utilize the PovertyMap-Wilds dataset from [16], which is a variant of the dataset collected by [50]. We use the training, validation-ID and test-OOD subsets of the data, giving us 9 797 train images, 1 000 val images and 3 963 test images. We resize the images from size  $224 \times 224$  to  $64 \times 64$ . Train/val and test contain satellite images from disjoint sets of African countries, creating a distribution shift.

**VentricularVolume** Given an echocardiogram image  $x$  of a human heart, the task is to predict the volume  $y$  of the left ventricle. We utilize the EchoNet-Dynamic dataset [51], which contains 10 030 echocardiogram videos. Each video captures a complete cardiac cycle and is labeled with the left ventricular volume at two separate time points, representing end-systole (at the end of contraction) and end-diastole (just before contraction). For each video, we extract just one of these volume measurements along with the corresponding video frame. To create a clear distribution shift between train/val and test, we select the end systolic volume (smaller volume) for train and val, but the end diastolic volume (larger volume) for test. We utilize the provided dataset splits, giving us 7 460 train images, 1 288 val images and 1 276 test images.

**BrainTumourPixels** Given an image slice  $x$  of a brain MRI scan, the task is to predict the number of pixels  $y$  in the image which are labeled as brain tumour. We utilize the brain tumour dataset of the medical segmentation decathlon [52, 53], which is a subset of the data used in the 2016 and 2017 BraTS challenges [54, 55, 56]. The dataset contains 484 brain MRI scans with corresponding tumour segmentation masks. We split these scans 80%/20%/20% into train, val and test sets. The scans are 3D volumes of size  $240 \times 240 \times 155$ . We convert each scan into 155 image slices of size  $240 \times 240$ , and create a regression target for each image by counting the number of labeled brain tumour pixels. This gives us 20 614 train images, 6 116 val images and 6 252 test images.

**SkinLesionPixels** Given a dermatoscopic image  $x$  of a pigmented skin lesion, the task is to predict the number of pixels  $y$  in the image which are labeled as lesion. We utilize the HAM10000 dataset by [57], which contains 10 015 dermatoscopic images with corresponding skin lesion segmentation masks. HAM10000 consists of four different sub-datasets, three of which were collected in Austria, while the fourth sub-dataset was collected in Australia. To create a clear distribution shift between train/val and test, we use the

Australian sub-dataset as our test set. After randomly splitting the remaining images 85%/15% into train and val sets, we obtain 6 592 train images, 1 164 val images and 2 259 test images.

**HistologyNucleiPixels** Given an H&E stained histology image  $x$ , the task is to predict the number of pixels  $y$  in the image which are labeled as nuclei. We utilize the CoNSeP dataset by [58], along with the pre-processed versions they provide of the Kumar [59] and TNBC [60] datasets. The datasets contain large H&E stained image tiles, with corresponding nuclear segmentation masks. The three datasets were collected at different hospitals/institutions, with differing procedures for specimen preservation and staining. By using CoNSeP and Kumar for train/val and TNBC for test, we thus obtain a clear distribution shift. From the large image tiles, we extract  $64 \times 64$  patches via regular gridding, and create a regression target for each image patch by counting the number of labeled nuclei pixels. In the end, we obtain 10 808 train images, 2 702 val images and 2 267 test images.

**AerialBuildingPixels** Given an aerial image  $x$ , the task is to predict the number of pixels  $y$  in the image which are labeled as building. We utilize the Inria aerial image labeling dataset [61], which contains 180 large aerial images with corresponding building segmentation masks. The images are captured at five different geographical locations. We use the images from two densely populated American cities for train/val, and the images from a more rural European area for test, thus obtaining a clear distribution shift. After preprocessing, we obtain 11 184 train images, 2 797 val images and 3 890 test images.

Constructing these custom datasets is one of our main contributions. This is what enables us to propose an extensive benchmark of large-scale yet convenient datasets (which are all publicly available), containing different types of challenging distribution shifts, specifically for image-based regression.

## 3.2 Evaluation

We propose to evaluate regression uncertainty estimation methods mainly in terms of prediction interval coverage (1): if a method outputs a prediction  $\hat{y}(x)$  and a 90% prediction interval  $C_{0.1}(x)$  for each input  $x$ , does the method actually achieve 90% coverage on the *test* set? I.e., are the prediction intervals calibrated?

**Motivation** Regression uncertainty estimation methods can be evaluated using various approaches. One alternative is sparsification [62], measuring how well the uncertainty can be used to sort predictions from worst to best. Perfect sparsification can however be achieved even if the absolute scale of the uncertainty is consistently underestimated. Therefore, a lot of previous work has instead evaluated methods in terms of calibration [22, 23]. Specifically

for regression, a common form of calibration is based on quantiles [63, 64, 65]. Essentially, a model is said to be calibrated if the interval coverage (1) equals  $1 - \alpha$  for *all* miscoverage rates  $\alpha \in ]0, 1[$ . This is measured by the expected calibration error,  $ECE = \frac{1}{m} \sum_{j=1}^m |Coverage(C_{\alpha_j}) - (1 - \alpha_j)|$ ,  $\alpha_j \sim U(0, 1)$ . Our proposed evaluation metric is thus a special case of this approach, considering just one specific miscoverage rate  $\alpha = 0.1$ . We argue that this results in a simpler and more interpretable metric, which also is motivated by how prediction intervals actually are used in real-world applications. There, one particular value of  $\alpha$  is selected ( $\alpha = 0.1$  is a common choice), and the corresponding intervals  $C_\alpha$  are then expected to achieve a coverage of  $1 - \alpha$  on unseen test data. Recent alternative calibration metrics directly measure how well the uncertainty aligns with the prediction errors [66, 67]. While these enable relative comparisons of different methods, they are not easily interpretable in terms of absolute performance.

**Implementation Details** For each dataset and method, we first train a DNN on the train set  $\mathcal{D}_{\text{train}}$ . Then, we run the method on the *val* set  $\mathcal{D}_{\text{val}}$ , resulting in a prediction interval  $C_\alpha(x) = [L_\alpha(x), U_\alpha(x)]$  for each input  $x$  ( $\alpha = 0.1$ ). Importantly, we then calibrate these prediction intervals on *val* using the procedure in [21]. This gives calibrated prediction intervals  $\tilde{C}_\alpha(x)$ , for which the interval coverage on *val* exactly equals  $1 - \alpha$ . Specifically,  $\tilde{C}_\alpha(x)$  is constructed from the original interval  $C_\alpha(x) = [L_\alpha(x), U_\alpha(x)]$ ,

$$\tilde{C}_\alpha(x) = [L_\alpha(x) - Q_{1-\alpha}(E, \mathcal{D}_{\text{val}}), U_\alpha(x) + Q_{1-\alpha}(E, \mathcal{D}_{\text{val}})], \quad (7)$$

where  $E = \{\max(L_\alpha(x_i) - y_i, y_i - U_\alpha(x_i)) : i \in \mathcal{D}_{\text{val}}\}$  are conformity scores computed on  $\mathcal{D}_{\text{val}}$ , and  $Q_{1-\alpha}(E, \mathcal{D}_{\text{val}})$  is the  $(1 - \alpha)$ -th quantile of these scores. Finally, we then run the method on the test set  $\mathcal{D}_{\text{test}}$ , outputting a calibrated prediction interval  $\tilde{C}_\alpha(x^*)$  for each input  $x^*$ . Ideally, the interval coverage of  $\tilde{C}_\alpha(x^*)$  does not change from *val* to *test*, i.e.  $Coverage(\tilde{C}_\alpha) = 1 - \alpha$  should be true also on *test*. If  $Coverage(\tilde{C}_\alpha) \neq 1 - \alpha$ , a conservative method ( $> 1 - \alpha$ ) is preferred compared to an *overconfident* method ( $< 1 - \alpha$ ). For methods based on selective prediction (Section 2.2), the only difference is that prediction intervals  $\tilde{C}_\alpha(x^*)$  are output only for some test inputs  $x^*$  (iff  $\kappa_f(x^*) \leq \tau$ ). The interval coverage is thus computed only on this subset of *test*. This is similar to the notion of “selective calibration” discussed for the classification setting by [68]. We set  $\alpha = 0.1$  since this is a commonly used miscoverage rate in practice.

**Secondary Metrics** We also evaluate methods in terms of mean absolute error (MAE) and average interval length on the *val* set. This measures the quality of the prediction  $\hat{y}(x)$  and the prediction interval  $C_\alpha(x)$ , respectively [22]. The average interval length is a natural secondary metric, since a method that achieves a coverage close to  $1 - \alpha$  but outputs extremely large intervals for all inputs  $x$ , would not be particularly useful in practice. Moreover, if two different methods both are perfectly calibrated, i.e.  $Coverage(C_\alpha) = 1 - \alpha$ , the

method producing smaller prediction intervals would be preferred. For methods based on selective prediction (which output predictions only for certain inputs  $x$ ), the proportion of inputs for which a prediction actually is output is another natural secondary metric. For these methods, we thus also evaluate in terms of the prediction rate (6) on test. If a coverage close to  $1 - \alpha$  is achieved with a very low prediction rate, the method might still not be practically useful in certain applications. For two perfectly calibrated methods, one with a higher prediction rate would be preferred.

## 4 Evaluated Methods

We evaluate five common regression uncertainty estimation methods from Section 2.1, which all output a prediction interval  $C_\alpha(x)$  and a predicted target  $\hat{y}(x) \in C_\alpha(x)$  for each input  $x$ . Two of these we also combine with selective prediction (Section 2.2), utilizing four different uncertainty functions  $\kappa_f(x)$ . In total, we evaluate 10 different methods. For all methods, we train models based on a ResNet34 [69] backbone DNN. This architecture is chosen because of its simplicity and widespread use. The ResNet takes an image  $x$  as input and outputs a feature vector  $g(x) \in \mathbb{R}^{512}$ . Below we specify and provide implementation details for each of the 10 evaluated methods, while we refer back to Section 2 for more general descriptions.

**Conformal Prediction** We create a standard direct regression model by feeding the ResNet feature vector  $g(x)$  into a network head of two fully-connected layers, outputting a scalar prediction  $f_\theta(x)$ . The model is trained using the L2 loss. We then utilize conformal prediction to create prediction intervals  $C_\alpha(x)$  according to (2). Instead of splitting the train images into  $\mathcal{I}_1$  and  $\mathcal{I}_2$ , we use the val images as the calibration set  $\mathcal{I}_2$ .

**Ensemble** We train an ensemble  $\{f_{\theta_1}, \dots, f_{\theta_M}\}$  of  $M = 5$  direct regression models and compute the ensemble mean  $\hat{\mu}(x) = \frac{1}{M} \sum_{i=1}^M f_{\theta_i}(x)$  and ensemble variance  $\hat{\sigma}^2(x) = \frac{1}{M} \sum_{i=1}^M (\hat{\mu}(x) - f_{\theta_i}(x))^2$ . By inserting these into equation 4, prediction intervals  $C_\alpha(x)$  of input-dependent length are then constructed.

**Gaussian** We create a Gaussian model  $p_\theta(y|x) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$  by feeding the ResNet feature vector  $g(x)$  into two separate network heads of two fully-connected layers. These output the mean  $\mu_\theta(x)$  and variance  $\sigma_\theta^2(x)$ , respectively. Prediction intervals  $C_\alpha(x)$  are then constructed according to (4).

**Gaussian Ensemble** We train an ensemble of  $M = 5$  Gaussian models, compute a single mean  $\hat{\mu}(x)$  and variance  $\hat{\sigma}^2(x)$  according to (5), and plug these into (4) to construct prediction intervals  $C_\alpha(x)$ .

**Quantile Regression** We create a quantile regression model by feeding the ResNet feature vector  $g(x)$  into two separate network heads. These output the quantiles  $q_\theta^{\alpha_{10}}(x), q_\theta^{\alpha_{90}}(x)$ , directly forming prediction intervals  $C_\alpha(x)$  according to (3). The model is trained by minimizing the average pinball loss of  $q_\theta^{\alpha_{10}}(x)$  and  $q_\theta^{\alpha_{90}}(x)$ .

**Gaussian + Selective GMM** We combine the *Gaussian* method with a selective prediction mechanism. After training a Gaussian model, we run it on each image in train to extract ResNet feature vectors  $\{g(x_i)\}_{i=1}^N$ . We then utilize scikit-learn [70] to fit a GMM (4 components, full covariance) to these train feature vectors. To compute an uncertainty score  $\kappa_f(x)$  for a given input  $x$ , we extract  $g(x)$  and evaluate its likelihood according to the fitted GMM,  $\kappa_f(x) = -\text{GMM}(g(x))$ . The prediction  $\mu_\theta(x)$  and corresponding prediction interval  $C_\alpha(x)$  of the Gaussian model are then output iff  $\kappa_f(x) \leq \tau$ . To set the user-specified threshold  $\tau$ , we compute  $\kappa_f(x)$  on all images in val and pick the 95% quantile. This choice of  $\tau$  is motivated by the commonly reported FPR95 OOD detection metric, but  $\tau$  could be set using other approaches.

**Gaussian + Selective kNN** Identical to *Gaussian + Selective GMM*, but  $\kappa_f(x) = \text{kNN}(g(x))$ . Specifically, the uncertainty score  $\kappa_f(x)$  is computed by extracting  $g(x)$  and computing the average distance to its  $k = 10$  nearest neighbors among the train feature vectors  $\{g(x_i)\}_{i=1}^N$ . Following [34], we utilize the Annoy<sup>1</sup> approximate neighbors library, with cosine similarity as the distance metric.

**Gaussian + Selective Variance** Identical to *Gaussian + Selective GMM*, but  $\kappa_f(x) = \sigma_\theta^2(x)$  (the variance of the Gaussian model). This is used as a simple baseline for the two previous methods.

**Gaussian Ensemble + Selective GMM** We combine *Gaussian Ensemble* with a selective prediction mechanism. After training an ensemble of  $M = 5$  Gaussian models, we run each model on each image in train to extract  $M$  sets of ResNet feature vectors. For each model, we then fit a GMM to its set of train feature vectors. I.e., we fit  $M$  different GMMs. To compute an uncertainty score  $\kappa_f(x)$  for a given input  $x$ , we extract a feature vector and evaluate its likelihood according to the corresponding GMM, for each of the  $M$  models. Finally, we compute the mean of the GMM likelihoods,  $\kappa_f(x) = \frac{1}{M} \sum_{i=1}^M -\text{GMM}_i(g_i(x))$ .

**Gaussian Ensemble + Selective Ensemble Variance** Identical to *Gaussian Ensemble + Selective GMM*, but  $\kappa_f(x) = \frac{1}{M} \sum_{i=1}^M (\hat{\mu}(x) - \mu_{\theta_i}(x))^2$ , where  $\hat{\mu}(x) = \frac{1}{M} \sum_{i=1}^M \mu_{\theta_i}(x)$ . Hence, the variance of the ensemble means is used as the uncertainty score. This constitutes a simple baseline for the previous method.

---

<sup>1</sup><https://github.com/spotify/annoy>

All models are trained for 75 epochs using the ADAM optimizer [71]. The same hyperparameters are used for all datasets, and neither the training procedure nor the models are specifically tuned for any particular dataset. All experiments are implemented using PyTorch [72], and our complete implementation is made publicly available. All models were trained on individual NVIDIA TITAN Xp GPUs. On one such GPU, training 20 models on one dataset took approximately 24 hours. We chose to train all models based on a ResNet34 backbone because it is widely used across various applications, yet simple to implement and quite computationally inexpensive. The proposed benchmark and the evaluated uncertainty estimation methods are however entirely independent of this specific choice of DNN backbone architecture. Exploring the use of other more powerful models and evaluating how this affects the reliability of uncertainty estimation methods is an interesting direction which we leave for future work.

## 5 Related Work

Out-of-distribution robustness of DNNs is an active area of research [15, 73, 26, 74, 75, 76, 77, 78]. All these previous works do however focus exclusively on *classification* tasks. Moreover, they consider no uncertainty measures but instead evaluate only in terms of accuracy. While [79, 80] evaluate uncertainty calibration, they also just consider the classification setting. In contrast, evaluation of uncertainty estimation methods is our main focus, and we do this specifically for *regression*.

The main sources of inspiration for our work are [16] and [27]. While [16] propose an extensive benchmark with various real-world distribution shifts, it only contains a single regression dataset. Moreover, methods are evaluated solely in terms of predictive performance. [27] perform a comprehensive evaluation of uncertainty estimation methods under distribution shifts, but only consider classification tasks. Inspired by this, we thus propose our benchmark for evaluating reliability of *uncertainty estimation* methods under *real-world distribution shifts* in the *regression* setting. Most similar to our work is that of [81]. However, their benchmark contains just two regression datasets (tabular weather prediction and a complex vehicle motion prediction task), they only evaluate ensemble-based uncertainty estimation methods, and these methods are not evaluated in terms of calibration.

## 6 Results

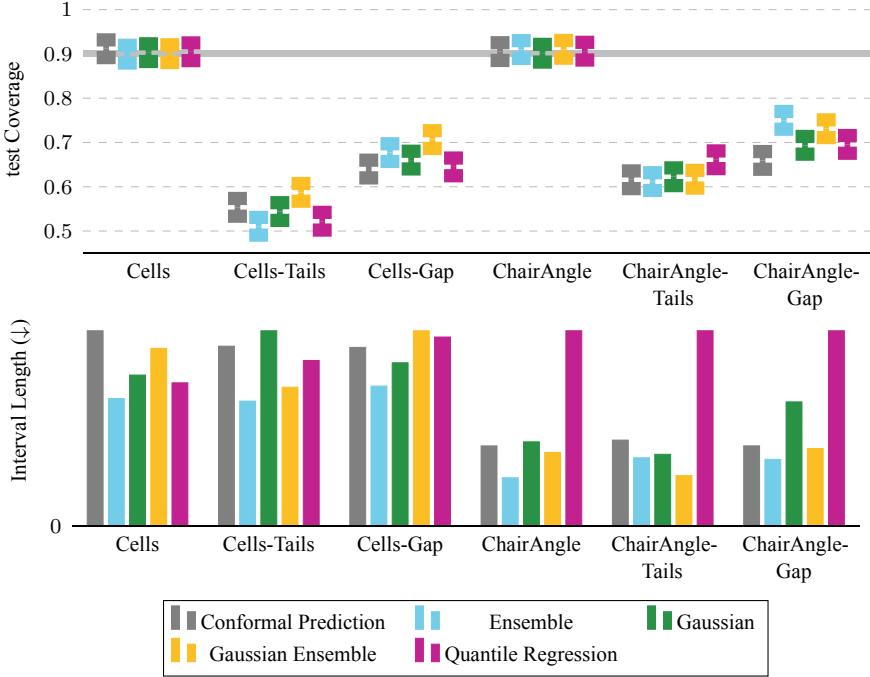
We evaluate the 10 methods specified in Section 4 on all 12 datasets from Section 3.1, according to the evaluation procedure described in Section 3.2. For each method we train 20 models, randomly select 5 of them for evaluation and report the averaged metrics. For the ensemble methods, we construct an ensemble by randomly selecting  $M = 5$  out of the 20 trained models, evaluate the ensemble and then repeat this 5 times in total. To ensure that the results do not depend on our specific choice of  $\alpha = 0.1$ , we also evaluate methods with two alternative miscoverage rates. While the main paper only contains results for  $\alpha = 0.1$ , we repeat most of the evaluation for  $\alpha = 0.2$  and  $\alpha = 0.05$  in Appendix B, observing very similar trends overall.

### 6.1 Common Uncertainty Estimation Methods

We start by evaluating the first five methods from Section 4, those which output predictions and corresponding 90% prediction intervals for all inputs. The results in terms of our main metric test coverage are presented in the upper part of Figure 2 for the synthetic datasets, and in Figure 3 for the six real-world datasets. In the lower parts of Figure 2 & 3, results in terms of average val interval length are presented. The complete results, including our other secondary metric val MAE, are provided in Table A1 - Table A12 in the appendix. Please note that, because we utilize a new benchmark consisting of custom datasets, we are not able to directly compare the MAE of our models with that of any previous work from the literature.

In Figure 2, the test coverage results on the first synthetic dataset *Cells* are found in the upper-left. As there is no distribution shift between train/val and test for this dataset, we use it as a baseline. We observe that all five methods have almost perfectly calibrated prediction intervals, i.e. they all obtain a test coverage very close to 90%. This is exactly the desired behaviour. On *Cells-Tails* however, on which we introduced a clear distribution shift, we observe in Figure 2 that the test coverage drops dramatically from the desired 90% for all five methods. Even the state-of-the-art uncertainty estimation method *Gaussian Ensemble* here becomes highly overconfident, as its test coverage drops down to  $\approx 59\%$ . On *Cells-Gap*, the test coverages are slightly closer to 90%, but all five methods are still highly overconfident. On the other synthetic dataset *ChairAngle*, we observe in Figure 2 that all five methods have almost perfectly calibrated prediction intervals. However, as we introduce clear distribution shifts on *ChairAngle-Tails* and *ChairAngle-Gap*, we can observe that the test coverage once again drops dramatically for all methods.

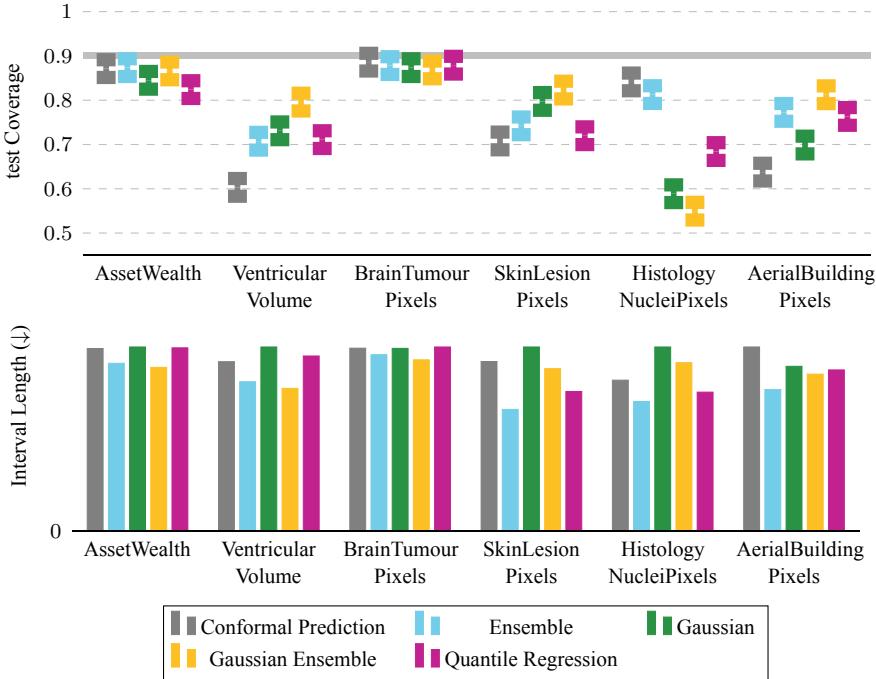
The results on the six real-world datasets are found in Figure 3. In the upper part, we observe that all five methods have quite well-calibrated prediction



**Figure 2:** Results for the five common regression uncertainty estimation methods (which output predictions and corresponding 90% prediction intervals for all inputs), on the six *synthetic* datasets. **Top:** Results in terms of our main metric test coverage. A perfectly calibrated method would achieve a test coverage of exactly 90%, as indicated by the solid line. **Bottom:** Results in terms of average val interval length.

intervals on *AssetWealth* and *BrainTumourPixels*, even though they all are consistently somewhat overconfident (test coverages of 82%-89%). On the four remaining datasets, the methods are in general more significantly overconfident. On *VentricularVolume*, we observe test coverages of 60%-80% for all methods, and on *SkinLesionPixels* the very best coverage is  $\approx 82\%$ . On *HistologyNucleiPixels*, most methods only obtain test coverages of 55%-70%, and on *AerialBuildingPixels* the very best coverage is  $\approx 81\%$ . In fact, not a single method actually reaches the desired 90% test coverage on any of these real-world datasets.

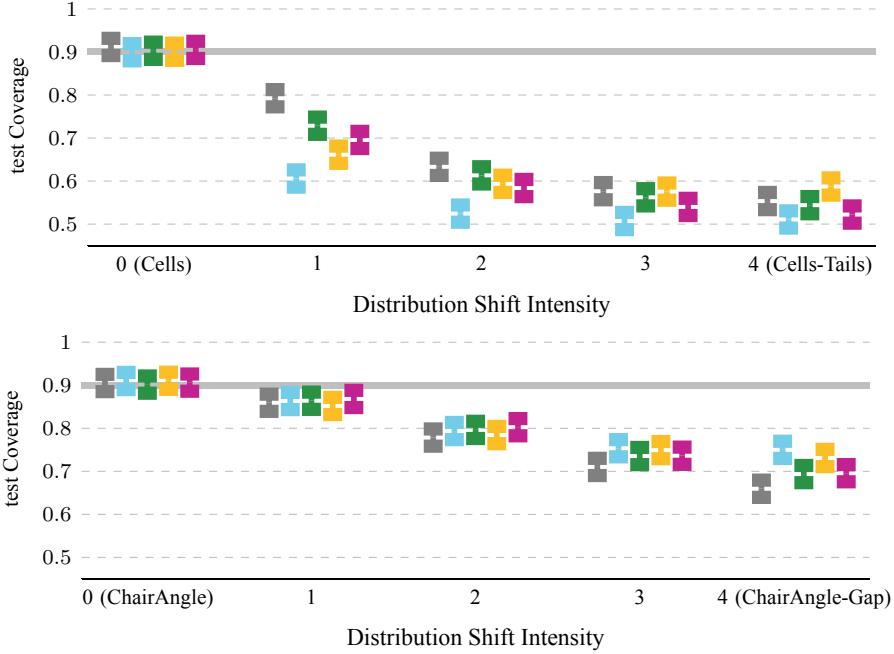
In terms of average val interval length, we observe in the lower parts of Figure 2 & 3 that *Ensemble* consistently produces smaller prediction intervals than *Conformal Prediction*. Moreover, the intervals of *Gaussian Ensemble* are usually smaller than those of *Gaussian*. When comparing the interval lengths of *Quantile Regression* and *Gaussian*, we observe no clear trend that is consistent across all datasets. Since the average interval lengths vary a lot between dif-



**Figure 3:** Results for the five common regression uncertainty estimation methods (which output predictions and corresponding 90% prediction intervals for all inputs), on the six *real-world* datasets. **Top:** Results in terms of our main metric test coverage. **Bottom:** Results in terms of average val interval length.

ferent datasets, Figure 2 & 3 only show relative comparisons of the methods. For absolute numerical scales, see Table A1 - Table A12.

To further study how the test coverage performance is affected by distribution shifts, we also apply the five methods to three additional variants of the *Cells* dataset. *Cells* has no difference in regression target range between train/val and test, whereas for *Cells-Tails* the target range is  $]50, 150]$  for train/val and  $[1, 200]$  for test. By creating three variants with intermediate target ranges, we thus obtain a sequence of five datasets with increasing degrees of distribution shifts, starting with *Cells* (no distribution shift) and ending with *Cells-Tails* (maximum distribution shift). The test coverage results on this sequence of datasets are presented in the upper part of Figure 4. We observe that as the degree of distribution shift is increased step-by-step, the test coverage also drops accordingly. The lower part of Figure 4 presents the results of a similar experiment, in which we construct a sequence of five datasets starting with *ChairAngle* (no distribution shift) and ending with *ChairAngle-Gap* (maximum distribution shift). Also in this case, we observe that the test coverage drops step-by-step along with the increased degree of distribution shift.

