



LUND UNIVERSITY

Uncertainty estimation for adaptable and reliable robotic vision

Kristoffersson Lind, Simon

2026

[Link to publication](#)

Citation for published version (APA):

Kristoffersson Lind, S. (2026). *Uncertainty estimation for adaptable and reliable robotic vision*. Computer Science, Lund University.

Total number of authors:

1

General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Uncertainty Estimation for Adaptable and Reliable Robotic Vision

Uncertainty Estimation for Adaptable and Reliable Robotic Vision

by Simon Kristoffersson Lind



LUND
UNIVERSITY

Thesis for the degree of Doctor of Philosophy
Thesis advisor: Prof. Volker Krüger
Faculty opponent: Prof. Fredrik Lindsten

To be presented, with the permission of the Faculty of Engineering of Lund University, for public criticism in the E:1406 lecture hall at the Department of Computer Science on Thursday, the 28th of May 2026 at 9:00.

Organization LUND UNIVERSITY Department of Computer Science Box 118 S-221 00 LUND Sweden		Document name DOCTORAL DISSERTATION	
		Date of disputation 2026-05-28	
Author(s) Simon Kristoffersson Lind		Sponsoring organization Excellence Center at Linköping-Lund in Information Technology (ELLIIT)	
Title and subtitle Uncertainty Estimation for Adaptable and Reliable Robotic Vision			
Abstract Throughout the past decade, neural network-based vision has been increasingly integrated into robots. It is well-known, however, that neural networks can be unreliable, especially when faced with inputs that differ from their training data. With the goal of making neural network-based vision more reliable for robotic applications, this thesis explores uncertainty estimation, and adaptability. Adaptability is explored in terms of the cameras themselves, and it is argued that their built-in parameters form a suitable basis for adapting to varying and difficult visual scenes. An optimization problem is formulated, to adjust camera parameters with the goal of minimizing uncertainty. Depending on the application, it can be beneficial to express uncertainty across local regions in an image. A gradient-based method is proposed based on normalizing flows, which provides uncertainty estimates at the pixel-level. The notion of adaptability based on uncertainty is further applied to visuomotor policy learning. A normalizing flow is used to directly produce fine-grained control sequences for a dual-arm robot, and its uncertainty estimate is used to improve the overall quality of generated sequences. Calibration for regression problems is non-trivial. Several different calibration metrics are used in literature, but they are lacking proper analysis. Such an analysis is provided by the use of toy datasets, to gain insight into what they measure, and whether they are stable estimates. Finally, a combined uncertainty estimate is proposed based on softmax and normalizing flows, that aims to mimic the uncertainty estimate from Gaussian processes. This proposed uncertainty estimate is shown to perform well in the problem of selective classification in the presence of out-of-distribution data.			
Key words Robotics, Vision, Uncertainty			
Classification system and/or index terms (if any)			
Supplementary bibliographical information		Language English	
ISSN and key title 1404-1219		ISBN 978-91-8104-973-2 (print) 978-91-8104-974-9 (pdf)	
Recipient's notes		Number of pages 183	Price
		Security classification	

I, the undersigned, being the copyright owner of the abstract of the above-mentioned dissertation, hereby grant to all reference sources the permission to publish and disseminate the abstract of the above-mentioned dissertation.

Signature _____

Date _____

Uncertainty Estimation for Adaptable and Reliable Robotic Vision

by Simon Kristoffersson Lind



LUND
UNIVERSITY

Cover illustration: My dog, after various visual transformations.

Funding information: Thesis work was funded by ELLIIT Grant C08.

© Simon Kristoffersson Lind 2026

Faculty of Engineering, Department of Computer Science

ISBN: 978-91-8104-973-2 (print)

ISBN: 978-91-8104-974-9 (pdf)

ISSN: 1404-1219

Printed in Sweden by E-Husets Tryckeri, Lund 2026

Dedicated to my family

Contents

Acknowledgements	ii
Popular science summary in English	iii
Populärvetenskaplig sammanfattning på svenska	iv
List of publications	v
Author contributions	vi
Uncertainty Estimation for Adaptable and Reliable Robotic Vision	1
1 Introduction	1
2 Robotic Vision	5
3 Uncertainty	10
4 Normalizing Flows	15
5 Uncertainty as a Driver for Adaptability	18
6 Calibration for Regression	27
7 Combined uncertainty estimates	34
8 Uncertainty-Aware Trajectory Generation	42
9 Conclusions	49
10 Limitations and Future Work	50
11 References	51
Scientific publications	59
Paper I: Out-of-Distribution Detection for Adaptive Computer Vision . . .	61
Paper II: Making the Flow Glow – Robot Perception under Severe Lighting Conditions using Normalizing Flow Gradients	81
Paper III: Uncertainty quantification metrics for deep regression	99
Paper IV: GPify: Leveraging the combined strength of Normalizing Flow and Softmax for an out-of-distribution aware confidence score . . .	121
Paper V: Normalizing Flows are Capable Bi-manual Visuomotor Policy Mod- els	147

Acknowledgements

I will start with the easy part and thank my two dogs, Koda and Grim, for being exceptional models throughout my work. Koda especially has been by my side throughout my entire studies, and he was the first example I ever used to motivate my work.

Now for the more difficult part: people. I hope I do not forget anyone who feels they deserved a mention. If I do, forgive me.

Above all, I would like to extend my deepest gratitude to my loving wife, Tea, who has stood by me for over 10 years now. As a direct result of my studies, she has endured a lot of divided attention from my side, and many late nights of “just one more paragraph”, and I am eternally grateful for her patience and support.

Second, my parents-in-law, Sara and Aje. For a long time, they have been some of my biggest role models, and helped me see the best sides of academia.

On the academic side of things, it is only right that I begin by thanking my supervisor, Volker. I am grateful that he entrusted me as a PhD candidate, and I am grateful for all his guidance along the way. Not once throughout these 5 years were we stuck without any interesting ideas to discuss.

My unofficial supervisor, Rudolph, has been a great help throughout large parts of my thesis. He was especially helpful when I first started out, by offering concrete and directed guidance when I needed it most, and for that, I am incredibly grateful.

I want to extend my gratitude to the RSS group, particularly Elin and Jacek, for creating an open and engaging working environment.

Finally, my gratitude goes to all my fellow PhD Students, especially my closest colleagues, Ayesha, Esra, Jialong, Hashim, Momina, Faseeh. They have all contributed to an inviting and fun workplace, and I can always count on them for an engaging lunch discussion about the most random topics.

Popular science summary in English

Modern robots often rely on cameras to interpret and navigate their surroundings. However, interpreting the actual contents in a digital picture from a camera is a daunting task. In practice, pictures are interpreted using deep neural networks. While these neural networks are incredibly powerful, they cannot always be trusted.

Since robots exist in a physical environment, often in contact with humans, it is imperative to ensure that they do not cause any harm, neither to objects nor humans. An extreme, and tragic, example of what can go wrong, is that of a self-driving Tesla. In 2016, such a Tesla drove straight into the side of a truck, which caused the death of its driver. This accident was a direct result of a neural network that was unable to see the truck against the bright sky.

In this thesis, the overarching goal is to make neural networks more reliable. The main idea is that neural networks must produce a measure of uncertainty. Roughly speaking, instead of saying “The road is clear. It is safe to drive.”, a neural network should say “I believe it is safe to drive. I am $X\%$ sure that the road is clear.”.

Cameras contain parameters that can be adjusted to change the way a picture looks. The first method explored in this thesis, to improve reliability, is to use those parameters. If a neural network reports high uncertainty, camera parameters can be adjusted to get a better picture. As a result, the robot behaves in a way that is not too different from the way we humans use cameras. For example, you might change the exposure time to get a good picture of a sunset. Additionally, a method is created that allows a neural network to express uncertainty over a specific region in a picture. That way, the camera parameters can be adjusted differently for different objects in a picture.

Blindly trusting the uncertainty from a neural network is not a good idea either. Part of this thesis provides an analysis of different methods to verify the uncertainty from a neural network, to ensure that it actually measures uncertainty properly.

Uncertainty can come from different sources, and research surrounding uncertainty commonly discusses two main types called aleatoric and epistemic. Dealing with two different uncertainty measurements is difficult, and therefore a method is created to combine them into one.

Finally, there is a family of methods in robotics, where a neural network controls a robot directly, rather than relying on a separate control module. In this thesis, a model called Normalizing Flows is applied in this context, because it can produce a statistically sound measure of uncertainty. This uncertainty is then used to control the robot with improved reliability.

Populärvetenskaplig sammanfattning på svenska

Moderna robotar använder sig ofta av kameror för att tolka och navigera sin omgivning. Att ta en bild från en kamera och tolka dess innehåll är ingen enkel uppgift, och detta görs i praktiken med hjälp av neurala nätverk. Neurala nätverk är extremt kapabla, men de går inte alltid att lita på.

Eftersom robotar rör sig i en fysisk miljö, inte sällan tillsammans med människor, är det av yttersta vikt att de inte skadar saker eller människor i sin omgivning. Ett extremt och tragiskt exempel på vad som kan gå fel återfinns i en självkörande Tesla, som 2016 körde rakt in i sidan av en lastbil vilket ledde till att föraren avled. Detta på grund av att bilens neurala nätverk inte kunde urskilja lastbilen från en ljus himmel.

Det övergripande målet med denna avhandling är att bidra till göra neurala nätverk mer pålitliga. Denna avhandling grundas i tesen att neurala nätverk måste användas i samband med ett mått av osäkerhet. Kort sagt krävs att neurala nätverk berättar för oss hur säkra de är. I stället för att säga "Det är fritt fram att köra här", så bör ett neuralt nätverk säga "Jag tror att det är fritt fram att köra här, och jag är $X\%$ säker".

En första metod som utforskas för att öka pålitligheten är att optimera kameraparametrar. Ifall ett neuralt nätverk ger en indikation på att osäkerheten är hög, så kan kameraparametrar justeras för att få en tydligare bild. Resultatet är en process som inte är helt olik hur människor använder kameror, där man justerar till exempel exponeringstiden för att få en klar bild av en solnedgång. Detta utökas sedan med en metod för att uttrycka osäkerhet för en specifik region i en bild, så att kamerans parametrar kan justeras för att se olika saker i bilden.

Att förlita sig blint på osäkerhetsmått från neurala nätverk är inte heller en bra idé. Därför undersöks metoder för att verifiera osäkerhetsmått, så att de faktiskt mäter osäkerhet på ett adekvat sätt.

Det finns också olika typer av osäkerhet, och traditionellt sett har det använts två olika mått för att mäta dessa olika typer av osäkerheter. På många sätt är det komplicerat att hantera två olika mått samtidigt, och därför skapas en metod för att kombinera dessa två mått, så att ett och samma mått kan användas för att summera all osäkerhet i ett neuralt nätverk.

Slutligen finns en särskild grupp av metoder inom robotik, där neurala nätverk kontrollerar en robot direkt, i stället för att endast tolka omgivningen. En modell som kallas Normalizing Flows utforskas i denna kontext, eftersom de kan producera ett välgrundat mått av osäkerhet. Experimenten i denna avhandling visar att detta mått av osäkerhet kan användas för att styra roboten på ett pålitligt sätt.

List of publications

This thesis is based on the following publications, referred to by their Roman numerals:

- I **Out-of-Distribution Detection for Adaptive Computer Vision**
S. Kristoffersson Lind, R. Triebel, L. Nardi, V. Krüger
Image Analysis, R. Gade, M. Felsberg, and J.-K. Kämäräinen, Eds., in Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 311–325
- II **Making the Flow Glow – Robot Perception under Severe Lighting Conditions using Normalizing Flow Gradients**
S. Kristoffersson Lind, R. Triebel, V. Krüger
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2024, pp. 11195–11201
- III **Uncertainty quantification metrics for deep regression**
S. Kristoffersson Lind, Z. Xiong, P.-E. Forssén, V. Krüger
Pattern Recognition Letters, vol. 186, pp. 91–97, 2024
- IV **GPify: Leveraging the combined strength of Normalizing Flow and Softmax for an out-of-distribution aware confidence score**
S. Kristoffersson Lind, R. Triebel, V. Krüger
International Journal of Computer Vision, vol. 134, 2026
- V **Normalizing Flows are Capable Bi-manual Visuomotor Policy Models**
J. Li, **S. Kristoffersson Lind**, W. Xie, M. Stenmark, V. Krüger
Submitted to IEEE Robotics and Automation Letters.

All papers are reproduced with permission of their respective publishers.

Author contributions

Table 1: Overview of my contributions for each publication included in this thesis.

	Concept	Implementation	Evaluation	Analysis	Writing
Paper I					
Paper II					
Paper III					
Paper IV					
Paper V					

Symbol table:

-  I did roughly half of the work.
-  I did a clear majority of the work.
-  I did almost all of the work.
-  I did all of the work.

Co-authors are abbreviated as follows:

Volker Krüger:	VK	Luigi Nardi:	LN
Rudolph Triebel:	RT	Ziliang Xiong:	ZX
Per-Erik Forssén:	PEF	Jialong Li:	JL
Wenrui Xie:	WX	Maj Stenmark:	MS

Throughout my doctoral studies, Volker Krüger has served as my main supervisor. Luigi Nardi served as my advisor through the initial phase of my studies. Rudolph Triebel has served as my advisor throughout specific projects: Papers I, II, and IV.

Paper I: Out-of-Distribution Detection for Adaptive Computer Vision

The idea of camera parameter optimization was carried over from my Master’s thesis. I formulated the concept of using out-of-distribution detection as an uncertainty estimate. RT proposed normalizing flows as the main model. I implemented the method, and performed the evaluations. I analyzed evaluation results jointly with VK, LN, and RT. I wrote the initial manuscript. VK, LN, and RT helped review and edit the manuscript.

Paper II: Making the Flow Glow – Robot Perception under Severe Lighting Conditions using Normalizing Flow Gradients

I formulated the idea of a local uncertainty estimate in discussion with VK. I independently discovered the normalizing flow gradient as a method for expressing local uncertainty. I performed the initial evaluations. I analyzed the initial results jointly with RT. I implemented the method and performed the main evaluations. I analyzed the results jointly with VK and RT. I wrote the initial manuscript. VK and RT helped review and edit the manuscript.

Paper III: Uncertainty quantification metrics for deep regression

I formulated the idea jointly with ZX and his advisor PEF. I designed the toy datasets jointly with ZX. I implemented and evaluated the energy-based model, and ZX did the same for the ensemble. I analyzed the results jointly with ZX and PEF. I wrote the initial manuscript jointly with ZX. PEF and VK helped review and edit the manuscript.

Paper IV: GPify: Leveraging the combined strength of Normalizing Flow and Softmax for an out-of-distribution aware confidence score

I formulated the idea of combining uncertainty estimates independently. I constructed the GPify formula independently. I implemented the method and performed the evaluations independently. I analyzed the results mostly independently with help from RT. I wrote the initial manuscript. VK helped review and edit the manuscript.

Paper V: Normalizing Flows are Capable Bi-manual Visuomotor Policy Models

The idea of testing normalizing flows against diffusion policy was formulated by JL, VK, and MS. I formulated and implemented everything relating to the proposed normalizing flow policy. I performed all the simulated evaluations for normalizing flow policy. JL implemented and performed all the real robot evaluations with help from WX. I wrote the initial manuscript jointly with JL. VK and MS helped review and edit the manuscript.

GenAI Statement

The term *GenAI* here refers broadly to large language models and other forms of content generation models, such as, but not limited to, ChatGPT, Google Gemini, Microsoft Copilot, and Anthropic's Claude.

No GenAI has been used throughout my creation of this thesis. Nor has any of my above stated contributions been made by, or with the help of GenAI. All my contributions are entirely my own.

Uncertainty Estimation for Adaptable and Reliable Robotic Vision

1 Introduction

Imbuing robots with the ability to see and interpret their surroundings has been a long standing goal of robotics research. Within the field of computer vision, modern neural network-based methods have achieved impressive performance across many different tasks, even outperforming humans in certain contexts (Szeliski, 2022). As a result, neural network-based vision is being increasingly integrated in robotic systems (Li et al., 2023; Mousavian et al., 2019; Chavez-Garcia et al., 2017). It is well known, however, that neural networks are difficult to interpret and perform unreliably when faced with inputs that differ significantly from their training data. While unreliable performance and lacking interpretability may not pose a problem in many applications, robots tend to require strict safety guarantees since they are physical entities capable of causing harm to their surroundings. An example to illustrate, albeit a tragic one, is that of a Tesla that crashed into the side of a truck due to a failure in its visual systems, which caused the death of its driver (Tesla, 2016).

Any robotic vision pipeline consists of at least two parts: a camera to capture an image, and an algorithm to analyze the image. A failure in either of these parts will result in a failure of the overall vision pipeline. Thus, in order to ensure reliable operation, it is insufficient to solely focus on the neural networks performing the image analysis.

Generally, there is a mindset among roboticists that everything needs to be adaptable. As such, adaptability is treated as a central function in most robot components. For example:

- If an object has moved, dynamic movement primitives (Saveriano et al., 2023) can be used to adapt the robot’s arm to the new location.
- If the robot is in a new location, localization and mapping techniques (Taheri and Xia, 2021) allow the robot to regain its bearings.
- If there is a physical obstacle, impedance controllers (Mayr and Salt-Ducaju, 2022) can adapt the robot’s force to avoid damages.

Adaptability is also one of the central arguments for legged robots (Raibert et al., 2008). Cameras, on the other hand, have not been subject to the same mindset of adaptability. Instead, cameras are assumed to always operate flawlessly, even though it is trivial to find situations, such as sunsets, where they produce images that are unfit for image analysis.

One key insight in this thesis is the fact that cameras are already capable of adapting to different scenarios. In fact, all modern cameras feature built-in parameters that can be adjusted to produce vastly differing image outputs. Some examples include saturation, contrast, and exposure. While the exact parameters differ between camera manufacturers, most cameras include between 5 to 10 adjustable parameters, for example: 6 parameters in an Axis camera (Axis Communication, 2025), 9 parameters in StereoLabs cameras (Stereolabs, 2026), 10 parameters in TierIV cameras (TIER IV, 2026), and 9 in RealSense D400 cameras (Intel RealSense, 2019). In other words, these parameters offer several degrees of freedom that can be used to integrate adaptive behaviors in cameras. Neglecting these degrees of freedom can be compared to having an extra arm on a robot that is never used.

Perhaps the first and most obvious issue with implementing adaptability for cameras is the fact that there is no well-defined goal state. Randomly changing camera parameters is unlikely to lead to any reliability improvements, and therefore a metric is required to indicate which parameter configurations are more or less suitable for a given scenario. There exist metrics that aim to gauge the visual quality of an image, but they tend to be based on human perception (Mittal et al., 2012), and might therefore not be suitable for neural network-based vision. Instead, this thesis presents the argument that uncertainty estimation is a suitable measure for adaptability in vision. By constructing an optimization problem, camera parameters can be adjusted to minimize uncertainty, thus allowing the camera and neural network to work in conjunction. Such an optimization problem forms the first contribution of this thesis.

It is not trivial to estimate uncertainty in neural networks, and many different methods exist (Gawlikowski et al., 2021). Throughout the earliest work in this thesis, out-of-distribution detection (Yang et al., 2024) is used as an uncertainty estimate based on the argument that neural networks perform unreliably when faced with out-of-distribution inputs.

While out-of-distribution detection has merits as an uncertainty estimate, it only addresses part of the problem. Throughout literature, uncertainty is divided into two distinct categories: aleatoric, and epistemic (Kendall and Gal, 2017), also known as data and model uncertainty. Out-of-distribution detection is only capable of representing epistemic uncertainty. Aleatoric uncertainty must be represented using other means, for example using the softmax function for classification problems. However, representing uncertainty using two separate estimates is not ideal, as it increases complexity and makes analysis more difficult. For example, in the aforementioned camera parameter optimization problem, using two separate uncertainty estimates results in a multi-objective optimization problem. Not only is multi-objective optimization more difficult (Deb et al., 2016), it also shifts more responsibility onto the end-user who has to select an optimal solution based on the Pareto front (Deb et al., 2016). Therefore, it is desirable to create a combined metric to represent both aleatoric and epistemic uncertainty, which constitutes another contribution of this thesis.

Relying on uncertainty estimation to increase reliability in vision raises an important follow-up question regarding the reliability of the uncertainty estimate itself. Producing yet another uncertainty estimate is not a viable solution. For classification tasks, there exists the concept of *calibration* (Guo et al., 2017). An uncertainty estimate is said to be calibrated if the estimated uncertainty reflects the true uncertainty, or more formally if $P_{\text{estimate}}(\text{class}) = P_{\text{true}}(\text{class})$. It follows that a natural *calibration error* can be constructed, which conveys how reliable a given uncertainty estimate is: $|P_{\text{estimate}}(\text{class}) - P_{\text{true}}(\text{class})|$.

Regression tasks, however, do not lend themselves to any equally natural concept of calibration. A few calibration metrics exist, that have been used in regression literature, specifically: Area Under Sparsification Error (Ilg et al., 2018), Spearman’s rank correlation (Spearman, 1904), Negative Log-Likelihood (Lakshminarayanan et al., 2017), and a regression version of calibration error (Kuleshov et al., 2018). While all these metrics are commonly used, there is no rigorous analysis to determine their efficacy. Therefore, this thesis contributes to the field by using toy datasets to investigate what these metrics actually measure, whether they are stable and unbiased under varying amounts of data, and whether they are interpretable by the end-user.

A relatively recent trend in robotics is to use neural networks for end-to-end learning of fine-grained robot control. One such task is visuomotor policy learning (Chi et al., 2024), where a neural network is trained to produce a movement trajectory based on camera input. This is in contrast to earlier methods which would only produce a goal location, and then rely on a separate motion planner to generate the actual movement trajectory (Li et al., 2023). Visuomotor policy learning, and similar tasks, present an interesting case in the context of uncertainty estimation, as they are commonly implemented using generative models, such as diffusion models (Chi et al., 2024). In generative modelling, the goal is to model the distribution of movement trajectories, and then subsequently sample from it. Since samples are drawn from a learned distribution, there is no notion of aleatoric uncertainty, and therefore sample trajectories are usually executed without a corresponding uncertainty estimate. However, even if the model has learned a perfect representation of the distribution of movement trajectories, it can still produce poor samples from low-density regions. Therefore, this thesis argues that epistemic uncertainty is a valuable tool to produce high-quality trajectories. Specifically, normalizing flows (Papamakarios et al., 2021) are explored as a compelling candidate for visuomotor policy learning, as they are capable of both sampling and evaluating their learned distribution.

To summarize, this thesis addresses the following research objectives:

- **Objective 1:** Introduce adaptability in robotic vision pipelines, using uncertainty estimates as a guide.
- **Objective 2:** Construct a combined metric encompassing both aleatoric and epistemic uncertainty.
- **Objective 3:** Analyze calibration metrics for regression problems.
- **Objective 4:** Apply epistemic uncertainty for end-to-end robot policy learning.

2 Robotic Vision

From a historical point of view, robots have been used primarily in manufacturing (Chen et al., 2025a). Manufacturing robots are typically deployed in a closed working area, where they can execute repetitive tasks in a pre-programmed manner without any outside influence. As a result, manufacturing robots have little need for vision.

More recent technological advancements, such as force-feedback sensing, which paved the way for safer robots, have led to an increased focus on collaborative robotics (Vicentini, 2020; Hentout et al., 2019). Collaborative robots, as the name suggests, are designed with the goal of operating in a shared environment with humans. Such a shared environment is vastly more complex in terms of robot deployment. First and foremost, a shared environment demands strong safety guarantees to ensure the robot does not harm itself, its surrounding environment, and most importantly nearby humans. Second, since the environment is assumed to be dynamic, a collaborative robot requires various sensors, like camera-based vision, in order to react to changes in the environment.

For the purposes of this thesis, it is important to distinguish between robotic perception, and vision. Perception is commonly used to refer to the use of any sensor with the goal of interpreting a robot’s surrounding environment, such as the LiDAR-based human tracking system presented by Topp and Christensen (2005). Vision, on the other hand, is used throughout this thesis to refer specifically to camera-based visual perception.

Camera-based visual perception on its own is no different than computer vision (Szeliski, 2022), and indeed many computer vision methods have been applied in robots to, for example, detect terrain traversability (Sevastopoulos and Konstantopoulos, 2022), detect objects (Song et al., 2021), and to detect human actions and intentions (Hentout et al., 2019).

As of this writing there exist no computer vision methods that are 100% reliable, and therefore camera-based vision often needs to be supplemented to safeguard the system when visual perception fails. Autonomous vehicles, for example, tend to supplement their camera-based vision with LiDAR sensors to ensure reliable collision avoidance (Li and Ibanez-Guzman, 2020).

2.1 Neural Network-Based Vision

Starting with AlexNet in 2012 (Krizhevsky et al.), computer vision research has been increasingly focused on neural network-based methods, and today the state of the art for most, if not all, vision tasks is a neural network (Szeliski, 2022). Consequently, this thesis revolves around neural network-based methods and the challenges they present. Therefore, a brief introduction is provided here.

A neural network is a parameterized function approximator (Goodfellow et al., 2016). Arguably the simplest form, and the origin of the name *neural network*, is the *multilayer perceptron*, or *feed-forward network*. A feed-forward network is a composition of functions:

$$\begin{aligned} \text{NN} : \mathbb{R}^n &\mapsto \mathbb{R}^m \\ \text{NN}(\mathbf{x}) &= f_L \circ a_{L-1} \circ f_{L-1} \circ \dots \circ a_1 \circ f_1(\mathbf{x}) . \end{aligned} \quad (1)$$

Here, each *layer* f_l is an affine function $f_l(\mathbf{x}) = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l$. $\mathbf{W}_l$ and $\mathbf{b}_l$ are parameters commonly called *weights* and *biases*. Each a_l is an element-wise non-linear *activation function*, most commonly the *rectified linear unit* (ReLU):

$$\text{ReLU}(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} .$$

The primary purpose of an activation function is to prevent the neural network from reducing to another affine function. Given

$$g_1(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \text{ and } g_2(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} ,$$

the composition $g_1 \circ g_2(\mathbf{x})$ reduces to another affine function:

$$g_1 \circ g_2(\mathbf{x}) = g_1(g_2(\mathbf{x})) = \mathbf{A}(\mathbf{C}\mathbf{x} + \mathbf{d}) + \mathbf{b} = (\mathbf{AC})\mathbf{x} + (\mathbf{Ad} + \mathbf{b}) .$$

Thus, the activation functions ensure that each consecutive layer f_l increases the representational power of the neural network. Provided an arbitrary number of layers f_l , it has been shown that feed-forward networks with ReLU activations are universal function approximators (Lu et al., 2017).

AlexNet (Krizhevsky et al., 2012) and its many successors are not plain feed-forward networks, but rather *convolutional* neural networks. As the name suggests, convolutional neural networks are based on the convolution operation, rather than affine functions. For images, and other similar types of data, a 2-dimensional convolution operation can be used for each f_l :

$$f_l(\mathbf{x}) = \mathbf{x} * \mathbf{w}_l + \mathbf{b}_l , \quad (2)$$

where $\mathbf{w}_l$ and $\mathbf{b}_l$ are once again parameters. This construction allows for the weight $\mathbf{w}_l$ to be significantly smaller, while also allowing the same weight to be applied across the spatial dimension of the input image $\mathbf{x}$. It is worth noting that each layer in a convolutional network is effectively the same type of convolutional filter used in many classic image processing methods (Szeliski, 2022), with the primary difference being that the convolutional weights are learned rather than hand-crafted.

Naively implementing a neural network based on convolutions, as in equation (2), will produce another image, which is not suitable for many tasks. In order to accommodate different tasks, the resulting image is typically flattened into a vector, which is then transformed by one or more affine layers to produce the desired output size (Goodfellow et al., 2016).

A more recent development within computer vision is that of transformer models (Vaswani et al., 2017). Originally developed for natural language processing, transformers have been adapted and applied to various computer vision tasks (Dosovitskiy et al., 2021). As transformers are significantly more complex compared to multilayer perceptrons and convolutional networks, implementation details are deemed out of scope for the purposes of this thesis. Importantly, transformers follow the same type of layer-wise composition of linear functions with interleaved non-linearities. As such, there is no reason to analyze transformers in a separate context from other neural network types.

Classification

Equations (1) and (2), while relatively simplistic, work without modifications for regression problems. Classification tasks, however, require extra processing. The most widely adopted method for classification is to construct an output vector $\mathbf{y} \in \mathbb{R}^C$ where C is the number of classes (Bishop, 2006). An output classification is then chosen according to the maximum element of $\mathbf{y}$:

$$\text{class}(\mathbf{y}) = \underset{c}{\operatorname{argmax}} y_c ,$$

where y_c denote the elements of $\mathbf{y}$.

While it is not necessary, it is common to apply the *softmax* function to the output $\mathbf{y}$, which effectively constructs a probability distribution over the classes (Bishop, 2006):

$$\text{softmax}(\mathbf{y}) = \frac{e^{y_c}}{\sum_k e^{y_k}} \quad \forall \text{ classes } c .$$

Viewing Neural Networks as Feature Extractors

Among many older machine learning methods, especially in computer vision, it is common to use a hand-crafted feature extractor such as SIFT (Lowe, 2004). Extracted features would then be used as input to a simple model, such as linear models or decision trees, for subsequent regression or classification (Szeliski, 2022). Neural networks can be viewed as learned feature extractors, where features $\mathbf{z}_{L-1}$ are computed as a result of the first $L - 1$ layers, and then used as input to an affine model f_L (Goodfellow et al., 2016).

Viewing neural networks as feature extractors is useful for many applications, particularly because it allows for *pre-training* and *fine-tuning* (Goodfellow et al., 2016). In other words, a large, highly expressive neural network can be trained to extract features, which can then easily be adapted to different tasks by changing only the final layer f_L . Additionally, because of the layered structure of neural networks, features can be extracted after any layer as illustrated in figure 1. There is some evidence that layers further into the neural network produce more semantically meaningful features (Krizhevsky et al., 2012). As such, features from different layers may be useful for different applications.

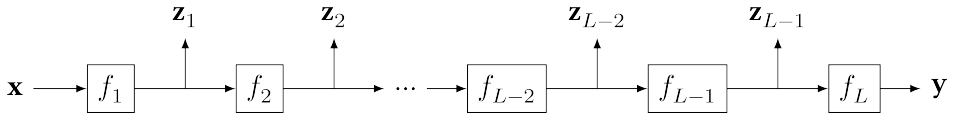


Figure 1: Illustration of neural networks as feature extractors. Here $\mathbf{z}_l$ denotes the features at each layer f_l . Activation functions are omitted for brevity.

2.2 Challenges with Neural Networks

Most real-world problems are infeasible, perhaps even impossible, to solve with 100% accuracy. For example, the problem of pedestrian trajectory prediction (Golchoubian et al., 2023), which is of interest for autonomous vehicles, requires predicting future events, which is arguably impossible to do with 100% accuracy given the stochastic nature of human behavior. As such, it is unreasonable to expect neural networks to

ever achieve 100% accuracy, and thus it is necessary to reason about when and how they may fail. However, modern neural networks commonly consist of hundreds of layers (He et al., 2016), and millions of parameters (Liu et al., 2021), which makes analysis non-trivial.

Some early attempts at analyzing neural networks, specifically for vision problems, are based on the observation that convolutional layers are effectively equivalent to classical convolutional filter methods (Krizhevsky et al., 2012). By visualizing the learned filters in the layers of a convolutional network, some insight can be gained into what each layer is computing. For example it has been observed that the first layer tends to consist of simple edge and corner filters. However, as the number of layers in a network increases, it becomes increasingly difficult to reason about what each filter means.

Other attempts exploit the fact that neural networks have a tractable gradient (Simonyan et al., 2014). One method uses the gradients in each layer to produce an image that maximizes the activation of a single filter, which gives some insight into what that specific filter represents. As networks grow larger, the number of filters increases as well, which diminishes the information gained from a single filter given that the final computation is a product of all filters. Another, similar method, is that of *saliency maps* (Simonyan et al., 2014), which computes the gradient of the input image with respect to the final output of the network, which can provide some insight into which pixels in an image contributed to a certain output. Saliency maps are limited mainly by their lacking generality, as a saliency map is only valid for a single configuration of pixels, and any change in the input image has the potential to greatly change the saliency map.

Arguably the most effective way to gain insight into when and how a neural network may fail is by uncertainty estimation (Grimmett et al., 2016). Many neural network constructions, especially for classification tasks, produce an uncertainty estimate by design. Other types of networks can be augmented to include post-hoc uncertainty estimation (Abdar et al., 2021). In an ideal scenario, an uncertainty estimate should reflect both the likelihood and severity of an error, which makes uncertainty estimation a desirable tool for real-world deployment of neural network-based methods. Naturally, uncertainty estimation is not without issues. It has been show, for example, that the uncertainty estimates produced by neural networks tend to poorly reflect the actual likelihood of errors, especially as the networks increase in size and complexity (Guo et al., 2017). Additionally, neural networks perform unreliably when an input is different compared to their training data, and this is not reflected in the uncertainty estimates produced by the neural networks themselves (Grimmett et al., 2016).

3 Uncertainty

For the purposes of this thesis, confidence and uncertainty are equivalent, as one is effectively the inverse of the other, and therefore the terms are occasionally used interchangeably depending on the context.

As outlined in the previous section, uncertainty estimation has the potential to be a valuable tool for improving and reasoning about the reliability of neural network-based robotic vision. In order to understand how to utilize uncertainty estimates, it is necessary to first understand what is meant by uncertainty, and how it can be estimated.

Neural networks are statistical models (Goodfellow et al., 2016). Earlier sections introduced neural networks with a primary focus on their formulation as function approximators. Nonetheless, any function approximator that is learned from data is a statistical model. For example, it is trivial to show that least squares is equivalent to maximum likelihood estimation (Bishop, 2006). As such, probability theory is a suitable lens to view and discuss the concept of uncertainty.

Any statistical model approximates a probability distribution over the data it is trained with. In the example of a least squares regressor, it forms a maximum likelihood estimate of $P(y|x)$. On the other end, the training process itself can be viewed as a maximum likelihood estimate of $P(\theta|\mathcal{D})$, where θ and $\mathcal{D}$ denote the model parameters and the training data respectively (Bishop, 2006). These two probabilities form the basis of the modern discourse around uncertainty (Abdar et al., 2021).

$P(y|x)$ is otherwise referred to as *aleatoric uncertainty*, or *data uncertainty*, and encompasses uncertainty that is inherent in the data itself (Abdar et al., 2021). In regression tasks, a common source of aleatoric uncertainty is measurement noise. For classification tasks, on the other hand, aleatoric uncertainty is inherent similarities between two or more classes. Importantly, aleatoric uncertainty is inherent in the data, and thus cannot be reduced or removed.

$P(\theta|\mathcal{D})$, otherwise known as *epistemic uncertainty* or *model uncertainty*, refers to uncertainty about the model parameters, or equivalently, uncertainty stemming from lacking training data (Abdar et al., 2021). Since statistical models are learned from data, it is impossible to know whether or not a model is correct over regions in the input space where there is no training data. In contrast to aleatoric uncertainty, epistemic uncertainty can be reduced by acquiring more training data. Figure 2 illustrates both types of uncertainty for a toy regression problem.

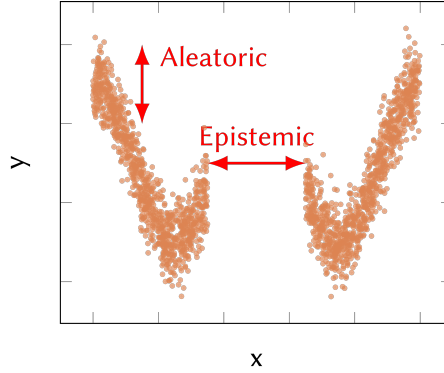


Figure 2: Aleatoric and epistemic uncertainty in regression.

3.1 Uncertainty Estimation

In terms of estimating uncertainties, aleatoric uncertainty is commonly estimated within the trained model itself. For classification models, aleatoric uncertainty is trivialized by the softmax function, which directly estimates the aleatoric uncertainty $P(y|x)$ (Bishop, 2006).

By default, most regression models do not incorporate any uncertainty estimate, as they only produce a scalar or vector which approximates the maximum likelihood prediction y . Therefore, regression models must be explicitly designed to produce an uncertainty estimate. Perhaps the most straightforward way of incorporating aleatoric uncertainty is to output a Gaussian distribution $\mathcal{N}(\mu, \sigma)$, by estimating the mean μ and standard deviation σ (Lakshminarayanan et al., 2017), where μ is used as the prediction, and σ is used as a measure of aleatoric uncertainty. Another, more involved method, is energy-based regression (Gustafsson et al., 2020), which directly approximates $P(x, y)$, and produces a maximum likelihood estimate of $P(y|x)$ by optimizing over y while keeping x fixed.

As epistemic uncertainty concerns a model’s parameters, it has to be estimated by external means. One of the earliest and most common methods of modelling epistemic uncertainty in neural networks is by using an *ensemble*. An ensemble is a collection of multiple, usually identical, models that are trained independently on the same data, effectively forming a number of different samples from $P(\theta|\mathcal{D})$ (Gawlikowski et al., 2021). A similar method is that of Monte Carlo dropout (Gal and Ghahramani, 2016). Dropout is a layer that can be added to a neural network that randomly sets intermediate features to 0 (Srivastava et al., 2014). Thus, computing several predictions with dropout enabled effectively simulates different samples from $P(\theta|\mathcal{D})$.

Another possibility for epistemic uncertainty is to approximate it using a separate model, which is the method adopted throughout this thesis. Since epistemic uncertainty is related to the amount of data that is available during training, scoring functions for out-of-distribution detection can be used as an estimate (Gawlikowski et al., 2021).

3.2 Out-of-Distribution Detection

Given a probability distribution $p(\mathbf{x})$, called the in-distribution, out-of-distribution detection is a classification problem with the goal of determining whether a data point is likely to have been sampled from $p(\mathbf{x})$ (Yang et al., 2024). This is illustrated in figure 3.