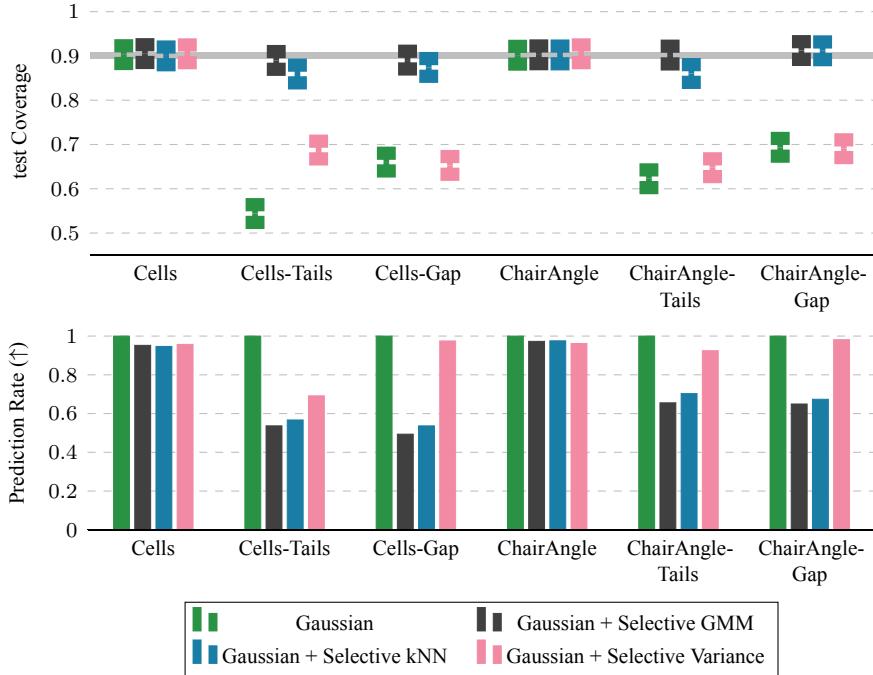


**Figure 4:** Test coverage results for the five common regression uncertainty estimation methods, on synthetic datasets with *increasing degrees of distribution shifts*. **Top:** From *Cells* (no distribution shift) to *Cells-Tails* (maximum distribution shift). **Bottom:** From *ChairAngle* to *ChairAngle-Gap*.

A study of the relative performance of the five methods on the real-world datasets, in terms of all three metrics (test coverage, average val interval length, val MAE), is finally presented in Figure A1 - Figure A3 in the appendix. One can clearly observe that *Ensemble* and *Gaussian Ensemble* achieve the best performance, thus indicating that ensembling multiple models indeed helps to improve the performance.

## 6.2 Selective Prediction Methods

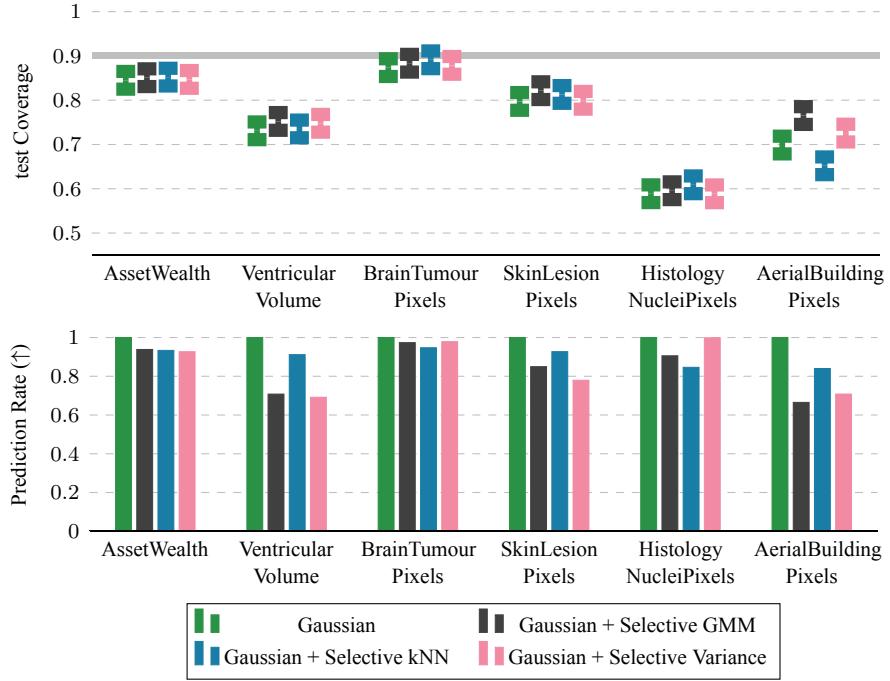
Next, we evaluate the methods with an added selective prediction mechanism. We start with the three methods based on *Gaussian*. The results in terms of test coverage and test prediction rate are available in Figure 5 for the synthetic datasets, and in Figure 6 for the six real-world datasets. While a complete evaluation of these methods also should include the average val interval length, we note that the selective prediction mechanism does not modify the intervals of the underlying *Gaussian* method (which already has been evaluated in terms of interval length in Section 6.1). Here, we therefore focus on the test coverage



**Figure 5:** Results for the three selective prediction methods based on *Gaussian*, on the six *synthetic* datasets. **Top:** Results in terms of our main metric test coverage. **Bottom:** Results in terms of test prediction rate (the proportion of test inputs for which a prediction actually is output).

and test prediction rate. Complete numerical results are provided in Table A1 - Table A12 in the appendix.

In the upper part of Figure 5, we observe that selective prediction based on feature-space density significantly improves the test coverage of *Gaussian* on the synthetic datasets. While *Gaussian* has well-calibrated prediction intervals only on *Cells* and *ChairAngle*, which are baseline datasets without any distribution shift, *Gaussian + Selective GMM* is almost perfectly calibrated across all six datasets. On *Cells-Tails*, for example, it improves the test coverage from  $\approx 54\%$  up to  $\approx 89\%$ . *Gaussian + Selective kNN* also significantly improves the test coverages, but not quite to the same extent. In the lower part of Figure 5, we can observe that when *Gaussian + Selective GMM* significantly improves the test coverage, there is also a clear drop in its test prediction rate. For example, the prediction rate drops from  $\approx 0.95$  on *Cells* down to  $\approx 0.54$  on *Cells-Tails*. By rejecting nearly 50% of all inputs as OOD in this case, *Gaussian + Selective GMM* can thus remain well-calibrated on the subset of test it actually outputs predictions for. In Figure 5, we also observe that *Gaussian + Selective Variance* only marginally improves the test coverage.



**Figure 6:** Results for the three selective prediction methods based on *Gaussian*, on the six *real-world* datasets. **Top:** Results in terms of test coverage. **Bottom:** Results in terms of test prediction rate.

While *Gaussian + Selective GMM* significantly improves the test coverage of *Gaussian* and has well-calibrated prediction intervals across the synthetic datasets, we observe in Figure 6 that this is not true for the six real-world datasets. *Gaussian + Selective GMM* does consistently improve the test coverage, but only marginally, and it still suffers from significant overconfidence in many cases. On *VentricularVolume*, for example, the test prediction rate of *Gaussian + Selective GMM* is as low as  $\approx 0.71$ , but the test coverage only improves from  $\approx 73\%$  to  $\approx 75\%$  compared to *Gaussian*.

For the two methods based on *Gaussian Ensemble*, the results are presented in Figure A4 & A5 in the appendix. Overall, we observe very similar trends. *Gaussian Ensemble + Selective GMM* significantly improves the test coverage of *Gaussian Ensemble* and is almost perfectly calibrated across the synthetic datasets. However, when it comes to the real-world datasets, it often remains significantly overconfident.

Finally, Figure A6 presents a relative comparison of the five selective prediction methods across all 12 datasets, in terms of average test coverage error (absolute difference between empirical and expected interval coverage) and

average test prediction rate. We observe that *Gaussian Ensemble + Selective GMM* achieves the best test coverage error, but also has the lowest test prediction rate. In fact, each improvement in terms of test coverage error also corresponds to a decrease in test prediction rate for these five methods, meaning that there seems to be an inherent trade-off between the two metrics.

## 7 Discussion

Let us now analyze the results from Section 6 in more detail, and discuss what we consider the most important findings and insights. First of all, we can observe that among the 10 considered methods, not a single one was close to producing perfectly calibrated prediction intervals across all 12 datasets. We thus conclude that our proposed benchmark indeed is challenging and interesting. Moreover, the results in Figure 2 & 3 demonstrate that while common uncertainty estimation methods are well calibrated when there is no distribution shift (*Cells* and *ChairAngle*), they can all break down and become highly overconfident in many realistic scenarios. This highlights the importance of employing sufficiently realistic and thus challenging benchmarks when evaluating uncertainty estimation methods. Otherwise, we might be lead to believe that methods will be more reliable during practical deployment than they actually are.

**Coverage Guarantees Might Instill a False Sense of Security** We also want to emphasize that *Conformal Prediction* and *Quantile Regression*<sup>2</sup> have theoretical coverage guarantees, but still are observed to become highly overconfident for many datasets in Figure 2 & 3. Since the guarantees depend on the assumption that all data points are exchangeable (true for i.i.d. data, for instance), which generally does not hold under distribution shifts, these results should actually not be surprising. The results are however a good reminder that we always need to be aware of the underlying assumptions, and whether or not they are likely to hold in common practical applications. Otherwise, such theoretical guarantees might just instill a false sense of security, making us trust methods more than we actually should.

**Clear Performance Differences between Synthetic and Real-World Data** We find it interesting that selective prediction based on feature-space density, in particular *Gaussian + Selective GMM*, works almost perfectly in terms of test coverage across the synthetic datasets (Figure 5), but fails to give significant improvements on the real-world datasets (Figure 6). The results on *VentricularVolume* are particularly interesting, as the prediction rate drops quite a lot without significantly improving the test coverage. This means that while a

---

<sup>2</sup>Since all prediction intervals are calibrated on val, we are using *Conformalized Quantile Regression* [21].

relatively large proportion of the test inputs are deemed OOD and thus rejected by the method, the test coverage is barely improved. On the synthetic datasets, there is a corresponding improvement in test coverage whenever the prediction rate drops significantly (Figure 5). It is not clear why such an obvious performance difference between synthetic and real-world datasets is observed. One possible explanation is that real-world data requires better models for  $p(x)$ , i.e. that the relatively simple approaches based on feature-space density not are sufficient. Properly explaining this performance difference is an important problem, but we will here leave this as an interesting direction for future work.

**OOD Uncertainty Scores Perform Well, but Not Well Enough** Comparing the selective prediction methods, we observe that *Gaussian + Selective GMM* consistently outperforms *Gaussian + Selective Variance* (Figure 5 & 6) and that *Gaussian Ensemble + Selective GMM* outperforms *Gaussian Ensemble + Selective Ensemble Variance* in most cases (Figure A4 & A5). Relative to common uncertainty estimation baselines, methods based on feature-space density thus achieve very strong performance. This is in line with the state-of-the-art OOD detection performance that has been demonstrated recently. In our results, we can however clearly observe that while feature-space methods perform well *relative* to common baselines, the resulting selective prediction methods are still overconfident in many cases – the *absolute* performance is still far from perfect. Using our benchmark, we are thus able to not only compare the relative performance of different OOD uncertainty scores, but also evaluate their performance in an absolute sense.

**Performance Differences among Real-World Datasets are Mostly Logical** When we compare the performance on the different real-world datasets in Figure 3, all methods are relatively well-calibrated on *BrainTumourPixels* and *AssetWealth*. For *BrainTumourPixels*, the train, val and test splits were created by randomly splitting the original set of MRI scans. The distribution shift between train/val and test is thus also fairly limited. For *AssetWealth* (satellite images from different African countries), the shift is likely quite limited at least compared to *AerialBuildingPixels* (satellite images from two different continents). Finally, the results for *HistologyNucleiPixels* are interesting, as this is the only dataset where *Conformal Prediction* clearly obtains the best test coverage. It is not clear why the methods which output prediction intervals of input-dependent length struggle on this particular dataset.

Finally, we should emphasize that while test coverage is our main metric, this by itself is not sufficient for a method to be said to “perform well” in a general sense. For example, a perfectly calibrated method with a low test prediction rate might not be particularly useful in practice. While even very low test prediction rates likely would be tolerated in many medical applications and other safety-critical domains (as long as the method stays perfectly calibrated), one can imagine more low-risk settings where this calibration versus prediction

rate trade-off is a lot less clear. Since not a single one of the evaluated methods was close to being perfectly calibrated across all 12 datasets, we did however mainly focus on analyzing the test coverage in this paper. If multiple methods had performed well in terms of test coverage, a more detailed analysis and discussion of the secondary metrics performance would have been necessary.

**The main actionable takeaways from our work can be summarized as:**

- All methods are well calibrated on baseline datasets with no distribution shift, but become highly overconfident in many realistic scenarios. Uncertainty estimation methods must therefore be evaluated using sufficiently challenging benchmarks. Otherwise, one might believe that methods will be more reliable during real-world deployment than they actually are.
- Conformal prediction methods have commonly promoted theoretical coverage guarantees, but these depend on an assumption that is unlikely to hold in many practical applications. Consequently, also these methods can become highly overconfident in realistic scenarios. If the underlying assumptions are not examined critically by practitioners, such theoretical guarantees risk instilling a false sense of security – making these models even less suitable for safety-critical deployment.
- The clear performance difference between synthetic and real-world datasets observed for selective prediction methods based on feature-space density is a very interesting direction for future work. If the reasons for this performance gap can be understood, an uncertainty estimation method that stays well calibrated across all datasets could potentially be developed.
- Selective prediction methods based on feature-space density perform well relative to other methods (as expected based on their state-of-the-art OOD detection performance), but are also overconfident in many cases. Only comparing the relative performance of different methods is therefore not sufficient. To track if actual progress is being made towards the ultimate goal of truly reliable uncertainty estimation methods, benchmarks must also evaluate method performance in an absolute sense.

## 8 Conclusion

We proposed an extensive benchmark for testing the reliability of regression uncertainty estimation methods under real-world distribution shifts. The benchmark consists of 8 publicly available image-based regression datasets with different types of challenging distribution shifts. We employed our benchmark to evaluate many of the most common uncertainty estimation methods, as well

as two state-of-the-art uncertainty scores from OOD detection. We found that while all methods are well calibrated when there is no distribution shift, they become highly overconfident on many of the benchmark datasets. Methods based on the OOD uncertainty scores performed well relative to other methods, but the absolute performance is still far from perfect. This uncovers important limitations of current regression uncertainty estimation methods. Our work thus serves as a challenge to the research community, to develop methods which actually produce calibrated prediction intervals across all benchmark datasets. To that end, future directions include exploring the use of more sophisticated models for  $p(x)$  within selective prediction – hopefully closing the performance gap between synthetic and real-world datasets – and employing alternative DNN backbone architectures. We hope that our benchmark will spur more work on how to develop truly reliable regression uncertainty estimation methods.

**Acknowledgments** This research was supported by the Swedish Research Council via the projects *NewLEADS – New Directions in Learning Dynamical Systems* (contract number: 621-2016-06079) and *Deep Probabilistic Regression – New Models and Learning Algorithms* (contract number: 2021-04301), and by the *Kjell & Märta Beijer Foundation*. We also thank Ludvig Hult and Dave Zachariah for providing helpful feedback on an early manuscript.

## References

- [1] R. Rothe, R. Timofte, and L. Van Gool. “Deep expectation of real and apparent age from a single image without facial landmarks.” In: *International Journal of Computer Vision (IJCV)* (2016).
- [2] H. Law and J. Deng. “Cornernet: Detecting objects as paired keypoints.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [3] B. Xiao, H. Wu, and Y. Wei. “Simple baselines for human pose estimation and tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [4] Z. Zhu, Q. Wang, B. Li, W. Wu, J. Yan, and W. Hu. “Distractor-aware siamese networks for visual object tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [5] S. Shi, X. Wang, and H. Li. “Pointrcnn: 3d object proposal generation and detection from point cloud.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.
- [6] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).

- [7] J. H. Cole, R. P. Poudel, D. Tsagkrasoulis, M. W. Caan, C. Steves, T. D. Spector, and G. Montana. “Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker.” In: *NeuroImage* (2017).
- [8] B. A. Jónsson, G. Björnsdóttir, T. Thorgeirsson, L. M. Ellingsen, G. B. Walters, D. Gudbjartsson, H. Stefansson, K. Stefansson, and M. Ulfarsdóttir. “Brain age prediction using deep learning uncovers associated sequence variants.” In: *Nature Communications* (2019).
- [9] W. Shi, G. Yan, Y. Li, H. Li, T. Liu, C. Sun, G. Wang, Y. Zhang, Y. Zou, and D. Wu. “Fetal brain age estimation and anomaly detection using attention-based deep ensembles with uncertainty.” In: *NeuroImage* (2020).
- [10] W. Xue, A. Islam, M. Bhaduri, and S. Li. “Direct multitype cardiac indices estimation via joint representation and regression learning.” In: *IEEE Transactions on Medical Imaging* (2017).
- [11] R. Poplin, A. V. Varadarajan, K. Blumer, Y. Liu, M. V. McConnell, G. S. Corrado, L. Peng, and D. R. Webster. “Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning.” In: *Nature Biomedical Engineering* (2018).
- [12] T. Langner, R. Strand, H. Ahlström, and J. Kullberg. “Large-scale biometry with interpretable neural network regression on UK Biobank body MRI.” In: *Scientific Reports* (2020).
- [13] T. Langner, F. K. Gustafsson, B. Avelin, R. Strand, H. Ahlström, and J. Kullberg. “Uncertainty-aware body composition analysis with deep regression ensembles on UK Biobank MRI.” In: *Computerized Medical Imaging and Graphics* (2021).
- [14] E. J. Topol. “High-performance medicine: the convergence of human and artificial intelligence.” In: *Nature Medicine* (2019).
- [15] D. Hendrycks and T. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.” In: *International Conference on Learning Representations (ICLR)*. 2019.
- [16] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. “Wilds: A benchmark of in-the-wild distribution shifts.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2021.
- [17] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. “Weight Uncertainty in Neural Network.” In: *International Conference on Machine Learning (ICML)*. 2015.
- [18] Y. Gal. “Uncertainty in Deep Learning.” PhD thesis. University of Cambridge, 2016.

- [19] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [20] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [21] Y. Romano, E. Patterson, and E. Candes. “Conformalized quantile regression.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).
- [22] T. Gneiting, F. Balabdaoui, and A. E. Raftery. “Probabilistic forecasts, calibration and sharpness.” In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (2007).
- [23] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. “On calibration of modern neural networks.” In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017.
- [24] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. 2009.
- [25] S. G. Finlayson, A. Subbaswamy, K. Singh, J. Bowers, A. Kupke, J. Zittrain, I. S. Kohane, and S. Saria. “The clinician and dataset shift in artificial intelligence.” In: *New England Journal of Medicine* (2021).
- [26] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, et al. “The many faces of robustness: A critical analysis of out-of-distribution generalization.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.
- [27] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [28] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2020.
- [29] Y. Geifman and R. El-Yaniv. “Selective classification for deep neural networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2017).
- [30] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou. “A Unified Survey on Anomaly, Novelty, Open-Set, and Out-of-Distribution Detection: Solutions and Future Challenges.” In: *Transactions on Machine Learning Research (TMLR)* (2022).

- [31] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal. “Deep Deterministic Uncertainty: A Simple Baseline.” In: *arXiv preprint* (2021).
- [32] G. Pleiss, A. Souza, J. Kim, B. Li, and K. Q. Weinberger. *Neural Network Out-of-Distribution Detection for Regression Tasks*. 2020.
- [33] Y. Sun, Y. Ming, X. Zhu, and Y. Li. “Out-of-distribution Detection with Deep Nearest Neighbors.” In: *International Conference on Machine Learning (ICML)*. 2022.
- [34] J. Kuan and J. Mueller. “Back to the Basics: Revisiting Out-of-Distribution Detection Baselines.” In: *arXiv preprint arXiv:2207.03061* (2022).
- [35] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman. “Inductive confidence machines for regression.” In: *European Conference on Machine Learning*. Springer. 2002.
- [36] H. Papadopoulos. “Inductive conformal prediction: Theory and application to neural networks.” In: *Tools in artificial intelligence*. Citeseer, 2008.
- [37] R. Koenker and G. Bassett Jr. “Regression quantiles.” In: *Econometrica: journal of the Econometric Society* (1978).
- [38] V. Jensen, F. M. Bianchi, and S. N. Anfinsen. “Ensemble Conformalized Quantile Regression for Probabilistic Time Series Forecasting.” In: *arXiv preprint arXiv:2202.08756* (2022).
- [39] I. Steinwart and A. Christmann. “Estimating conditional quantiles with the help of the pinball loss.” In: *Bernoulli* (2011).
- [40] K. Chua, R. Calandra, R. McAllister, and S. Levine. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.
- [41] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.
- [42] R. M. Neal. “Bayesian learning for neural networks.” PhD thesis. University of Toronto, 1995.
- [43] Y.-A. Ma, T. Chen, and E. Fox. “A complete recipe for stochastic gradient MCMC.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2015.
- [44] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning.” In: *International Conference on Machine Learning (ICML)*. 2016.

- [45] R. Herbei and M. H. Wegkamp. “Classification with reject option.” In: *The Canadian Journal of Statistics/La Revue Canadienne de Statistique* (2006).
- [46] W. Jiang, Y. Zhao, and Z. Wang. “Risk-Controlled Selective Prediction for Regression Deep Neural Network Models.” In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020.
- [47] J. Serrà, D. Álvarez, V. Gómez, O. Slizovskaia, J. F. Núñez, and J. Luque. “Input Complexity and Out-of-distribution Detection with Likelihood-based Generative Models.” In: *International Conference on Learning Representations (ICLR)*. 2020.
- [48] X. Ding, Y. Wang, Z. Xu, W. J. Welch, and Z. J. Wang. “CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation.” In: *International Conference on Learning Representations (ICLR)*. 2021.
- [49] X. Ding, Y. Wang, Z. Xu, W. J. Welch, and Z. J. Wang. “Continuous Conditional Generative Adversarial Networks for Image Generation: Novel Losses and Label Input Mechanisms.” In: *arXiv preprint arXiv:2011.07466* (2020).
- [50] C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke. “Using publicly available satellite imagery and deep learning to understand economic well-being in Africa.” In: *Nature communications* (2020).
- [51] D. Ouyang, B. He, A. Ghorbani, N. Yuan, J. Ebinger, C. P. Langlotz, P. A. Heidenreich, R. A. Harrington, D. H. Liang, E. A. Ashley, et al. “Video-based AI for beat-to-beat assessment of cardiac function.” In: *Nature* (2020).
- [52] A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. Van Ginneken, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, et al. “A large annotated medical image dataset for the development and evaluation of segmentation algorithms.” In: *arXiv preprint arXiv:1902.09063* (2019).
- [53] M. Antonelli, A. Reinke, S. Bakas, K. Farahani, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers, et al. “The medical segmentation decathlon.” In: *Nature Communications* (2022).
- [54] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki, et al. “Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge.” In: *arXiv preprint arXiv:1811.02629* (2018).

- [55] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos. “Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features.” In: *Scientific data* (2017).
- [56] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, et al. “The multimodal brain tumor image segmentation benchmark (BRATS).” In: *IEEE Transactions on Medical Imaging* (2014).
- [57] P. Tschandl, C. Rosendahl, and H. Kittler. “The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions.” In: *Scientific data* (2018).
- [58] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot. “HoVer-Net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images.” In: *Medical Image Analysis* (2019).
- [59] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane, and A. Sethi. “A dataset and a technique for generalized nuclear segmentation for computational pathology.” In: *IEEE Transactions on Medical Imaging* (2017).
- [60] P. Naylor, M. Laé, F. Reyal, and T. Walter. “Segmentation of nuclei in histopathology images by deep regression of the distance map.” In: *IEEE Transactions on Medical Imaging* (2018).
- [61] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. “Can semantic labeling methods generalize to any city? The Inria aerial image labeling benchmark.” In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE. 2017.
- [62] E. Ilg, O. Cicek, S. Galessio, A. Klein, O. Makansi, F. Hutter, and T. Bro. “Uncertainty estimates and multi-hypotheses networks for optical flow.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [63] V. Kuleshov, N. Fenner, and S. Ermon. “Accurate uncertainties for deep learning using calibrated regression.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2018.
- [64] P. Cui, W. Hu, and J. Zhu. “Calibrated reliable regression using maximum mean discrepancy.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2020).
- [65] Y. Chung, W. Neiswanger, I. Char, and J. Schneider. “Beyond pinball loss: Quantile methods for calibrated uncertainty quantification.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).
- [66] D. Levi, L. Gispan, N. Giladi, and E. Fetaya. “Evaluating and calibrating uncertainty prediction in regression tasks.” In: *Sensors* (2022).

- [67] E. Pickering and T. P. Sapsis. “Structure and Distribution Metric for Quantifying the Quality of Uncertainty: Assessing Gaussian Processes, Deep Neural Nets, and Deep Neural Operators for Regression.” In: *arXiv preprint arXiv:2203.04515* (2022).
- [68] A. Fisch, T. S. Jaakkola, and R. Barzilay. “Calibrated Selective Classification.” In: *Transactions on Machine Learning Research (TMLR)* (2022).
- [69] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [70] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research (JMLR)* (2011).
- [71] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014).
- [72] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [73] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song. “Natural adversarial examples.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.
- [74] P. Izmailov, P. Nicholson, S. Lotfi, and A. G. Wilson. “Dangers of bayesian model averaging under covariate shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).
- [75] H. Yao, Y. Wang, S. Li, L. Zhang, W. Liang, J. Zou, and C. Finn. “Improving out-of-distribution robustness via selective augmentation.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2022.
- [76] L. Schwinn, L. Bungert, A. Nguyen, R. Raab, F. Pulsmeyer, D. Precup, B. Eskofier, and D. Zanca. “Improving Robustness against Real-World and Worst-Case Distribution Shifts through Decision Region Quantification.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2022.
- [77] O. Wiles, S. Gowal, F. Stimberg, S.-A. Rebuffi, I. Ktena, K. D. Dvijotham, and A. T. Cemgil. “A Fine-Grained Analysis on Distribution Shift.” In: *International Conference on Learning Representations (ICLR)*. 2022.
- [78] M. M. Zhang, S. Levine, and C. Finn. “MEMO: Test Time Robustness via Adaptation and Augmentation.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

- [79] S. Zaidi, A. Zela, T. Elsken, C. C. Holmes, F. Hutter, and Y. Teh. “Neural ensemble search for uncertainty estimation and dataset shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).
- [80] D. Hendrycks, A. Zou, M. Mazeika, L. Tang, B. Li, D. Song, and J. Steinhardt. “Pixmix: Dreamlike pictures comprehensively improve safety measures.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.
- [81] A. Malinin et al. “Shifts: A Dataset of Real Distributional Shift Across Multiple Large-Scale Tasks.” In: *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*. 2021.
- [82] F. Wenzel, A. Dittadi, P. Gehler, C.-J. Simon-Gabriel, M. Horn, D. Zietlow, D. Kurnert, C. Russell, T. Brox, B. Schiele, et al. “Assaying out-of-distribution generalization in transfer learning.” In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).
- [83] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal. “Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty.” In: *arXiv preprint arXiv:2102.11582* (2021).
- [84] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. “Spectral Normalization for Generative Adversarial Networks.” In: *International Conference on Learning Representations (ICLR)*. 2018.

# Appendix

## A Dataset Details

More detailed descriptions of the 12 datasets from Section 3.1 are provided below.

**Cells** Given a synthetic fluorescence microscopy image  $x$ , the task is to predict the number of cells  $y$  in the image. We utilize the Cell-200 dataset from [48, 49], consisting of 200 000 grayscale images of size  $64 \times 64$ . We randomly draw 10 000 train images, 2 000 val images and 10 000 test images. Thus, there is no distribution shift between train/val and test. We therefore use this as a baseline dataset.

**Cells-Tails** We create a variant of *Cells* with a clear distribution shift between train/val and test. For the 10 000 train images and 2 000 val images, the regression targets  $y$  are limited to the range  $]50, 150]$ . For the 10 000 test images, the targets instead lie in the full original range  $[1, 200]$ .

**Cells-Gap** Another variant of *Cells* with a clear distribution shift between train/val and test. For the 10 000 train images and 2 000 val images, the regression targets  $y$  are limited to  $[1, 50] \cup [150, 200]$ . For the 10 000 test images, the targets instead lie in the full original range  $[1, 200]$ .

**ChairAngle** Given a synthetic image  $x$  of a rendered chair model, the task is to predict the yaw angle  $y$  of the chair. We utilize the RC-49 dataset from [48, 49], which contains  $64 \times 64$  RGB images of different chair models rendered at 899 yaw angles ranging from  $0.1^\circ$  to  $89.9^\circ$ , with step size  $0.1^\circ$ . We randomly split their training set and obtain 17 640 train images and 4 410 val images. By sub-sampling their test set we also get 11 225 test images. There is no obvious distribution shift between train/val and test, and we therefore use this as a second baseline dataset.