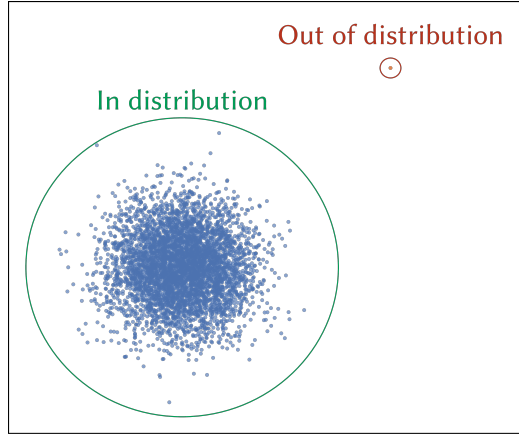


Figure 3: Illustration of out-of-distribution detection.

There are other classification problems with slightly different end goals: *anomaly detection*, *outlier detection* and *novelty detection* (Yang et al., 2024). However, all three problems utilize similar methods, and for the purposes of epistemic uncertainty estimation, they can be considered to be equivalent. The term out-of-distribution detection is used throughout this thesis, because it most accurately reflects the goal of estimating epistemic uncertainty.

Out-of-distribution methods broadly fall into two categories: those that rely on reconstruction, and those that rely on data modelling (Yang et al., 2024). Reconstruction-based methods rely on the assumption that an autoencoder trained to reconstruct samples from $p(\mathbf{x})$ will likely produce higher errors when attempting to reproduce out-of-distribution samples. Data modelling approaches, on the other hand, con-

struct approximate models of the probability distribution $p(\mathbf{x})$ either directly or indirectly. Indirect methods include the one-class SVM, which transforms the in-distribution such that it forms a hypersphere with minimal radius, and classifies samples based on the distance to the center of the hypersphere.

It is worth mentioning here that there are methods that perform out-of-distribution detection, with relative success, by using the softmax outputs $P(y|x)$ of a trained classifier (Yang et al., 2024). However, it has been shown that such methods perform poorly compared to other methods, and it is trivial to construct examples where they fail entirely (Xia and Bouganis, 2024).

3.3 Gaussian Processes

It is difficult to discuss uncertainty in the context of machine learning without also discussing Gaussian processes (Rasmussen and Williams, 2005). Gaussian processes are a class of non-parametric models that are commonly cited as the gold standard in terms of machine learning methods with uncertainty estimates. Two main features set Gaussian processes apart from most other methods: they are non-parametric, and they are constructed with a probabilistic notion of both aleatoric and epistemic uncertainty.

Instead of producing a singular maximum likelihood estimate, like most machine learning methods, predictions from Gaussian processes are Gaussian distributions:

$$y \sim \mathcal{N}(\mu, \sigma) .$$

Here, the final prediction is typically $y = \mu$, which corresponds to the maximum likelihood, and σ is taken to represent the uncertainty.

A Gaussian process is defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$. The choice of $m(\mathbf{x})$ is relatively inconsequential, and therefore it is usually defined to be a constant function $m(\mathbf{x}) = \mathbf{0}$. For the covariance function, the most common choice is the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{1}{2}|\mathbf{x}-\mathbf{x}'|^2} .$$

Shared among all kernel functions is the idea that data points that are close to each other should have a high covariance value, while points farther away should have a lower covariance.

Given training data $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ and $Y = [y_1, \dots, y_n]$, aleatoric uncertainty is modelled directly by assuming that each y_i is sampled with Gaussian noise:

$$y_i \sim \mathcal{N}(y_i^{\text{true}}, \epsilon) .$$

Tuning ϵ can either constrain or relax the resulting model, and setting $\epsilon = 0$ results in exact interpolation with regard to the predicted mean μ .

Using $m(\mathbf{X})$ to denote element-wise application of $m(\mathbf{x}_i)$, and $k(\mathbf{X}, \mathbf{X})$ to denote pair-wise application of $k(\mathbf{x}_i, \mathbf{x}_j)$, results in the following formulas for the predictive mean μ and variance σ :

$$\begin{aligned} \mu &= m(\mathbf{x}^*) + k(\mathbf{x}^*, \mathbf{X}) [k(\mathbf{X}, \mathbf{X}) + \epsilon^2 \mathbf{I}]^{-1} (Y - m(\mathbf{X})), \\ \sigma &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, \mathbf{X}) [k(\mathbf{X}, \mathbf{X}) + \epsilon^2 \mathbf{I}]^{-1} k(\mathbf{X}, \mathbf{x}^*) , \end{aligned}$$

where $\mathbf{x}^*$ denotes a previously unseen point, for which a prediction $y^* \sim \mathcal{N}(\mu, \sigma)$ is made.

Importantly, the formula for the predictive σ explicitly encodes both aleatoric and epistemic uncertainty. Specifically, aleatoric uncertainty is modelled by ϵ , and epistemic uncertainty is modelled by the covariance with regard to the training data points $k(\mathbf{x}^*, \mathbf{X})$. As a result, Gaussian processes produce a unified scalar that encapsulates both forms of uncertainty. It is also interpretable in the form of a variance, which makes it highly desirable for real-world applications.

The primary drawback of Gaussian processes, hindering their wider deployment, is the term $[k(\mathbf{X}, \mathbf{X}) + \epsilon^2 \mathbf{I}]^{-1}$, which is the inverse of a large matrix that scales with the number of data samples. Thus, computing a Gaussian process is intractable for most modern datasets, given their size.

4 Normalizing Flows

Normalizing flows are a class of models for density estimation, with the capability of both sampling and evaluating their target probability density (Papamakarios et al., 2021). These properties make them a versatile model for applications dealing with uncertainty, and as such they are used extensively throughout this thesis.

Most, if not all, interesting machine learning problems form complex probability distributions that are difficult to reason about directly. Normalizing flows are constructed based on the idea of transforming a complex distribution into a simple one that is easy to reason about, and tractable to compute (Papamakarios et al., 2021). Formally, a normalizing flow is a diffeomorphism: a bijective transformation $T : \mathbb{R}^n \mapsto \mathbb{R}^n$, such that both T and T^{-1} are differentiable.

Given a target probability density $p_x(\mathbf{x})$, a normalizing flow is constructed with the goal of transforming $p_x(\mathbf{x})$ into a different distribution $p_u(\mathbf{u})$. It can be shown that such a transformation can be constructed for any $p_u(\mathbf{u})$, and hence $p_u(\mathbf{u})$ can be chosen freely, with the most common choice being $p_u = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Formally:

$$\begin{aligned} \mathbf{u} &= T(\mathbf{x}), \quad \mathbf{x} = T^{-1}(\mathbf{u}) \quad \text{where} \\ \mathbf{x} &\sim p_x(\mathbf{x}), \quad \mathbf{u} \sim p_u(\mathbf{u}) \quad . \end{aligned}$$

Additionally, T being a diffeomorphism allows for an exact expression to evaluate $p_x(\mathbf{x})$ (Papamakarios et al., 2021):

$$p_x(\mathbf{x}) = p_u(T(\mathbf{x})) |\det J_T(\mathbf{x})| \quad ,$$

where $J_T(\mathbf{x})$ denotes the Jacobian matrix for T . Thus, normalizing flows can sample from $p_x(\mathbf{x})$ by drawing samples $\mathbf{u} \sim p_u(\mathbf{u})$ and applying the inverse transform $\mathbf{x} = T^{-1}(\mathbf{u})$, in addition to being able to evaluate the probability density $p_x(\mathbf{x})$ by applying the transform $T(\mathbf{x})$.

While it is possible, in theory, to construct a transform T for any distributions $p_x(\mathbf{x})$ and $p_u(\mathbf{u})$, it is not trivial in practice. Modern implementations typically rely on neural networks to construct T . However, both the inverse transform T^{-1} and the Jacobian determinant $|\det J_T(\mathbf{x})|$ are intractable for neural networks of any meaningful size. Therefore, the structure of the neural network has to be altered to make these operations tractable.

The most common neural network-based implementation of T , and the one adopted throughout this thesis, is so-called *coupling layers*, popularized by RealNVP (Dinh et al., 2017). Given an input $\mathbf{x} = [x_1, \dots, x_n]$, a coupling layer splits $\mathbf{x}$ into disjoint parts (Papamakarios et al., 2021). Although $\mathbf{x}$ can be split arbitrarily, the standard choice is to split it into two equal halves:

$$\mathbf{x}_1 = [x_1, \dots, x_{n/2}], \quad \mathbf{x}_2 = [x_{n/2+1}, \dots, x_n] \ .$$

$\mathbf{x}_1$ is then modified by an element-wise invertible and differentiable function, while keeping $\mathbf{x}_2$ fixed. Though other choices exist, the most common is an element-wise linear function:

$$\begin{aligned} \mathbf{x}'_1 &= \mathbf{x}_1 \cdot \boldsymbol{\alpha} + \boldsymbol{\beta} \\ \mathbf{x}'_2 &= \mathbf{x}_2 \ . \end{aligned}$$

Finally, the output is computed by concatenation:

$$\mathbf{x}' = [\mathbf{x}'_1 | \mathbf{x}'_2] \ .$$

This construction results in a triangular Jacobian:

$$J_T(\mathbf{x}) = \begin{bmatrix} \alpha_1 & \mathbf{0} & \mathbf{0} & \dots & \dots & \dots \\ \mathbf{0} & \ddots & \mathbf{0} & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \alpha_{n/2} & \dots & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 1 \end{bmatrix} \ ,$$

for which the determinant is a product along the diagonal:

$$\det J_T(\mathbf{x}) = \prod_{i=1}^{n/2} \alpha_i \ .$$

In turn, this also allows for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to be computed arbitrarily. A common choice is to let them be neural networks operating on $\mathbf{x}_2$:

$$\boldsymbol{\alpha} = \text{NN}_{\alpha}(\mathbf{x}_2), \quad \boldsymbol{\beta} = \text{NN}_{\beta}(\mathbf{x}_2) \ . \quad (3)$$

Applying a simple element-wise function to $\mathbf{x}_1$ may seem too restrictive for a normalizing flow of this type to model any complex probability densities. Indeed, it is unknown whether coupling layers are universal function approximators (Papamakarios et al., 2021). Nonetheless, there is evidence that they are capable enough for many practical applications, for example image generation (Dinh et al., 2017).

4.1 Normalizing Flows for Out-of-Distribution Detection

While normalizing flows have been successful in sampling from complex densities (Dinh et al., 2017), it has been observed that they often fail at evaluating densities for images (Kirichenko et al., 2020). Work by Kirichenko et al. suggests that normalizing flows tend to learn local correlations between pixels, which produces misrepresentative density estimates for out-of-distribution images. One of their proposed solutions to alleviate this issue, and the one adopted in this thesis, is to utilize a pre-trained neural network, typically a convolutional network, as a feature extractor, and then training the normalizing flow to approximate the density of those features. This is illustrated in figure 4.

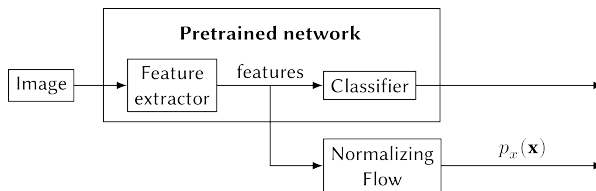


Figure 4: Illustration of how normalizing flows are applied to features from a pretrained network.

In addition to improving the density evaluation for images, operating on features from a pre-trained network allows the normalizing flow to be smaller. Consequently, both training and inference times are significantly reduced.

5 Uncertainty as a Driver for Adaptability

Figure 5 displays an example where camera-based vision fails. In the leftmost picture there is a dog clearly visible. A yellow bounding box is also shown, which is an output detection from YOLOv4 (Bochkovski et al., 2020), thus indicating that the dog has been successfully detected. On the right, there is another picture of the same scene but from a different viewing angle. This change in viewing angle results in different lighting conditions, which in turn renders the objects inside the room completely imperceptible, including the dog. As a result, the rightmost image is entirely unsuitable for robotic vision tasks.

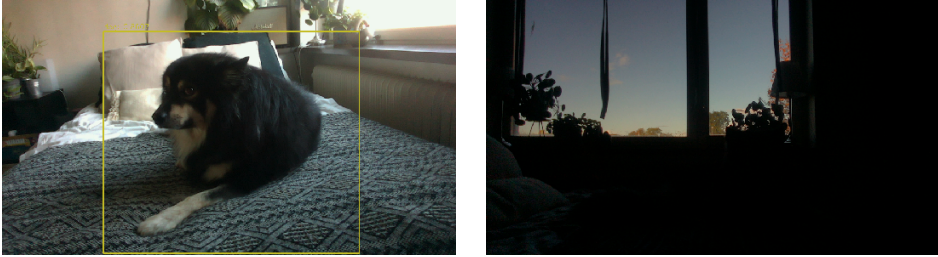


Figure 5: An example where taking a picture of the same scene from two different viewpoints produces entirely different images, one of which is entirely unsuitable for downstream vision tasks.

As mentioned in section 1, all cameras have built-in parameters that can be adjusted to change the final image. The particular images in figure 5 were taken with a Real-sense D435, which exposes a total of 9 parameters: *backlight compensation*, *brightness*, *contrast*, *exposure*, *gain*, *gamma*, *saturation*, *sharpness*, and *white balance*. By adjusting these parameters, the inside of the room can once again be made visible, as shown in figure 6. Note especially that, after adjusting camera parameters, the dog is successfully detected by YOLOv4, which serves as an indication that the resulting image is suitable for robotic vision. Based on this observation, **Paper I** proposes to use the parameters available in a camera to introduce adaptability in robotic vision.

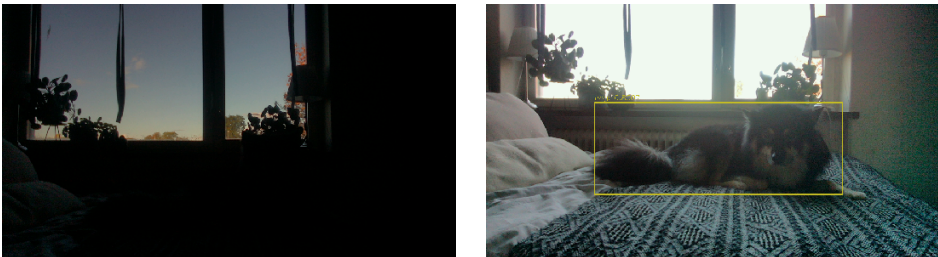


Figure 6: The scene in the rightmost image of figure 5 can be recovered by adjusting camera parameters.

If camera parameters are to be integrated into robots as a tool for adaption, the parameter adjustment needs to be automated, which is non-trivial. There exists no single parameter configuration that works well for all scenarios, as highlighted by figure 5. Additionally, there may be several parameter configurations that work well for a given scene, depending on what part of the image should be visible. For example, in figure 6, it can be argued that both parameter configurations work well depending on whether the outside or inside of the room should be visible. Complexity of this kind makes it impractical to construct a step-by-step algorithm for parameter adjustment, and therefore it is implemented as an optimization problem.

Ultimately, the goal of adjusting camera parameters in the context of robotic vision is to produce an image suitable for neural network-based analysis. It is well known that neural networks perform unreliably for out-of-distribution inputs. Conversely, it is not unreasonable to assume that neural networks perform better for inputs that are more likely to be in-distribution. Therefore, out-of-distribution detection is used as the optimization target metric. Specifically, a normalizing flow is used to approximate the log-likelihood $\ln p(\mathbf{x})$ of an image, and the optimization is formulated as:

$$\max_{\theta} \ln p(\mathbf{x}) ,$$

where θ denotes the available camera parameters.

5.1 Out-of-Distribution Detection for Robot Vision

Out-of-distribution has been used previously in conjunction with robot vision (Yuh et al., 2021; Wellhausen et al., 2020; McAllister et al., 2019). However, in previous works, out-of-distribution has only been applied as a fail-safe, which results in the robot ceasing operation when an input is deemed to be out-of-distribution. For example, Yuh et al. (2021) use out-of-distribution detection in an autonomous car to engage emergency braking when an input is deemed out-of-distribution.

Instead of ceasing operation, our work allows the robot to adapt its camera parameters and continue normal operation.

5.2 Initial Experiments

With the goal of evaluating the proposed camera parameter adaption, a robot experiment setup is constructed. A robot from our lab is used, pictured in figure 7. It is a UR5 arm mounted on a mobile platform, which allows it to easily be moved to explore a variety of visual scenes. A Realsense D435 is mounted at the gripper, which is the robot’s only visual input. Throughout this section, the downstream task is assumed to be object detection with YOLOv4 (Bochkovski et al., 2020).

The camera parameter optimization problem is constructed as illustrated in figure 7. Based on prior experience, our method of choice is evolutionary optimization, as it is easy to implement while offering competitive performance to other more complex algorithms (Kristoffersson Lind and Tykesson, 2021). Throughout these experiments, elitism selection is used, with a population of 50, mutation rate of 20%, no crossover, and a maximum allowance of 200 parameter evaluations. Since YOLOv4 is trained on the COCO dataset (Lin et al., 2015), our normalizing flow is trained to recognize COCO as in-distribution.

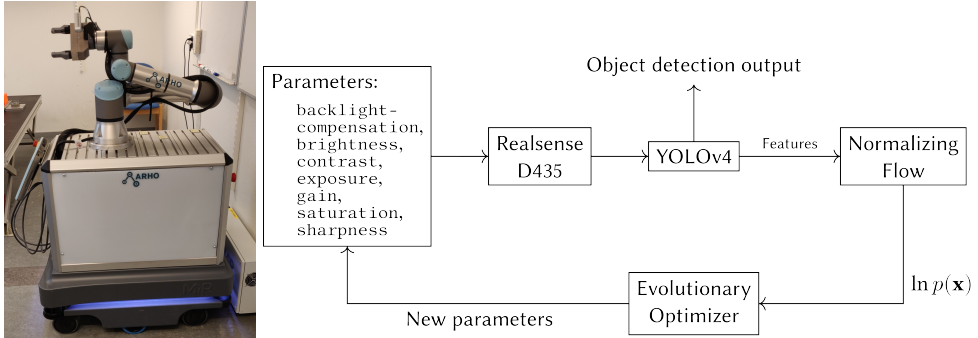


Figure 7: Left: Our robot, Heron. Right: Overview of the parameter optimization pipeline.

Before implementing and testing the full parameter optimization problem, an initial experiment is performed to verify the expressive capabilities of the camera parameters. Our robot is placed in our lab and images are captured with 10000 randomized parameter settings. This is repeated with the light turned off, to get both a light and dark scene. All of the collected images are run through the normalizing flow, and histograms of the resulting log-likelihoods $\ln p_x(\mathbf{x})$ are shown in figure 8, with log-likelihoods from the COCO training data for comparison. The main takeaway from these histograms is the fact that varying the camera parameters results in different images that cover a wide range of log-likelihood values. Importantly, both light and dark images can be transformed into something the normalizing flow recognizes as in-distribution. In turn, this gives confidence that the camera parameters are indeed capable of adapting to a wide range of visual scenarios.

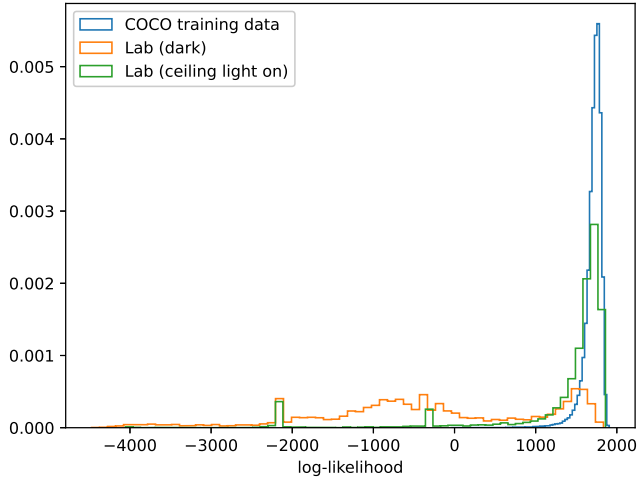


Figure 8: Histogram of log-likelihood values from 10000 different parameter settings in both a light and dark lab setting. The blue histogram shows log-likelihoods from the COCO training data, as the reference in-distribution.

With confidence in the camera parameters, a small-scale experiment is constructed with the full parameter optimization pipeline. Our robot is placed in 13 different scenes with the task of detecting various objects among the COCO object classes. Each scene is specifically constructed to include difficult visual elements, such as bright lights, reflections, and general background clutter. A comparison is made between the camera’s default parameter configuration, and the best configuration found by our evolutionary optimization procedure. On average, our parameter optimization led to an improvement in object detection performance of approximately 5% in the standard mAP metric¹, and 10% in the mAR metric.

¹Both the mAP and mAR metrics have been adopted as standards in the object detection literature. They are usually attributed to the COCO dataset, but they are not included in the paper. Instead, see <https://cocodataset.org/#detection-eval> for a description of the actual metrics.

5.3 Global vs. Local Uncertainty Estimation

While the experiments presented above were successful, they serve as little more than proof of concept, and more data is required to decidedly show the efficacy of the proposed parameter optimization method. In order to gather more data, a fully automated experiment is constructed, in which the same robot is deployed in a tabletop object picking task. A tabletop picking task is constructed with 8 different objects from the COCO categories surrounded by various clutter found in our lab. To make the task more difficult, the ceiling lights are turned off, and a lamp² is placed in the middle of the table facing up, which results in a very difficult visual input. This tabletop picking scene is pictured in figure 9. Note that the leftmost picture is taken with the ceiling light on only for illustrative purposes, and the ceiling light is kept off for the entirety of the picking experiment. Finally, the experiment is automated by letting the robot arm move autonomously to random positions on a semi-circle above the tabletop, with the camera angled directly at the scene.

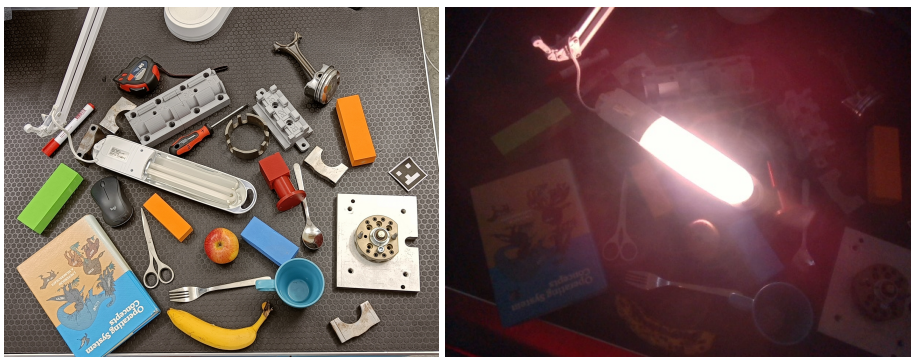


Figure 9: Cluttered tabletop picking scene. Left: ceiling lights on. Right: Ceiling lights off.

Upon deploying our previously proposed camera parameter optimization pipeline in this experiment, it often produced poor images with only the lamp visible, as shown in figure 10. One possible explanation for this behaviour is that the tabletop scene itself, especially in the dark, is out-of-distribution in reference to the COCO training data. Hence, the normalizing flow might produce low log-likelihoods for all images, regardless of their parameter configuration, which in turn might invalidate the ordering of log-likelihood values. It was observed, however, that for many parameter configurations, such as the one shown in figure 9, many of the objects were visible, and would likely be classified as in-distribution when viewed in isolation. This led to the hypothesis that our parameter optimization could be improved by constructing an uncertainty estimate that can be evaluated locally in a region of the input image.

²Side note: the original lamp broke and was replaced roughly halfway through the experiments, that's why you'll see two different lamps in the figures.



Figure 10: Example where our initial parameter optimization failed in the new automated experiment.

Normalizing flows, by default, produce a singular log-likelihood value for an entire image, and it is not obvious how they could be used to construct a local uncertainty estimate. The work presented in **Paper II** draws inspiration from saliency maps (Simonyan et al., 2014) used in image classification tasks, and investigates the pixel-wise gradient with regard to the log-likelihood calculated by the normalizing flow.

Saliency maps are produced by calculating the pixel-wise gradient with respect to a specific class label (Simonyan et al., 2014), with the reasoning that they highlight which pixels contributed to choosing that specific label. Normalizing flows do not produce class labels, but rather a single log-likelihood score. In spite of this difference, a similar reasoning can be applied, in that the pixel-wise gradient highlights which pixels contribute to the log-likelihood score. Since gradients tend to be small close to an optimum, it is reasonable to assume that regions of an image that are out-of-distribution (far from an optimum) should exhibit larger gradient values. Thus, given an input image I , the absolute pixel-wise gradients are computed to form a gradient image ∇I :

$$\nabla I_{yx} = \left| \frac{\delta \ln p(I)}{\delta I_{yx}} \right| ,$$

where I_{yx} denotes the pixel at coordinate (x, y) . This is illustrated in figure 11.

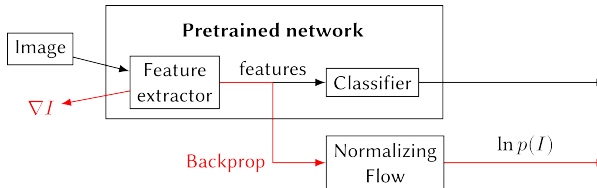


Figure 11: Illustration of how the normalizing flow absolute gradient image ∇I is computed using backpropagation.

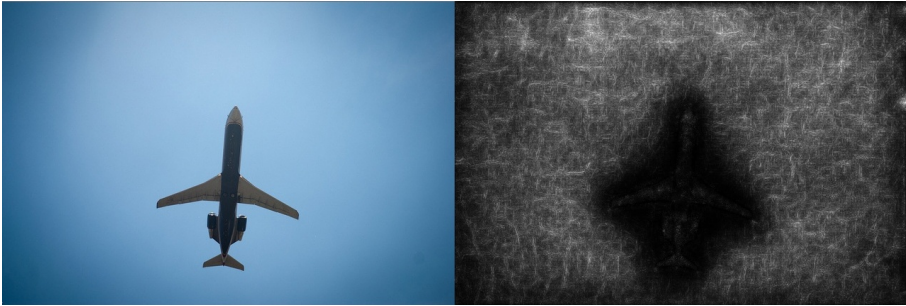


Figure 12: Example visualization of the absolute image gradient with respect to the normalizing flow log-likelihood.

Upon visual inspection of this absolute gradient image, our reasoning appears to hold, as shown in figure 12. Of course, visual inspection is insufficient as evidence. Therefore, two large-scale empirical tests are performed.

First, a classical out-of-distribution experiment is performed using the Cifar10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and CelebA (Liu et al., 2015) datasets, where one dataset is trained as in-distribution and the others should be classified as out-of-distribution. An EfficientNet (Tan and Le, 2020) pre-trained on ImageNet (Russakovsky et al., 2015) is used as the backbone for feature extraction. Since this experiment classifies whole images as in- or out-of-distribution, the average of a gradient image is computed to produce a singular score for a whole image. Results in table 1 show that the absolute gradient is able to classify out-of-distribution images with a similar accuracy compared to the log-likelihood.

Table 1: Success rates for out-of-distribution detection at a fixed 5% false-positive rate. Comparison between direct log-likelihood and our proposed absolute gradient image. Rows denote the training dataset used as in-distribution, and columns denote the out-of-distribution datasets.

		$\ln p(I)$			∇I		
		Cifar10	SVHN	CelebA	Cifar10	SVHN	CelebA
Train	Cifar10	-	0.4148	0.9953	-	0.6688	0.9205
	SVHN	0.999	-	1.0	0.972	-	0.9999
	CelebA	1.0	1.0	-	0.9464	0.9747	-

Second, an experiment is performed with the COCO dataset, in which the absolute gradient is compared between correct and incorrect object bounding boxes. Here, the average gradient is computed only inside each object’s predicted bounding box. For YOLOv4, the absolute gradients within correct bounding boxes are roughly 19% smaller on average compared to incorrect bounding boxes. These results are reproduced for Faster-RCNN (Ren et al., 2016), for which the difference is roughly 21%. Both results are shown in table 2.

Table 2: Average absolute gradients for COCO object detection.

	Correct bounding box	Incorrect bounding box
YOLOv4	0.4447	0.5640
Faster-RCNN	0.7658	0.9510

Together, these experiments show empirically that the proposed absolute gradient is capable of estimating uncertainty at the pixel-level. A noteworthy consequence of providing uncertainty at a pixel-level is the fact that only a single forward and backward pass through the feature extractor and normalizing flow is required, which then allows for an uncertainty estimate over any given region within an image by averaging over pixels.

Returning to our automated tabletop object detection experiment, the absolute gradient image is integrated as an alternative uncertainty estimate in our camera parameter optimization pipeline. To leverage the local nature of the gradient image, the average gradient is computed only within the predicted bounding boxes from YOLOv4. When no bounding boxes are predicted, the average is instead computed over the entire gradient image as a fallback. Throughout 83 object detection trials, with randomized viewing angles, optimizing camera parameters based on the gradient image resulted in 114 successful object detections, which is a 60% improvement compared to optimizing based on the log-likelihood directly, and a 150% improvement compared to the camera’s default parameters. Counterintuitively, the camera’s default parameters with auto-exposure enabled performed the worst out of the 4 configurations tested. The total successful detections for each configuration are summarized in table 3.

Table 3: Total successful object detections for the automated experiment.

Default Parameters Autoexposure off	Default Parameters Autoexposure on	Optimization w.r.t. $\ln p(I)$	Optimization w.r.t. ∇I
45	42	71	114

5.4 Qualitative Observations

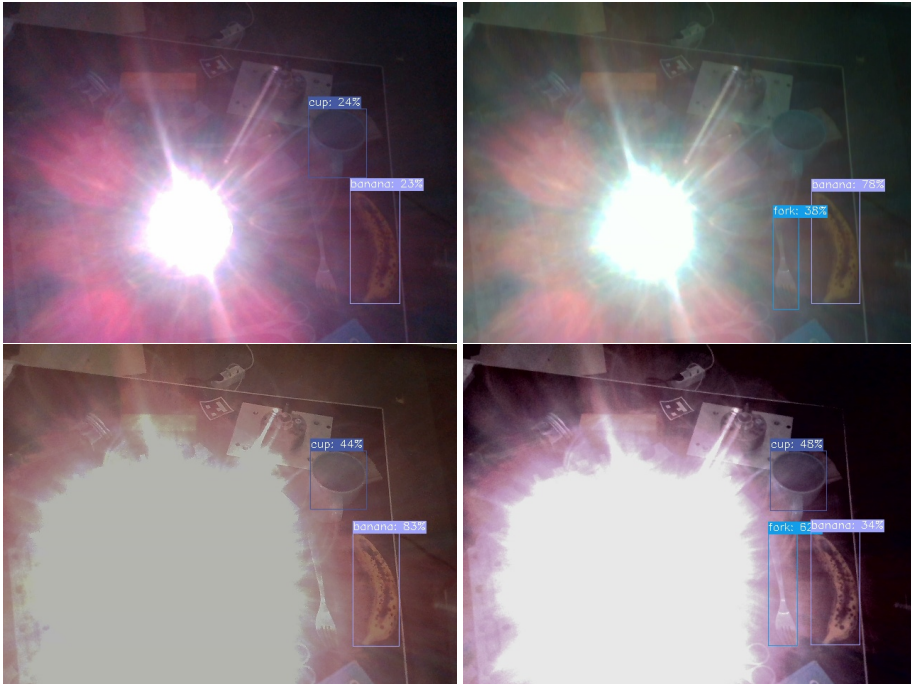


Figure 13: Example images from the automated tabletop object detection experiment. Top left: Default camera parameters. Top right: Default camera parameters with autoexposure. Bottom left: Parameters optimized w.r.t. $\ln p(I)$. Bottom Right: Parameters optimized w.r.t. ∇I .

Figure 13 shows images captured in one of the 83 trials with YOLO’s predicted bounding boxes drawn on top. It is immediately clear that the parameter configuration optimized based on ∇I is the only one where 3 objects are successfully detected, even though the image produced by optimizing based on $\ln p(I)$ looks similar to the human eye. Interestingly, both images produced by our camera parameter optimization feature a larger region that is over-exposed. Nonetheless, these two images both feature a clearer view of the visible COCO objects compared to the two images with default parameters. An important observation, highlighted by these example images, is the fact that the parameter configurations that are most suitable for downstream vision tasks, are not necessarily the same ones that a human would select. This reinforces the argument from section 1, that image quality metrics based on human preferences are likely not ideal in the context of adaptable vision.

6 Calibration for Regression

Given that the work presented in section 5 relies on uncertainty estimation, it is reasonable to analyze the uncertainty estimate itself. Uncertainty estimation is ultimately a question about trust. Predictions from a neural network cannot naively be trusted, and thus uncertainty estimates are used to convey a notion of how trustworthy a given prediction is. However, it can be argued that the uncertainty estimate itself cannot be naively trusted either. Therefore, this section explores methods of assessing how accurate and trustworthy an uncertainty estimate is.

As outlined in section 3.2, epistemic uncertainty is closely related to the problem of out-of-distribution detection, and indeed out-of-distribution detection is an appropriate setting to verify epistemic uncertainty estimates. If an estimate can accurately determine whether data samples are in- or out-of-distribution, then it is deemed trustworthy. Since out-of-distribution detection deals with data distributions directly, it is valid for both classification and regression tasks.

Aleatoric uncertainty is dealt with in different ways for classification and regression tasks. For classification tasks, aleatoric uncertainty is evaluated by measuring *calibration* (Guo et al., 2017). A model is said to be calibrated if the predictive confidence matches the true probability:

$$P(y|x) = p \text{ .}$$

Here, p denotes the true probability that the label y is correct. This definition leads to a natural notion of *calibration error*:

$$E[|P(y|x) - p|] \text{ .}$$

In practice, p has to be empirically estimated, which leads to a metric called *expected calibration error* (ECE) (Guo et al., 2017). Given an evaluation dataset $\mathcal{D} = \{(x_1, y_1), \dots (x_n, y_n)\}$, ECE is calculated by splitting the data into bins $b_1, \dots, b_m$:

$$b_i = \left\{ (x_k, y_k) \mid \frac{i-1}{m} < P(y_k|x_k) \leq \frac{i}{m} \right\} \text{ ,}$$

and subsequently calculating the difference between the average confidence and accuracy for each bin:

$$\text{ECE}(\mathcal{D}) = \sum_{i=1}^m \frac{|b_i|}{n} |\text{conf}(b_i) - \text{acc}(b_i)| \text{ ,}$$

where

$$\begin{aligned}\text{conf}(b_i) &= \frac{1}{|b_i|} \sum_{(x_k, y_k) \in b_i} P(\hat{y}_k | x_k) , \\ \text{acc}(b_i) &= \frac{1}{|b_i|} \sum_{(x_k, y_k) \in b_i} \mathbf{1}(\hat{y}_k = y_k) .\end{aligned}$$

Throughout these formulas, $|b_i|$ denotes the number of data points in bin b_i , $\hat{y}_k$ denotes the predicted label corresponding to (x_k, y_k) , and $\mathbf{1}$ denotes the indicator function.

While there have been criticisms of ECE (Popordanoska et al., 2022), those criticisms are directed at the use of binning rather than the notion of calibration error itself. In other words, there have been different methods proposed for computing calibration error, but the concept itself remains undisputed.

In contrast to classification, regression tasks do not lend themselves to the same ease of analysis, the primary reason being the fact that continuous spaces require probability densities. Hence, evaluating calibration requires estimating the probability density of both the prediction, and the evaluation dataset itself, both of which are non-trivial. Naturally, evaluation is only possible for models that output an uncertainty, and therefore only models with uncertainty estimates are considered throughout this section.

There is no de-facto method of measuring calibration for regression models, but rather several methods used in different works. A rigorous analysis of these methods is lacking, however. As such, the goal of **Paper III** is not the creation of a new method to measure calibration, but rather to provide an analysis of existing methods.

6.1 Calibration Metrics for Regression Tasks

Four primary metrics are identified throughout literature, that each serve the purpose of measuring calibration for regression models: *Area Under Sparsification Error* (AUSE) (Ilg et al., 2018), *Spearman’s Rank Correlation* (Spearman, 1904), *Negative Log-Likelihood* (NLL) (Lakshminarayanan et al., 2017), and *Calibration Error* (CE) (Kuleshov et al., 2018).

Note that, while they borrow the same name, the calibration error proposed by Kuleshov et al. (2018) is not identical to the concept of calibration error for classification. Hence, the shorthand CE is used to differentiate it from the ECE metric for classification.

Area Under Sparsification Error

Given an evaluation dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and corresponding predictions $\hat{y}_1, \dots, \hat{y}_n$, a *sparsification curve* is computed based on the *mean absolute error* (MAE):

$$\text{MAE}(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| .$$

The sparsification curve is generated by recording the MAE, while iteratively removing the data point with the highest predicted uncertainty. Under this procedure, the resulting sparsification curve should ideally be monotonically decreasing, which implies that the predicted uncertainty is correlated with the error. A so-called *oracle* curve is generated by repeating the same procedure of recording MAE, but instead iteratively removing the data point with the highest error. No sparsification curve can decrease faster than the oracle curve, and thus AUSE is formulated as the area between the sparsification curve and the oracle.

Spearman's Rank Correlation

Given a list $L = [l_1, \dots, l_n]$, the *rank* of an element l_i is the index that element would have if the list were sorted, formally (Spearman, 1904):

$$\text{rank}(l_i) = 1 + |\{l_k \mid l_k < l_i, l_k \in L\}| ,$$

where $|\{\dots\}|$ denotes the number of elements in the set. A corresponding sequence can be constructed:

$$\text{Rank Seq}(L) = [\text{rank}(l_1), \dots, \text{rank}(l_n)] .$$

As the name suggests, Spearman's rank correlation is the correlation between two rank sequences. While this is an old method, originally constructed by Spearman for his small-scale psychology experiments, it has received newfound attention as a calibration metric (Tan et al., 2023). In this context, the rank correlation is computed between the predicted uncertainty, and the absolute prediction errors.

Negative Log-Likelihood

NLL is a common scoring function, used across many different machine learning applications (Bishop, 2006). Formally, it is nothing more than the sum of negative log-likelihoods over an evaluation dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$:

$$\text{NLL}(\mathcal{D}) = - \sum_{(x_i, y_i) \in \mathcal{D}} \log p(y_i | x_i) .$$

It has been shown that NLL is a *strictly proper scoring rule* (Gneiting and Raftery, 2007), which means that it is minimal if and only if the predicted distribution $p(y|x)$ is equal to the true data distribution. This property makes NLL useful as a measure of calibration, since it measures how well the learned uncertainty matches the data distribution. Its only notable caveat, in principle, is the fact that it requires the uncertainty estimate to be a probability density, which prohibits its use with other type of uncertainty estimates, like loss prediction (Yoo and Kweon, 2019).