**ChairAngle-Tails** We create a variant of *ChairAngle* with a clear distribution shift between train/val and test. For train and val, we limit the regression targets  $y$  to the range  $]15, 75[$ . For test, the targets instead lie in the full original range  $]0, 90[$ . We obtain 11 760 train images, 2 940 val images and 11 225 test images.

**ChairAngle-Gap** Another variant of *ChairAngle* with a clear distribution shift between train/val and test. For the 11 760 train images and 2 940 val images, the regression targets  $y$  are limited to  $]0, 30] \cup [60, 90[$ . For the 11 225 test images, the targets instead lie in the full original range  $]0, 90[$ .

**AssetWealth** Given a satellite image  $x$  (8 image channels), the task is to predict the asset wealth index  $y$  of the region. We utilize the PovertyMap-Wilds

dataset [16], which is a variant of the dataset collected by [50]. We use the training, validation-ID and test-OOD subsets of the data, giving us 9 797 train images, 1 000 val images and 3 963 test images. We resize the images from size  $224 \times 224$  to  $64 \times 64$ . Train/val and test contain satellite images from disjoint sets of African countries, creating a distribution shift.

**VentricularVolume** Given an echocardiogram image  $x$  of a human heart, the task is to predict the volume  $y$  of the left ventricle. We utilize the EchoNet-Dynamic dataset by [51], which contains 10 030 echocardiogram videos. Each video captures a complete cardiac cycle and is labeled with measurements of the left ventricular volume at two separate time points, representing end-systole (at the end of contraction - smaller volume) and end-diastole (just before contraction - larger volume). For each video, we extract just one of these volume measurements along with the corresponding video frame. To create a clear distribution shift between train/val and test, we select the end systolic volume (smaller volume) for train and val, but the end diastolic volume (larger volume) for test. We utilize the provided dataset splits, giving us 7 460 train images, 1 288 val images and 1 276 test images. We resize the images from size  $112 \times 112$  to  $64 \times 64$ .

**BrainTumourPixels** Given an image slice  $x$  of a brain MRI scan, the task is to predict the number of pixels  $y$  in the image which are labeled as brain tumour. We utilize the brain tumour dataset of the medical segmentation decathlon [52, 53], which is a subset of the data used in the 2016 and 2017 BraTS challenges [54, 55, 56]. The dataset contains 484 brain MRI scans with corresponding tumour segmentation masks. We split these scans 80%/20%/20% into train, val and test sets. The scans are 3D volumes of size  $240 \times 240 \times 155$ . We convert each scan into 155 image slices of size  $240 \times 240$ , and create a regression target for each image by counting the number of labeled brain tumour pixels. We then also remove all images which contain no tumour pixels. The original image slices have 4 channels (FLAIR, T1w, T1gd, T2w), but we only use the first three and convert the slices into standard RGB images. This gives us 20 614 train images, 6 116 val images and 6 252 test images. We also resize the images from size  $240 \times 240$  to  $64 \times 64$ .

**SkinLesionPixels** Given a dermatoscopic image  $x$  of a pigmented skin lesion, the task is to predict the number of pixels  $y$  in the image which are labeled as lesion. We utilize the HAM10000 dataset by [57], which contains 10 015 dermatoscopic images with corresponding skin lesion segmentation masks. HAM10000 consists of four different sub-datasets, three of which (ViDIR Legacy, ViDIR Current and ViDIR MoleMax) were collected in Austria, while the fourth sub-dataset (Rosendahl) was collected in Australia. To create a clear distribution shift between train/val and test, we use the Australian sub-dataset (Rosendahl) as our test set. After randomly splitting the remaining images 85%/15% into train and val sets, we obtain 6 592 train images, 1 164

val images and 2 259 test images. We then create a regression target for each image by counting the number of labeled skin lesion pixels. We also resize the images from size  $450 \times 600$  to  $64 \times 64$ .

**HistologyNucleiPixels** Given an H&E stained histology image  $x$ , the task is to predict the number of pixels  $y$  in the image which are labeled as nuclei. We utilize the CoNSeP dataset by [58], along with the pre-processed versions they provide of the Kumar [59] and TNBC [60] datasets. The datasets contain large H&E stained image tiles (of size  $1\,000 \times 1\,000$  or  $512 \times 512$ ) at  $40\times$  objective magnification, with corresponding nuclear segmentation masks. The three datasets were collected at different hospitals/institutions, with differing procedures for specimen preservation and staining. By using CoNSeP and Kumar for train/val and TNBC for test, we thus obtain a clear distribution shift. From the large image tiles, we extract  $64 \times 64$  patches via regular gridding, and create a regression target for each image patch by counting the number of labeled nuclei pixels. We then also remove all images which contain no nuclei pixels. In the end, we obtain 10 808 train images, 2 702 val images and 2 267 test images.

**AerialBuildingPixels** Given an aerial image  $x$ , the task is to predict the number of pixels  $y$  in the image which are labeled as building. We utilize the Inria aerial image labeling dataset [61], which contains 180 large aerial images with corresponding building segmentation masks. The images are of size  $5\,000 \times 5\,000$ , and are captured at five different geographical locations. We use the images from two densely populated American cities (Austin and Chicago) for train/val, and the images from a more rural European area (West Tyrol, Austria) for test, thus obtaining a clear distribution shift. We first resize the images to size  $1\,000 \times 1\,000$ , and then extract  $64 \times 64$  patches via regular gridding. We also create a regression target for each image patch by counting the number of labeled building pixels. After removal of all images which contain no building pixels, we finally obtain 11 184 train images, 2 797 val images and 3 890 test images.

The additional variants of *Cells* and *ChairAngle* in Figure 4 are specified as follows. *Cells*: no difference in regression target range between train/val and test. *Cells-Tails*: target range  $]50, 150]$  for train/val,  $[1, 200]$  for test. We create three versions with intermediate target ranges (1:  $[37.5, 163.5]$ , 2:  $[25, 176]$ , 3:  $[12.5, 188.5]$ ) for test. *ChairAngle*: no difference in target range between train/val and test. *ChairAngle-Gap*: target range  $]0, 30[ \cup ]60, 90[$  for train/val,  $]0, 90[$  for test. We create three versions with intermediate target ranges (1:  $]0, 33.75[ \cup ]56.25, 90[$ , 2:  $]0, 37.5[ \cup ]52.5, 90[$ , 3:  $]0, 41.25[ \cup ]48.75, 90[$ ) for test.

## B Additional Results & Method Variations

*Please note that the results in Table A1 - Table A42 not are rounded/truncated to only significant digits.*

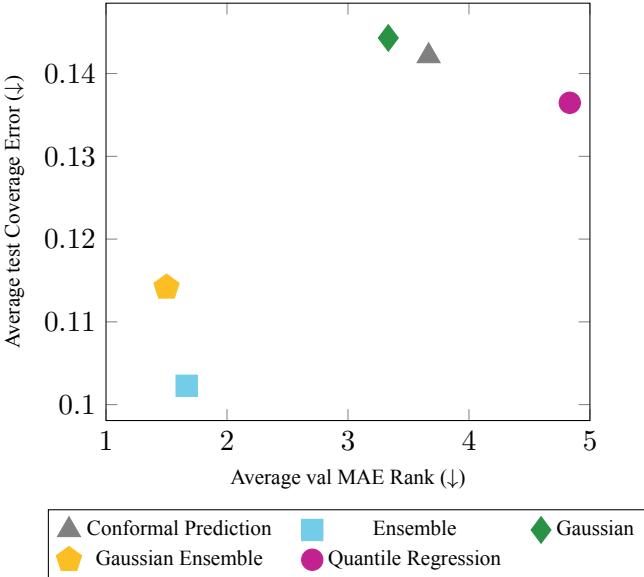
To ensure that the test coverage results do not depend on our specific choice of studying 90% prediction intervals ( $\alpha = 0.1$ ), we repeat most of the evaluation for two alternative miscoverage rates  $\alpha$ . Specifically, we redo the evaluation of 6/10 methods on 9/12 datasets with 80% ( $\alpha = 0.2$ ) and 95% ( $\alpha = 0.05$ ) prediction intervals. The results for 80% prediction intervals are given in Figure A7 - Figure A10 and Table A13 - Table A21, while the results for 95% prediction intervals are given in Figure A11 - Figure A14 and Table A22 - Table A30. We observe very similar trends overall. For example, all methods still have almost perfectly calibrated prediction intervals on *Cells* and *ChairAngle*, i.e. they all obtain a test coverage very close to 80%/95%, but only *Gaussian + Selective GMM* remains well-calibrated on *Cells-Tails* and *ChairAngle-Gap*. With the exception of *BrainTumourPixels*, all methods are also significantly overconfident on all the real-world datasets.

Figure A15 shows test coverage results for the five common regression uncertainty estimation methods on the *AerialBuildingPixels* dataset, and on two versions with different test sets. For all three datasets, train/val contains images from Austin and Chicago. For *AerialBuildingPixels*, test contains images from West Tyrol, Austria. For *AerialBuildingPixels-Kitsap*, test instead contains images from Kitsap County, WA. For *AerialBuildingPixels-Vienna*, test contains images from Vienna, Austria. Intuitively, the distribution shift between train/val and test could potentially be smaller for *AerialBuildingPixels-Kitsap* and *AerialBuildingPixels-Vienna* than for the original *AerialBuildingPixels*, but we observe no clear trends in Figure A15.

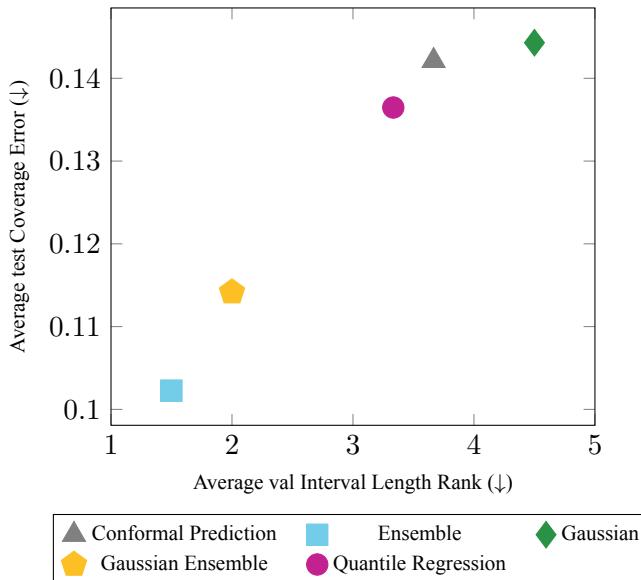
Figure A16 & A17 present a study in which we aim to relate the test coverage performance to a quantitative measure of distribution shift (“distance” between the distributions of train/val and test), complementing our qualitative discussion in the *Performance Differences among Real-World Datasets are Mostly Logical* paragraph of Section 7. How to quantify the level of distribution shift in real-world datasets is however far from obvious, see e.g. Appendix E.1 in [82]. We here explore if the difference in regression accuracy (MAE) on val and test can be adopted as such a measure, extending the approach by [82] to our regression setting. We compute the average test coverage error for the five common regression uncertainty estimation methods, whereas the val/test MAE is for the *Conformal Prediction* method (standard direct regression models). The results for the six synthetic datasets are presented in Figure A16, and for the six real-world datasets in Figure A17. We observe that, in general, a larger distribution shift measure does indeed seem to correspond to worse test

coverage performance. Among the 12 datasets, *HistologyNucleiPixels* is the only one that quite clearly breaks this general trend.

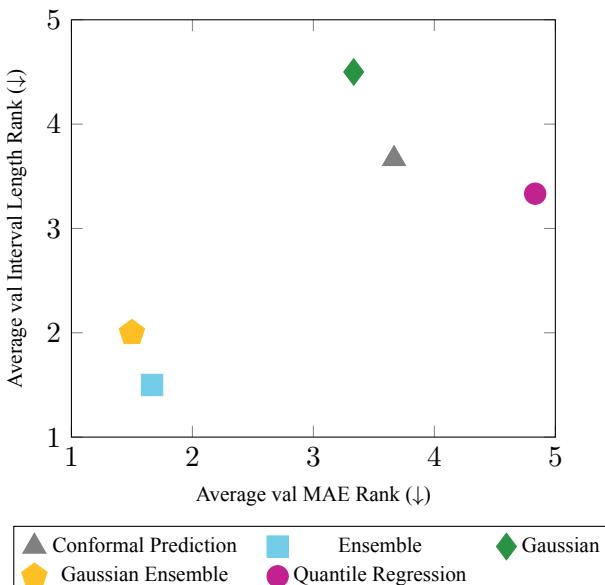
Apart from the main comparison of the 10 methods specified in Section 4, we also evaluate a few method variations. The results for these experiments are provided in Table A31 - Table A42. For *Gaussian + Selective GMM*, we vary the number of GMM mixture components from the standard  $k = 4$  to  $k = 2$  and  $k = 8$ , but observe no particularly consistent or significant trends in the results. Similarly, we vary the number of neighbors from the standard  $k = 10$  to  $k = 5$  and  $k = 20$  for *Gaussian + Selective kNN*, but observe no clear trends here either. For *Gaussian + Selective kNN*, we also explore replacing the cosine similarity distance metric with L2 distance, again obtaining very similar results. Following [83], we finally add spectral normalization [84] for further feature-space regularization, but observe no significant improvements for *Gaussian + Selective GMM*.



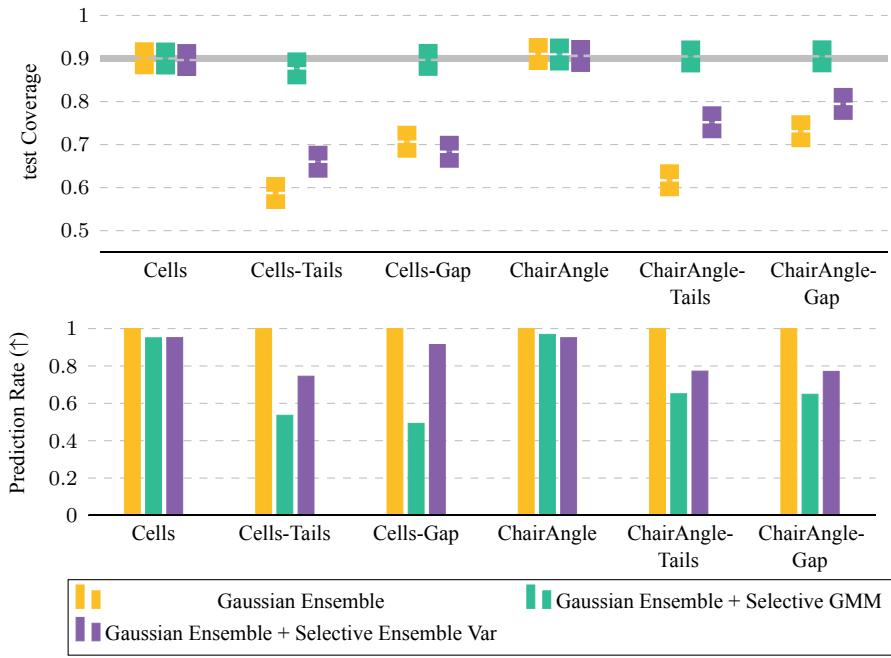
**Figure A1:** Performance comparison of the five common regression uncertainty estimation methods on the six *real-world* datasets, in terms of average test coverage error and average val MAE rank (the five methods are ranked 1 - 5 in terms of val MAE on each dataset, and then the average rank is computed). *Ensemble* and *Gaussian Ensemble* achieve the best performance.



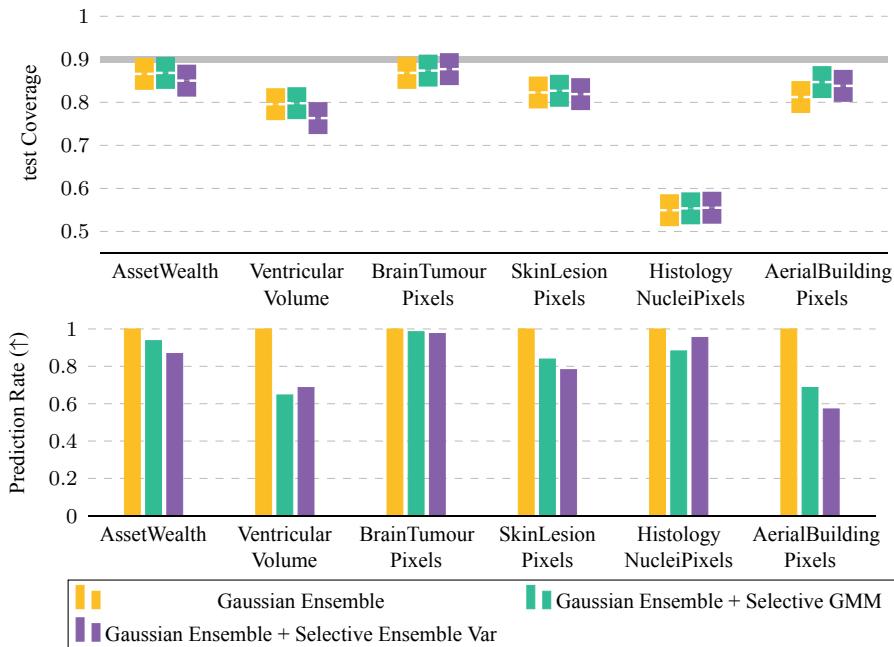
**Figure A2:** Performance comparison of the five common regression uncertainty estimation methods on the six *real-world* datasets, in terms of average test coverage error and average val interval length rank (the five methods are ranked 1 - 5 in terms of val interval length on each dataset, and then the average rank is computed). *Ensemble* and *Gaussian Ensemble* achieve the best performance.



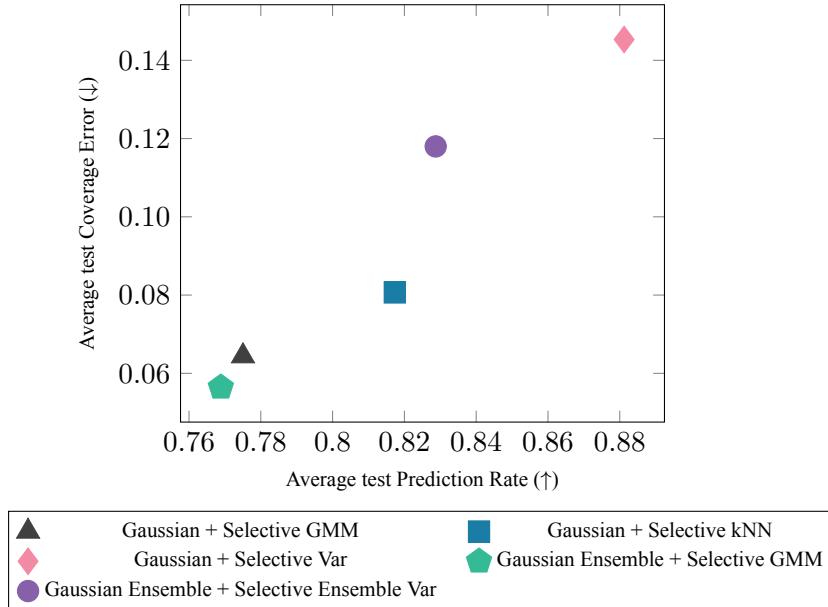
**Figure A3:** Performance comparison of the five common regression uncertainty estimation methods on the six *real-world* datasets, in terms of average val interval length rank and average val MAE rank. *Ensemble* and *Gaussian Ensemble* achieve the best performance.



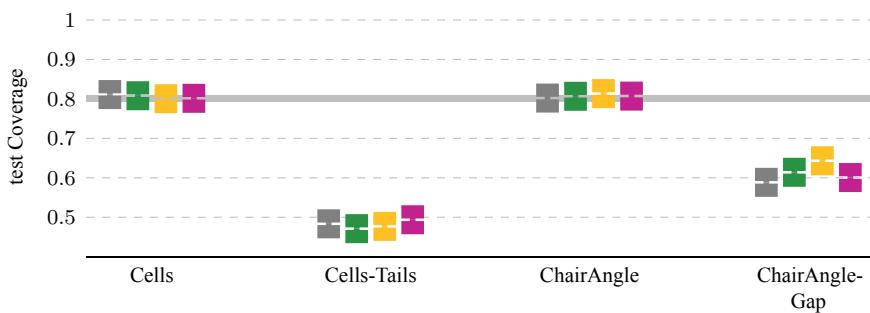
**Figure A4:** Results for the two selective prediction methods based on *Gaussian Ensemble*, on the six *synthetic* datasets.



**Figure A5:** Results for the two selective prediction methods based on *Gaussian Ensemble*, on the *real-world* datasets.



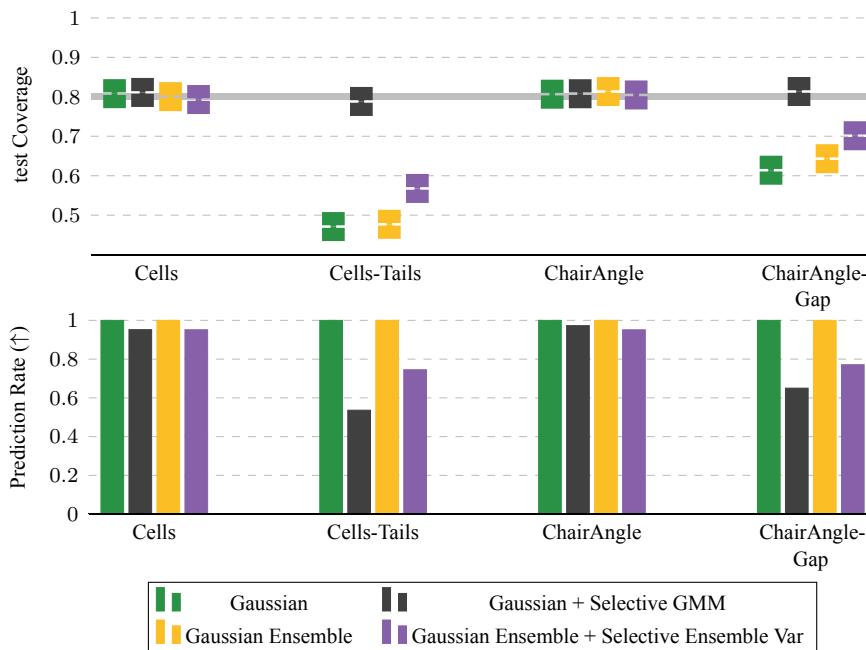
**Figure A6:** Performance comparison of the selective prediction methods across all 12 datasets, in terms of average test coverage error and test prediction rate. *Gaussian Ensemble + Selective GMM* achieves the best coverage error, but also has the lowest prediction rate. For these five methods, each improvement in terms of coverage error also corresponds to a decrease in prediction rate.



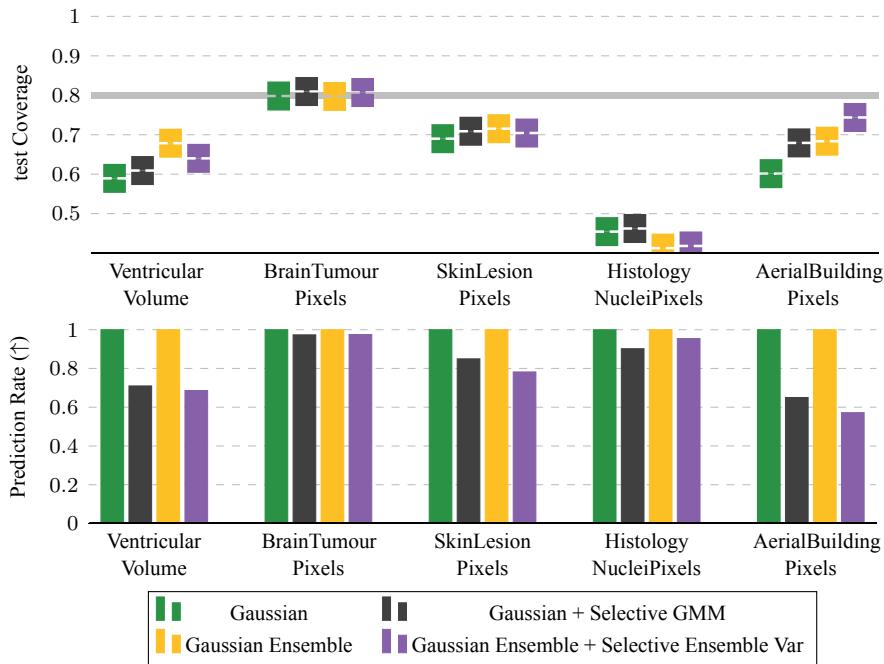
**Figure A7:** Miscoverage rate  $\alpha = 0.2$ : Results in terms of test coverage for four of the common regression uncertainty estimation methods (Conformal Prediction, Gaussian, Gaussian Ensemble, Quantile Regression), on four of the *synthetic* datasets. See Table A13 - Table A16 for other metrics.



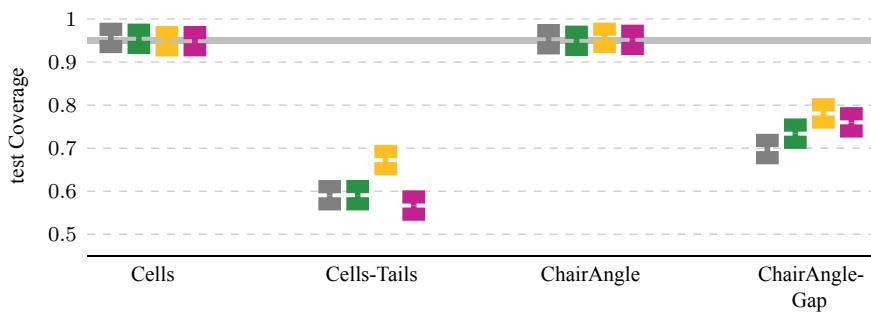
**Figure A8:** Miscoverage rate  $\alpha = 0.2$ : Results in terms of test coverage for four of the common regression uncertainty estimation methods (Conformal Prediction, Gaussian, Gaussian Ensemble, Quantile Regression), on five of the *real-world* datasets. See Table A17 - Table A21 for other metrics.



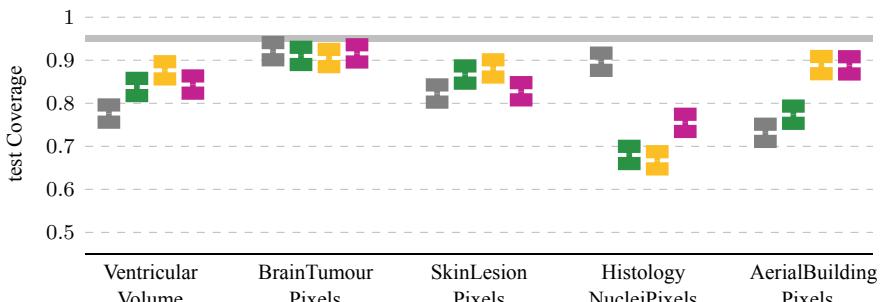
**Figure A9:** Miscoverage rate  $\alpha = 0.2$ : Results for two of the selective prediction methods, on four of the *synthetic* datasets. See Table A13 - Table A16 for other metrics.



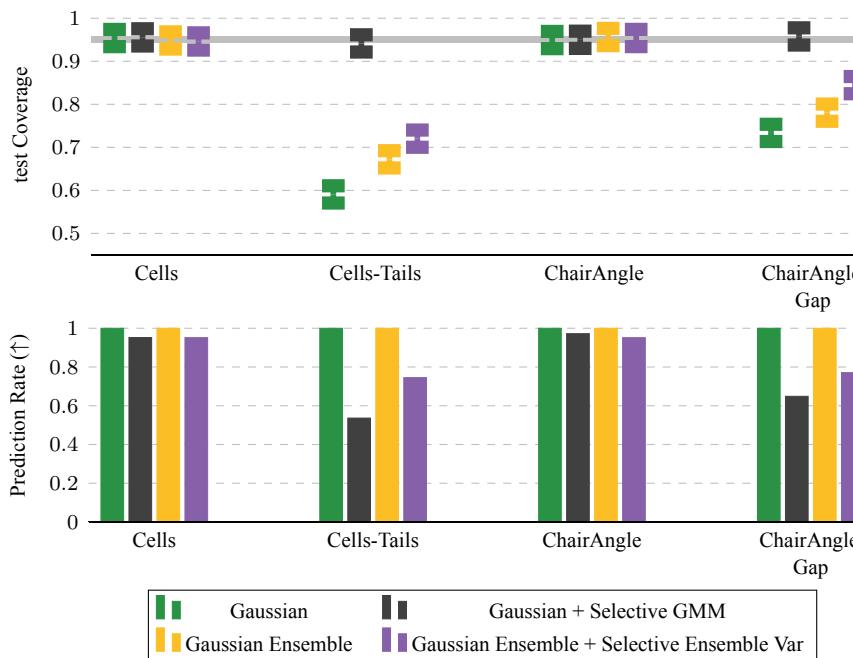
**Figure A10:** Miscoverage rate  $\alpha = 0.2$ : Results for two of the selective prediction methods, on five of the *real-world* datasets. See Table A17 - Table A21 for other metrics.