Calibration Error

Kuleshov et al. (2018) present a variant of ECE that is adapted for regression. Since regression tasks do not allow for direct probability estimation, Kuleshov et al. instead choose to formulate a score based on the cumulative distribution of the estimated uncertainty:

$$P(y_i | x_i) = p(y < y_i | x_i) \ .$$

This formulation allows for a binning approach similar to the one used in ECE. Bins $b_1, \dots, b_m$ are constructed from the evaluation dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$:

$$b_i = \{y_k | P(y_k | x_k) \leq p_i\} \ ,$$

where $p_i = \frac{i}{m}$ denotes the upper bound for bin b_i . Note the absence of a lower bound. These bins are turned into empirical frequencies:

$$\hat{p}_i = \frac{|b_i|}{n} \ ,$$

and subsequently compared to the true frequencies p_i :

$$\text{CE}(\mathcal{D}) = \sum_{i=1}^m w_i (\hat{p}_i - p_i)^2 \ ,$$

where w_i are arbitrary scaling weights. Intuitively, this formula follows a similar logic compared to the original calibration formulation for classification: for a given threshold, say 50%, the cumulative distribution should report $P(y_k | x_k) < 50\%$ for approximately 50% of the samples.

6.2 Analysis

For the purpose of analyzing calibration metrics, 4 toy problems are constructed, each featuring a different form of uncertainty:

- *homoscedastic*: Gaussian noise with fixed standard deviation,
- *heteroscedastic*: Gaussian noise with variable standard deviation,
- *multimodal*: two distinct modes in the data distribution,
- *epistemic*: missing training data in a fixed region.

Datasets are created from each toy problem, consisting of 100000 training, 1000 validation, and 1000 test samples. Two different models are trained for each toy dataset: a Gaussian ensemble model (Lakshminarayanan et al., 2017), and an energy-based regression model (Gustafsson et al., 2020). These models exhibit some differences in terms of learning the training data distribution, which makes it interesting to observe whether these differences can be accurately detected by each calibration metric.

All 4 toy datasets, along with the models' predicted uncertainties, are shown in figure 14. Note especially the central region in the epistemic dataset, where the predictions from both models deviate significantly from the evaluation data. That is the region where data was omitted during training to create epistemic uncertainty.

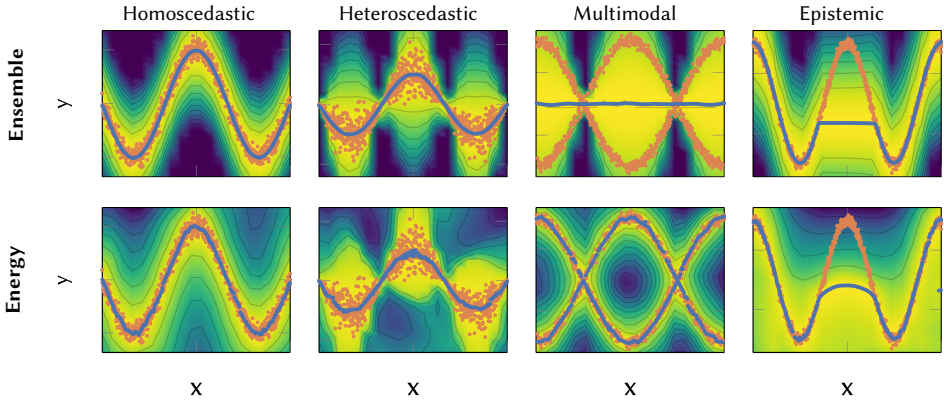


Figure 14: Toy datasets with predicted uncertainties. Countour plots: predicted uncertainty. Yellow indicates low uncertainty, and dark blue indicates high uncertainty. Orange scatter plot: evaluation data. Blue scatter plot: predictions $\hat{y}_i$. Top row: Gaussian ensemble model. Bottom row: energy-based model.

Results from calculating all 4 calibration metrics for both models, and all 4 toy datasets are presented in table 4. One of the benefits of using toy datasets is that the ground truth data distribution is known and can be evaluated. Metrics for the ground truth distributions are also displayed in table 4 for comparison.

Table 4: All calibration metrics for both models on all toy datasets. ↓: lower value is better. ↑: higher value is better.

		AUSE↓	CE↓	NLL↓	Spearman↑
Homoscedastic:	Ensemble	0.5915	0.0023	-0.8819	0.0444
	Energy-Based	0.5707	0.0032	-0.8568	-0.0264
	Ground truth	0.5917	0.0003	-0.8965	0.0076
Heteroscedastic:	Ensemble	0.2334	0.0001	-0.1091	0.0005
	Energy-Based	0.2422	0.0001	-0.0990	-0.0419
	Ground truth	0.2305	0.0001	-0.1472	0.0280
Multimodal:	Ensemble	0.0071	0.0229	0.8098	-0.0420
	Energy-Based	0.5821	0.0018	-0.6535	0.0093
	Ground truth	0.5180	0.0001	-0.7935	0.0073
Epistemic:	Ensemble	0.6016	0.0145	36.4332	0.0298
	Energy-Based	1.3888	0.0298	31.2425	0.0067
	Ground truth	0.5454	0.0001	-1.5871	-0.0194

From the plots, it can be observed that both models learn an uncertainty that looks to fit the homoscedastic and heteroscedastic data well. The energy based model appears to also fit the multimodal dataset well, whereas the ensemble fails spectacularly due to only being able to output a singular Gaussian, though it accurately represents the error in its predictions. Both models fail to model the out-of-distribution region in the epistemic dataset, which is entirely expected.

Combining the visual observations with the table leads to a few interesting conclusions. First and foremost, Spearman’s rank correlation fails completely, producing values that are seemingly random, and all close to 0, even for the ground truth. Since Spearman’s rank correlation computes a direct correlation between ranks, it is directly unsuitable for large and dense datasets, as small differences in error or uncertainty can lead to large differences in rank. Second, AUSE appears to reflect our visual observations well, except in the case of the multimodal dataset, where it seems to imply that the ensemble is better than the ground truth. This highlights the fact that AUSE correlates uncertainty with prediction errors, and does not directly compare the probability densities, which may or may not be desirable depending on the application. Finally, both NLL and calibration error match our visual observations well across all datasets, though they are not always in agreement as to which model is better. A major drawback of NLL in the context of calibration is the fact that it is difficult to interpret on its own, since it has no clear bounds. Hence, NLL is useful for comparing different models, but might not be ideal for measuring calibration with the goal of determining whether an uncertainty estimate is trustworthy.

6.3 Convergence and Bias

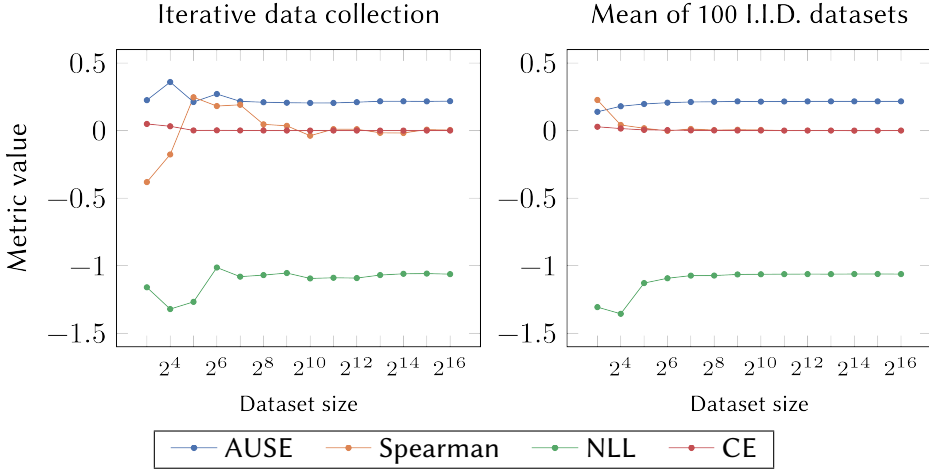


Figure 15: Left: simulated iterative data collection. Right: average of 100 I.I.D. datasets of different sizes.

All of the metrics investigated in this section are estimated based on a finite evaluation dataset. Hence, the size of the evaluation dataset has the potential to greatly impact the resulting estimate. First, all estimates are assumed to converge to a “true” value for a large enough evaluation dataset, and therefore it is of interest to evaluate how much data is required for each metric to reach convergence. Second, an ideal metric should be an unbiased estimator, even for small data samples.

In order to investigate the convergence of each metric, a data collection procedure is simulated by iteratively sampling a validation dataset from our heteroscedastic toy problem. Starting from 4 samples, each metric is evaluated with the ensemble model at multiples of 2 up to 2^{16} samples.

A similar procedure is used to investigate bias. Instead of iteratively sampling a single dataset, 100 datasets are sampled I.I.D. from our heteroscedastic toy problem at each size, again from 4 up to 2^{16} samples.

Results for both of these experiments are shown in figure 15. In terms of convergence, it is clear that none of the metrics are converged at only 4 or 16 evaluation samples, which is entirely expected. CE appears to reach convergence the fastest, already at 2^5 samples, though, all metrics appear to have converged at a dataset size of 2^{10} . As such, convergence should pose no issue for modern machine learning applications, unless data is particularly difficult to acquire. Somewhat surprisingly, all of the metrics appear to be biased at very small dataset sizes. AUSE and CE seem to be the least biased at small dataset sizes. Like with convergence, bias appears to pose no real issue at any meaningful dataset size.

7 Combined uncertainty estimates

Looking back at the camera parameter adaption presented in section 5, especially in hindsight after the analysis of calibration in section 6, it is clear that out-of-distribution detection is not a complete solution for uncertainty estimation. While the original reasoning still holds, out-of-distribution detection entirely neglects any notion of aleatoric uncertainty. Initially, it may seem reasonable to simply add an aleatoric uncertainty estimate, resulting in one estimate for each type of uncertainty. However, if integrated into the context of camera parameter adaption, our optimization problem in section 5 would turn into a multi-objective optimization problem, thus greatly increasing its complexity (Deb et al., 2016). Therefore, it is desirable to instead create a singular, unified uncertainty estimate that models both epistemic and aleatoric uncertainty.

7.1 SCOD

Throughout this thesis, uncertainty estimates are used as a means for adapting and overcoming difficult situations. Parallel to this, there exists a research problem called *selective classification* (El-Yaniv and Wiener, 2010), which is essentially a regular classification problem with the added option to reject a data sample. As the name suggests, classification is the primary task, and rejection is merely a mechanism to improve classification accuracy over the non-rejected samples. Thus, selective classification is a clear use-case for aleatoric uncertainty estimates as a rejection criterion.

Selective classification has later been extended into a problem called *unknown detection* (Kim et al., 2023) or *Selective Classification in the presence of Out-of-distribution Data* (SCOD) (Xia and Bouganis, 2024). These two terms refer to the same problem, and the term SCOD is used throughout this thesis as it more clearly conveys the focus on classification as the primary task. Again, as the name suggests, SCOD is an extension of selective classification where out-of-distribution samples should also be rejected.

SCOD is formalized as a dual classification problem. The evaluation data is split into three distinct categories:

- $ID_{\checkmark}$: data points that are in-distribution, and correctly classified.
- $ID_{\times}$: data points that are in-distribution, but incorrectly classified.
- OOD: data points that are out-of-distribution.

Based on these categories, AUROC scores are computed separately for two classification tasks: ID_✓ vs. ID_×, and ID_✓ vs. OOD. Essentially, ID_✓ vs. ID_× reflects the original selective classification problem, and ID_✓ vs. OOD is added to account for out-of-distribution samples.

Viewed in the context of uncertainty estimation, the classification problem ID_✓ vs. ID_× measures the efficacy of an aleatoric uncertainty estimate, and ID_✓ vs. OOD measures the efficacy of an epistemic uncertainty estimate. However, it is worth noting that the standalone problem of out-of-distribution detection is a classification problem of ID vs. OOD, where ID denotes the union ID_✓ ∪ ID_×. Hence, the ID_✓ vs. OOD classification task offers a different, potentially less nuanced, approach of evaluating an epistemic uncertainty estimate.

7.2 SIRC

Softmax Information Retaining Combination (SIRC) proposed by Xia and Bouganis (2024), is a method of combining two confidence scores s_1 and s_2 , where s_1 is assumed to perform well for ID_✓ vs. ID_× classification, and s_2 is assumed to perform well for ID_✓ vs. OOD classification. In the context of this thesis, s_1 represents an aleatoric uncertainty estimate, and s_2 represents an epistemic uncertainty estimate. Recall that confidence and uncertainty can be considered equivalent, but inverse. SIRC is formulated as follows:

$$SIRC(s_1, s_2) = -(s_1^{max} - s_1) (1 + e^{-b(s_2 - a)}) ,$$

where s_1^{max} denotes a known upper bound for the s_1 score. This equation forms a sigmoid in the (s_1, s_2) -plane, and the parameters a and b control its location and shape.

Xia and Bouganis evaluate SIRC using different combinations of scores s_1 and s_2 . For s_1 they evaluate both the *maximum softmax probability* (MSP)

$$\text{MSP} = \max_i \pi_i ,$$

and the negative softmax entropy

$$\mathcal{H} = - \sum_i \pi_i \log \pi_i .$$

In both of these formulas π_i denotes the output probabilities from a softmax operation.

All scores for s_2 operate on a feature vector $\mathbf{z}$, which denotes the input before the last hidden layer of the classification model. They evaluate the following scores for s_2 : the L_1 -norm $\|\mathbf{z}\|_1$, the *residual score* $\|\mathbf{z}^{P^\perp}\|_2$ by Wang et al. (2022), and the L_2 -normalized distance to $\mathbf{z}$'s k th nearest neighbor in the training data. Note that $\mathbf{z}$ is the same type of features used with normalizing flows throughout our work.

7.3 GPify

On the surface, SIRC appears fit our requirements. It is, after all, a singular score that combines both aleatoric and epistemic uncertainty estimates. SIRC would likely work well if integrated into our camera parameter optimization in section 5. However, SIRC is difficult to interpret, mainly because it has an upper bound of 0 and entirely lacks a lower bound. In robotics, there is generally a human operator in charge of overseeing the robot's operation. As such, it is desirable for an uncertainty estimate to be interpretable by a human to facilitate efficient and reliable decision-making. Therefore, **Paper IV** proposes *GPify*, which is a combined uncertainty estimate with a focus on interpretability.

Like our previous work in section 5, normalizing flows are chosen here to represent epistemic uncertainty. Normalizing flows are a good fit for our requirements in this section since they implement an exact formula for the learned probability density. Additionally, as they represent densities directly, they can be made interpretable with maintained probabilistic integrity. For aleatoric uncertainty, we follow SIRC and use the maximum softmax probability. Based on the general availability of softmax probabilities in classification models, there is little need to replace it with a different estimate.

Making Normalizing Flows Interpretable

Normalizing flows, by default, output log-likelihoods $\ln p(\mathbf{x})$, which are not easily interpretable for a human operator. Therefore, the first step in designing an interpretable score is to make the normalizing flow interpretable. This is done by converting its log-likelihoods into probabilities by using an empirical distribution given a dataset S :

$$\Pr(\mathbf{x}) = \frac{|\{\mathbf{x}_i \mid \ln p(\mathbf{x}_i) \leq \ln p(\mathbf{x}), \mathbf{x}_i \in S\}|}{|S|}.$$

As usual, $||$ denotes the number of elements in a set. S can either be taken from the training or validation data for the normalizing flow, or it can be sampled from the normalizing flow itself. Sampling S from the normalizing flow is likely beneficial when data is sparse.

Designing a Combined Score

GPify is designed with inspiration from Gaussian processes (GP), hence the name. As outlined in section 3.3, Gaussian processes produce an interpretable uncertainty estimate by design, that reflects both aleatoric and epistemic uncertainty.

In order to analyze, and ultimately mimic Gaussian processes, a Gaussian process is trained on a simple two-class classification problem, and its estimated confidence is plotted in figure 16. Figure 16 also visualizes the confidences from a sigmoid classifier, and a normalizing flow with an empirical probability as described above. Note that the sigmoid is used in place of softmax since this is a two-class problem. The Gaussian process exhibits two primary features that are desirable for an interpretable confidence: first, it has a baseline uncertainty $= \frac{1}{2}$ in regions where data is sparse, and second, it has a negative mode to express that a data point explicitly is *not* class 1.

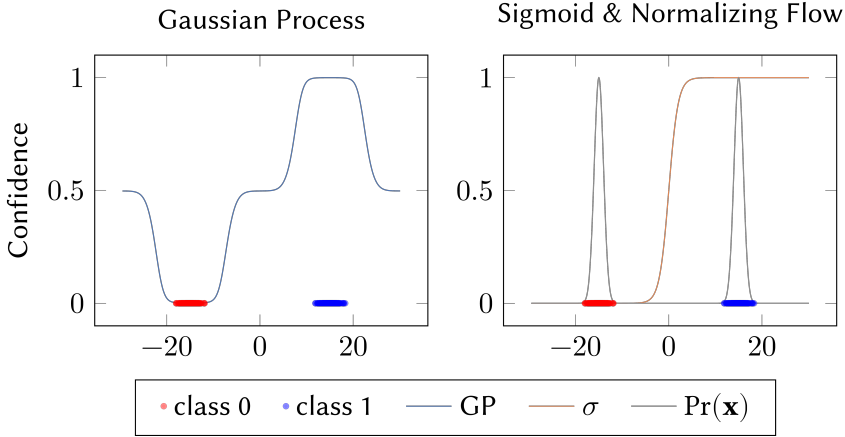


Figure 16: A trivial two-class classification problem. Left: Gaussian process confidence. Right: Sigmoid and normalizing flow confidence.

Normalizing flows have no notion of classes, which results in one mode of confidence for each class. Conversely, the sigmoid is able to differentiate between classes, but does not possess a baseline uncertainty like the Gaussian process.

A first step towards combining the softmax and normalizing flow comes from the observation that multiplying their confidences effectively *selects* a mode. Additionally, the sigmoid can be shifted by a constant $-\frac{1}{2}$ without loss of classification performance. Multiplying the normalizing flow's confidence by a shifted sigmoid results in a negative mode similar to the Gaussian process. Of course, this results in confidence values < 0 , which is undesirable from an interpretability standpoint. Shifting the result back into the range $[0, 1]$ by adding the baseline uncertainty $\frac{1}{2}$ results in our

formulation of GPify:

$$\text{GPify}(\mathbf{x}) = \left(\sigma(\mathbf{x}) - \frac{1}{2} \right) \cdot \Pr(\mathbf{x}) + \frac{1}{2} ,$$

where σ is used to denote the sigmoid function. This construction is illustrated in figure 17. Notice that our GPify score mimics the exact structure of the Gaussian process, with the only major difference being the fact that GPify’s modes are significantly narrower.

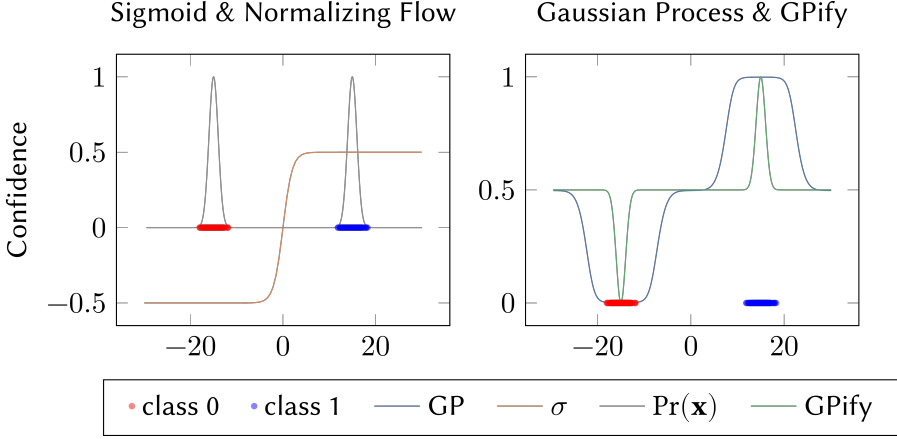


Figure 17: A trivial two-class classification problem. Left: Sigmoid shifted by $-\frac{1}{2}$, and normalizing flow. Right: Gaussian process and GPify.

Extending GPify to multi-class problems is done by replacing sigmoid with the maximum softmax probability, and adjusting the baseline uncertainty to be $\frac{1}{C}$ where C denotes the number of classes. This leads to our final, generalized formulation of GPify:

$$\text{GPify}(\mathbf{x}) = \left(\text{MSP}(\mathbf{x}) - \frac{1}{C} \right) * \Pr(\mathbf{x}) + \frac{1}{C} .$$

In order to verify that GPify is well-behaved even for multi-class problems, an experiment is constructed using the MNIST dataset (Lecun et al., 1998). MNIST is visualized in 2D using UMAP (McInnes et al., 2020). Specifically, a parametric version of UMAP (Sainburg et al., 2021) allows for a bijective transformation between digits and the 2D visualization space. In turn, the bijective UMAP allows contour plots to be made from each respective models’ confidence estimates. Figure 18 shows a comparison between GPify and a Gaussian process when applied to MNIST. Note, again, the similarity between the predicted confidences. Additionally, here GPify’s narrower confidence modes appear to be an asset compared to the Gaussian process, which assigns more confidence mass outside of the data regions.

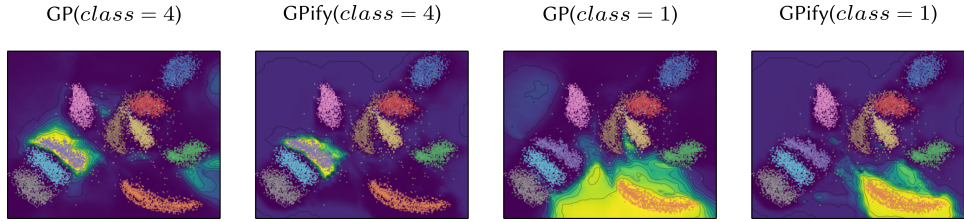


Figure 18: Contour plots from MNIST, mapped into the 2D plane using UMAP. Each color of the scatter points indicate a digit class. The two leftmost plots display their respective model’s confidence in $class = 4$, and the two rightmost display confidence for $class = 1$.

7.4 Interpretability

Since the foundation of GPify is a multiplication of the softmax probability $P(y|\mathbf{x})$, and the normalizing flow probability $\Pr(\mathbf{x})$, a possible interpretation is that GPify forms a joint distribution, roughly:

$$\text{GPify}(\mathbf{x}) = P(y, \mathbf{x} \sim p_x(\mathbf{x})) .$$

In the context of robotics, this can in turn be interpreted as a probability that both the image capture and image analysis are correct.

7.5 Empirical Evaluation

While the above visualizations inspire confidence that GPify is well-behaved, they are insufficient for any real conclusions. Therefore, a large-scale SCOD experiment is constructed. For this purpose, a subset of 150 classes from the ImageNet dataset (Russakovsky et al., 2015) is used. These classes are further divided into an in-distribution dataset with 100 classes, which leaves 50 classes out-of-distribution.

A classifier and corresponding normalizing flow are trained on the in-distribution dataset. GPify is then evaluated and compared to SIRC and all its variations in terms of: $\text{ID}_\checkmark$ vs. $\text{ID}_\times$ classification, $\text{ID}_\checkmark$ vs. OOD classification, and ID vs. OOD classification. Note the addition of ID vs. OOD classification, which is not present in the original SCOD problem setting. This is added for the purpose of ensuring that GPify does not degrade the performance of the standalone normalizing flow in terms of pure out-of-distribution detection.

In order to verify that GPify is not dependent on a specific classifier architecture, this SCOD experiment is repeated four times with different classifier architectures: ResNet (He et al., 2016), EfficientNet (Tan and Le, 2020), ViT (Dosovitskiy et al., 2021), and Swin (Liu et al., 2021).

Section 7.3 briefly mentioned that there are different possibilities for constructing the dataset S , that is used as the basis of the empirical distribution $\Pr(\mathbf{x})$. Different choices of S may impact the resulting performance of GPify, and therefore 3 different methods are evaluated for constructing S . First, S is taken to be the normalizing flow’s training data. Second, S is sampled from the normalizing flow. When sampling from a normalizing flow, a sample is drawn from a standard Gaussian $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and transformed to produce the sample $\mathbf{x} = T^{-1}(\mathbf{u})$. Thus, another way of changing the composition of S is to change the standard deviation in the Gaussian samples: $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \sigma \mathbf{I})$. Our third method optimizes the standard deviation σ to maximize the ID $_{\checkmark}$ vs. ID $_{\times}$ classification accuracy on the normalizing flow’s training data. These 3 variants of GPify are denoted, respectively: GPify $_{|\text{train}}$, GPify $_{|\sigma=1}$, and GPify $_{|\sigma=\text{opt}}$.

Table 5 displays AUROC scores for each evaluated classification task. In the interest of providing easily readable results, table 5 provides a reduced table with scores averaged across the 4 classifier architectures. Additionally, only one variant of SIRC is provided for comparison, specifically the variant where the maximum softmax probability is used for s_1 , and the L $_1$ -norm $\|\mathbf{z}\|_1$ is used for s_2 . This variant of SIRC is chosen as it performed the best overall throughout our experiments. Complete tables of results are available in **Paper IV**.

Table 5: AUROC scores for the SCOD evaluation for GPify compared to best variant of SIRC, averaged over the 4 classifier architectures.

	ID $_{\checkmark}$ vs. ID $_{\times}$	ID $_{\checkmark}$ vs. OOD	ID vs. OOD
SIRC ($s_1 = \text{MSP}$, $s_2 = \ \mathbf{z}\ _1$)	88.32%	85.06%	77.90%
GPify $_{ \text{train}}$	86.61%	86.45%	80.33
GPify $_{ \sigma=1}$	88.18%	87.72%	82.50
GPify $_{ \sigma=\text{opt}}$	88.74%	87.88%	82.20%

It is immediately clear from table 5 that our GPify is a strong contender in the SCOD framework. Overall, it appears that GPify performs marginally worse than SIRC on average for ID $_{\checkmark}$ vs. ID $_{\times}$ classification, though it compensates with superior OOD classification. Since table 5 omits many details, the purpose here is not to claim that GPify is decidedly better than SIRC. Rather, the only goal in this evaluation is to show that GPify offers competitive SCOD performance, while offering advantages in terms of interpretability.

7.6 Adversarial Detection

Adversarial examples are data samples that are purposefully changed to produce a wrong classification (Szegedy et al., 2014). Specifically, they are created by taking an existing image, and adding an imperceptible amount of noise to it. In practice, this is done by a few steps of gradient descent, to minimize the softmax probability of the correct class (Kurakin et al., 2017). As such, Adversarial examples can be considered a special case of out-of-distribution samples. Furthermore, since they are constructed by manipulating the softmax probabilities, they cannot be accurately detected by softmax alone. Therefore, it is interesting to investigate whether GPify can be used to detect adversarial examples despite relying on softmax for aleatoric uncertainty.

An experiment is constructed using the same 4 classifier architectures, and the same dataset from our SCOD evaluation. For each image in the in-distribution dataset, 4 different adversarial examples are created: one using the minimal number of gradient descent steps to produce a wrong classification, and one each when a wrong class is predicted with 90%, 95%, and 99% softmax confidence. $\text{GPify}|_{\sigma=1}$ is then used to classify whether each example is in- or out-of-distribution.

Resulting AUROC scores, in table 6, show convincingly that GPify does possess the capability of detecting adversarial examples. However, the classification performance appears to vary greatly depending on the classifier architecture. Especially EfficientNet appears to stand out among the 4 architectures tested.

Table 6: AUROC scores for adversarial detection.

	ResNet	EfficientNet	ViT	Swin
Minimum noise	96.04%	95.79%	89.45%	79.01%
Wrong class confidence > 90%	97.75%	23.59%	79.64%	80.26%
Wrong class confidence > 95%	97.81%	13.65%	74.84%	83.61%
Wrong class confidence > 99%	98.81%	1.89%	80.43%	92.63%

More thorough evaluation is necessary to determine the extent of GPify’s adversarial detection abilities, why it differs between classifier architectures, and how it compares to dedicated adversarial detectors (Aldahdooh et al., 2022).

8 Uncertainty-Aware Trajectory Generation

Visuomotor policy learning is a recent robotic vision task, where a model is trained to produce fine-grained control commands for a robot given only visual input (Chi et al., 2024). Formally, given an input image I_t , and the current state of the robot s_t , the goal of a visuomotor policy is to produce an action a_t for the robot to execute (Chi et al., 2024). These are indexed by t to indicate time steps. While different tasks may require different forms of the state s_t and action a_t , this section focuses on dual-arm manipulation for which it is convenient to let both s_t and a_t be joint states of the robot’s arms. As such, unless otherwise specified, s_t and a_t are $2 \cdot (J+1)$ -dimensional vectors of the form:

$$[\alpha_1, \dots, \alpha_J, g_1, \beta_1, \dots, \beta_J, g_2]$$

where J indicates the number of joints in each arm, α and β denote joint angles corresponding to the respective arms, and g denotes one gripper state per arm.

While there exist different approaches, a popular implementation of visuomotor policy learning is that of Diffusion Policy (Chi et al., 2024), which uses a diffusion model (Song et al., 2020) to generate the actions a_t . Diffusion models, however, feature two severe weaknesses in the context of robotics: inference is slow, which makes real-time operation difficult, and they do not produce any uncertainty estimates, which makes it difficult to reason about their outputs. Therefore, **Paper V** proposes to use normalizing flows for visuomotor policy learning, since they feature both an exact formula for density estimation and efficient inference.

8.1 Diffusion Models

Although diffusion models are not part of the contribution in this work, a brief introduction is provided here in order to highlight their weaknesses.

On the surface, diffusion models appear remarkably similar to normalizing flows (Ho et al., 2020). When used for sampling, both models draw from a standard Gaussian and subsequently transform the Gaussian sample into their target distribution space. The difference is in the transformation from Gaussian noise. Like normalizing flows, there exists different implementations of diffusion models, but this section only focuses on the two most common: *Denoising Diffusion Probabilistic Models* (DDPM) (Ho et al., 2020) and, *Denoising Diffusion Implicit Models* (DDIM) (Song et al., 2020).

DDPM works by modelling the transformation of a clean sample $\mathbf{x}_0$ to Gaussian noise as a Markov chain (Ho et al., 2020):

$$p(\mathbf{x}_i | \mathbf{x}_{i-1}) = \mathcal{N}(\sqrt{\alpha_i} \mathbf{x}_{i-1}, (1 - \alpha_i) \mathbf{I}) , \quad (4)$$

which can be simplified to

$$p(\mathbf{x}_i | \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_i} \mathbf{x}_0, (1 - \bar{\alpha}_i) \mathbf{I}), \quad \bar{\alpha}_i = \prod_{s=1}^i \alpha_s .$$

Here, α_i is called the *noise schedule* and controls how quickly the Markov chain converges to pure Gaussian noise. A neural network is implemented to model this Markov chain by first reparameterizing it:

$$\mathbf{x}_i = \sqrt{\bar{\alpha}_i} \mathbf{x}_0 + (1 - \bar{\alpha}_i) \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) ,$$

and subsequently training a neural network $\epsilon_\theta(\mathbf{x}_0, i)$ to predict ϵ using the standard L_2 loss

$$\mathcal{L} = \|\epsilon - \epsilon_\theta(\mathbf{x}_0, i)\|^2 .$$

In order to sample from the target data distribution, a sample $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is drawn, from which noise is iteratively estimated and removed using the trained $\epsilon_\theta(\mathbf{x}_i, i)$:

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{\alpha_i}} \left(\mathbf{x}_i - \frac{(1 - \alpha_i)^2}{\sqrt{1 - \bar{\alpha}_i}} \epsilon_\theta(\mathbf{x}_i, i) \right) + \sigma_i^2 \epsilon ,$$

until eventually a noise-free sample $\mathbf{x}_0$ is recovered. The term $\sigma_i^2 \epsilon$ serves to make the sampling process stochastic, but a common choice is to maintain a deterministic sampling process by setting $\sigma_i = 0$.

DDIM is a direct extension of DDPM (Song et al., 2020). A major constraint in DDPM is the fact that the forward and backward Markov chain must consist of the same number of steps n . By reformulating the backward Markov chain as $p(\mathbf{x}_{i-1} | \mathbf{x}_i, \mathbf{x}_0)$, DDIM decouples the forward and backward process. Naturally, the clean sample $\mathbf{x}_0$ is unknown during sampling, and therefore it is replaced with an estimate:

$$\mathbf{x}_0 \approx f_\theta(\mathbf{x}_i) = \frac{\mathbf{x}_i - \sqrt{1 - \bar{\alpha}_i} \epsilon_\theta(\mathbf{x}_i, i)}{\sqrt{\bar{\alpha}_i}} ,$$

which is then used to step through the backward Markov chain:

$$\mathbf{x}_{i-k} = \sqrt{\bar{\alpha}_{i-k}} f_\theta(\mathbf{x}_i) + \sqrt{1 - \bar{\alpha}_{i-k} - \sigma_i^2} \epsilon_\theta(\mathbf{x}_i, i) + \sigma_i \epsilon .$$

Note the index $i - k$, which allows for arbitrarily large jumps through the generating process.

It has been observed that a large number of steps n is beneficial during training, although the number of sampling steps can be reduced somewhat without significantly degrading the sample quality (Song et al., 2020). Hence, DDIM allows for faster sampling, while keeping a high number of training steps.

Since diffusion models generate samples by iterative application of the neural network $\epsilon_\theta(\mathbf{x}_i, i)$, their inference time scales linearly with the number of denoising iterations. In practice, it is rare to see fewer than 10 denoising iterations. Diffusion Policy, for example, uses 15 as the lowest number of denoising iterations (Chi et al., 2024). Hence, any practical diffusion-based model is an order of magnitude slower compared to models that only require a single neural network computation.

Based on the above formula (4), it is theoretically possible, but intractable to evaluate the probability density $p(\mathbf{x}_0)$. As such, no common implementation of diffusion models feature any kind of uncertainty estimation. It is possible to implement a diffusion model as a continuous-time ODE instead of a Markov chain (Yang et al., 2023), which in turn allows for density estimation. However, the continuous-time ODE is an equivalent formulation for normalizing flows (Papamakarios et al., 2021), and it is seldomly used in practice as it relies on numerical ODE solvers.

8.2 Normalizing Flow Policy

In order for normalizing flows to be applied to visuomotor policy learning, they need to be adapted to model a conditional distribution:

$$p(a_t | o_t) ,$$

where o_t represents the *observation*, which contains both the image I_t and the robot state s_t . Modifying a normalizing flow based on coupling layers to model a conditional distribution is trivial, by adding the observation as an additional input to the neural networks from equation (3):

$$\text{NN}_\alpha(\mathbf{x}_2, o_t), \text{NN}_\beta(\mathbf{x}_2, o_t) .$$

Note that adding an additional input to the NN-functions does not affect the Jacobian determinant, nor the invertibility of the normalizing flow.

A common strategy, also employed by Diffusion Policy (Chi et al., 2024) is called *action chunking*, and it involves predicting a sequence of actions, based on a sequence of past observations:

$$p(a_t, \dots, a_{t+n} | o_t, \dots, o_{t-m}) .$$

By operating on sequences rather than singular actions and observations, temporal consistency is improved over the generated actions.

Strided Training

During initial testing, it was observed that the normalizing flow would often learn to model unwanted motions, believed to stem primarily from stationary periods and noise in the training data. Therefore a stride s is incorporated when gathering samples from the training data. With this stride, the normalizing flow is instead trained to model:

$$p(a_t, a_{t+s}, \dots, a_{t+ns} \mid o_t, o_{t-s}, \dots, o_{t-ms}) .$$

This is illustrated in figure 19 for a stride of $s = 2$. Effectively, adding a stride in this way allows the normalizing flow to bypass small, unwanted motions present in the training data, and hence increase focus on learning the intended behavior.

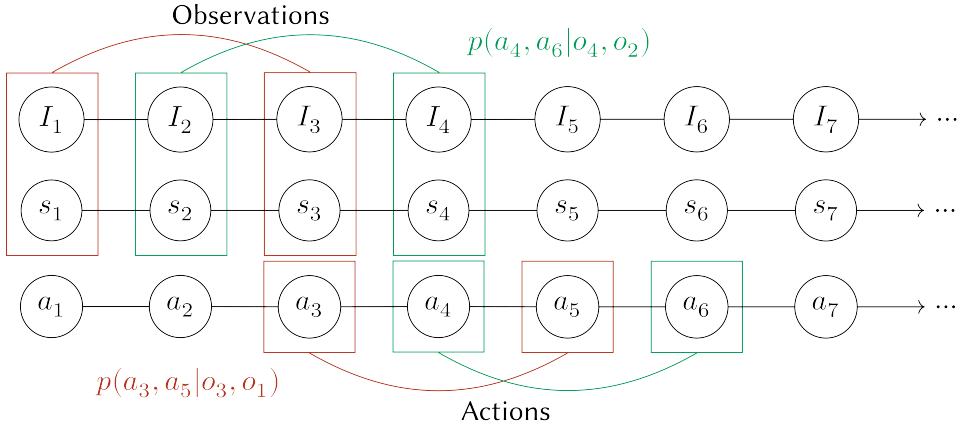


Figure 19: Illustration of our strided sampling of observations and actions during training. Here with $s = 2$.

Likelihood Optimization

Since both diffusion models and normalizing flows produce action sequences by sampling the target distribution $p(a|o)$, there is no guarantee that the sampled action is of any quality. It is reasonable to expect that actions sampled from high-density regions will be better compared to samples from low-density regions, and therefore two primary methods are explored with the goal of producing samples from higher-density regions in the action space. First, by sampling with a smaller standard deviation $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \sigma \mathbf{I})$ with $\sigma < 1$, the samples should, in expectation, correspond to higher-density regions in the action space. Second, optimization is performed to increase the likelihood of the sampled action, in order to explicitly produce higher-quality samples.