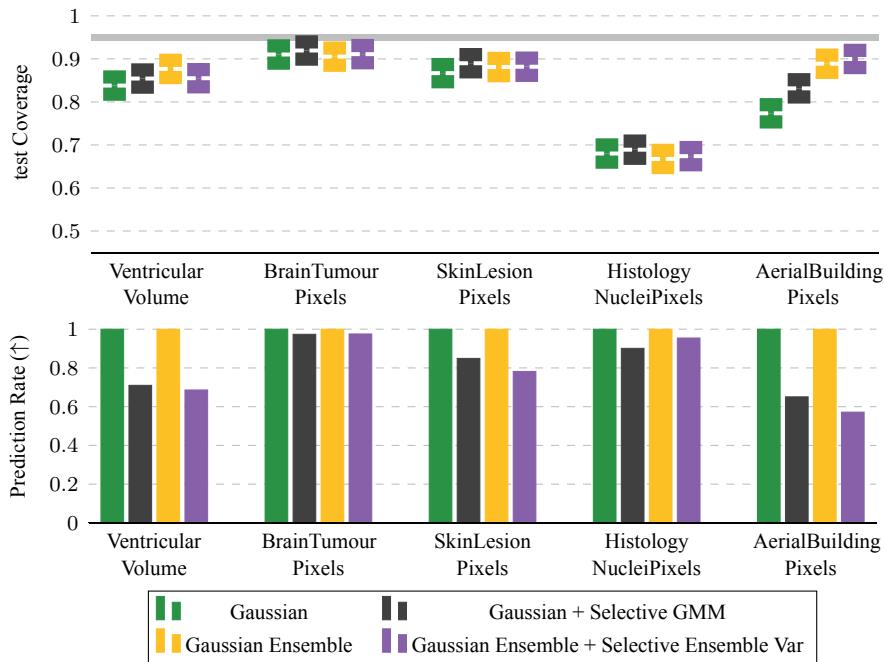
**Figure A11:** Miscoverage rate  $\alpha = 0.05$ : Results in terms of test coverage for four of the common regression uncertainty estimation methods (Conformal Prediction, Gaussian, Gaussian Ensemble, Quantile Regression), on four of the *synthetic* datasets. See Table A22 - Table A25 for other metrics.



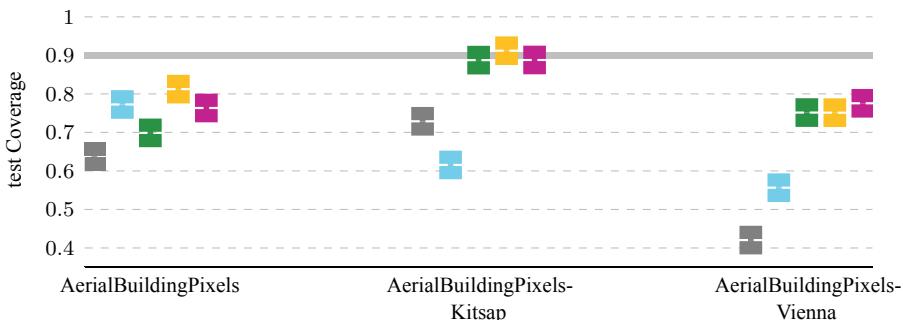
**Figure A12:** Miscoverage rate  $\alpha = 0.05$ : Results in terms of test coverage for four of the common regression uncertainty estimation methods (Conformal Prediction, Gaussian, Gaussian Ensemble, Quantile Regression), on five of the *real-world* datasets. See Table A26 - Table A30 for other metrics.



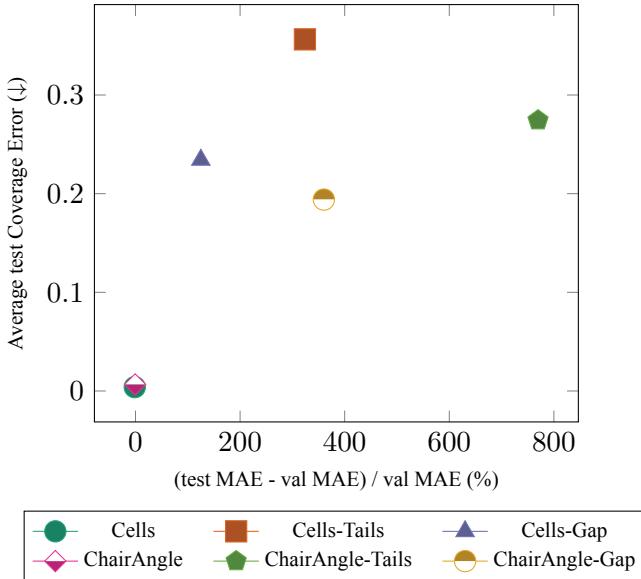
**Figure A13:** Miscoverage rate  $\alpha = 0.05$ : Results for two of the selective prediction methods, on four of the *synthetic* datasets. See Table A22 - Table A25 for other metrics.



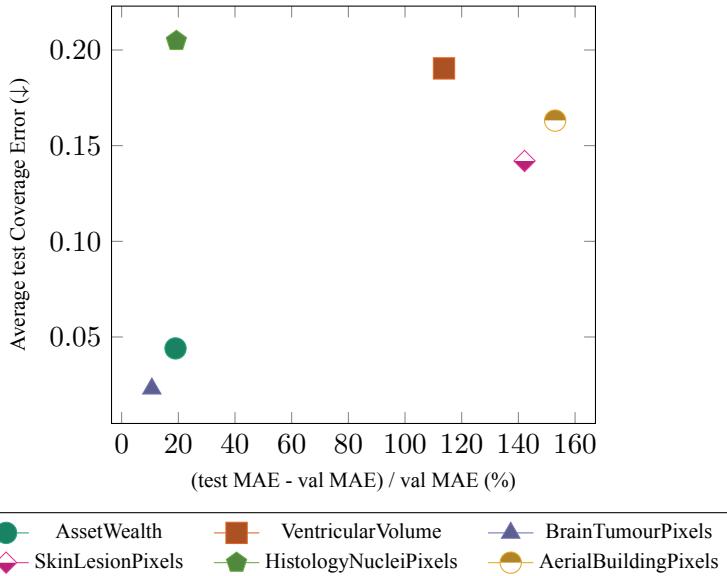
**Figure A14:** Miscoverage rate  $\alpha = 0.05$ : Results for two of the selective prediction methods, on five of the *real-world* datasets. See Table A26 - Table A30 for other metrics.



**Figure A15:** Results for the five common regression uncertainty estimation methods (Conformal Prediction, Ensemble, Gaussian, Gaussian Ensemble, Quantile Regression) on the *AerialBuildingPixels* dataset, and on two versions with different test sets. For all three datasets, train/val contains images from Austin and Chicago. For *AerialBuildingPixels*, test contains images from West Tyrol, Austria. For *AerialBuildingPixels-Kitsap*, test instead contains images from Kitsap County, WA. For *AerialBuildingPixels-Vienna*, test contains images from Vienna, Austria.



**Figure A16:** Using the difference in regression accuracy (MAE) on val and test as a quantitative measure of distribution shift in each dataset (inspired by [82], extended to our regression setting), and comparing this to the test coverage performance. Results for the six *synthetic* datasets. In general, a larger distribution shift measure corresponds to worse test coverage performance.



**Figure A17:** Exactly the same comparison as in Figure A16, but for the six *real-world* datasets. A larger distribution shift measure generally corresponds to worse test coverage performance also in this case. The *HistologyNucleiPixels* dataset is somewhat of an outlier.

Paper VII – How Reliable is Your Regression Model’s Uncertainty?

**Table A1:** Complete results on the *Cells* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	$4.03121 \pm 2.78375$	$18.8216 \pm 11.8302$	$0.9117 \pm 0.00275173$	1.0
Ensemble	$2.22525 \pm 0.471309$	$12.2881 \pm 2.78795$	$0.8994 \pm 0.00689986$	1.0
Gaussian	$3.61704 \pm 1.13624$	$14.5492 \pm 4.44927$	$0.90286 \pm 0.00594629$	1.0
Gaussian Ensemble	$2.78757 \pm 1.16951$	$17.1285 \pm 4.33367$	$0.90062 \pm 0.0027571$	1.0
Quantile Regression	$3.70405 \pm 0.81647$	$13.8023 \pm 2.19085$	$0.90486 \pm 0.006771$	1.0
Gaussian + Selective GMM	$3.61704 \pm 1.13624$	$14.5492 \pm 4.44927$	$0.905241 \pm 0.00448315$	$0.95216 \pm 0.00290558$
Gaussian + Selective kNN	$3.61704 \pm 1.13624$	$14.5492 \pm 4.44927$	$0.900069 \pm 0.00677637$	$0.94656 \pm 0.00470472$
Gaussian + Selective Variance	$3.61704 \pm 1.13624$	$14.5492 \pm 4.44927$	$0.904839 \pm 0.00795449$	$0.95712 \pm 0.00200938$
Gaussian Ens + Selec GMM	$2.78757 \pm 1.16951$	$17.1285 \pm 4.33367$	$0.899999 \pm 0.00311328$	$0.95066 \pm 0.0023105$
Gaussian Ens + Selec Ens Var	$2.78757 \pm 1.16951$	$17.1285 \pm 4.33367$	$0.896465 \pm 0.0015705$	$0.95156 \pm 0.00215277$

**Table A2:** Complete results on the *Cells-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	$3.56733 \pm 1.0818$	$14.2128 \pm 4.23582$	$0.55356 \pm 0.0299096$	1.0
Ensemble	$1.83461 \pm 0.178937$	$9.8697 \pm 1.6672$	$0.50078 \pm 0.007608$	1.0
Gaussian	$4.05446 \pm 1.33153$	$15.4321 \pm 4.98796$	$0.54402 \pm 0.0347525$	1.0
Gaussian Ensemble	$2.40691 \pm 0.580524$	$10.9696 \pm 1.70051$	$0.5874 \pm 0.0479263$	1.0
Quantile Regression	$3.32571 \pm 1.24578$	$13.0848 \pm 3.54239$	$0.52222 \pm 0.0284102$	1.0
Gaussian + Selective GMM	$4.05446 \pm 1.33153$	$15.4321 \pm 4.98796$	$0.889825 \pm 0.0193021$	$0.53654 \pm 0.0101012$
Gaussian + Selective kNN	$4.05446 \pm 1.33153$	$15.4321 \pm 4.98796$	$0.859179 \pm 0.0255173$	$0.56692 \pm 0.0127107$
Gaussian + Selective Variance	$4.05446 \pm 1.33153$	$15.4321 \pm 4.98796$	$0.687475 \pm 0.0659807$	$0.6914 \pm 0.0651196$
Gaussian Ens + Selec GMM	$2.40691 \pm 0.580524$	$10.9696 \pm 1.70051$	$0.877068 \pm 0.0222502$	$0.53604 \pm 0.0125903$
Gaussian Ens + Selec Ens Var	$2.40691 \pm 0.580524$	$10.9696 \pm 1.70051$	$0.660212 \pm 0.0325832$	$0.74512 \pm 0.0438128$

**Table A3:** Complete results on the *Cells-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	$3.67702 \pm 1.33587$	$17.1152 \pm 6.49211$	$0.64002 \pm 0.0620287$	1.0
Ensemble	$2.7261 \pm 0.705274$	$13.4061 \pm 2.40513$	$0.6771 \pm 0.0544788$	1.0
Gaussian	$3.53089 \pm 1.0619$	$15.6396 \pm 6.23458$	$0.66028 \pm 0.114703$	1.0
Gaussian Ensemble	$3.46118 \pm 0.95429$	$18.707 \pm 4.13287$	$0.7066 \pm 0.0317638$	1.0
Quantile Regression	$4.75328 \pm 1.73499$	$18.108 \pm 4.21716$	$0.64488 \pm 0.12875$	1.0
Gaussian + Selective GMM	$3.53089 \pm 1.0619$	$15.6396 \pm 6.23458$	$0.890569 \pm 0.00953089$	$0.49372 \pm 0.0025926$
Gaussian + Selective kNN	$3.53089 \pm 1.0619$	$15.6396 \pm 6.23458$	$0.874032 \pm 0.0432364$	$0.53646 \pm 0.0139864$
Gaussian + Selective Variance	$3.53089 \pm 1.0619$	$15.6396 \pm 6.23458$	$0.652766 \pm 0.117192$	$0.9748 \pm 0.000940213$
Gaussian Ens + Selec GMM	$3.46118 \pm 0.95429$	$18.707 \pm 4.13287$	$0.896848 \pm 0.0127879$	$0.49278 \pm 0.00192914$
Gaussian Ens + Selec Ens Var	$3.46118 \pm 0.95429$	$18.707 \pm 4.13287$	$0.68326 \pm 0.0299799$	$0.9147 \pm 0.0182427$

**Table A4:** Complete results on the *ChairAngle* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	$0.289127 \pm 0.081643$	$1.08752 \pm 0.202247$	$0.905265 \pm 0.00339915$	1.0
Ensemble	$0.189858 \pm 0.0548452$	$0.788401 \pm 0.137465$	$0.909915 \pm 0.003399101$	1.0
Gaussian	$0.376692 \pm 0.171928$	$1.37757 \pm 0.382191$	$0.901577 \pm 0.00308472$	1.0
Gaussian Ensemble	$0.361834 \pm 0.165348$	$1.2044 \pm 0.442422$	$0.910414 \pm 0.00316257$	1.0
Quantile Regression	$0.851253 \pm 0.544717$	$3.19741 \pm 1.9751$	$0.906209 \pm 0.00204038$	1.0
Gaussian + Selective GMM	$0.376692 \pm 0.171928$	$1.37757 \pm 0.382191$	$0.902482 \pm 0.00436054$	$0.972739 \pm 0.00191733$
Gaussian + Selective kNN	$0.376692 \pm 0.171928$	$1.37757 \pm 0.382191$	$0.90222 \pm 0.0045694$	$0.975465 \pm 0.00201785$
Gaussian + Selective Variance	$0.376692 \pm 0.171928$	$1.37757 \pm 0.382191$	$0.904805 \pm 0.00765098$	$0.961782 \pm 0.0193133$
Gaussian Ens + Selec GMM	$0.361834 \pm 0.165348$	$1.2044 \pm 0.442422$	$0.9093 \pm 0.00297263$	$0.969033 \pm 0.000940447$
Gaussian Ens + Selec Ens Var	$0.361834 \pm 0.165348$	$1.2044 \pm 0.442422$	$0.905905 \pm 0.00336481$	$0.951359 \pm 0.00453556$

**Table A5:** Complete results on the *ChairAngle-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	$0.358162 \pm 0.168257$	$1.31102 \pm 0.531401$	$0.615804 \pm 0.00527695$	1.0
Ensemble	$0.21931 \pm 0.0596705$	$1.04074 \pm 0.245846$	$0.611706 \pm 0.00345611$	1.0
Gaussian	$0.241214 \pm 0.091736$	$1.09417 \pm 0.382907$	$0.622592 \pm 0.00714065$	1.0
Gaussian Ensemble	$0.13365 \pm 0.0189933$	$0.769172 \pm 0.0814039$	$0.617016 \pm 0.0063698$	1.0
Quantile Regression	$0.820934 \pm 0.653268$	$2.9815 \pm 2.15473$	$0.660953 \pm 0.0379624$	1.0
Gaussian + Selective GMM	$0.241214 \pm 0.091736$	$1.09417 \pm 0.382907$	$0.901946 \pm 0.00382993$	$0.655448 \pm 0.001058$
Gaussian + Selective kNN	$0.241214 \pm 0.091736$	$1.09417 \pm 0.382907$	$0.860311 \pm 0.00617031$	$0.703038 \pm 0.00691227$
Gaussian + Selective Variance	$0.241214 \pm 0.091736$	$1.09417 \pm 0.382907$	$0.647559 \pm 0.0360179$	$0.924383 \pm 0.0751205$
Gaussian Ens + Selec GMM	$0.13365 \pm 0.0189933$	$0.769172 \pm 0.0814039$	$0.904807 \pm 0.00394199$	$0.651314 \pm 0.0013893$
Gaussian Ens + Selec Ens Var	$0.13365 \pm 0.0189933$	$0.769172 \pm 0.0814039$	$0.751845 \pm 0.0131765$	$0.772401 \pm 0.00878839$

**Table A6:** Complete results on the *ChairAngle-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.35034 $\pm$ 0.161819	1.35 $\pm$ 0.547089	0.659546 $\pm$ 0.00916108	1.0
Ensemble	0.22898 $\pm$ 0.0853825	1.1222 $\pm$ 0.174147	0.749951 $\pm$ 0.0155327	1.0
Gaussian	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.69363 $\pm$ 0.0345876	1.0
Gaussian Ensemble	0.226352 $\pm$ 0.0677413	1.30588 $\pm$ 0.136235	0.731065 $\pm$ 0.0126216	1.0
Quantile Regression	0.639151 $\pm$ 0.296536	3.29137 $\pm$ 1.53269	0.695697 $\pm$ 0.0413613	1.0
Gaussian + Selective GMM	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.91215 $\pm$ 0.00604745	0.649372 $\pm$ 0.00334919
Gaussian + Selective kNN	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.911574 $\pm$ 0.00376608	0.673764 $\pm$ 0.0113432
Gaussian + Selective Variance	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.690436 $\pm$ 0.0366025	0.981292 $\pm$ 0.0152406
Gaussian Ens + Selec GMM	0.226352 $\pm$ 0.0677413	1.30588 $\pm$ 0.136235	0.905039 $\pm$ 0.00229837	0.648624 $\pm$ 0.00094348
Gaussian Ens + Selec Ens Var	0.226352 $\pm$ 0.0677413	1.30588 $\pm$ 0.136235	0.794531 $\pm$ 0.014876	0.7711 $\pm$ 0.0191172

**Table A7:** Complete results on the *AssetWealth* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.346532 $\pm$ 0.00578306	1.5838 $\pm$ 0.0318086	0.87136 $\pm$ 0.0090944	1.0
Ensemble	0.320002 $\pm$ 0.00264202	1.45568 $\pm$ 0.0129312	0.87348 $\pm$ 0.00598963	1.0
Gaussian	0.367501 $\pm$ 0.0416437	1.597 $\pm$ 0.207599	0.844966 $\pm$ 0.0456831	1.0
Gaussian Ensemble	0.3295 $\pm$ 0.00783906	1.42071 $\pm$ 0.0485437	0.866162 $\pm$ 0.00711672	1.0
Quantile Regression	0.404279 $\pm$ 0.0683226	1.58957 $\pm$ 0.161689	0.823921 $\pm$ 0.0313343	1.0
Gaussian + Selective GMM	0.367501 $\pm$ 0.0416437	1.597 $\pm$ 0.207599	0.850824 $\pm$ 0.047533	0.93838 $\pm$ 0.0170443
Gaussian + Selective kNN	0.367501 $\pm$ 0.0416437	1.597 $\pm$ 0.207599	0.852107 $\pm$ 0.0474734	0.933586 $\pm$ 0.0203541
Gaussian + Selective Variance	0.367501 $\pm$ 0.0416437	1.597 $\pm$ 0.207599	0.846981 $\pm$ 0.0430247	0.926874 $\pm$ 0.0159098
Gaussian Ens + Selec GMM	0.3295 $\pm$ 0.00783906	1.42071 $\pm$ 0.0485437	0.868444 $\pm$ 0.00727312	0.937371 $\pm$ 0.0138213
Gaussian Ens + Selec Ens Var	0.3295 $\pm$ 0.00783906	1.42071 $\pm$ 0.0485437	0.85047 $\pm$ 0.00397082	0.868282 $\pm$ 0.0299149

**Table A8:** Complete results on the *VentricularVolume* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	11.2471 $\pm$ 0.201399	47.4505 $\pm$ 1.556	0.603135 $\pm$ 0.0125842	1.0
Ensemble	10.2476 $\pm$ 0.113707	41.8445 $\pm$ 0.750822	0.707367 $\pm$ 0.00877743	1.0
Gaussian	12.7238 $\pm$ 1.52197	51.566 $\pm$ 3.52739	0.730878 $\pm$ 0.0619045	1.0
Gaussian Ensemble	10.1141 $\pm$ 0.180661	39.9817 $\pm$ 1.91308	0.795768 $\pm$ 0.0231243	1.0
Quantile Regression	12.4944 $\pm$ 0.676265	49.0448 $\pm$ 3.33314	0.710972 $\pm$ 0.045171	1.0
Gaussian + Selective GMM	12.7238 $\pm$ 1.52197	51.566 $\pm$ 3.52739	0.752046 $\pm$ 0.0529087	0.707994 $\pm$ 0.0208741
Gaussian + Selective kNN	12.7238 $\pm$ 1.52197	51.566 $\pm$ 3.52739	0.735105 $\pm$ 0.0612413	0.911599 $\pm$ 0.0447475
Gaussian + Selective Variance	12.7238 $\pm$ 1.52197	51.566 $\pm$ 3.52739	0.747868 $\pm$ 0.0569984	0.691693 $\pm$ 0.0341016
Gaussian Ens + Selec GMM	10.1141 $\pm$ 0.180661	39.9817 $\pm$ 1.91308	0.798094 $\pm$ 0.0166674	0.646865 $\pm$ 0.0366201
Gaussian Ens + Selec Ens Var	10.1141 $\pm$ 0.180661	39.9817 $\pm$ 1.91308	0.763412 $\pm$ 0.0286772	0.686207 $\pm$ 0.0431835

**Table A9:** Complete results on the *BrainTumourPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	21.3163 $\pm$ 0.45997	93.7169 $\pm$ 2.99061	0.885925 $\pm$ 0.00395743	1.0
Ensemble	21.1133 $\pm$ 0.210209	90.3825 $\pm$ 0.825858	0.878183 $\pm$ 0.00318872	1.0
Gaussian	21.0625 $\pm$ 0.358012	93.6284 $\pm$ 2.29916	0.873544 $\pm$ 0.00871896	1.0
Gaussian Ensemble	20.5336 $\pm$ 0.211421	87.7414 $\pm$ 0.818979	0.868426 $\pm$ 0.00320847	1.0
Quantile Regression	22.0348 $\pm$ 0.697606	94.3249 $\pm$ 3.07265	0.879079 $\pm$ 0.00380396	1.0
Gaussian + Selective GMM	21.0625 $\pm$ 0.358012	93.6284 $\pm$ 2.29916	0.883515 $\pm$ 0.0138279	0.973576 $\pm$ 0.0187252
Gaussian + Selective kNN	21.0625 $\pm$ 0.358012	93.6284 $\pm$ 2.29916	0.891264 $\pm$ 0.00734602	0.947185 $\pm$ 0.0178434
Gaussian + Selective Variance	21.0625 $\pm$ 0.358012	93.6284 $\pm$ 2.29916	0.878666 $\pm$ 0.00735197	0.978791 $\pm$ 0.00694672
Gaussian Ens + Selec GMM	20.5336 $\pm$ 0.211421	87.7414 $\pm$ 0.818979	0.873824 $\pm$ 0.00466519	0.985349 $\pm$ 0.00461984
Gaussian Ens + Selec Ens Var	20.5336 $\pm$ 0.211421	87.7414 $\pm$ 0.818979	0.877211 $\pm$ 0.00302057	0.975368 $\pm$ 0.00323397

**Table A10:** Complete results on the *SkinLesionPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	107.514 $\pm$ 1.87464	492.922 $\pm$ 14.1865	0.708012 $\pm$ 0.00917777	1.0
Ensemble	99.3156 $\pm$ 0.997867	353.842 $\pm$ 4.73577	0.742098 $\pm$ 0.00852236	1.0
Gaussian	105.417 $\pm$ 1.15178	535.139 $\pm$ 215.446	0.797255 $\pm$ 0.0270734	1.0
Gaussian Ensemble	100.639 $\pm$ 0.464183	472.14 $\pm$ 85.2654	0.822931 $\pm$ 0.0134328	1.0
Quantile Regression	113.076 $\pm$ 3.20875	405.904 $\pm$ 4.75158	0.719788 $\pm$ 0.0213826	1.0
Gaussian + Selective GMM	105.417 $\pm$ 1.15178	535.139 $\pm$ 215.446	0.821515 $\pm$ 0.0137705	0.849579 $\pm$ 0.0206181
Gaussian + Selective kNN	105.417 $\pm$ 1.15178	535.139 $\pm$ 215.446	0.813027 $\pm$ 0.0306586	0.927047 $\pm$ 0.00830718
Gaussian + Selective Variance	105.417 $\pm$ 1.15178	535.139 $\pm$ 215.446	0.799821 $\pm$ 0.0150855	0.77946 $\pm$ 0.059169
Gaussian Ens + Selec GMM	100.639 $\pm$ 0.464183	472.14 $\pm$ 85.2654	0.826921 $\pm$ 0.00719649	0.839132 $\pm$ 0.0108815
Gaussian Ens + Selec Ens Var	100.639 $\pm$ 0.464183	472.14 $\pm$ 85.2654	0.819129 $\pm$ 0.00910128	0.782116 $\pm$ 0.0109776

**Table A11:** Complete results on the *HistologyNucleiPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	217.887 $\pm$ 3.71766	993.665 $\pm$ 18.3282	0.841288 $\pm$ 0.0128314	1.0
Ensemble	197.078 $\pm$ 1.99489	853.361 $\pm$ 14.1904	0.812704 $\pm$ 0.00664074	1.0
Gaussian	211.795 $\pm$ 10.0239	1211.83 $\pm$ 396.946	0.588796 $\pm$ 0.0912794	1.0
Gaussian Ensemble	196.785 $\pm$ 2.14454	1108.53 $\pm$ 145.034	0.54936 $\pm$ 0.05995	1.0
Quantile Regression	227.895 $\pm$ 9.37415	914.909 $\pm$ 45.995	0.684076 $\pm$ 0.0428704	1.0
Gaussian + Selective GMM	211.795 $\pm$ 10.0239	1211.83 $\pm$ 396.946	0.59554 $\pm$ 0.0956742	0.90569 $\pm$ 0.0453555
Gaussian + Selective kNN	211.795 $\pm$ 10.0239	1211.83 $\pm$ 396.946	0.608929 $\pm$ 0.0805954	0.846228 $\pm$ 0.0482832
Gaussian + Selective Variance	211.795 $\pm$ 10.0239	1211.83 $\pm$ 396.946	0.588598 $\pm$ 0.0913593	0.998765 $\pm$ 0.00134954
Gaussian Ens + Selec GMM	196.785 $\pm$ 2.14454	1108.53 $\pm$ 145.034	0.553725 $\pm$ 0.0642274	0.882488 $\pm$ 0.0156395
Gaussian Ens + Selec Ens Var	196.785 $\pm$ 2.14454	1108.53 $\pm$ 145.034	0.555149 $\pm$ 0.0620052	0.954477 $\pm$ 0.0256246

**Table A12:** Complete results on the *AerialBuildingPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	235.417 $\pm$ 7.16096	1038.45 $\pm$ 51.5565	0.637584 $\pm$ 0.0691102	1.0
Ensemble	199.206 $\pm$ 2.76543	798.312 $\pm$ 6.80986	0.772494 $\pm$ 0.0227288	1.0
Gaussian	217.877 $\pm$ 1.72493	929.562 $\pm$ 47.6606	0.698766 $\pm$ 0.0662263	1.0
Gaussian Ensemble	208.487 $\pm$ 1.03581	885.349 $\pm$ 27.8584	0.812339 $\pm$ 0.0617386	1.0
Quantile Regression	242.284 $\pm$ 6.0122	909.191 $\pm$ 24.9528	0.763342 $\pm$ 0.0902358	1.0
Gaussian + Selective GMM	217.877 $\pm$ 1.72493	929.562 $\pm$ 47.6606	0.76535 $\pm$ 0.0388677	0.652082 $\pm$ 0.0990489
Gaussian + Selective kNN	217.877 $\pm$ 1.72493	929.562 $\pm$ 47.6606	0.651867 $\pm$ 0.0771154	0.840103 $\pm$ 0.0463989
Gaussian + Selective Variance	217.877 $\pm$ 1.72493	929.562 $\pm$ 47.6606	0.725714 $\pm$ 0.0779575	0.708226 $\pm$ 0.103803
Gaussian Ens + Selec GMM	208.487 $\pm$ 1.03581	885.349 $\pm$ 27.8584	0.847101 $\pm$ 0.038266	0.686787 $\pm$ 0.0278702
Gaussian Ens + Selec Ens Var	208.487 $\pm$ 1.03581	885.349 $\pm$ 27.8584	0.838352 $\pm$ 0.0312236	0.571928 $\pm$ 0.0402727

**Table A13:** Miscoverage rate  $\alpha = 0.2$ : Results on the *Cells* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	4.03121 $\pm$ 2.78375	14.042 $\pm$ 10.2791	0.811 $\pm$ 0.0100485	1.0
Gaussian	3.61704 $\pm$ 1.13624	11.544 $\pm$ 3.59513	0.808 $\pm$ 0.00859046	1.0
Gaussian Ensemble	2.78757 $\pm$ 1.16951	13.1298 $\pm$ 3.34997	0.80044 $\pm$ 0.00316582	1.0
Quantile Regression	4.40758 $\pm$ 1.47411	13.3314 $\pm$ 4.39511	0.80152 $\pm$ 0.00667005	1.0
Gaussian + Selective GMM	3.61704 $\pm$ 1.13624	11.544 $\pm$ 3.59513	0.811076 $\pm$ 0.00507358	0.95216 $\pm$ 0.00280043
Gaussian Ens + Selective Ens Var	2.78757 $\pm$ 1.16951	13.1298 $\pm$ 3.34997	0.793124 $\pm$ 0.00576659	0.95156 $\pm$ 0.00215277

**Table A14:** Miscoverage rate  $\alpha = 0.2$ : Results on the *Cells-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	3.56733 $\pm$ 1.0818	11.3841 $\pm$ 3.54242	0.48332 $\pm$ 0.0337429	1.0
Gaussian	4.05446 $\pm$ 1.33153	12.7086 $\pm$ 4.39552	0.47106 $\pm$ 0.0319074	1.0
Gaussian Ensemble	2.40691 $\pm$ 0.580524	8.62187 $\pm$ 1.5177	0.4768 $\pm$ 0.0461347	1.0
Quantile Regression	4.20398 $\pm$ 1.61554	13.1241 $\pm$ 4.61204	0.49358 $\pm$ 0.0341114	1.0
Gaussian + Selective GMM	4.05446 $\pm$ 1.33153	12.7086 $\pm$ 4.39552	0.788228 $\pm$ 0.0239896	0.5364 $\pm$ 0.0100584
Gaussian Ens + Selective Ens Var	2.40691 $\pm$ 0.580524	8.62187 $\pm$ 1.5177	0.567734 $\pm$ 0.0270648	0.74512 $\pm$ 0.0438128

**Table A15:** Miscoverage rate  $\alpha = 0.2$ : Results on the *ChairAngle* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.289127 $\pm$ 0.081643	0.872808 $\pm$ 0.205426	0.80212 $\pm$ 0.00337798	1.0
Gaussian	0.376692 $\pm$ 0.171928	1.14193 $\pm$ 0.40832	0.806646 $\pm$ 0.00565869	1.0
Gaussian Ensemble	0.361834 $\pm$ 0.165348	1.01037 $\pm$ 0.399644	0.813452 $\pm$ 0.00329742	1.0
Quantile Regression	0.648062 $\pm$ 0.181932	1.87322 $\pm$ 0.428951	0.807127 $\pm$ 0.00380433	1.0
Gaussian + Selective GMM	0.376692 $\pm$ 0.171928	1.14193 $\pm$ 0.40832	0.807801 $\pm$ 0.0066405	0.972829 $\pm$ 0.0019256
Gaussian Ens + Selective Ens Var	0.361834 $\pm$ 0.165348	1.01037 $\pm$ 0.399644	0.804719 $\pm$ 0.00318856	0.951359 $\pm$ 0.00453556

**Table A16:** Miscoverage rate  $\alpha = 0.2$ : Results on the *ChairAngle-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.35034 $\pm$ 0.161819	1.09877 $\pm$ 0.497591	0.588472 $\pm$ 0.0119766	1.0
Gaussian	0.454516 $\pm$ 0.280174	1.69802 $\pm$ 0.840083	0.613951 $\pm$ 0.0358797	1.0
Gaussian Ensemble	0.226352 $\pm$ 0.0677413	0.972076 $\pm$ 0.118477	0.6431 $\pm$ 0.0136145	1.0
Quantile Regression	0.692597 $\pm$ 0.269476	2.29958 $\pm$ 0.738934	0.600606 $\pm$ 0.0136385	1.0
Gaussian + Selective GMM	0.454516 $\pm$ 0.280174	1.69802 $\pm$ 0.840083	0.813442 $\pm$ 0.00720691	0.650227 $\pm$ 0.00138861
Gaussian Ens + Selective Ens Var	0.226352 $\pm$ 0.0677413	0.972076 $\pm$ 0.118477	0.701301 $\pm$ 0.0146886	0.7711 $\pm$ 0.0191172

**Table A17:** Miscoverage rate  $\alpha = 0.2$ : Results on the *VentricularVolume* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	11.2471 $\pm$ 0.201399	33.0996 $\pm$ 0.991594	0.457053 $\pm$ 0.00583106	1.0
Gaussian	12.7238 $\pm$ 1.52197	37.3172 $\pm$ 4.91536	0.589342 $\pm$ 0.0793156	1.0
Gaussian Ensemble	10.1141 $\pm$ 0.180661	29.9914 $\pm$ 1.12007	0.678527 $\pm$ 0.0277009	1.0
Quantile Regression	12.5465 $\pm$ 0.941364	35.8073 $\pm$ 1.5139	0.537774 $\pm$ 0.0444887	1.0
Gaussian + Selective GMM	12.7238 $\pm$ 1.52197	37.3172 $\pm$ 4.91536	0.609177 $\pm$ 0.0682058	0.709875 $\pm$ 0.0232873
Gaussian Ens + Selective Ens Var	10.1141 $\pm$ 0.180661	29.9914 $\pm$ 1.12007	0.640078 $\pm$ 0.0277429	0.686207 $\pm$ 0.0431835

**Table A18:** Miscoverage rate  $\alpha = 0.2$ : Results on the *BrainTumourPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	21.3163 $\pm$ 0.45997	66.146 $\pm$ 2.75637	0.824632 $\pm$ 0.0128289	1.0
Gaussian	21.0625 $\pm$ 0.358012	64.2423 $\pm$ 1.9975	0.798113 $\pm$ 0.0133438	1.0
Gaussian Ensemble	20.5336 $\pm$ 0.211421	62.5706 $\pm$ 0.713494	0.796961 $\pm$ 0.00658989	1.0
Quantile Regression	21.9545 $\pm$ 0.149787	66.2654 $\pm$ 1.33091	0.807646 $\pm$ 0.0165337	1.0
Gaussian + Selective GMM	21.0625 $\pm$ 0.358012	64.2423 $\pm$ 1.9975	0.809477 $\pm$ 0.0139729	0.973576 $\pm$ 0.0187252
Gaussian Ens + Selective Ens Var	20.5336 $\pm$ 0.211421	62.5706 $\pm$ 0.713494	0.806998 $\pm$ 0.00641546	0.975368 $\pm$ 0.00323397

**Table A19:** Miscoverage rate  $\alpha = 0.2$ : Results on the *SkinLesionPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	107.514 $\pm$ 1.87464	275.494 $\pm$ 5.60123	0.553254 $\pm$ 0.0122434	1.0
Gaussian	105.417 $\pm$ 1.15178	395.661 $\pm$ 174.897	0.689951 $\pm$ 0.0406148	1.0
Gaussian Ensemble	100.639 $\pm$ 0.464183	351.961 $\pm$ 70.4308	0.715272 $\pm$ 0.0278757	1.0
Quantile Regression	106.382 $\pm$ 2.70593	251.346 $\pm$ 8.67954	0.590792 $\pm$ 0.0115523	1.0
Gaussian + Selective GMM	105.417 $\pm$ 1.15178	395.661 $\pm$ 174.897	0.708767 $\pm$ 0.0274691	0.849668 $\pm$ 0.0189464
Gaussian Ens + Selective Ens Var	100.639 $\pm$ 0.464183	351.961 $\pm$ 70.4308	0.704273 $\pm$ 0.0198608	0.782116 $\pm$ 0.0109776

**Table A20:** Miscoverage rate  $\alpha = 0.2$ : Results on the *HistologyNucleiPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	217.887 $\pm$ 3.71766	688.269 $\pm$ 13.6761	0.744155 $\pm$ 0.0107073	1.0
Gaussian	211.795 $\pm$ 10.0239	902.884 $\pm$ 295.983	0.454345 $\pm$ 0.0986975	1.0
Gaussian Ensemble	196.785 $\pm$ 2.14454	840.287 $\pm$ 113.005	0.412528 $\pm$ 0.0544736	1.0
Quantile Regression	221.136 $\pm$ 9.7918	679.117 $\pm$ 30.6895	0.580503 $\pm$ 0.0463751	1.0
Gaussian + Selective GMM	211.795 $\pm$ 10.0239	902.884 $\pm$ 295.983	0.462248 $\pm$ 0.105935	0.901985 $\pm$ 0.0400294
Gaussian Ens + Selective Ens Var	196.785 $\pm$ 2.14454	840.287 $\pm$ 113.005	0.417977 $\pm$ 0.0572495	0.954477 $\pm$ 0.0256246

**Table A21:** Miscoverage rate  $\alpha = 0.2$ : Results on the *AerialBuildingPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.80$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	235.417 $\pm$ 7.16096	685.829 $\pm$ 31.3669	0.52437 $\pm$ 0.0610316	1.0
Gaussian	217.877 $\pm$ 1.72493	685.899 $\pm$ 43.1749	0.60129 $\pm$ 0.0717427	1.0
Gaussian Ensemble	208.487 $\pm$ 1.03581	658.243 $\pm$ 27.1206	0.683599 $\pm$ 0.0674139	1.0
Quantile Regression	229.243 $\pm$ 6.21038	662.104 $\pm$ 18.5495	0.635733 $\pm$ 0.0350343	1.0
Gaussian + Selective GMM	217.877 $\pm$ 1.72493	685.899 $\pm$ 43.1749	0.679246 $\pm$ 0.0462652	0.65018 $\pm$ 0.0999173
Gaussian Ens + Selective Ens Var	208.487 $\pm$ 1.03581	658.243 $\pm$ 27.1206	0.743714 $\pm$ 0.0354023	0.571928 $\pm$ 0.0402727

**Table A22:** Miscoverage rate  $\alpha = 0.05$ : Results on the *Cells* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	4.03121 $\pm$ 2.78375	22.3973 $\pm$ 12.7194	0.95612 $\pm$ 0.00460235	1.0
Gaussian	3.61704 $\pm$ 1.13624	17.0459 $\pm$ 4.98381	0.95402 $\pm$ 0.00275928	1.0
Gaussian Ensemble	2.78757 $\pm$ 1.16951	20.5445 $\pm$ 5.00176	0.9484 $\pm$ 0.00331843	1.0
Quantile Regression	3.5478 $\pm$ 1.25834	17.8244 $\pm$ 3.57608	0.94858 $\pm$ 0.00565346	1.0
Gaussian + Selective GMM	3.61704 $\pm$ 1.13624	17.0459 $\pm$ 4.98381	0.955558 $\pm$ 0.0032579	0.95212 $\pm$ 0.00276362
Gaussian Ens + Selective Ens Var	2.78757 $\pm$ 1.16951	20.5445 $\pm$ 5.00176	0.946085 $\pm$ 0.00337562	0.95156 $\pm$ 0.00215277

**Table A23:** Miscoverage rate  $\alpha = 0.05$ : Results on the *Cells-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	3.56733 $\pm$ 1.0818	16.3831 $\pm$ 4.80611	0.59048 $\pm$ 0.0307226	1.0
Gaussian	4.05446 $\pm$ 1.33153	17.5443 $\pm$ 5.3978	0.59066 $\pm$ 0.0417946	1.0
Gaussian Ensemble	2.40691 $\pm$ 0.580524	13.1205 $\pm$ 1.81604	0.67218 $\pm$ 0.0308246	1.0
Quantile Regression	3.34375 $\pm$ 0.642763	14.5539 $\pm$ 2.01592	0.5666 $\pm$ 0.0295055	1.0
Gaussian + Selective GMM	4.05446 $\pm$ 1.33153	17.5443 $\pm$ 5.3978	0.941551 $\pm$ 0.0164539	0.53636 $\pm$ 0.0100057
Gaussian Ens + Selective Ens Var	2.40691 $\pm$ 0.580524	13.1205 $\pm$ 1.81604	0.720221 $\pm$ 0.0374625	0.74512 $\pm$ 0.0438128

Paper VII – How Reliable is Your Regression Model’s Uncertainty?

**Table A24:** Miscoverage rate  $\alpha = 0.05$ : Results on the *ChairAngle* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.289127 $\pm$ 0.081643	1.27437 $\pm$ 0.18759	0.953087 $\pm$ 0.000729425	1.0
Gaussian	0.376692 $\pm$ 0.171928	1.59112 $\pm$ 0.367248	0.949381 $\pm$ 0.00143361	1.0
Gaussian Ensemble	0.361834 $\pm$ 0.165348	1.38384 $\pm$ 0.481345	0.95633 $\pm$ 0.0016841	1.0
Quantile Regression	0.888707 $\pm$ 0.472522	4.04185 $\pm$ 1.50356	0.951608 $\pm$ 0.00184683	1.0
Gaussian + Selective GMM	0.376692 $\pm$ 0.171928	1.59112 $\pm$ 0.367248	0.949963 $\pm$ 0.00265095	0.97208 $\pm$ 0.00166628
Gaussian Ens + Selective Ens Var	0.361834 $\pm$ 0.165348	1.38384 $\pm$ 0.481345	0.9541 $\pm$ 0.00167267	0.951359 $\pm$ 0.00453556

**Table A25:** Miscoverage rate  $\alpha = 0.05$ : Results on the *ChairAngle-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	0.35034 $\pm$ 0.161819	1.56264 $\pm$ 0.582954	0.697817 $\pm$ 0.00918537	1.0
Gaussian	0.454516 $\pm$ 0.280174	2.39892 $\pm$ 1.02472	0.733595 $\pm$ 0.0367327	1.0
Gaussian Ensemble	0.226352 $\pm$ 0.0677413	1.59703 $\pm$ 0.159795	0.780615 $\pm$ 0.0141415	1.0
Quantile Regression	1.43807 $\pm$ 0.99796	5.57113 $\pm$ 1.71783	0.759929 $\pm$ 0.034465	1.0
Gaussian + Selective GMM	0.454516 $\pm$ 0.280174	2.39892 $\pm$ 1.02472	0.95774 $\pm$ 0.00281998	0.64882 $\pm$ 0.00276083
Gaussian Ens + Selective Ens Var	0.226352 $\pm$ 0.0677413	1.59703 $\pm$ 0.159795	0.844587 $\pm$ 0.0165508	0.7711 $\pm$ 0.0191172

**Table A26:** Miscoverage rate  $\alpha = 0.05$ : Results on the *VentricularVolume* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	11.2471 $\pm$ 0.201399	70.211 $\pm$ 1.37665	0.776176 $\pm$ 0.00853621	1.0
Gaussian	12.7238 $\pm$ 1.52197	67.1562 $\pm$ 6.31002	0.83793 $\pm$ 0.0396691	1.0
Gaussian Ensemble	10.1141 $\pm$ 0.180661	51.9065 $\pm$ 3.94789	0.876646 $\pm$ 0.0140944	1.0
Quantile Regression	12.0793 $\pm$ 0.130032	61.623 $\pm$ 3.04835	0.843574 $\pm$ 0.0243659	1.0
Gaussian + Selective GMM	12.7238 $\pm$ 1.52197	67.1562 $\pm$ 6.31002	0.854302 $\pm$ 0.0279176	0.710188 $\pm$ 0.0225912
Gaussian Ens + Selective Ens Var	10.1141 $\pm$ 0.180661	51.9065 $\pm$ 3.94789	0.855053 $\pm$ 0.0125036	0.686207 $\pm$ 0.0431835

**Table A27:** Miscoverage rate  $\alpha = 0.05$ : Results on the *BrainTumourPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	21.3163 $\pm$ 0.45997	123.719 $\pm$ 2.07495	0.921113 $\pm$ 0.00123648	1.0
Gaussian	21.0625 $\pm$ 0.358012	123.587 $\pm$ 3.3168	0.910141 $\pm$ 0.00537523	1.0
Gaussian Ensemble	20.5336 $\pm$ 0.211421	113.592 $\pm$ 1.37398	0.905182 $\pm$ 0.00359141	1.0
Quantile Regression	24.6897 $\pm$ 2.49388	126.788 $\pm$ 7.00165	0.915995 $\pm$ 0.00956349	1.0
Gaussian + Selective GMM	21.0625 $\pm$ 0.358012	123.587 $\pm$ 3.3168	0.919455 $\pm$ 0.0105219	0.973672 $\pm$ 0.0186442
Gaussian Ens + Selective Ens Var	20.5336 $\pm$ 0.211421	113.592 $\pm$ 1.37398	0.910995 $\pm$ 0.00267557	0.975368 $\pm$ 0.00323397

**Table A28:** Miscoverage rate  $\alpha = 0.05$ : Results on the *SkinLesionPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	107.514 $\pm$ 1.87464	845.109 $\pm$ 31.2572	0.822753 $\pm$ 0.0104666	1.0
Gaussian	105.417 $\pm$ 1.15178	691.768 $\pm$ 232.071	0.866844 $\pm$ 0.0166304	1.0
Gaussian Ensemble	100.639 $\pm$ 0.464183	582.719 $\pm$ 95.3488	0.881009 $\pm$ 0.00686587	1.0
Quantile Regression	112.07 $\pm$ 5.13986	627.938 $\pm$ 68.3371	0.827977 $\pm$ 0.017595	1.0
Gaussian + Selective GMM	105.417 $\pm$ 1.15178	691.768 $\pm$ 232.071	0.889891 $\pm$ 0.00940058	0.849225 $\pm$ 0.0194809
Gaussian Ens + Selective Ens Var	100.639 $\pm$ 0.464183	582.719 $\pm$ 95.3488	0.881749 $\pm$ 0.00534917	0.782116 $\pm$ 0.0109776

**Table A29:** Miscoverage rate  $\alpha = 0.05$ : Results on the *HistologyNucleiPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	217.887 $\pm$ 3.71766	1282.78 $\pm$ 25.9915	0.896603 $\pm$ 0.0113207	1.0
Gaussian	211.795 $\pm$ 10.0239	1491.62 $\pm$ 472.378	0.680018 $\pm$ 0.0860417	1.0
Gaussian Ensemble	196.785 $\pm$ 2.14454	1358.12 $\pm$ 174.189	0.66749 $\pm$ 0.0522544	1.0
Quantile Regression	251.562 $\pm$ 12.244	1191.96 $\pm$ 65.3122	0.754566 $\pm$ 0.0379023	1.0
Gaussian + Selective GMM	211.795 $\pm$ 10.0239	1491.62 $\pm$ 472.378	0.68897 $\pm$ 0.0895173	0.901191 $\pm$ 0.0405698
Gaussian Ens + Selective Ens Var	196.785 $\pm$ 2.14454	1358.12 $\pm$ 174.189	0.673945 $\pm$ 0.0533738	0.954477 $\pm$ 0.0256246

**Table A30:** Miscoverage rate  $\alpha = 0.05$ : Results on the *AerialBuildingPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.95$ )	test Prediction Rate ( $\uparrow$ )
Conformal Prediction	235.417 $\pm$ 7.16096	1474.93 $\pm$ 53.6301	0.731414 $\pm$ 0.070291	1.0
Gaussian	217.877 $\pm$ 1.72493	1181.55 $\pm$ 41.5859	0.773522 $\pm$ 0.0548329	1.0
Gaussian Ensemble	208.487 $\pm$ 1.03581	1108.87 $\pm$ 21.3551	0.88874 $\pm$ 0.0490213	1.0
Quantile Regression	294.181 $\pm$ 19.9268	1281.37 $\pm$ 51.1001	0.888175 $\pm$ 0.028968	1.0
Gaussian + Selective GMM	217.877 $\pm$ 1.72493	1181.55 $\pm$ 41.5859	0.831501 $\pm$ 0.0374173	0.650797 $\pm$ 0.0987431
Gaussian Ens + Selective Ens Var	208.487 $\pm$ 1.03581	1108.87 $\pm$ 21.3551	0.898938 $\pm$ 0.0263409	0.571928 $\pm$ 0.0402727

**Table A31:** Method variation results on the *Cells* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.905241 $\pm$ 0.00448315	0.95216 $\pm$ 0.00290558
Gaussian + Selective GMM, $k = 2$	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.9044 $\pm$ 0.00481955	0.952 $\pm$ 0.00270259
Gaussian + Selective GMM, $k = 8$	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.904928 $\pm$ 0.00469093	0.9528 $\pm$ 0.00331662
Gaussian + Sel. GMM, Spec. Norm	4.80289 $\pm$ 1.89759	18.5596 $\pm$ 9.13391	0.905 $\pm$ 0.00703097	0.95118 $\pm$ 0.00220672
Gaussian + Selective kNN	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.900069 $\pm$ 0.00677637	0.94656 $\pm$ 0.00470472
Gaussian + Selective kNN, $k = 5$	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.900146 $\pm$ 0.00671325	0.94736 $\pm$ 0.00395049
Gaussian + Selective kNN, $k = 20$	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.900048 $\pm$ 0.00691667	0.94624 $\pm$ 0.00248564
Gaussian + Selective kNN, L2	3.61704 $\pm$ 1.13624	14.5492 $\pm$ 4.44927	0.906295 $\pm$ 0.00579854	0.95166 $\pm$ 0.0042734

**Table A32:** Method variation results on the *Cells-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.889825 $\pm$ 0.0193021	0.53654 $\pm$ 0.0101012
Gaussian + Selective GMM, $k = 2$	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.891933 $\pm$ 0.0175265	0.525 $\pm$ 0.00829409
Gaussian + Selective GMM, $k = 8$	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.893648 $\pm$ 0.0182098	0.53516 $\pm$ 0.00975635
Gaussian + Sel. GMM, Spec. Norm	4.91114 $\pm$ 2.46961	17.3945 $\pm$ 7.3177	0.881351 $\pm$ 0.021587	0.5331 $\pm$ 0.0157885
Gaussian + Selective kNN	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.859179 $\pm$ 0.0255173	0.56692 $\pm$ 0.0127107
Gaussian + Selective kNN, $k = 5$	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.855387 $\pm$ 0.0261636	0.57206 $\pm$ 0.0129226
Gaussian + Selective kNN, $k = 20$	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.862429 $\pm$ 0.0264602	0.56236 $\pm$ 0.0122017
Gaussian + Selective kNN, L2	4.05446 $\pm$ 1.33153	15.4321 $\pm$ 4.98796	0.898932 $\pm$ 0.00921808	0.51486 $\pm$ 0.00474662

**Table A33:** Method variation results on the *Cells-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.890569 $\pm$ 0.00953089	0.49372 $\pm$ 0.0025926
Gaussian + Selective GMM, $k = 2$	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.892265 $\pm$ 0.00867262	0.48902 $\pm$ 0.00420828
Gaussian + Selective GMM, $k = 8$	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.88967 $\pm$ 0.0111239	0.50196 $\pm$ 0.00492041
Gaussian + Sel. GMM, Spec. Norm	2.81679 $\pm$ 0.42988	12.194 $\pm$ 1.10907	0.883644 $\pm$ 0.0167566	0.4991 $\pm$ 0.00761236
Gaussian + Selective kNN	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.874032 $\pm$ 0.0432364	0.53646 $\pm$ 0.0139864
Gaussian + Selective kNN, $k = 5$	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.874822 $\pm$ 0.0424219	0.5368 $\pm$ 0.0137332
Gaussian + Selective kNN, $k = 20$	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.873876 $\pm$ 0.0443446	0.536 $\pm$ 0.014724
Gaussian + Selective kNN, L2	3.53089 $\pm$ 1.0619	15.6396 $\pm$ 6.23458	0.887052 $\pm$ 0.0144573	0.49576 $\pm$ 0.00474578

**Table A34:** Method variation results on the *ChairAngle* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.902482 $\pm$ 0.00436054	0.972739 $\pm$ 0.00191733
Gaussian + Selective GMM, $k = 2$	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.902972 $\pm$ 0.00540708	0.96392 $\pm$ 0.00121759
Gaussian + Selective GMM, $k = 8$	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.901822 $\pm$ 0.00328732	0.981238 $\pm$ 0.00113892
Gaussian + Sel. GMM, Spec. Norm	0.516487 $\pm$ 0.227015	1.82929 $\pm$ 0.809339	0.903828 $\pm$ 0.00534803	0.970993 $\pm$ 0.000466666
Gaussian + Selective kNN	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.90222 $\pm$ 0.00459694	0.975465 $\pm$ 0.00201785
Gaussian + Selective kNN, $k = 5$	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.902561 $\pm$ 0.00342563	0.983804 $\pm$ 0.00152586
Gaussian + Selective kNN, $k = 20$	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.902137 $\pm$ 0.00457917	0.966592 $\pm$ 0.00129222
Gaussian + Selective kNN, L2	0.376692 $\pm$ 0.171928	1.37757 $\pm$ 0.382191	0.903183 $\pm$ 0.00504343	0.977247 $\pm$ 0.00149241

**Table A35:** Method variation results on the *ChairAngle-Tails* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.901946 $\pm$ 0.00382993	0.655448 $\pm$ 0.001058
Gaussian + Selective GMM, $k = 2$	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.903353 $\pm$ 0.00471599	0.645452 $\pm$ 0.00155062
Gaussian + Selective GMM, $k = 8$	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.902103 $\pm$ 0.00382668	0.660472 $\pm$ 0.00115112
Gaussian + Sel. GMM, Spec. Norm	0.375118 $\pm$ 0.24509	1.24343 $\pm$ 0.611561	0.910147 $\pm$ 0.00315697	0.65518 $\pm$ 0.00176257
Gaussian + Selective kNN	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.860311 $\pm$ 0.00617031	0.703038 $\pm$ 0.00691227
Gaussian + Selective kNN, $k = 5$	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.88625 $\pm$ 0.00498864	0.683742 $\pm$ 0.00547083
Gaussian + Selective kNN, $k = 20$	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.809551 $\pm$ 0.00764886	0.743287 $\pm$ 0.0107676
Gaussian + Selective kNN, L2	0.241214 $\pm$ 0.091736	1.09417 $\pm$ 0.382907	0.89682 $\pm$ 0.00698641	0.662076 $\pm$ 0.00289246

**Table A36:** Method variation results on the *ChairAngle-Gap* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.91215 $\pm$ 0.00604745	0.649372 $\pm$ 0.00334919
Gaussian + Selective GMM, $k = 2$	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.912708 $\pm$ 0.00541793	0.642138 $\pm$ 0.00143869
Gaussian + Selective GMM, $k = 8$	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.911374 $\pm$ 0.00475979	0.654824 $\pm$ 0.00368979
Gaussian + Sel. GMM, Spec. Norm	0.294511 $\pm$ 0.0861797	1.19904 $\pm$ 0.193646	0.910836 $\pm$ 0.00442911	0.649996 $\pm$ 0.00233428
Gaussian + Selective kNN	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.911574 $\pm$ 0.00376608	0.673764 $\pm$ 0.0113432
Gaussian + Selective kNN, $k = 5$	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.91224 $\pm$ 0.00426519	0.670646 $\pm$ 0.00691764
Gaussian + Selective kNN, $k = 20$	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.907543 $\pm$ 0.00204724	0.680107 $\pm$ 0.0162927
Gaussian + Selective kNN, L2	0.454516 $\pm$ 0.280174	2.09212 $\pm$ 0.933756	0.913383 $\pm$ 0.00539352	0.658744 $\pm$ 0.00274677