For the optimization, two methods are evaluated. *Stochastic batch selection* draws several samples from $p(a|o)$ at once, and returns the one with the highest likelihood, thus selecting higher-density samples. *Gradient refinement* generates a single sample that is refined by a small number of iterations of gradient ascent to maximize the likelihood. Note that neither of these methods are designed with the goal of converging to an optimum likelihood. Stochasticity in the generated action sequence is generally thought to be beneficial for generalization, and therefore our proposed methods are designed with the goal of improving the likelihood without reaching convergence.

Experiments in Simulation

An ablation study is performed using the *Beat Block Hammer* task in the RoboTwin 2.0 simulation framework (Chen et al., 2025b) to validate the design elements in our proposed normalizing flow policy. Specifically the sampling stride s , the sampling standard deviation σ and the likelihood optimization are evaluated. Results indicate that, for RoboTwin, the best configuration is to use gradient refinement with 10 gradient ascent iterations, a sampling stride $s = 4$, and a sampling standard deviation $\sigma = 0.5$, which resulted in a success rate of 51%. Stochastic batch sampling, using 128 samples, and the same stride and standard deviation, performed marginally worse with a 50% success rate. However, stochastic batch sampling is an order of magnitude faster at inference time since it only requires a single passthrough of the normalizing flow.

All 50 simulated tasks in the RoboTwin 2.0 framework are evaluated with our proposed normalizing flow policy, which results in an average success rate of 38.06% using gradient refinement, and 37.16% using stochastic batch sampling. Both of these results mark a decisive improvement over Diffusion Policy, which has an average success rate of 28.04% listed on the official RoboTwin 2.0 leaderboard (Chen et al., 2025c).

One concern that has hindered the adoption of normalizing flows across different fields is the fact that there is no proof of their capabilities in terms of universal function approximation (Papamakarios et al., 2021), at least for coupling layers. While the above experiments show promising results, they were trained using only 50 episodes of data, which might not be enough to reach the limit of the model’s expressive capabilities. With the goal of showing that normalizing flows are capable of scaling to larger data quantities, an experiment is performed using 4 of the RoboTwin 2.0 tasks. Each task is repeated using 10, 25, 50, 100, 200, 300, and 500 episodes of training data. Results, shown in figure 20, indicate that normalizing flow policy exhibits similar scaling properties compared to Diffusion Policy, which alleviates any concerns regarding model capacity.

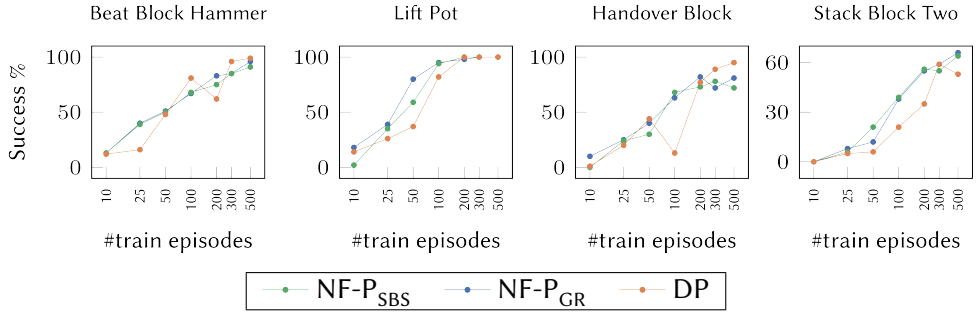


Figure 20: Results from 4 RoboTwin tasks using varying amounts of training data. NF-P_{SBS}: normalizing flow policy with stochastic batch sampling. NF-P_{GR}: normalizing flow policy with gradient refinement. DP: Diffusion Policy.

Real Robot Experiment

To verify that the performance of our normalizing flow policy generalizes to real robots, a small-scale experiment is performed using a dual-arm KUKA iiwa in our lab. Two tasks are set up: one in which two wooden cubes are to be stacked on top of each other in a pre-defined order, and one where a towel is to be folded. Both tasks are shown in figure 21.

Training data is collected by human tele-operation. 150 training episodes are collected for the block stacking task, and 100 training episodes for the towel folding task. For these experiments, normalizing flow policy is used with the stochastic batch sampling strategy, with 128 batched samples, $s = 6$ as the training stride³ and a sampling standard deviation $\sigma = 0.5$. Diffusion Policy is also evaluated for comparison.

20 trials were performed for each task, and the number of successful attempts are reported in table 7 for both models. Our normalizing flow policy appears to perform markedly better than Diffusion Policy for both tasks.

Throughout these experiments, it was observed that Diffusion Policy struggled significantly with stalling, especially when transitioning between sub-goals of each task. For example, Diffusion Policy would often get stuck on the block stacking task after having placed the first block, where it would repeat the motion to place the first block ad infinitum rather than moving on to place the other block. Our normalizing flow policy did exhibit some minor stalling between sub-goals, but significantly less than Diffusion Policy. One factor that highlights this difference in stalling, is the fact that Diffusion Policy was given a time window of 8 minutes to progress between sub-goals, whereas our normalizing flow policy was only given 30 seconds.

³During initial testing, $s = 6$ was found to work better on the real robot, though no thorough ablation study was performed in this setting.



Figure 21: The two real-world robot experiments. Top: block stacking. Bottom towel folding. The red circles in the leftmost pictures highlight the camera used to view the scene.

Table 7: Number of successful attempts in our real-world experiment. Out of 20 total attempts.

	Block Stacking	Towel Folding
Diffusion Policy	3	11
Normalizing Flow Policy	12	15

8.3 Inference Time, Training and Convergence

One of the primary benefits of using normalizing flows as a base model over diffusion models, is the fact that they can perform sampling using only one passthrough of the model. In this work, two variants of normalizing flow policy are proposed. Gradient refinement runs in the same order of magnitude as Diffusion Policy, as they both rely on an iterative procedure to generate their actions. For reference, inference with Diffusion Policy takes approximately 700ms on our Nvidia A100 GPU. Our normalizing flow policy with gradient refinement is somewhat faster, taking approximately 450ms. Stochastic batch selection, on the other hand, runs in 19ms, which enables real-time policy execution with only a marginal decrease in performance compared to gradient refinement.

A factor that is often overlooked in modern neural network applications is training and convergence time. Throughout all the above experiments, our normalizing flow policy has been trained for 300 epochs, which takes between 30 minutes to 2 hours depending on the size of the training dataset. Diffusion Policy, however, when trained for our real-world experiment, did not reach convergence until roughly 2400 epochs of training, which took approximately 12 hours. As such, normalizing flow offers not only improved inference times, but also improved training times, thus allowing for a significantly faster deployment cycle.

9 Conclusions

This thesis presents the overarching argument that uncertainty estimation is a valuable tool for robotic vision. Not only can uncertainty estimation help us reason about our vision models, it can also directly improve the performance and reliability of our vision models by enabling adaptive behaviors.

First, the work presented in section 5 lays the foundation for adaptable vision. Built-in camera parameters are proposed as a basis for adaptation, and it is shown that the built-in parameters in most cameras can adapt to a wide variety of visual scenarios. In order to facilitate automated adaption, an optimization problem is formulated, where camera parameters are optimized to minimize uncertainty estimated by a normalizing flow. Empirical results show that our proposed parameter optimization is an effective tool to improve the reliability of robotic vision. Additionally, the gradients from normalizing flows are shown to reflect uncertainty at a pixel-level, which can be used to estimate uncertainty of local regions in an image. This local uncertainty estimate is shown to further improve reliability in tasks where only a part of the image needs to be considered.

Further, our work presented in section 8 applies the same line of reasoning to visuo-motor policy learning. A normalizing flow policy is proposed, which uses a normalizing flow to produce fine-grained action trajectories based on visual input. The likelihood output from the normalizing flow is used as a guiding metric to produce more high-quality action trajectories. It is shown that our normalizing flow policy achieves, on average, better performance across 50 simulated and 2 real-world robotic tasks when compared to Diffusion Policy.

Section 6 presents an analysis of calibration metrics for regression tasks. By using toy datasets, our analysis shows that area under sparsification error, negative log-likelihood, and calibration error, are all useful metrics, though negative log-likelihood is difficult to interpret. Area under sparsification error measures correlation between uncertainty and prediction error, while negative log-likelihood and calibration error measure how well the predicted distribution matches the ground-truth. As such, they are likely ideal for different tasks, based on the exact goals. Spearman’s rank correlation is also evaluated, but it is shown to be highly unstable.

Finally, a combined uncertainty estimate is presented in section 7, based on the maximum softmax probability and normalizing flows. Our proposed score, GPify, mimics the uncertainty estimate from a Gaussian process, and can be interpreted as a joint distribution $P(y, x)$. It is shown that GPify accurately reflects both aleatoric and epistemic uncertainty, while also possessing some capabilities for adversarial detection.

10 Limitations and Future Work

Our camera parameter optimization problem implemented in section 5 requires the robot to pause its operation while the parameters are being optimized. While pausing operation is not necessarily a problem in many applications, it would be desirable for the optimization to happen continuously, such that the robot never has to pause for extended periods of time. Achieving such a continuous-time optimization is likely incredibly difficult, though, as the visual conditions might change through the course of optimization. Nonetheless, it is an interesting direction for future research.

Additionally, our camera parameter optimization is only evaluated with object detection tasks, which limits the generality of our conclusions. Intuitively, there is no reason to doubt that our optimization approach would improve reliability for other tasks as well, but more experiments with different tasks would be beneficial.

In hindsight, after our work with GPify, perhaps the most glaring limitation throughout section 5 is the fact that only epistemic uncertainty is considered. Arguably, GPify should be used in any future experiments to also incorporate aleatoric uncertainty.

Throughout the entirety of the work in this thesis, normalizing flows has been our model of choice for representing epistemic uncertainty. Although it is our belief that normalizing flows are ideal models for our purposes, our work provides no direct proof of their superiority. Thus, it would be beneficial for future research to evaluate it comparatively alongside other methods.

Finally, and arguably the biggest limitation in this thesis, is the fact that uncertainty estimation is never really applied in any regression tasks. Indeed, visuomotor policy learning can be viewed as a regression problem, and our work in section 8 does apply uncertainty estimation to improve the performance of our normalizing flow policy. However, by modelling it as a generative problem, rather than discriminative, aleatoric uncertainty is conveniently avoided altogether. While many methods exist to estimate aleatoric uncertainty for regression models, it is difficult to verify them because calibration is more complex than it may initially seem. Our work in section 6 provides useful insights for evaluating calibration, though it is largely limited to 1-dimensional problems. Generalizing calibration to multi-dimensional problems results in so-called Copula calibration (Ziegel and Gneiting, 2013), which is theoretically sound, but quickly becomes intractable due to calculating multi-dimensional integrals. An approach that remains tractable is HDR calibration (Chung et al., 2024), though it does not guarantee calibrated marginal distributions. Part of our ongoing work investigates how calibration can be evaluated in a way that is tractable while ensuring calibrated marginals. Likely, a lot more work is required before multi-dimensional calibration for regression can be considered solved.

11 References

- Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V., and Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297.
- Aldahdooh, A., Hamidouche, W., Fezza, S. A., and Déforges, O. (2022). Adversarial example detection for dnn models: a review and experimental comparison. *Artificial Intelligence Review*, 55(6):4403–4462.
- Axis Communication (2025). Axis f9 104-b mk ii main unit. <https://www.axis.com/products/axis-f9104-b-mk-ii-main-unit>.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, cop, New York, NY.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection.
- Chavez-Garcia, R. O., Guzzi, J., Gambardella, L. M., and Giusti, A. (2017). Image classification for ground traversability estimation in robotics. In Blanc-Talon, J., Penne, R., Philips, W., Popescu, D., and Scheunders, P., editors, *Advanced Concepts for Intelligent Vision Systems*, page 325–336, Cham. Springer International Publishing.
- Chen, C., Au, W., and Liu, S. (2025a). *Robotics: From Theory to Practice*. CRC Press, Boca Raton.
- Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Liang, Q., Li, Z., Lin, X., Ge, Y., Gu, Z., et al. (2025b). Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*.
- Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Liang, Q., Li, Z., Lin, X., Ge, Y., Gu, Z., et al. (2025c). Robotwin 2.0 official leaderboard. <https://robotwin-platform.github.io/leaderboard> Fetched 2026-02-23.
- Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. (2024). Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*.
- Chung, Y., Char, I., and Schneider, J. (2024). Sampling-based multi-dimensional recalibration. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F., editors, *Proceedings of the 41st International Conference on*

- Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 8919–8940. PMLR.
- Deb, K., Sindhya, K., and Hakanen, J. (2016). *Multi-Objective Optimization*. CRC Press.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real nvp. (arXiv:1605.08803). arXiv:1605.08803 [cs, stat].
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. (arXiv:2010.11929). arXiv:2010.11929 [cs].
- El-Yaniv, R. and Wiener, Y. (2010). On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641.
- Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.
- Gawlikowski, J., Rovile Njietcheu Tassi, C., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. (2021). A survey of uncertainty in deep neural networks. *arXiv e-prints*. ADS Bibcode: 2021arXiv210703342G.
- Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378.
- Golchoubian, M., Ghafurian, M., Dautenhahn, K., and Azad, N. L. (2023). Pedestrian trajectory prediction in pedestrian-vehicle mixed environments: A systematic review. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):11544–11567.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- Grimmett, H., Triebel, R., Paul, R., and Posner, I. (2016). Introspective classification for robot perception. *The International Journal of Robotics Research*, 35(7):743–762.
- Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, page 1321–1330, Sydney, NSW, Australia. JMLR.org.

- Gustafsson, F. K., Danelljan, M., Bhat, G., and Schön, T. B. (2020). Energy-based models for deep probabilistic regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 770–778. ISSN: 1063-6919.
- Hentout, A., Aouache, M., Maoudj, A., and Akli, I. (2019). Human–robot interaction in industrial collaborative robotics: a literature review of the decade 2008–2017. *Advanced Robotics*, 33(15–16):764–799.
- Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*.
- Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., and Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667.
- Intel RealSense (2019). D400 series datasheet. <https://support.intelrealsense.com/hc/en-us/articles/360022894693>.
- Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.
- Kim, J., Koo, J., and Hwang, S. (2023). A unified benchmark for the unknown detection capability of deep neural networks. *Expert Systems with Applications*, 229:120461. arXiv:2112.00337 [cs].
- Kirichenko, P., Izmailov, P., and Wilson, A. G. (2020). Why normalizing flows fail to detect out-of-distribution data. In *Advances in Neural Information Processing Systems*, volume 33, page 20578–20589. Curran Associates, Inc.
- Kristoffersson Lind, S. and Tykesson, J. (2021). Self-optimization of camera hardware. Student Paper.
- Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.
- Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning*, page 2796–2804. PMLR.

- Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial examples in the physical world. (arXiv:1607.02533). arXiv:1607.02533 [cs].
- Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 6405–6416, Red Hook, NY, USA. Curran Associates Inc.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Li, Y. and Ibanez-Guzman, J. (2020). Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61.
- Li, Z., Xu, B., Wu, D., Zhao, K., Chen, S., Lu, M., and Cong, J. (2023). A yolo-ggcnn based grasping framework for mobile robots in unknown environments. *Expert Systems with Applications*, 225:119993.
- Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context. (arXiv:1405.0312). arXiv:1405.0312 [cs].
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.
- Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Mayr, M. and Salt-Ducaju, J. M. (2022). A c++ implementation of a cartesian impedance controller for robotic manipulators. arXiv:2212.11215.
- McAllister, R., Kahn, G., Clune, J., and Levine, S. (2019). Robustness to out-of-distribution inputs via task-aware generative uncertainty. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2083–2089.
- McInnes, L., Healy, J., and Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction. (arXiv:1802.03426). arXiv:1802.03426 [stat].

- Mittal, A., Moorthy, A. K., and Bovik, A. C. (2012). No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708.
- Mousavian, A., Eppner, C., and Fox, D. (2019). 6-dof graspnet: Variational grasp generation for object manipulation. page 2901–2910.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
- Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 22(57):1–64.
- Popordanoska, T., Sayer, R., and Blaschko, M. B. (2022). A consistent and differentiable l_p canonical calibration error estimator. In *Advances in Neural Information Processing Systems*.
- Raibert, M., Blankespoor, K., Nelson, G., and Playter, R. (2008). Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825.
- Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.
- Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv:1506.01497 [cs].
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.
- Sainburg, T., McInnes, L., and Gentner, T. Q. (2021). Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907.
- Saveriano, M., Abu-Dakka, F. J., Kramberger, A., and Peternel, L. (2023). Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13):1133–1184.
- Sevastopoulos, C. and Konstantopoulos, S. (2022). A survey of traversability estimation for mobile robots. *IEEE Access*, 10:96331–96347.

- Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. (arXiv:1312.6034). arXiv:1312.6034 [cs].
- Song, J., Meng, C., and Ermon, S. (2020). Denoising diffusion implicit models. *arXiv:2010.02502*.
- Song, Q., Li, S., Bai, Q., Yang, J., Zhang, X., Li, Z., and Duan, Z. (2021). Object detection method for grasping robot based on improved yolov5. *Micromachines*, 12(11):1273.
- Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.
- Stereolabs (2026). Camera controls. <https://www.stereolabs.com/docs/video/camera-controls>.
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. (arXiv:1312.6199). arXiv:1312.6199 [cs].
- Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer International Publishing, Cham.
- Taheri, H. and Xia, Z. C. (2021). Slam; definition and evolution. *Engineering Applications of Artificial Intelligence*, 97:104032.
- Tan, A. R., Urata, S., Goldman, S., Dietschreit, J. C. B., and Gómez-Bombarelli, R. (2023). Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles. *npj Computational Materials*, 9(1):1–11.
- Tan, M. and Le, Q. V. (2020). Efficientnet: Rethinking model scaling for convolutional neural networks. (arXiv:1905.11946). arXiv:1905.11946 [cs, stat].
- Tesla (2016). A tragic loss. <https://web.archive.org/web/20230209082350/https://www.tesla.com/blog/tragic-loss/>. Original accessed 2023-09-15, since removed from Tesla’s own website, hence the archived link.
- TIER IV (2026). Camera control tool documentation. https://tier4.github.io/edge-auto-docs/user_manual/camera-control-tool-user-manual.html.

- Topp, E. and Christensen, H. (2005). Tracking for following and passing persons. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 2321–2327. ISSN: 2153-0866.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Vicentini, F. (2020). Collaborative robotics: A survey. *Journal of Mechanical Design*, 143(040802).
- Wang, H., Li, Z., Feng, L., and Zhang, W. (2022). Vim: Out-of-distribution with virtual-logit matching. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 4911–4920.
- Wellhausen, L., Ranftl, R., and Hutter, M. (2020). Safe robot navigation via multi-modal anomaly detection. *IEEE Robotics and Automation Letters*, 5(2):1326–1333.
- Xia, G. and Bouganis, C.-S. (2024). Augmenting the softmax with additional confidence scores for improved selective classification with out-of-distribution data. *International Journal of Computer Vision*, 132(9):3714–3752.
- Yang, J., Zhou, K., Li, Y., and Liu, Z. (2024). Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 132(12):5635–5662.
- Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2023). Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, 56(4):105:1–105:39.
- Yoo, D. and Kweon, I. S. (2019). Learning loss for active learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 93–102.
- Yuhas, M., Feng, Y., Ng, D. J. X., Rahiminasab, Z., and Easwaran, A. (2021). Embedded out-of-distribution detection on an autonomous robot platform. In *Proceedings of the Workshop on Design Automation for CPS and IoT*, Destion '21, page 13–18, New York, NY, USA. Association for Computing Machinery.
- Ziegel, J. F. and Gneiting, T. (2013). Copula calibration. arXiv:1307.7650 [stat].