**Table A37:** Method variation results on the *AssetWealth* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.850824 \pm 0.047533$	$0.93838 \pm 0.0170443$
Gaussian + Selective GMM, $k = 2$	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.850232 \pm 0.0469534$	$0.933687 \pm 0.0176348$
Gaussian + Selective GMM, $k = 8$	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.85067 \pm 0.0456954$	$0.932879 \pm 0.0102523$
Gaussian + Sel. GMM, Spec. Norm	$0.354647 \pm 0.00973911$	$1.55594 \pm 0.0421589$	$0.872742 \pm 0.0250168$	$0.941812 \pm 0.0101895$
Gaussian + Selective kNN	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.852107 \pm 0.0474734$	$0.933586 \pm 0.0203541$
Gaussian + Selective kNN, $k = 5$	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.85256 \pm 0.0469571$	$0.933232 \pm 0.02149$
Gaussian + Selective kNN, $k = 20$	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.851914 \pm 0.0460655$	$0.933636 \pm 0.0200087$
Gaussian + Selective kNN, L2	$0.367501 \pm 0.0416437$	$1.597 \pm 0.207599$	$0.853365 \pm 0.0410864$	$0.897149 \pm 0.0162693$

**Table A38:** Method variation results on the *VentricularVolume* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.752046 \pm 0.0529087$	$0.707994 \pm 0.0208741$
Gaussian + Selective GMM, $k = 2$	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.745271 \pm 0.0582068$	$0.790752 \pm 0.0307666$
Gaussian + Selective GMM, $k = 8$	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.747625 \pm 0.0584117$	$0.656583 \pm 0.0258853$
Gaussian + Sel. GMM, Spec. Norm	$11.6311 \pm 0.483357$	$45.5475 \pm 0.747647$	$0.734907 \pm 0.00853342$	$0.719279 \pm 0.0147984$
Gaussian + Selective kNN	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.735105 \pm 0.0612413$	$0.911599 \pm 0.0447475$
Gaussian + Selective kNN, $k = 5$	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.735857 \pm 0.0604531$	$0.916614 \pm 0.0419073$
Gaussian + Selective kNN, $k = 20$	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.734752 \pm 0.0616982$	$0.909404 \pm 0.0469634$
Gaussian + Selective kNN, L2	$12.7238 \pm 1.52197$	$51.566 \pm 3.52739$	$0.740812 \pm 0.049802$	$0.740439 \pm 0.0222073$

**Table A39:** Method variation results on the *BrainTumourPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.883515 \pm 0.0138279$	$0.973576 \pm 0.0187252$
Gaussian + Selective GMM, $k = 2$	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.881874 \pm 0.0118346$	$0.977863 \pm 0.0145103$
Gaussian + Selective GMM, $k = 8$	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.880823 \pm 0.01273$	$0.973417 \pm 0.01856$
Gaussian + Sel. GMM, Spec. Norm	$22.0729 \pm 1.25273$	$95.0605 \pm 3.40124$	$0.890506 \pm 0.0112498$	$0.9738 \pm 0.00857087$
Gaussian + Selective kNN	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.891264 \pm 0.00734602$	$0.947185 \pm 0.0178434$
Gaussian + Selective kNN, $k = 5$	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.891103 \pm 0.00680399$	$0.949392 \pm 0.0173229$
Gaussian + Selective kNN, $k = 20$	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.891759 \pm 0.00804523$	$0.94453 \pm 0.0188456$
Gaussian + Selective kNN, L2	$21.0625 \pm 0.358012$	$93.6284 \pm 2.29916$	$0.879639 \pm 0.0084247$	$0.981862 \pm 0.0068585$

**Table A40:** Method variation results on the *SkinLesionPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.821515 \pm 0.0137705$	$0.849579 \pm 0.0206181$
Gaussian + Selective GMM, $k = 2$	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.825054 \pm 0.0122698$	$0.844799 \pm 0.0191092$
Gaussian + Selective GMM, $k = 8$	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.815696 \pm 0.010454$	$0.862683 \pm 0.0259293$
Gaussian + Sel. GMM, Spec. Norm	$107.531 \pm 2.11824$	$607.435 \pm 407.758$	$0.821797 \pm 0.0350093$	$0.837273 \pm 0.0205022$
Gaussian + Selective kNN	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.813027 \pm 0.0306586$	$0.927047 \pm 0.00830718$
Gaussian + Selective kNN, $k = 5$	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.812763 \pm 0.0309198$	$0.928375 \pm 0.0100103$
Gaussian + Selective kNN, $k = 20$	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.812419 \pm 0.0306294$	$0.929172 \pm 0.0109976$
Gaussian + Selective kNN, L2	$105.417 \pm 1.15178$	$535.139 \pm 215.446$	$0.808667 \pm 0.0112832$	$0.831784 \pm 0.0181658$

**Table A41:** Method variation results on the *HistologyNucleiPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.59554 \pm 0.0956742$	$0.90569 \pm 0.0453555$
Gaussian + Selective GMM, $k = 2$	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.597998 \pm 0.094508$	$0.912836 \pm 0.0323147$
Gaussian + Selective GMM, $k = 8$	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.595718 \pm 0.0973458$	$0.943449 \pm 0.0409511$
Gaussian + Sel. GMM, Spec. Norm	$206.492 \pm 7.82601$	$896.779 \pm 61.4921$	$0.656762 \pm 0.0824111$	$0.855668 \pm 0.0632114$
Gaussian + Selective kNN	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.608929 \pm 0.0805954$	$0.846228 \pm 0.0482832$
Gaussian + Selective kNN, $k = 5$	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.610025 \pm 0.0809455$	$0.842258 \pm 0.046231$
Gaussian + Selective kNN, $k = 20$	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.607876 \pm 0.0817183$	$0.858491 \pm 0.0496696$
Gaussian + Selective kNN, L2	$211.795 \pm 10.0239$	$1211.83 \pm 396.946$	$0.588514 \pm 0.0914286$	$0.975562 \pm 0.0190356$

**Table A42:** Method variation results on the *AerialBuildingPixels* dataset.

Method	val MAE ( $\downarrow$ )	val Interval Length ( $\downarrow$ )	test Coverage ( $\geq 0.90$ )	test Prediction Rate ( $\uparrow$ )
Gaussian + Selective GMM	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.76535 \pm 0.0388677$	$0.652082 \pm 0.0990489$
Gaussian + Selective GMM, $k = 2$	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.751721 \pm 0.0454772$	$0.634602 \pm 0.122618$
Gaussian + Selective GMM, $k = 8$	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.850213 \pm 0.025791$	$0.617738 \pm 0.0879951$
Gaussian + Sel. GMM, Spec. Norm	$220.763 \pm 7.23119$	$1014.77 \pm 164.155$	$0.817517 \pm 0.0505544$	$0.638149 \pm 0.046255$
Gaussian + Selective kNN	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.651867 \pm 0.0771154$	$0.840103 \pm 0.0463989$
Gaussian + Selective kNN, $k = 5$	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.654256 \pm 0.0771566$	$0.846067 \pm 0.0456245$
Gaussian + Selective kNN, $k = 20$	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.647779 \pm 0.0795474$	$0.832596 \pm 0.0542738$
Gaussian + Selective kNN, L2	$217.877 \pm 1.72493$	$929.562 \pm 47.6606$	$0.815075 \pm 0.0278439$	$0.641851 \pm 0.11347$

# Paper VIII

## Title

ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods

## Authors

Philipp Von Bachmann, Daniel Gedon, Fredrik K. Gustafsson, Antônio H. Ribeiro, Erik Lampa, Stefan Gustafsson, Johan Sundström, Thomas B. Schön

## Edited version of

P. Von Bachmann, D. Gedon, F. K. Gustafsson, A. H. Ribeiro, E. Lampa, S. Gustafsson, J. Sundström, and T. B. Schön. “ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods.” In Preparation. 2023



# **ECG-Based Electrolyte Prediction: Evaluating Regression and Probabilistic Methods**

## **Abstract**

Imbalances in electrolyte concentrations can have severe consequences, but accurate and accessible measurements could improve patient outcomes. The current measurement method based on blood tests is accurate but invasive and time-consuming, and is often unavailable for example in remote locations or in an ambulance setting. In this paper, we explore the use of deep neural networks (DNNs) for regression tasks to accurately predict continuous electrolyte concentrations from electrocardiograms (ECGs), a quick and widely adopted tool. We analyze our DNN models on a novel dataset of over 290 000 ECGs across four major electrolytes and compare their performance with traditional machine learning models. For improved understanding, we also study the full spectrum from continuous predictions to a binary classification of extreme concentration levels. Finally, we investigate probabilistic regression approaches and explore uncertainty estimates for enhanced clinical usefulness. Our results show that DNNs outperform traditional models but model performance varies significantly across different electrolytes. While discretization leads to good classification performance, it does not address the original problem of continuous concentration level prediction. While probabilistic regression has practical potential, our uncertainty estimates are not perfectly calibrated. Our study is therefore a first step towards developing an accurate and reliable ECG-based method for electrolyte concentration level prediction—a method with high potential impact within multiple clinical scenarios.

## **1 Introduction**

Electrolytes such as potassium or calcium influence the water and acid-base balance in the human body and ensure the proper functioning of muscles, brain and heart [1, 2]. Electrolyte imbalances are frequently observed in hospitalized

patients, with abnormal potassium levels affecting around 25 % of them [3, 4]. These imbalances can lead to severe heart conditions, such as arrhythmia and cardiac arrest [5].

Electrolyte imbalances are challenging to detect as symptoms often do not appear until the imbalance is severe. Blood tests provide accurate measurements of electrolyte concentrations but are invasive, slow, and inaccessible from remote locations. Electrolytes have known but complex relationships with the electrocardiogram (ECG) since they directly influence heart function [6]. An ECG measures the electrical activity of the heart, it is low-cost and a widely available routine diagnostic tool for heart-related conditions in primary and specialized care. Developing an automated method to extract accurate electrolyte concentration measurements directly from the ECG could provide non-invasive, convenient, and rapid electrolyte monitoring for a large population. Such ECG-based methods would be particularly useful in rural areas relying on telehealth setups, or in an ambulance setting where blood laboratory analysis equipment typically is unavailable.

Computer-based automatic processing of ECGs is an established technology [7]. Recently, deep neural networks (DNNs) have shown promise as an alternative to traditional methods, which use hand-crafted features in combination with simple models, in the classification of cardiac diseases with known ECG patterns [8, 9, 10]. DNNs have also demonstrated success in detecting patterns that are not easily identifiable by traditional electrocardiographic analysis. For instance, models can detect myocardial infarction in ECG exams without ST-elevation [11] and predict the risk of mortality [12, 13], atrial fibrillation [14, 15], and left ventricular dysfunction [16] directly from the ECG. Most of these models are predictive of outcomes even for seemingly normal ECGs.

DNNs have been extensively studied for ECG-based classification problems and have consistently outperformed traditional machine learning models [15, 17], prompting interest in using DNNs for automatic electrolyte prediction. However, since electrolyte concentration levels are *continuous*, the prediction is naturally formulated as a *regression* problem. While there are several regression methods using DNNs in the general literature [18, 19, 20, 21, 22], few have been applied to ECG-based electrolyte prediction. The most common and simple method of DNN-based regression is known as *deep direct regression*, in which a DNN directly predicts continuous values by minimizing the mean-squared error between predicted and observed values [23]. In contrast, earlier studies on electrolyte prediction either used manually engineered ECG features combined with simple models [24, 25], focused on classifying abnormal hypo (low) and hyper (high) concentration levels [26, 17], or discretized concentration levels and applied an approach similar to ordinal regression [27]. Moreover, [27] only studied the prediction of a single electrolyte (potassium).

In this work, we investigate the feasibility of utilizing DNNs to predict the continuous concentration levels of electrolytes directly from ECGs. Initially, we employ the deep direct regression approach and evaluate its regression accuracy for four major electrolytes. Our analysis reveals that the performance of the DNN model varies considerably across different electrolytes. Furthermore, we compare the performance of DNNs with that of traditional models, such as Gradient Boosting and Random Forest, and demonstrate the superior performance of DNNs in ECG-based regression. To perform this analysis, we utilize a novel large-scale dataset comprising over 290 000 ECGs.

In cases when deep direct regression fails to accurately predict the continuous level, we study the prediction problem in more detail. We discretize the electrolyte concentration level and train classification models, with an increasing number of classes, studying the full spectrum: from continuous prediction to a binary classification of extreme concentration levels. This provides insights into the inherent difficulty of the prediction problem and enables us to extract as fine-grained predictions as possible, for different electrolytes. If the direct regression model learns a clear relationship between inputs and targets, we also extend the model to a *probabilistic regression* approach [28], which provides uncertainty estimates for the predictions and enhances the clinical usefulness of the regression model. We evaluate these uncertainty estimates on both in-distribution and out-of-distribution data.

Our main contributions can be summarized as follows:

- We utilize a novel large-scale dataset of more than 290 000 ECGs collected from adult patients who visited emergency departments in Swedish hospitals.
- We train deep direct regression models for ECG-based prediction of continuous electrolyte concentration levels, and demonstrate that DNNs outperform traditional machine learning models on this task.
- We investigate probabilistic regression approaches to provide continuous predictions with uncertainty estimates.
- We evaluate our regression models on four important electrolytes (potassium, calcium, sodium, and creatinine<sup>1</sup>), and find notable performance variations.
- We discretize the electrolyte concentration levels and train classification models, studying the full spectrum from continuous prediction to binary classification.

**Generalizable Insights about Machine Learning for Healthcare** We find that accurate ECG-based prediction of electrolyte concentration levels is in-

---

<sup>1</sup>Potassium, calcium and sodium are electrolytes, while creatinine is a blood biomarker. In this study, however, we refer to creatinine as an electrolyte for simplicity.

herently more difficult for some electrolytes than others, despite not having prior expectations of clear performance differences. Retrospectively, we can justify why in our context potassium levels can be predicted better than e.g. calcium levels—yielding insights into the manifestation of electrolytes on the ECG. Consistent with prior work on ECG-based *classification* tasks, our results demonstrate that DNNs outperform traditional machine learning models, also in our *regression* setting. Therefore, while DNN models generally are more complex and less interpretable, their superior performance is a strong argument for their use also within medical domains. Moreover, by simplifying the original problem via discretization of the electrolyte concentration levels, we find that good classification accuracy can be achieved even in cases when regression models struggle. By progressively increasing the number of discretized classes, we also demonstrate how to extract as fine-grained predictions as possible. This is a general approach that might be suitable for other problems which are naturally formulated as regression tasks, within a wider range of applications. Finally, we extend our direct regression model to the probabilistic regression approach and analyze the resulting uncertainty estimates, finding that they are not particularly well-calibrated. This calls for further investigations into improved uncertainty estimation methods in order to achieve the ultimate goal of accurate, reliable and clinically useful regression models.

## 2 Background

We formulate ECG-based electrolyte prediction as a regression problem and explore different regression approaches. We also study the prediction problem further by discretizing the concentration levels and training models on the simplified classification task.

### 2.1 Regression & Uncertainty Estimation

The goal in a regression problem is to predict a continuous target  $y \in \mathbb{R}$  for an input  $x$ , given a training dataset  $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$  of  $n$  data points. This is achieved by training a model  $m_\theta$  with parameters  $\theta$  such that a loss function is minimized on  $\mathcal{D}$ . In the common deep direct regression approach, the model is a DNN outputting continuous predictions,  $\hat{y} = m_\theta(x)$ , using the mean-squared error (MSE) as loss function. These predictions do not capture any measure of uncertainty. In medical applications, where predictions might impact the treatment of patients, this lack of uncertainty is especially problematic.

Probabilistic regression aims to solve this problem by estimating different types of uncertainties [29, 28, 30]. (1) *Aleatoric uncertainty* captures irreducible ambiguity from the experiment itself. One example is noise from a measurement

device. (2) *Epistemic uncertainty* refers to a lack of knowledge and is therefore reducible. Out-of-distribution (OOD) data is one example where epistemic uncertainty is expected to be high.

Aleatoric uncertainty can be estimated by explicitly modelling the conditional distribution  $p(y|x)$ . Assuming a Gaussian likelihood leads to the parametric model  $p(y|x; \theta) = \mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$ , where the DNN  $m_\theta$  outputs both the mean  $\mu$  and variance  $\sigma^2$ , i.e.  $m_\theta(x) = [\mu_\theta(x) \ \sigma_\theta^2(x)]^\top$ . The mean is used as a target prediction,  $\hat{y} = \mu_\theta(x)$ , whereas the variance  $\sigma_\theta^2(x)$  is interpreted as an estimate of input-dependent aleatoric uncertainty. The DNN is trained by minimizing the corresponding negative log-likelihood  $-\sum_{i=1}^n \log p(y_i|x_i; \theta)$ .

This approach does however not capture epistemic uncertainty. One way to add this uncertainty is by treating the model parameters  $\theta$  according to the Bayesian framework [31] and learning a *posterior* probability distribution over the parameters. Ensemble methods [32, 30] constitute a simple approach to estimating epistemic uncertainty. Multiple models are trained and the uncertainty is estimated by the variance of the prediction over all models. Ensemble methods usually improve the regression accuracy of the model and have been shown to be highly competitive baselines for uncertainty estimation methods [33, 34]. Another common method is the Laplace approximation [35], which approximates the *posterior* distribution  $p(\theta|\mathcal{D})$  with a Gaussian,

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta; \theta_{\text{MAP}}, \Sigma),$$

where the inverse of the covariance matrix  $\Sigma^{-1} = -\nabla_\theta^2 \log p(\mathcal{D}, \theta)|_{\theta_{\text{MAP}}}$  is the negative Hessian matrix, evaluated at the *maximum-a-posteriori* estimate  $\theta_{\text{MAP}}$ . Laplace approximations can be applied post-hoc to a pre-trained model with reduced computational complexity [36, 37].

## 2.2 Simplifying Regression via Discretization

If the accurate prediction of the continuous target  $y \in \mathbb{R}$  is unachievable or not required, a regression problem can be simplified into a standard classification problem by discretizing the targets. Specifically, the target range is divided into  $k$  intervals and each target  $y$  is assigned to the respective interval [38]. The model now outputs a distribution over  $k$  classes and predictions are made by the class with maximum probability.

Usually, the Cross-Entropy (CE) loss is used to train the model, which can however lead to rank inconsistency of the original continuous problem. This implies that for a predicted class  $\tilde{y}$  corresponding to a certain target interval, the probability for the classes corresponding to neighbouring intervals do not necessarily decrease monotonically away from the predicted class. To address this issue, [39] proposed rank-consistency *ordinal regression*. The output of

the model is changed to denote the probability that the target  $y$  is larger than or equal to the lower bound of the corresponding interval of class  $j$ . The model is trained with binary CE loss, and class predictions are computed as  $\tilde{y} = 1 + \sum_{j=1}^k p_\theta(x)_j$ , where  $p_\theta(x)_j = \mathbb{1}_{m_\theta(x)_j > 0.5}$ .

## 2.3 Clinical Relevance

In an in-hospital scenario, blood measurement with laboratory analysis is the gold standard to accurately and reliably determine the electrolyte concentration levels of patients. However, there are multiple scenarios where ECG-based prediction models are desired.

First, in the ambulance setting, there is typically no access to onboard blood laboratory analysis equipment but it is possible to acquire ECGs. Many countries apply a telehealth setup in this scenario and send the ECGs to a coronary care unit for reading and decision-making. If the patient in the ambulance has presented with arrhythmia, which is potentially lethal, it is important to quickly identify its cause. If electrolyte disturbances could be estimated, and identified as the cause of the arrhythmia, then life-saving treatment could be started directly in the ambulance. For example, an insulin-glucose infusion, an intravenous calcium injection, or an inhalation of a beta-2-agonist are all treatments for hyperkalemia (high potassium) that could be administered. Onboard automated ECG analysis would be highly useful in this scenario.

Second, in rural areas without specialists, a remote setting with telehealth care centres is built up. One such example is the Telehealth Network of Minas Gerais, Brazil [40] which receives up to 5 000 ECGs per day. In many locations, obtaining an ECG may be easier than obtaining a blood sample for electrolyte analysis. Our model could provide crucial care for these patients which would otherwise not be possible at all.

Third, in an in-hospital setting, an ECG-based electrolyte prediction could be useful for monitoring the treatment of hyperkalemia or hypernatremia (high sodium). For hyperkalemia, monitoring that potassium is decreased fast enough is an important objective; for hypernatremia, monitoring that sodium is decreased slowly enough (to prevent potentially lethal brain edema due to osmosis) is crucial. A real-time ECG-based prediction could replace frequent blood draws in these scenarios.

Extending the prediction model with uncertainty estimation increases its clinical usefulness. In Section 2.1 we defined two types of uncertainty, each of which plays a crucial role. (1) *Aleatoric uncertainty* captures inherent ambiguity in the data itself. For example, due to measurement noise, it is inherently more difficult to determine the electrolyte levels for some ECGs than for others.

Hence, an accurate prediction of concentration level might not be possible for some ECGs. A model that properly captures aleatoric uncertainty could automatically detect such cases, enabling doctors to take appropriate action such as acquiring a new ECG or asking for blood test analysis instead. If a model captures (2) *epistemic uncertainty*, it could detect cases when the ECG being analysed during clinical deployment is out-of-distribution compared to the training data. Failing to detect such cases could lead to highly incorrect model predictions, with potentially catastrophic consequences, since the accuracy of DNN models can drop significantly on out-of-distribution examples [41, 42].

### 3 Related Work

Most previous work on ECG-based electrolyte prediction relies on hand-crafted ECG features. These are specific characteristics such as the time between two waves, and the amplitude or slope of a wave. [24] were the first to manually develop a relation between such features and electrolyte concentrations. More recently, [43, 44, 25] rely on hand-crafted features but automatically fit the model parameters to data. Their performance is however limited. DNNs offer a different approach by jointly learning features and predictions. Convolutional neural networks (CNNs) have shown promising results for classifying different ECG patterns [8, 9, 10]. For electrolyte prediction, [26] used an 11-layer CNN to classify hyperkalemia. [27] were the first to develop a DNN for regression on potassium, using an approach similar to ordinal regression by discretizing the model outputs. Hence, despite recent work on ECG-based predictions for electrolytes, it remains unclear if the common deep direct regression approach can be applied to accurately predict electrolyte concentration levels from ECGs.

Little work has gone into deep prediction models for electrolytes other than potassium. [17] studied potassium, calcium and sodium, but they only considered the simplified problem of classifying hypo and hyper conditions. We are the first to study these electrolytes in the original problem setting of regressing continuous concentration levels. Moreover, we also consider the prediction of creatinine. We further apply probabilistic regression methods for uncertainty estimation. Closest to this probabilistic setting is [45], which proposed dataset shifts and compared the change of uncertainty for different models but concentrated exclusively on classification problems.

**Table 1:** Characteristics of our datasets.

		Potassium	Calcium	Sodium	Creatinine
Patients		165 508	79 577	163 610	166 908
ECG recordings		290 889	125 970	288 891	295 606
% Male		49.38	48.71	49.07	49.22
Age	mean	61.26	60.47	61.41	61.34
	sd	19.61	20.03	19.69	19.61
Minutes diff (abs)	mean	16.28	12.68	15.92	16.24
	sd	15.04	14.05	14.91	15.01
Concentration	mean	3.99	2.29	138.93	90.55
	sd	0.50	0.13	3.82	71.00

## 4 Dataset

We use data from adult patients attending six emergency departments in the Stockholm area, Sweden, between 2009 and 2017. The ECG recordings are linked through unique patient identifiers to blood measurements of electrolyte concentration levels of potassium, calcium, sodium and creatinine, extracted from electronic health records with laboratory measurements. Inclusion filters are applied to only include data where the ECG and blood measurement are acquired within  $\pm 60$  minutes. Larger time frames would enable more patients in our study, but at the cost of lower label quality.

Standard 10-second 12-lead ECGs are recorded, where we use the 8 independent leads since the remaining ones are mathematically redundant. The data is sampled, producing an ECG trace of size  $leads \times samples$ . We pre-process all ECG recordings to a sampling frequency of 400 Hz and pad with zeros to obtain 4 096 samples. We further apply a high-pass filter to remove biases and low-frequency trends, and finally, remove possible power line noise using a notch filter. The ground truth electrolyte concentration levels are obtained by blood tests. Details on pre-processing are provided in Appendix A.3.

We split our datasets into training, validation and test sets. 70 % of the patients are used for model development including training and validation. The remaining 30 % are split into 20 % for a *random test set*, where the recorded ECGs overlap in time with the development set. The last 10 % are used for a *temporal test set*, where the ECGs do not overlap in time with the development set, which is used to observe changes in recordings over time. We removed patients from the temporal test set who already had recordings in other datasets to avoid data leakage. In the main paper, we present results on the random test set with the exception of Table 3, while complementing results for the temporal test set are in the appendix.

We obtain four datasets—one for each electrolyte. The number of patients ranges between 79 577 and 166 908, with between 126 970 and 295 606 ECGs

in total, see Table 1 for more characteristics. Sometimes multiple blood measurements and multiple ECGs are recorded within the selected  $\pm 60$  minute time frame. We select the median electrolyte value and assign it to all ECGs for training. We consider multiple ECGs during training as a form of data augmentation. In the validation and test sets, we use only the first ECG. Details and comparisons with datasets from literature are in Appendix A.2.

## 5 Methods: Models & Training Procedures

We train regression and classification models for each electrolyte. The study has been approved by relevant ethical review authorities; details will be provided upon acceptance.

**Baseline Comparison** We first conduct a baseline performance comparison of our DNN model with different machine learning models. The raw ECG trace is of size  $leads \times samples$  (i.e.,  $8 \times 4096$ ), yielding 32 768 features. For instance for potassium, we have 290 889 ECG traces in our dataset. In contrast to training deep models, most traditional machine learning models use the full dataset in every iteration of training. However, given our dataset, this would be computationally infeasible. Therefore, we conduct two types of baseline comparisons. First, we *reduce the feature dimension* to ensure computational feasibility. The reduced data are applied to traditional machine learning models, namely linear regression, Gradient Boosting [46], and Random Forest [47]. To reduce the feature dimension, we employ principal component analysis (PCA) as a standard pre-processing step for the raw ECG traces. While PCA-based pre-processing is established for ECGs, it is typically applied to individual ECGs separately [48]. Given our large dataset, this approach would however be very computationally expensive. Thus we instead perform PCA on the entire dataset by vectorizing the  $8 \times 4096$  features and then reducing them. The reduced dimension is set to 256, based on the eigenvalue distribution shown in Figure A3, which explains approximately 60 % of the variance. However, this dimensionality reduction might still remove important information from the raw data. Hence, we perform a second baseline comparison, where we keep all features but ensure computational feasibility by *training in a batch-wise manner*—the usual training strategy for deep networks. In this scenario, we compare with batch-wise linear regression as well as a small 3-layer multi-layer perceptron (MLP) on the raw ECG input. The batch-wise training is done equally to our DNN.

**Model Architecture** For the choice of DNN architecture, reviews note that convolutional models such as ResNets are the dominant deep architecture for ECG-based prediction modelling [49, 50]. The authors in [10] also experimented with vectogram linear transformation for dimensionality reduction,

LSTMs and a VGG convolutional architecture, but used a ResNet. Hence, we use the ResNet backbone network from [10, 13] as a feature extractor in all DNN models. Our methodological approach is however model-agnostic and any architecture with high performance could be utilized instead.

**Model Training** For *deep direct regression*, the DNN  $m_\theta$  consists of the ResNet backbone and a network head that outputs target predictions,  $\hat{y} = m_\theta(x)$ . The DNN is trained from scratch using the MSE loss. During training, we normalize the targets  $y$  with z-transformation to obtain a similar target distribution across all electrolytes. We then discretize the targets into  $k$  intervals, and train both classification and ordinal regression models, as described in Section 2.2. The network head of the direct regression DNN is modified to output  $k$  values instead. For ordinal regression, we train using binary CE loss and for classification using the CE loss. All models are trained for 30 epochs and the final model is selected from the best validation loss. If not specified otherwise, we train each model with 5 different seeds and report the mean and standard deviation (sd) for all results.

For *probabilistic regression*, we create a Gaussian model  $\mathcal{N}(y; \mu_\theta(x), \sigma_\theta^2(x))$  by extending the direct regression DNN with a second network head that outputs the variance  $\sigma_\theta^2(x)$ . The model is trained by minimizing the corresponding negative log-likelihood. We train an ensemble of 5 Gaussian models, and then extract three different uncertainty estimates: (1) *Aleatoric uncertainty* is given by the average predicted variance  $\sigma_\theta^2(x)$  (denoted aleatoric Gaussian). (2) *Epistemic uncertainty* is computed as the variance of the predicted mean  $\mu_\theta(x)$  over the 5 ensemble members (denoted epistemic ensemble). (3) We additionally define an *epistemic uncertainty* by fitting a Laplace approximation after training using the `Laplace` library [37] (denoted epistemic Laplace). The approximation is fit to the last layer of the mean network head by approximating the full Hessian. We report the average epistemic Laplace uncertainty over the 5 ensemble members.

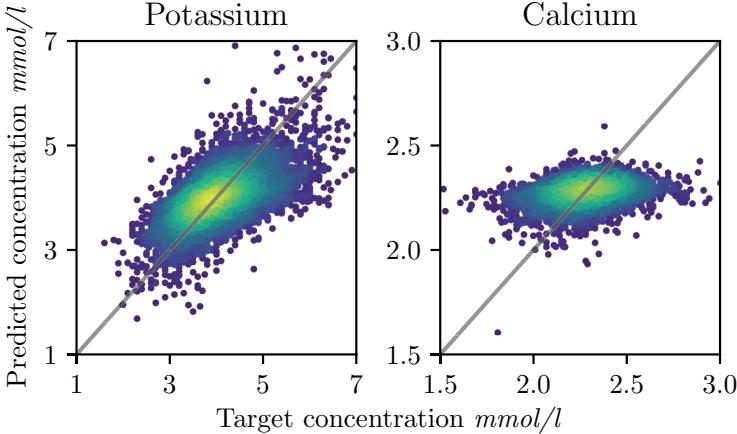
The code is implemented in PyTorch [51] and models are trained on a single Nvidia A100 GPU. Further training details are in Appendix B. Our complete implementation code and the trained models are publicly available at <https://github.com/philippvb/ecg-electrolyte-regression>.

## 6 Results

In this section, we start by discussing the results obtained through deep direct regression. Following that, we compare classification and ordinal regression in the context of discretized regression. Finally, we focus on potassium for probabilistic regression.

**Table 2:** Regression comparison with baseline models on the random test set.

	Model	MSE	MAE
Feature reduction with PCA to 256 dims.	Gradient Boosting	0.215	0.340
	Random Forest	0.220	0.346
	Linear Regression	0.220	0.344
Batch-wise training	Linear Regression	0.215	0.338
	3-layer MLP	0.218	0.346
	ResNet	0.152	0.285



**Figure 1:** *Regression scatter plot:* The diagonal depicts the optimal fit and the point density is indicated by colour (yellow: high, blue: low) from a Gaussian KDE.

## 6.1 Deep Direct Regression

**Baseline Comparison** We validate our ResNet architecture with baseline comparisons, using PCA-based feature reduction or batch-wise training to enable computational feasibility. The results obtained for potassium on the random test set are presented in Table 2. It is worth noting that the ground truth regression targets for potassium have a variance of 0.220. Simply predicting the mean target value for all inputs (resulting in an MSE equal to the variance) would thus achieve similar performance to all baseline models, indicating that the baselines fail to capture a clear relationship between ECG and potassium concentration. This behaviour is independent of the method used to ensure computational feasibility. In contrast, our DNN model achieves a significantly lower MSE and mean absolute error (MAE), implying that it has learned the true underlying relationship effectively. DNNs thus clearly outperform traditional machine learning models on this regression task.

**Table 3:** *Main regression results:* Results of our deep direct regression model on both the random and temporal test set, for all four electrolytes. Targets are not normalized.

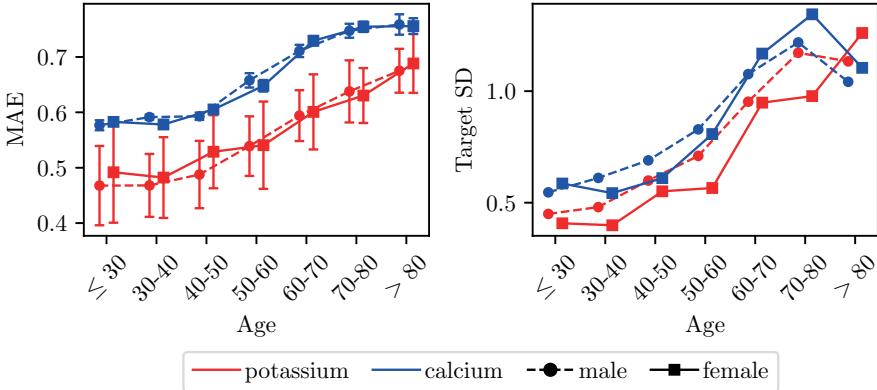
	Random test set MSE (sd)	Random test set MAE (sd)	Temporal test set MAE (sd)
Potassium	0.152 (0.026)	0.285 (0.015)	0.262 (0.013)
Calcium	0.015 (2e–4)	0.088 (5e–4)	0.106 (3e–4)
Sodium	12.59 (0.111)	2.512 (0.016)	2.390 (0.009)
Creatinine	3719 (86.04)	26.69 (1.118)	24.50 (1.298)

**Main Results** Figure 1 depicts the results of our deep direct regression model for potassium and calcium, plotting predictions  $\hat{y}$  against targets  $y$ . For potassium, the data points concentrate along the diagonal, indicating an overall good fit. For calcium, the predictions are horizontally aligned, meaning that the model mainly predicts the mean target value of the train dataset for all inputs  $x$ . Corresponding plots for sodium and creatinine are in Figure A4 in Appendix C. Sodium reflects the undesirable behaviour of calcium. While creatinine shows an overall positive trend, the model seems to suffer from the high variance for higher target values, making predictions for these values noisy. Since the main behaviour is captured by potassium and calcium, we will focus on them in the main text. The complete results for all four electrolytes are in Appendix C.

Quantitative results on both the random and temporal test set are presented in Table 3. There, the MSE and MAE do not directly reflect the performance difference between calcium and potassium observed in Figure 1, as calcium shows significantly lower errors. To understand these opposing results, we investigate the dataset distributions. The variance in the ground truth electrolyte levels is significantly lower for calcium with 0.016 compared to potassium with 0.220. Thus, predicting the mean training target value for calcium will result in a lower MSE without learning the relationship between input and target. Computing errors with normalized targets give MSE values which better reflect the performance difference, as Table A2 shows.

Returning to the results in Table 3, we do not observe a significant drop in performance when evaluating the temporal test set. In fact, the MAE is often slightly lower than for the random test set. This is the first indication that our model is agnostic to real-world data collection changes and distribution shifts. Other works that report regression performance of potassium concentrations obtain MAEs of 0.53 [27] and 0.50 [43], compared to which our model obtains superior performance.

**Stratification** To further analyze our results, we stratify them according to the age and sex of the patients in Figure 2 (left). We observe that our results are independent of sex, but that the MAE has a positive correlation with patient age. Comparing with Figure 2 (right) we see that this correlation is largely expected,



**Figure 2:** *Stratified regression results: Left:* MAE of regression task stratified for different electrolytes by age and sex. **Right:** Corresponding standard deviations of the target values stratified in a similar fashion.

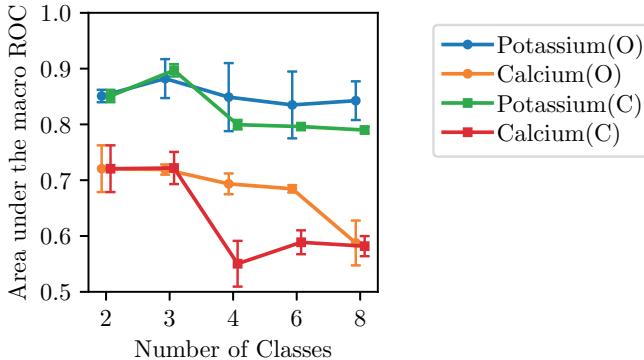
since the variance of the ground truth target values also increases with age. Correctly predicting the electrolyte concentration levels is thus inherently more difficult for older patients.

## 6.2 Classification & Ordinal Regression

We compare classification and ordinal regression in the simplified setting with discretized targets  $y$ . We consider increasingly fine-grained predictions by varying the number of intervals  $k$ . The class intervals are defined for each electrolyte separately: For  $k=3$  classes, we define the lower and upper interval bounds by  $\mu \pm 2\sigma$ . For  $k > 3$  classes we add evenly spaced interval bounds in between the extreme bounds. For binary classification ( $k=2$ ) we consider the hypo/hyper definitions from [17]<sup>2</sup>. For evaluation, we compute Receiver-Operating-Characteristic (ROC) curves for the cumulative classification  $p(\tilde{y} \leq i)$ ,  $i = 1, \dots, k$ , leading to  $k - 1$  individual curves.

Figure 3 shows the area under the macro averaged ROC (AUROC) for a different number of classes  $k$ . The AUROC simply averages the obtained  $k - 1$  AUROC values. For all  $k$ , the prediction performance on calcium is worse than on potassium. When increasing the number of classes, we observe a drop in AUROC which implies that fine-grained predictions are increasingly difficult. This effect is also stronger for calcium. Together with Figure 1, these results clearly suggest that accurate prediction of concentration levels is inher-

<sup>2</sup>Calcium: 2.0/2.75; potassium: 3.5/5.5; creatinine: 3.5/5.3; sodium: 130/150. Values in mmol/l. For creatinine, we default to  $\mu \pm 2\sigma$  as [17] do not consider creatinine.



**Figure 3:** *Macro ROC for varying number of classes:* O: Ordinal regression; C: classification models. For 2 classes we average the hypo and hyper results (see Table 4).

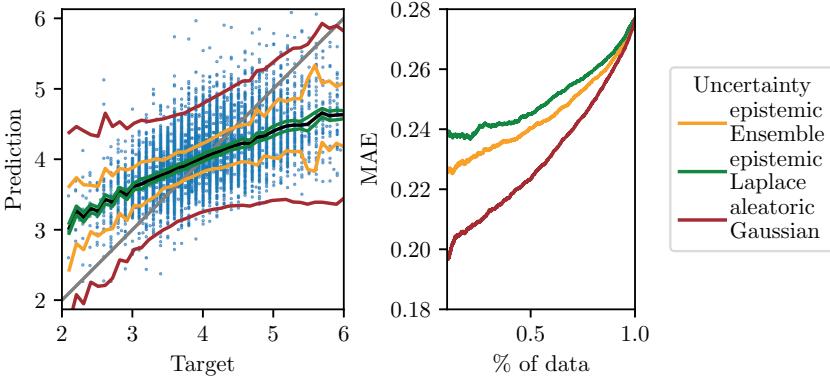
**Table 4:** Binary classification AUROC.

<b>Potassium</b>		(Bounds 3.5 / 5.5 mmol/l)	
	<i>n</i> Data	Hypo	Hyper
Ours	290 889	0.809 (0.003)	0.892 (0.009)
[27]	66 321	0.926	0.958
[26]	2 835 059	N/A	0.865
[17]	83 449	0.866	0.945
<b>Calcium</b>		(Bounds 2.0 / 2.75 mmol/l)	
	<i>n</i> Data	Hypo	Hyper
Ours	125 970	0.779 (0.012)	0.660 (0.036)
[17]	83 449	0.901	0.905

ently more difficult for calcium than for potassium. Comparing classification against ordinal regression in Figure 3, the latter decreases less in AUROC for more classes. In this discretized regression setting, ordinal regression can thus improve performance compared to standard classification models.

We now convert the class predictions into electrolyte concentration levels by mapping to the mean of the predicted class interval and computing the error to the continuous targets. The results in Figure A5 show that the MAE decreases with more classes for potassium but stays mostly constant for calcium. However, the MAE is never lower than the corresponding direct regression MAE from Table 3. While discretization thus can lead to good *classification* performance, it does not help solve the original problem of predicting continuous concentration levels.

For binary classification, we compare with results from literature in Table 4. The comparisons are not entirely fair since data collection and dataset size is different between all works. For potassium, our model reaches a slightly lower AUROC for both imbalances than 2 out of 3 works from the literature. For



**Figure 4:** *Regression uncertainty*: **Left:** Prediction vs target plot as in Figure 1. The black line indicates mean prediction  $\mu$ . The coloured lines show  $\mu \pm 2\sigma$  for different types of uncertainties. **Right:** Sparsification plot.

calcium, [17] reach a significantly higher AUROC, indicating that specialized models might outperform our approach which relies on a standard ResNet.

### 6.3 Probabilistic Regression

Here, we focus on potassium as the only electrolyte for which direct regression learns a clear relationship between inputs and targets. Figure 4 (left) shows the three uncertainty estimates defined in Section 5, for different target levels. Aleatoric Gaussian uncertainty fits the noise in the predictions quite well, since it increases towards the extremes, where predictions become noisy and the error increases. In comparison, both epistemic variances are smaller, which is expected due to the large size of the underlying training dataset. Epistemic Laplace is the smallest and almost constant. Epistemic from the ensemble increases towards the extreme values, similar to the aleatoric uncertainty.

Meaningful uncertainty estimates should correlate with the error – predictions with high error should also have high uncertainty. The sparsification plot in Figure 4 (right) shows that removing the most uncertain points lowers the MAE monotonically, as expected. This effect is strongest for aleatoric Gaussian uncertainty. However, the calibration plot in Figure A7 shows that the uncertainties are not particularly well-calibrated. Aleatoric Gaussian uncertainty has the highest correlation with the MSE, see Table A5. The correlation can be increased by adding epistemic ensemble uncertainty.

To further measure uncertainty, we perform OOD experiments similar to [45]. We first add Gaussian noise to the ECG traces, controlling the strength with the signal-to-noise ratio (SNR). In Table 5 we observe that with increasing

**Table 5:** Results for the OOD experiments.

	Baseline	SNR=10	SNR=1
MAE	0.304 (0.021)	0.330 (0.016)	0.368 (0.026)
Aleatoric Gaussian	0.389 (0.012)	0.399 (0.012)	0.480 (0.078)
Epistemic Ensemble	0.121 (0.048)	0.149 (0.041)	0.184 (0.075)
Epistemic Laplace	0.022 (0.003)	0.028 (0.009)	0.049 (0.031)

noise both the MAE and all uncertainty measures rise. This indicates that each uncertainty by itself is a useful indicator for this kind of OOD data. As a second experiment, we randomly mask a proportion of each ECG trace. The results for this experiment are provided in Table A6 and indicate that this type of OOD data is not detected by our uncertainty quantification.

## 7 Discussion

**Challenges in Predicting Calcium Levels** The varying performance of our models across electrolytes, see for instance Figure 1, requires discussion. The difficulty in predicting calcium levels in contrast to potassium levels can be explained with their manifestation in the ECG. There is a known relationship between both electrolyte levels and a change of the ECG, which for calcium is revealed mainly in a change of the QT interval [52, 53]. The range of values for calcium is however very narrow since extreme values can be lethal and it thus is tightly regulated by many mechanisms in the human body. In our dataset, about 95 % of the values are in the range  $2.29 \pm 0.26$ , see Table 1. The electrophysiological manifestation of this change in concentration level could be negligible, as Figure 2 in [54] indicates. Further, the calcium dataset size is less than half that of potassium, thus the number of patients with extreme calcium values could be insufficient to predict those reliably.

**Interlinked Effects of Electrolytes** The electrophysiological effects of potassium, calcium, sodium and creatinine are closely interlinked. For example, extracellular hypo- and hyperkalemia (potassium) levels promote cardiac arrhythmias, partly because of direct potassium effects, and partly because the intracellular balances of potassium, sodium and calcium are linked. Thus, hypo- and hyperkalemia directly impact sodium and calcium balances. Finally, creatinine levels can signify renal disease that can lead to hyperkalemia. This complex relationship between the studied electrolytes, together with the significantly better regression results achieved for potassium, could indicate that the summed electrophysiological effects are most tightly linked to potassium concentration. However, due to the known connection of calcium to the ECG,

it is unclear if the combined electrophysiological effects could also be linked to calcium if we had a similar dataset size as for potassium.

**Limitations** The primary focus of our work is to develop a methodology for performing regression in a medically realistic setup, rather than improving the model itself. Despite our model outperforming existing approaches, we observed that our model did not perform as well as some reference work for binary extreme value classification. Nevertheless, it is challenging to compare our results with those of other works due to differences in problem definition, such as the time interval between ECG and blood measurement. Therefore, we cannot definitively conclude that our model underperforms in this specific setting. Moreover, we should note that our model is based on certain assumptions, such as assigning ground-truth electrolyte concentration values to ECGs, which may introduce noise into our dataset. Further work still lies in improving the calibration of our uncertainty quantification for which we present a first step in the setting of ECG-based regression.

## 8 Conclusion

We trained deep models for direct regression of continuous electrolyte concentration levels from ECGs. While the model for potassium performed quite well, it struggled with the three other electrolytes. Simplifying the problem to binary classification, of clinically critical low or high levels, indicated that also those electrolytes for which the direct regression model struggled can achieve good classification performance. Defining more classes, for increasingly fine-grained predictions, we observed a sharp performance drop for electrolytes other than potassium. Our results thus strongly suggest that accurate ECG-based prediction of concentration levels is inherently more difficult for some electrolytes than for others. Future work should study this problem for an even larger set of electrolytes, and explore the possibility of combined models for all electrolytes due to their medial interconnection.

We also extended our deep direct regression model to the probabilistic regression approach and carefully analyzed the resulting uncertainty estimates. While especially the aleatoric uncertainty demonstrated potential practical usefulness e.g. via sparsification, the uncertainty estimates are not particularly well-calibrated. To achieve the ultimate goal of accurate, reliable and clinically useful prediction of electrolyte concentration levels, future work investigating possible approaches for improved uncertainty calibration is thus required.

**Acknowledgments** The study was funded by The Kjell and Märta Beijer Foundation; Anders Wiklöf; the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation; Uppsala University; and has received funding from the European Research

Council (ERC) under the European Union’s Horizon Europe research and innovation programme (grant agreement n° 101054643). The computations were enabled by resources in project sens2020005 and sens2020598 provided by the Swedish National Infrastructure for Computing at UPPMAX, partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

## References

- [1] I. Edelman and J. Leibman. “Anatomy of body water and electrolytes.” In: *The American Journal of Medicine* (1959).
- [2] G. P. Carlson and M. Bruss. “Chapter 17 - Fluid, Electrolyte, and Acid-Base Balance.” In: *Clinical Biochemistry of Domestic Animals (Sixth Edition)*. Ed. by J. J. Kaneko, J. W. Harvey, and M. L. Bruss. Sixth Edition. Academic Press, 2008.
- [3] B. Paice, K. Paterson, F. Onyanga-Omara, T. Donnelly, J. Gray, and D. Lawson. “Record linkage study of hypokalaemia in hospitalized patients.” In: *Postgraduate medical journal* (1986).
- [4] N. El-Sherif and G. Turitto. “Electrolyte disorders and arrhythmogenesis.” In: *Cardiology journal* (2011).
- [5] C. Fisch. “Relation of electrolyte disturbances to cardiac arrhythmias.” In: *Circulation* (1973).
- [6] B. Surawicz. “Relationship between electrocardiogram and electrolytes.” In: *American Heart Journal* (1967).
- [7] P. Macfarlane, B. Devine, and E. Clark. “The university of Glasgow (Uni-G) ECG analysis program.” In: *Computers in Cardiology, 2005*. 2005.
- [8] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng. “Cardiologist-level arrhythmia detection with convolutional neural networks.” In: *arXiv preprint arXiv:1707.01836* (2017).
- [9] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng. “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.” In: *Nature medicine* (2019).
- [10] A. H. Ribeiro, M. H. Ribeiro, G. M. Paixão, D. M. Oliveira, P. R. Gomes, J. A. Canazart, M. P. Ferreira, C. R. Andersson, P. W. Macfarlane, W. Meira Jr, et al. “Automatic diagnosis of the 12-lead ECG using a deep neural network.” In: *Nature communications* (2020).
- [11] S. Gustafsson, D. Gedon, E. Lampa, A. H. Ribeiro, M. J. Holzmann, T. B. Schön, and J. Sundström. “Development and validation of deep learning ECG-based prediction of myocardial infarction in emergency department patients.” In: *Scientific Reports* (2022).

- [12] S. Raghunath, A. E. Ulloa Cerna, L. Jing, D. P. VanMaanen, J. Stough, D. N. Hartzel, J. B. Leader, H. L. Kirchner, M. C. Stumpe, A. Hafez, et al. “Prediction of mortality from 12-lead electrocardiogram voltage data using a deep neural network.” In: *Nature medicine* (2020).
- [13] E. M. Lima, A. H. Ribeiro, G. M. Paixão, M. H. Ribeiro, M. M. Pinto-Filho, P. R. Gomes, D. M. Oliveira, E. C. Sabino, B. B. Duncan, L. Giatti, et al. “Deep neural network-estimated electrocardiographic age as a mortality predictor.” In: *Nature communications* (2021).
- [14] Z. I. Attia, P. A. Noseworthy, F. Lopez-Jimenez, S. J. Asirvatham, A. J. Deshmukh, B. J. Gersh, R. E. Carter, X. Yao, A. A. Rabinstein, B. J. Erickson, S. Kapa, and P. A. Friedman. “An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: a retrospective analysis of outcome prediction.” In: *The Lancet* (2019).
- [15] S. Biton, S. Gendelman, A. H. Ribeiro, G. Miana, C. Moreira, A. L. P. Ribeiro, and J. A. Behar. “Atrial fibrillation risk prediction from the 12-lead ECG using digital biomarkers and deep representation learning.” In: *European Heart Journal - Digital Health* (2021).
- [16] Z. I. Attia et al. “Screening for cardiac contractile dysfunction using an artificial intelligence-enabled electrocardiogram.” In: *Nature Medicine* (2019).
- [17] J.-m. Kwon, M.-S. Jung, K.-H. Kim, Y.-Y. Jo, J.-H. Shin, Y.-H. Cho, Y.-J. Lee, J.-H. Ban, K.-H. Jeon, S. Y. Lee, et al. “Artificial intelligence for detecting electrolyte imbalance using electrocardiography.” In: *Annals of Noninvasive Electrocardiology* (2021).
- [18] J. Gast and S. Roth. “Lightweight probabilistic deep networks.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [19] A. Varamesh and T. Tuytelaars. “Mixture Dense Regression for Object Detection and Human Pose Estimation.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
- [20] B. Xiao, H. Wu, and Y. Wei. “Simple baselines for human pose estimation and tracking.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [21] N. Ruiz, E. Chong, and J. M. Rehg. “Fine-grained head pose estimation without keypoints.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018.
- [22] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön. “Energy-Based Models for Deep Probabilistic Regression.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

- [23] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud. “A comprehensive analysis of deep regression.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2019).
- [24] P. P. Frohnert, E. R. Gluliani, M. Friedberg, W. J. Johnson, and W. N. Tauxe. “Statistical investigation of correlations between serum potassium levels and electrocardiographic findings in patients on intermittent hemodialysis therapy.” In: *Circulation* (1970).
- [25] V. Velagapudi, J. C. O’Horo, A. Vellanki, S. P. Baker, R. Pidikiti, J. S. Stoff, and D. A. Tighe. “Computer-assisted image processing 12 lead ECG model to diagnose hyperkalemia.” In: *Journal of electrocardiology* (2017).
- [26] C. D. Galloway, A. V. Valys, J. B. Shreibati, D. L. Treiman, F. L. Peterson, V. P. Gundotra, D. E. Albert, Z. I. Attia, R. E. Carter, S. J. Asirvatham, et al. “Development and validation of a deep-learning model to screen for hyperkalemia from the electrocardiogram.” In: *JAMA cardiology* (2019).
- [27] C.-S. Lin, C. Lin, W.-H. Fang, C.-J. Hsu, S.-J. Chen, K.-H. Huang, W.-S. Lin, C.-S. Tsai, C.-C. Kuo, T. Chau, et al. “A deep-learning algorithm (ECG12Net) for detecting hypokalemia and hyperkalemia by electrocardiography: algorithm development.” In: *JMIR medical informatics* (2020).
- [28] A. Kendall and Y. Gal. “What uncertainties do we need in Bayesian deep learning for computer vision?” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [29] Y. Gal. “Uncertainty in Deep Learning.” PhD thesis. University of Cambridge, 2016.
- [30] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.
- [31] R. M. Neal. “Bayesian learning for neural networks.” PhD thesis. University of Toronto, 1995.
- [32] T. G. Dietterich. “Ensemble methods in machine learning.” In: *International workshop on multiple classifier systems*. Springer. 2000.
- [33] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [34] F. K. Gustafsson, M. Danelljan, and T. B. Schön. “Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. 2020.

- [35] D. J. MacKay. “Bayesian interpolation.” In: *Neural computation* (1992).
- [36] A. Kristiadi, M. Hein, and P. Hennig. “Being Bayesian, even just a bit, fixes overconfidence in ReLU networks.” In: *International conference on machine learning*. PMLR. 2020.
- [37] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. “Laplace Redux–Effortless Bayesian Deep Learning.” In: *arXiv preprint arXiv:2106.14806* (2021).
- [38] L. Torgo and J. Gama. “Regression by classification.” In: *Brazilian symposium on artificial intelligence*. Springer. 1996.
- [39] W. Cao, V. Mirjalili, and S. Raschka. “Rank consistent ordinal regression for neural networks with application to age estimation.” In: *Pattern Recognition Letters* (2020).
- [40] M. B. Alkmim, R. M. Figueira, M. S. Marcolino, C. S. Cardoso, M. P. de Abreu, L. R. Cunha, D. F. da Cunha, A. P. Antunes, A. G. de A Resende, E. S. Resende, and A. L. P. Ribeiro. “Improving patient access to specialized health care: the Telehealth Network of Minas Gerais, Brazil.” In: *Bulletin of the World Health Organization* (2012).
- [41] D. Hendrycks and T. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.” In: *International Conference on Learning Representations (ICLR)*. 2019.
- [42] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, et al. “Wilds: A benchmark of in-the-wild distribution shifts.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2021.
- [43] Z. I. Attia, C. V. DeSimone, J. J. Dillon, Y. Sapir, V. K. Somers, J. L. Dugan, C. J. Bruce, M. J. Ackerman, S. J. Asirvatham, B. L. Striemer, et al. “Novel bloodless potassium determination using a signal-processed single-lead ECG.” In: *Journal of the American heart Association* (2016).
- [44] C. Corsi, M. Cortesi, G. Callisesi, J. De Bie, C. Napolitano, A. Santoro, D. Mortara, and S. Severi. “Noninvasive quantification of blood potassium concentration from ECG in hemodialysis patients.” In: *Scientific Reports* (2017).
- [45] T. Xia, J. Han, and C. Mascolo. “Benchmarking Uncertainty Qualification on Biosignal Classification Tasks under Dataset Shift.” In: *arXiv preprint arXiv:2112.09196* (2021).
- [46] J. H. Friedman. “Greedy function approximation: a gradient boosting machine.” In: *Annals of statistics* (2001).
- [47] L. Breiman. “Random forests.” In: *Machine learning* (2001).
- [48] F. Castells, P. Laguna, L. Sörnmo, A. Bollmann, and J. M. Roig. “Principal component analysis in ECG signal processing.” In: *EURASIP Journal on Advances in Signal Processing* (2007).

- [49] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi. “A review on deep learning methods for ECG arrhythmia classification.” In: *Expert Systems with Applications: X* (2020).
- [50] P. Xiong, S. M.-Y. Lee, and G. Chan. “Deep Learning for Detecting and Locating Myocardial Infarction by Electrocardiogram: A Literature Review.” In: *Frontiers in Cardiovascular Medicine* (2022).
- [51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.
- [52] J. D. Gardner, J. B. Calkins, and G. E. Garrison. “ECG Diagnosis: The Effect of Ionized Serum Calcium Levels on Electrocardiogram.” In: *The Permanente Journal* (2014).
- [53] E. Chorin, R. Rosso, and S. Viskin. “Electrocardiographic manifestations of calcium abnormalities.” In: *Annals of Noninvasive Electrocardiology: The Official Journal of the International Society for Holter and Noninvasive Electrocardiology, Inc* (2016).
- [54] N. Pilia, M. H. Mesa, O. Dössel, and A. Loewe. “ECG-based estimation of potassium and calcium concentrations: Proof of concept with simulated data.” In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2019.
- [55] A. K. Balci, O. Koksal, A. Kose, E. Armagan, F. Ozdemir, T. Inal, and N. Oner. “General characteristics of patients with electrolyte imbalance admitted to emergency department.” In: *World journal of emergency medicine* (2013).

# Appendix

## A Dataset

### A.1 Clarification of Electrolyte Definitions

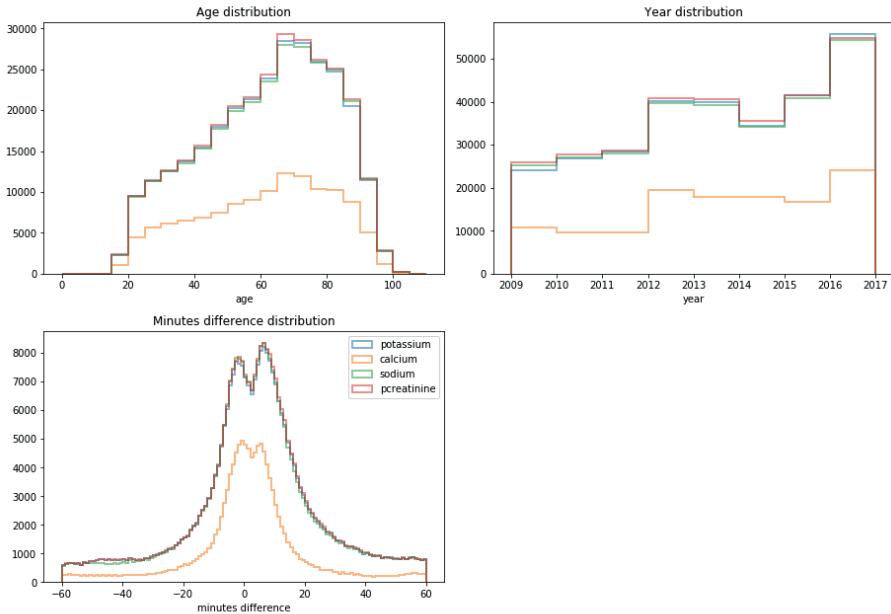
We note that potassium, calcium and sodium are by definition electrolytes but creatinine is an abundant blood biomarker. For ease of reading descriptions, we denote creatinine as an electrolyte in this study. The reason to include creatinine is the availability of large amounts of data and the general medical interest in its predictions. In some figures and tables in this appendix, creatinine is denoted “pcreatinine”.

### A.2 Dataset Characteristics

The characteristics of our four datasets are given in Table 1. More generally a population of emergency room patients with electrolyte imbalances has characteristics as described in [55]. We include data from all-comer patients to the emergency room with  $\geq 18$  years old with the only restriction that there is a blood biomarker test and ECG collected within 60 minutes. We have a varying number of patients in each dataset because not all electrolyte concentration values are available for all patients. Note that we use an inclusion filter of  $\pm 60$  minutes between ECG and blood measurement. We can compare our datasets with related work from the literature:

- [27] use 66 321 ECG recordings from 40 180 patients and related potassium concentration in a time frame of  $\pm 60$  minutes.
- [26] use 2 835 059 ECG recordings from 787 661 patients and related potassium concentrations. The authors develop their model on 60% (= 449 380) of the patients. All ECGs were recorded within 4 hours before potassium measurements.
- [17] have 92 140 patients, whereof 48 356 patients were used for model development with 83 449 ECGs. The study considered potassium, sodium and calcium within  $\pm 30$  minutes of ECG recordings.

We analysed our datasets in more detail to observe possible causes of errors or shortcuts for our model. In Figure A1 we show histograms of age, recording year and the time difference between ECG recording and blood measurement. In Figure A2 we show the distribution of electrolyte concentrations for all four electrolytes, which shows a Normal distribution for all electrolytes except for creatinine which is skewed towards large values. In order to validate our inclusion filter of  $\pm 60$  minutes, we analyze the concentration of electrolytes vs



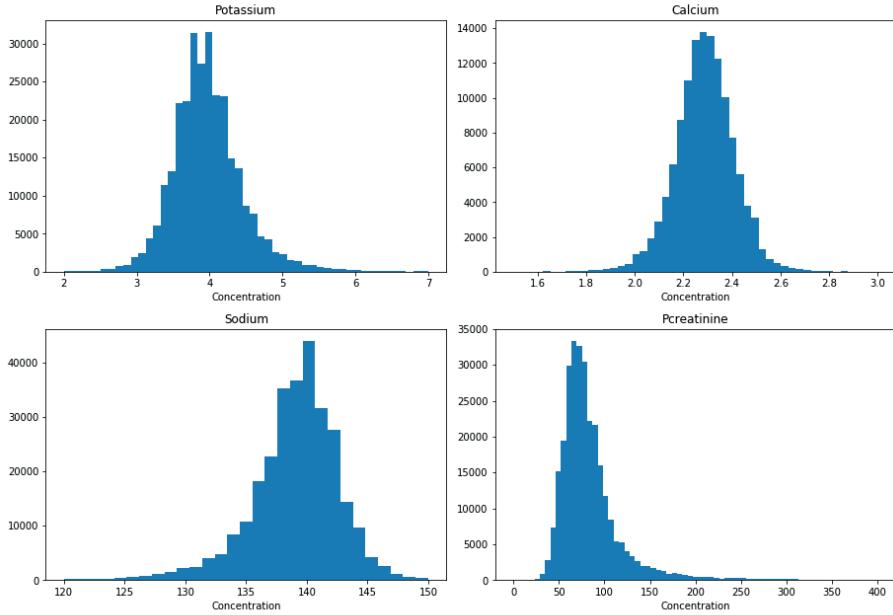
**Figure A1:** Histogram of metadata age (top left), recording year (top right) and minutes difference between ECG recording and blood measurement (bottom) for our four datasets.

the time difference and observe no clear change of concentration value over time. A similar analysis is done for age and sex. Here, we observe that older patients tend to have more extreme electrolyte concentration values for all four electrolytes.

### A.3 Pre-Processing

For the high-pass filter to remove the baseline (trends and low frequencies), we use an elliptic filter with a cut-off frequency of 0.8 Hz and an attenuation of 40 dB which is applied to the forward and reverse direction to avoid phase distortions. We additionally include a notch filter after observing that some ECGs are distorted by power line noise. The notch filter removes the 50 Hz with a quality factor of 30. Also, this filter is applied to the forward and reverse directions for the same reason. We use the pre-processing from the public library [github.com/antonior92/ecg-preprocessing](https://github.com/antonior92/ecg-preprocessing).

For the traditional machine learning methods, which we compare in Section 6.1, we further apply Principal Components Analysis (PCA) to reduce the dimensionality of the data. Here, we first concatenate all leads to get a 1D



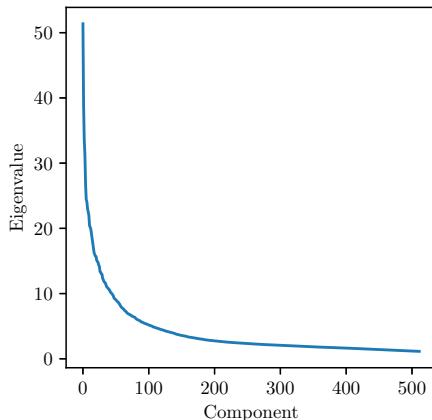
**Figure A2:** Histogram of the electrolyte concentration values for our four datasets.

signal of length  $leads \cdot samples = 8 \cdot 4096 = 32\,768$ . Then we fit PCA on our train dataset. We choose the number of principal components based on the eigenvalues in Figure A3. We see that the eigenvalues decrease fast and start to converge between 200 and 300, which is why we choose to use 256 components.

## B Training Details

### B.1 Network Architecture

We use a modified ResNet which was first developed in [10], and later also in [13], which also provides a public GitHub repository: <https://github.com/antonior92/ecg-age-prediction>. We adjust the last linear layer of the model for the different tasks, for example, different number of outputs for classification.



**Figure A3:** Eigenvalues of PCA components fit on train set. We show the first 512 eigenvalues of possible  $8 \cdot 4096 = 32\,768$  ones. We choose to reduce the dimensionality of our signal to 256 as it covers most information according to this figure.

**Table A1:** Hyperparameters for training the DNNs.

Hyperparameter	Value
optimizer	Adam
maximum epochs	30
batch size	32
initial learning rate	$10^{-3}$
learning rate scheduler	ReduceLROnPlateau
patience	7
min. learning rate	$10^{-7}$
learning rate factor	0.1

## B.2 Hyperparameters

We use the default training hyperparameters from the original network architecture repository. The only deviation is the number of epochs which we reduced from 70 to 30, since this is sufficient for our datasets to converge. The exact hyperparameters are listed in Table A1.

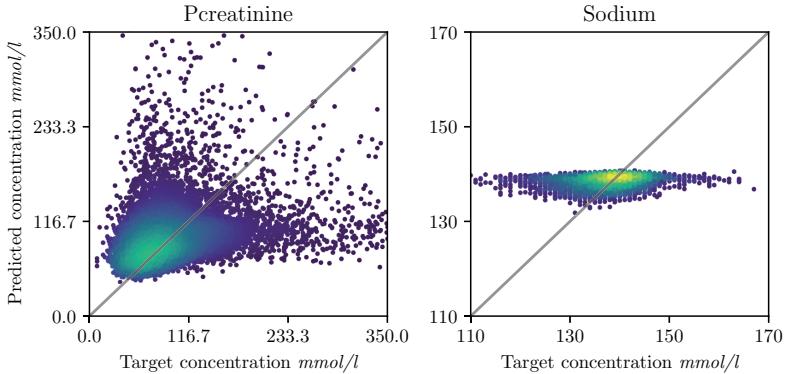
## C Additional Results

Below we present additional results. First, we list more regression results with the complementing scatter plots of Figure 1 (potassium and calcium) in Figure A4 (creatinine and sodium). Further we have a detailed performance table (more detailed than Table 3) for all electrolytes in Table A2 for the random test set and in Table A3 for the temporal test set. No significant difference in

performance between the test sets is observed which shows that our model is robust to shift and trends over time.

Second, we list more results for classification and ordinal regression. In Figure A5 we show the MAE for potassium and calcium which complements Figure 3 that shows the Macro ROC. Figure A6 complements the electrolytes by showing the Macro ROC and MAE for the other electrolytes (creatinine and sodium).

Third, we show additional results for probabilistic regression. Figure A7 gives the calibration plot for potassium. The tables in Table A4 and Table A5 contain numeric details about the sparsification plot for more uncertainties, and the correlation between MSE and the variance to quantify the uncertainty calibration. Table A6 lists the results of the OOD experiments. While the experiments for the SNR are expected (larger MAE and uncertainties for lower SNR), the results for masking are not as clear. While the MAE still increases, notably, especially the epistemic ensemble uncertainty decreases. This means that there is less variance in the mean predictions between the different ensemble members. Finally, Figure A8, Figure A9 and Figure A10 yield the results for the remaining electrolytes that were previously shown for potassium alone.



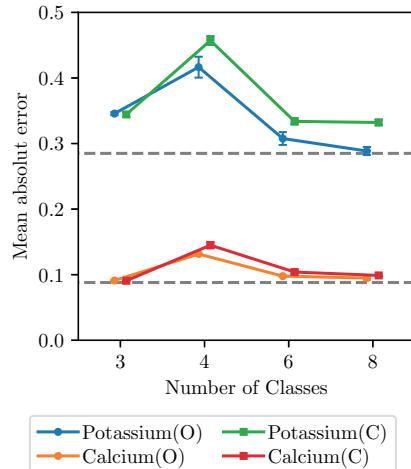
**Figure A4:** Regression scatter plot: Same as Figure 1, but for sodium and creatinine.

**Table A2: Regression performance on the random test dataset:** Table shows metrics for different electrolytes of the regression models from Section 6.1. Target variance refers to the variance of the dataset and therefore yields a worst case MSE performance (since a model with that MSE just predicts the mean of the dataset).

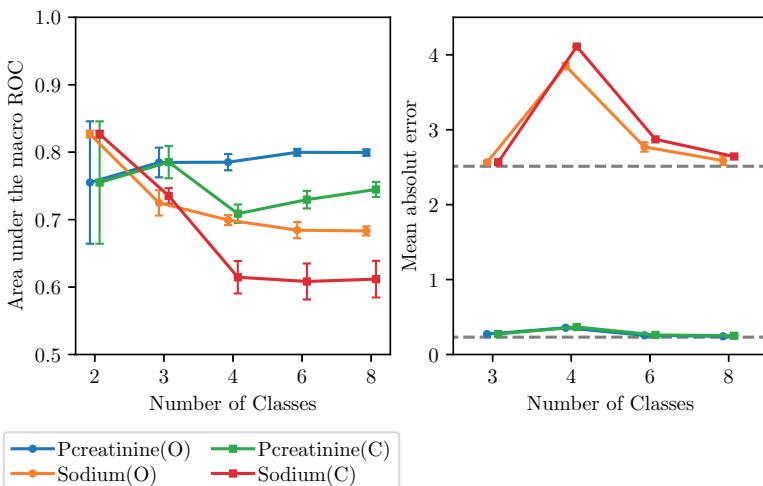
Electrolyte	MSE (sd)	MAE (sd)	Target variance	normalized MSE (sd)
Potassium	0.1524 (0.0259)	0.2846 (0.0152)	0.2158	0.6013 (0.1021)
Calcium	0.0148 (0.0002)	0.0877 (0.0005)	0.0159	0.8625 (0.0088)
Sodium	12.5933 (0.1108)	2.5123 (0.0156)	13.1026	0.8445 (0.0074)
Creatinine	3719.8727 (86.0351)	26.6929 (1.118)	4074.4715	0.7097 (0.0164)

**Table A3:** Regression performance on the temporal test dataset: Table shows metrics for different electrolytes of the regression models from Section 6.1. Target variance has same the meaning as in Table A2.

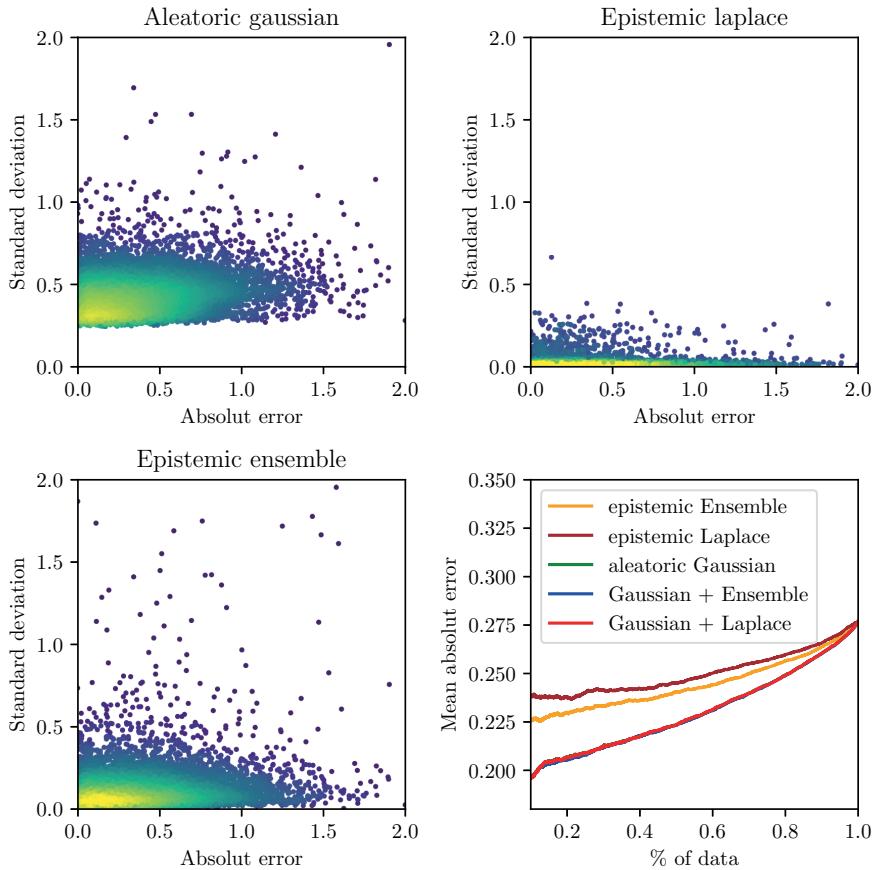
Electrolyte	MSE (sd)	MAE (sd)	Target variance	normalized MSE (sd)
Potassium	0.1319 (0.0171)	0.262 (0.0127)	0.1987	0.5203 (0.0675)
Calcium	0.0201 ( $\leq 0.0001$ )	0.1059 (0.0003)	0.0167	1.1742 (0.0028)
Sodium	12.0118 (0.084)	2.3903 (0.0093)	12.7672	0.8055 (0.0056)
Creatinine	2973.3104 (156.24)	24.5017 (1.2979)	3370.132	0.5673 (0.0298)



**Figure A5:** Classification (C) and Ordinal regression (O) MAE: Similar to Figure 3, we plot the MAE against the number of classes. The dashed line is the MAE of the corresponding deep direct regression model.



**Figure A6:** Classification (C) and Ordinal (O) regression: Same plot as Figure 3 and Figure A5 but for creatinine and sodium (only using 4 seeds for shown mean and sd).



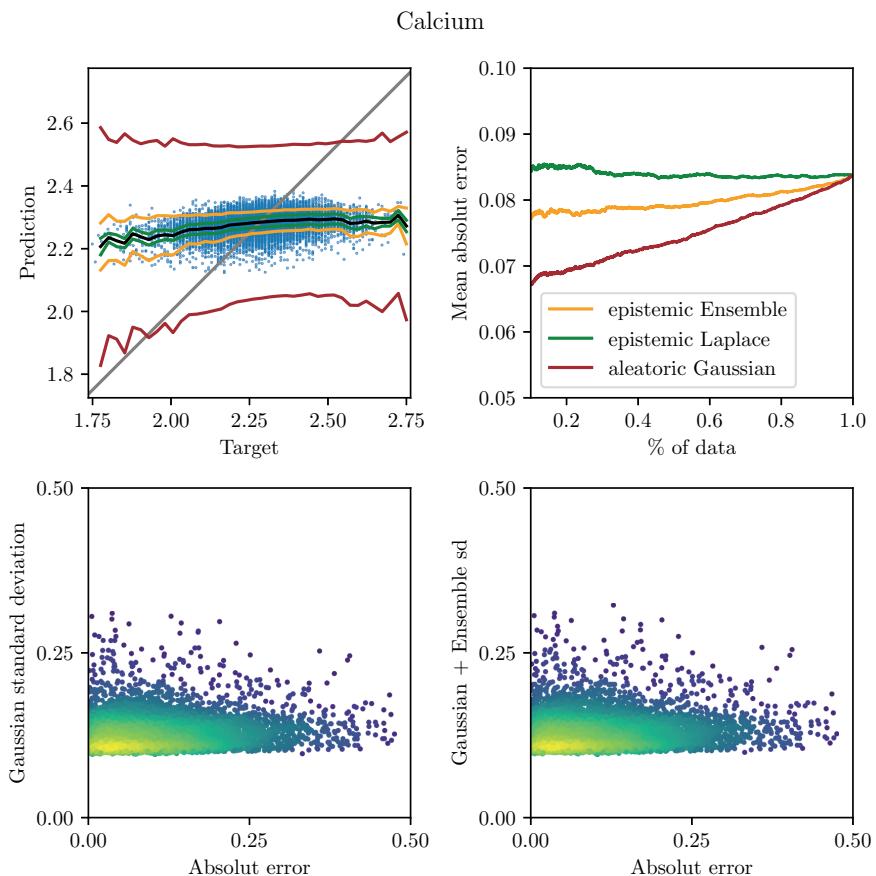
**Figure A7:** Calibration plot, potassium: **Top row and bottom left:** calibration plots as standard deviation vs. absolute error (to have the same units) for different uncertainties. Colours indicate frequency by a fitted Gaussian kernel density estimate. A perfectly calibrated model would follow the diagonal. **Bottom right:** sparsification plot with more results than in the main paper.

**Table A4:** Sparsification against MAE: Numbers in columns show different levels of sparsification (in per cent), and the corresponding row shows MAE values. This table gives the numeric values of the bottom right plot of Figure A7.

	25	50	75	100
Aleatoric Gaussian	0.213 (0.007)	0.228 (0.008)	0.250 (0.009)	0.283 (0.009)
Epistemic ensemble	0.235 (0.006)	0.246 (0.009)	0.259 (0.010)	0.283 (0.009)
Epistemic Laplace	0.249 (0.014)	0.260 (0.021)	0.271 (0.019)	0.283 (0.009)
Aleatoric Gaussian + Epistemic ensemble	0.211 (0.005)	0.227 (0.007)	0.249 (0.008)	0.283 (0.009)
Aleatoric Gaussian + Epistemic Laplace	0.212 (0.007)	0.228 (0.008)	0.250 (0.009)	0.283 (0.009)
Epistemic ensemble of direct reg.	0.227 (NA)	0.238 (NA)	0.249 (NA)	0.274 (NA)

**Table A5:** *Correlation between MSE and Variance:* we correlate the MSE with the variance from different uncertainties. A correlation of 1 would indicate perfect calibration.

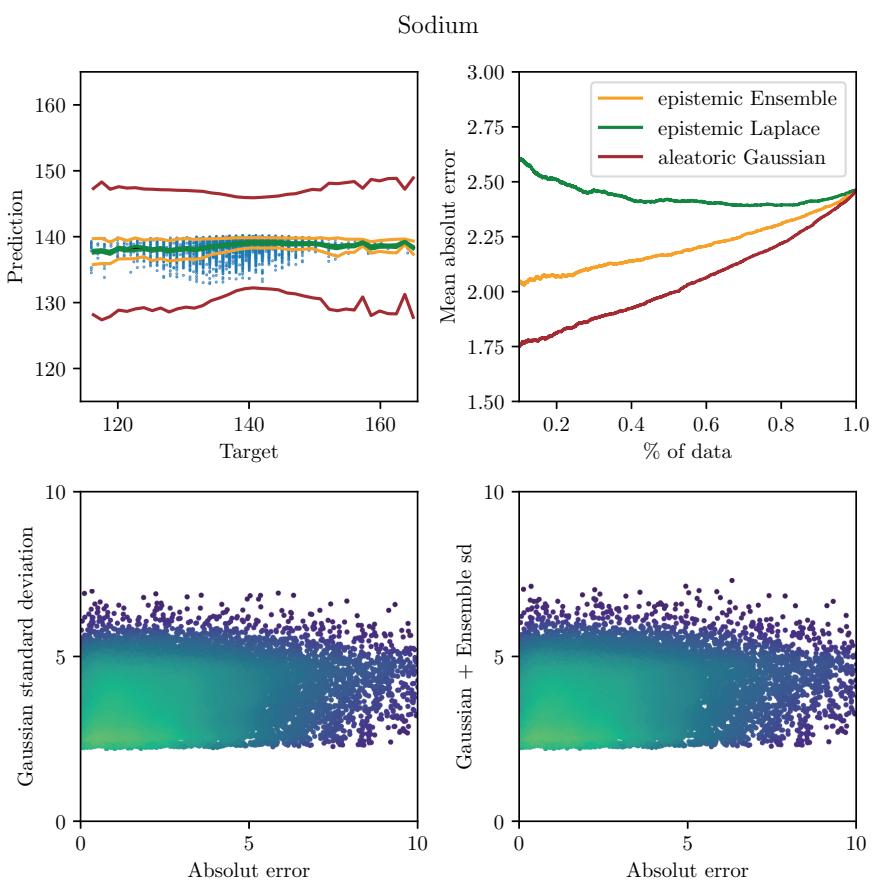
	Correlation
Aleatoric Gaussian	0.225 (0.066)
Epistemic ensemble	0.218 (0.010)
Epistemic Laplace	0.068 (0.034)
Aleatoric Gaussian + Epistemic ensemble	0.255 (0.039)
Aleatoric Gaussian + Epistemic Laplace	0.225 (0.066)
Epistemic ensemble of direct reg.	0.225 (N/A)



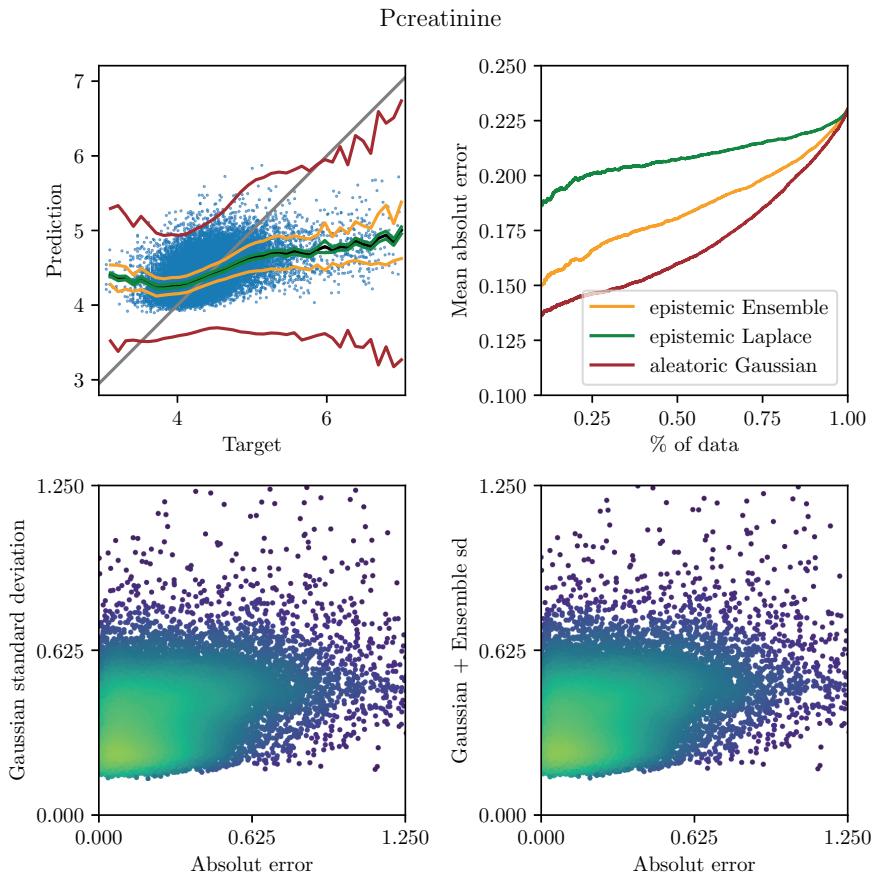
**Figure A8:** *Top left:* prediction vs target plot including various uncertainties. *Top right:* sparsification plot. *Bottom:* Calibration plots between different uncertainties and absolute error. The frequency of samples is highlighted by colour which is fitted with a Gaussian kernel density estimate.

**Table A6:** *OOD experiments*: This is an extended table from Table 5. SNR X refers to OOD experiments with varying SNR; Mask X refers to OOD experiments where X per cent of the data is masked.

	MAE	Aleatoric Gaussian	Epistemic ensemble	Epistemic Laplace	Epistemic direct reg.
Baseline	0.304 (0.021)	0.389 (0.012)	0.121 (0.048)	0.022 (0.003)	0.099
SNR 10	0.330 (0.016)	0.399 (0.012)	0.149 (0.041)	0.028 (0.009)	0.134
SNR 1	0.368 (0.026)	0.480 (0.078)	0.184 (0.075)	0.049 (0.031)	0.154
Mask 25	0.300 (0.008)	0.386 (0.015)	0.091 (0.015)	0.022 (0.001)	0.098
Mask 50	0.311 (0.005)	0.388 (0.008)	0.070 (0.010)	0.020 (0.002)	0.073
Mask 75	0.334 (0.001)	0.385 (0.004)	0.047 (0.005)	0.018 (0.002)	0.054



**Figure A9:** Same results as Figure A8 but for *sodium*.



**Figure A10:** Same results as Figure A8 but for *creatinine*, in the log-transformed space due to the heavily skewed distribution of creatinine.