Scientific publications

Paper I



Out-of-Distribution Detection for Adaptive Computer Vision

Simon Kristoffersson Lind

Lund University LTH

`simon.kristoffersson_lind@cs.lth.se`

Rudolph Triebel

German Aerospace Center and

Technical University of Munich

`rudolph.triebel@dlr.de`

Luigi Nardi

Lund University LTH

`luigi.nardi@cs.lth.se`

Volker Krüger

Lund University LTH

`volker.krueger@cs.lth.se`

Abstract

It is well known that computer vision can be unreliable when faced with previously unseen imaging conditions. This paper proposes a method to adapt camera parameters according to a normalizing flow-based out-of-distribution detector. A small-scale study is conducted which shows that adapting camera parameters according to this out-of-distribution detector leads to an average increase of 3 to 4 percentage points in mAP, mAR and F1 performance metrics of a YOLOv4 object detector. As a secondary result, this paper also shows that it is possible to train a normalizing flow model for out-of-distribution detection on the COCO dataset, which is larger and more diverse than most benchmarks for out-of-distribution detectors.

1 Introduction

In the past decade computer vision has become a de facto component in autonomous systems. However, it is well known that vision can be unreliable when faced with new, previously unseen situations. Uncertainty is especially abundant in the field of robotics, where the autonomous agents are expected to perform tasks in real-world scenarios that often differ significantly from any training data.

There is an abundance of research to suggest that *Out-of-Distribution* (OOD) detection can help improve the reliability of vision algorithms [6]. Relevant literature reasons that OOD detection allows systems to take action when uncertainty is encountered [6]. However, there is little research to show that OOD detection helps the reliability of autonomous systems in practice. Most literature on OOD detection tends to focus either on synthetic benchmarks, or controlled real-world scenarios where in- and out-of-distribution examples are very distinct. Neither of these scenarios are reflective of the diverse situations that may be encountered by autonomous systems in practice [6].

OOD detectors are commonly used by setting a threshold OOD score [9], and simply discarding any input that falls beyond this threshold. While this makes intuitive sense, on the basis that vision systems tend to be unreliable when faced with OOD data, it only utilises one end of the OOD scores. We hypothesize that a machine learning model will on average perform more reliably when an input has a lower OOD score. Therefore we propose to use an OOD detector as a quality metric, or un-normalized confidence; the lower the OOD score, the more certain we can be about our vision system’s output.

1.1 Our Contributions

- We expand the current knowledge on OOD detection by showing that it is possible to train normalizing flow models for OOD detection on large, diverse datasets while getting sensible results.
- We introduce a novel method for utilising OOD detection as a quality metric in autonomous systems and show that it can lead to reliability improvements for vision tasks.

2 Related Work

There has been some previous work investigating the uses of OOD detection in autonomous systems. For example Wellhausen *et al.* [17] perform anomaly detection on image data collected from a robot in real-world terrain. However, their main purpose is the evaluation of different OOD detection algorithms, and not an application of OOD detection to real-world operation.

Yuhas *et al.* [20] evaluate OOD detection as an emergency breaking system for an autonomous car, though not in real-world operation, but on a custom test track.

McAllister *et al.* [12] perform real-world crash avoidance experiments with an autonomous car. However, their method does not perform OOD detection directly. Instead they use a variational autoencoder to generate in-distribution samples from an out-of-distribution image, which they use as a measure of uncertainty.

Additionally, there is an abundance of work on OOD detection where the authors have taken care to ensure that in- and out-distributions are disjoint, for example [4, 7, 10, 18, 15, 13]. These works either use entirely different datasets as in- and out-distributions, or separate classes as in- and out-distributions. It is our belief that neither of these scenarios are reflective of real-world operation.

3 Background

3.1 Out-of-Distribution Detection

Conceptually, out-of-distribution detection is simple. First an in-distribution is defined, for example the training data for a machine learning-based vision system. Then, everything that deviates from said in-distribution is said to be out-of-distribution [19].

More formally, we assume that our vision system operates on data that is sampled i.i.d from some distribution $P_{train}(\mathbf{x})$. In practice, however, the system may encounter data that comes from a different distribution, in which case the system will produce unreliable results. Therefore, it is desirable to detect when data lies outside of the original distribution $P_{train}(\mathbf{x})$.

3.2 Normalizing Flows

This section aims to provide a basic introduction to normalizing flows. For more details we refer to Papamakarios *et al.* [14].

Normalizing flows have emerged in the last decade as a way to model complex probability distributions [14]. Let us assume that we want to evaluate or sample from a distribution $p_x(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^D$, and that p_x is intractable to evaluate or sample from. A normalizing flow can allow us to evaluate or sample from p_x by transforming $p_x(\mathbf{x})$ into a tractable distribution $p_u(\mathbf{u})$, $\mathbf{u} \in \mathbb{R}^D$.

In order to model $p_x(\mathbf{x})$, an invertible transformation T is constructed such that

$$\mathbf{x} = T(\mathbf{u}), \quad \mathbf{u} = T^{-1}(\mathbf{x})$$

where $\mathbf{u} \sim p_u(\mathbf{u})$. Here, we are free to define $p_u(\mathbf{u})$, for example $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.

By additionally requiring both T and T^{-1} to be differentiable, it is possible to recover $p_x(\mathbf{x})$ from $p_u(\mathbf{u})$ [16, 3, 14]:

$$p_x(\mathbf{x}) = p_u(\mathbf{u}) |\det J_T(\mathbf{u})|^{-1} = p_u(T^{-1}(\mathbf{x})) |\det J_{T^{-1}}(\mathbf{x})| \quad (1)$$

where J_T and $J_{T^{-1}}$ are the Jacobians of T and T^{-1} respectively.

This type of transformation is called a *diffeomorphism*. It has been shown that it is always possible to construct a diffeomorphism T under reasonable assumptions about $p_x(\mathbf{x})$ and $p_u(\mathbf{u})$ [14]. However, analytically constructing T is often infeasible in practice. Therefore, it is desirable to learn an approximation of T . Henceforth we will denote $T(\mathbf{x}; \Theta)$ as an approximation of T with learnable parameters Θ .

An important property is that the composition of two diffeomorphisms $T_2 \circ T_1$ is also a diffeomorphism [14]:

$$(T_2 \circ T_1)^{-1} = T_1^{-1} \circ T_2^{-1}$$

$$\det J_{T_2 \circ T_1}(\mathbf{u}) = \det J_{T_2}(T_1(\mathbf{u})) \cdot \det J_{T_1}(\mathbf{u})$$

This property opens up for many ways to construct powerful approximations $T(\mathbf{x}; \Theta)$. In this work, we focus on a particular model of so-called *affine coupling layers* [14].

Affine coupling layers first split the input $\mathbf{x}$ into two parts [14]:

$$\mathbf{x} = [x_i \mid i \in \mathcal{I}], \quad \mathcal{I} = 1, 2, \dots, D$$

$$\mathbf{x}_1 = [x_i \mid i \in \mathcal{A}], \quad \mathbf{x}_2 = [x_i \mid i \in \mathcal{B}], \quad \mathcal{A} \cup \mathcal{B} = \mathcal{I}, \quad \mathcal{A} \cap \mathcal{B} = \emptyset.$$

While $\mathcal{A}$ and $\mathcal{B}$ can be chosen arbitrarily, the most common choice is to simply split $\mathbf{x}$ into two halves:

$$\mathcal{A} = 1, 2, \dots, \frac{D}{2}, \quad \mathcal{B} = \frac{D}{2} + 1, \frac{D}{2} + 2, \dots, D.$$

Then, $\mathbf{x}_1$ is transformed as a function of $\mathbf{x}_2$, while $\mathbf{x}_2$ is left as-is:

$$\mathbf{x}'_1 = [x'_i | i \in \mathcal{A}] = \boldsymbol{\alpha} \cdot \mathbf{x}_1 + \boldsymbol{\beta},$$

where $\boldsymbol{\alpha} = \exp(F_1(\mathbf{x}_2; \Theta_1))$, $\boldsymbol{\beta} = F_2(\mathbf{x}_2; \Theta_2)$.

Here, $\cdot$ denotes element-wise multiplication, and F_1, F_2 are arbitrary functions, for example neural networks. The output of the coupling layer is then simply the reconstruction of $\mathbf{x}$ from $\mathbf{x}'_1$ and $\mathbf{x}_2$:

$$T_{\text{coupling}}(\mathbf{x}; \Theta_1, \Theta_2) = \mathbf{x}' = [x'_i | i \in \mathcal{I}], \quad \text{where } x'_i = x_i \forall i \in \mathcal{B}$$

It is trivial to see that this construction results in an invertible and differential transformation, since $\mathbf{x}_1$ is subject to an affine transformation and $\mathbf{x}_2$ remains the same. Coupling layers also offer the benefit of a simple Jacobian [14]:

$$J_{T_{\text{coupling}}} = \mathbf{1} F_1(\mathbf{x}_2; \Theta_1)$$

where $\mathbf{1}$ denotes a row vector of ones – in other words, summing the elements of $F_1(\mathbf{x}_2; \Theta_1)$.

A common way to construct powerful learnable transformations is to simply compose several of these coupling layers, while changing which parts of $\mathbf{x}$ are modified [14]. This approach is used in for example RealNVP [5], and it is what we use in our experiments.

3.3 Normalizing Flows for Image Out-of-Distribution Detection

Formulating OOD detection using normalizing flows is simple since we can directly compute the likelihood using (1), which represents the belief that an example $\mathbf{x}$ is in-distribution [9]. Note that a high value for $p_x(\mathbf{x})$ equals a low likelihood of being OOD. It is otherwise common for OOD detection systems to use the opposite formulation, where a high value equals high likelihood of being OOD.

Training a normalizing flow for OOD detection is done by minimizing the KL-divergence between $p_x(\mathbf{x})$ and $p_u(\mathbf{u})$, which results in the following loss function [14]:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^N \log p_u(T^{-1}(\mathbf{x}_n)) + \log |\det J_{T^{-1}}(\mathbf{x}_n)| + C$$

where C is a constant.

Since we only need the inverse transformation T^{-1} for OOD detection, we simply let T^{-1} denote the forward pass of our affine coupling layers.

Despite seeming like perfect candidates for OOD detection, since equation (1) allows for the direct computation of the approximate probability distribution, normalizing flows often fail in the OOD detection setting [9]. Kirichenko *et al.* [9] suggest that common image flow models, such as RealNVP[5], simply learn pixel correspondences, and as such fail to grasp any semantic information that makes up the distribution. Instead of training a flow model on images directly, Kirichenko *et al.* suggest training a flow model on features extracted from another pretrained image network.

4 Our Method

Based on our hypothesis that a lower OOD score would lead to better reliability, we propose to let a vision system adapt, instead of simply discarding OOD data. In order to bring this into practice, we create a framework for adaption around an existing vision task, namely object detection with a YOLOv4 [2] network.

Most commercially available cameras expose a number of parameters that can alter the visual appearance of a captured image. Common examples are: saturation, contrast, and exposure. These camera parameters are a natural candidate for adapting to various imaging conditions.

Our proposed method is to simply adapt camera parameters in order to minimize the OOD score. Intuitively, our method can be deployed in two ways:

- either: if an image is marked as OOD, adapt camera parameters to minimize OOD score,
- or: continuously adapt camera parameters to minimize OOD score.

While most cameras have some built-in measures to adapt, for example auto-exposure, these are often not sufficient. Figure 1 shows two images of the same scene to illustrate this. In one picture, the camera’s default settings fail to produce an image suitable for object detection. With hand-tuned parameters however, a more suitable image is acquired. Note also the yellow bounding box indicates that YOLOv4 can detect the dog in the hand-tuned image, which is not possible with the camera’s default settings.

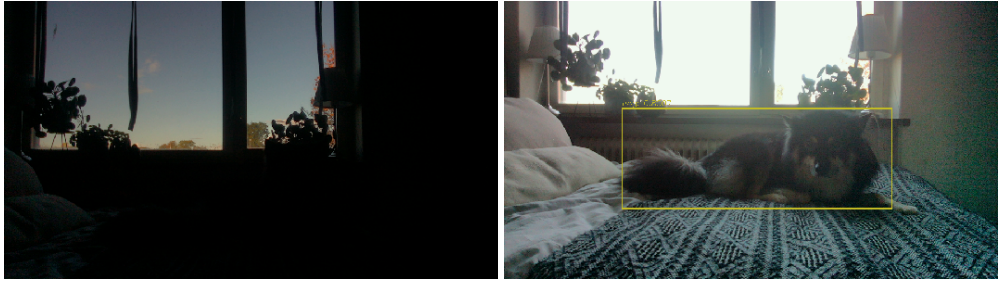


Figure 1: Left: camera’s default parameters, auto-exposure and auto-white-balance enabled. Right: hand-tuned camera parameters.

5 Experiments and Results

In this section we outline our two main experiments. Section 5.1 outlines the normalizing flow model used in our experiments. For our first experiment, in section 5.3, we train and validate a normalizing flow model as an OOD detector using the COCO dataset [11] as our in-distribution. Then, in section 5.4, we take the model from section 5.3 and apply it to a real-world experiment testing our proposed method from section 4.

5.1 Normalizing Flow Model

For our experiments we construct a normalizing flow model from affine coupling layers. We use a model very similar to the one used by Kirichenko *et al.* [9]. Our model consists of a series of 10 affine coupling layers, alternating between transforming the first and second half of the input vector. Each coupling layer consists of two shared linear layers with 512 hidden units followed by two parallel linear layers, one for α and one for β . This is illustrated in Figure 2.

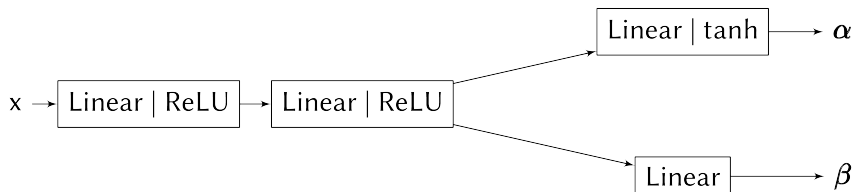


Figure 2: Illustration of coupling layer layout.

We train our model on features extracted from a YOLOv4 [2] object detection network. Training is done for 200 epochs with a batch size of 128, using the Adam [8] optimizer with a learning rate of 10^{-4} .

5.2 Intel RealSense D435

Our experiments use an Intel RealSense D435[1], which is a RGB+Depth camera. In our experiments, however, we only utilise the RGB sensor. Like most commercially available cameras, the D435 exposes a number of parameters that affect the resulting image in various ways. For our experiments we manipulate a selection of these parameters, specifically: backlight compensation, brightness, contrast, exposure, gain, saturation, and sharpness.

5.3 Training Normalizing Flows on a Large Diverse Dataset

In this section, we train a normalizing flow model using the entire COCO training dataset as our in-distribution. We then test the model by comparing the log-likelihood outputs from different input image distributions. Specifically, we divide this section into two smaller experiments:

- One where we compare log-likelihoods of COCO images to randomly generated images.
- One where we capture many images of a static scene with randomized camera parameters, and compare their log-likelihood to COCO images.

For our first experiment we run the trained model on all images in the COCO training, validation and testing datasets, and record the resulting log-likelihood scores. In order to show that the model can distinguish images that are *definitely not* part of the in-distribution, we also record log-likelihood scores from randomly generated images. These random images were generated according to:

$$\begin{aligned}\sigma &\sim \text{Uniform}(1, 256) \\ v_{i,j} &\sim \mathcal{N}(0, \sigma) \\ \text{pixel}_{i,j} &= \min(255, |v_{i,j}|), \quad \forall i, j.\end{aligned}$$

Our primary motivation behind using this formula is to generate images with varying pixel ranges, while also generating some mostly-black and mostly-white images. By varying σ per image, we vary the pixel range in different images. When σ is small, we get mostly-black images, and when σ is large we get mostly-white images due to the min operation. Figure 3 shows histograms of randomly generated images, along with the COCO training, validation, and testing datasets.

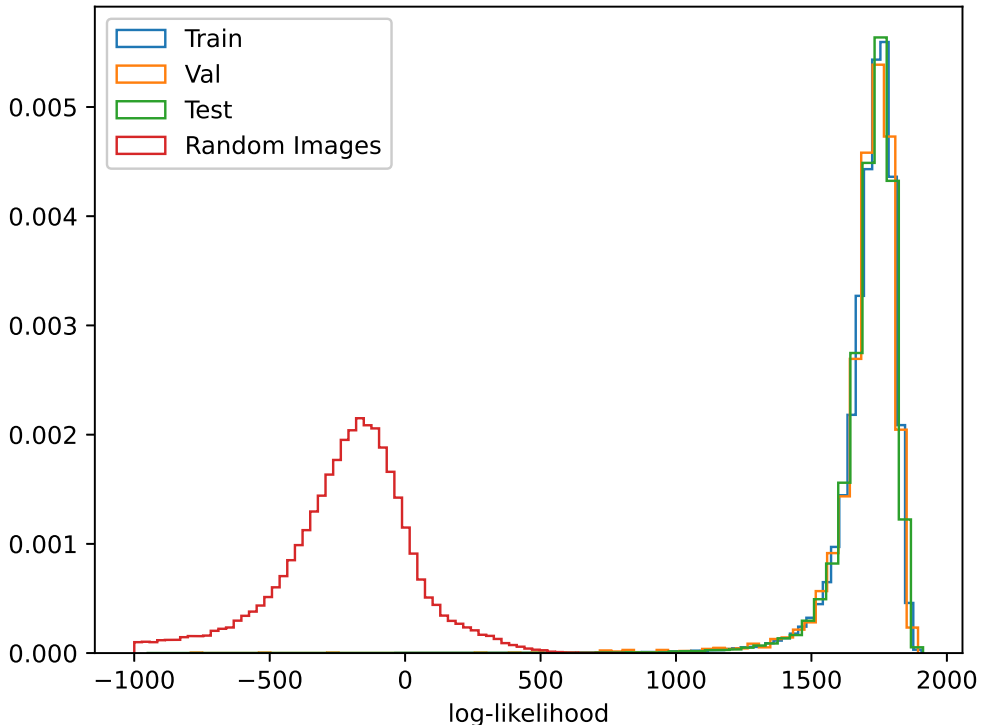


Figure 3: Normalized histogram plot of log-likelihoods from COCO training, validation and test datasets, as well as random images. The histogram has been cut at -1000 log-likelihood for improved readability.

For our second experiment we explore whether our normalizing flow can distinguish between different real images. We place a stationary D435 camera in our lab, and capture a large number of images. For each image, all camera parameters are randomized according to a uniform distribution. In order to be certain that all variance in log-likelihood scores are caused by changing camera parameters, we take precautions to minimize the amount of natural light entering our lab. Then, we capture 10000 images with the ceiling lamp on, and 10000 with the ceiling lamp off. The number 10000 was chosen arbitrarily. Figure 4 shows the corresponding histograms of log-likelihood scores, along with the COCO training data for comparison.

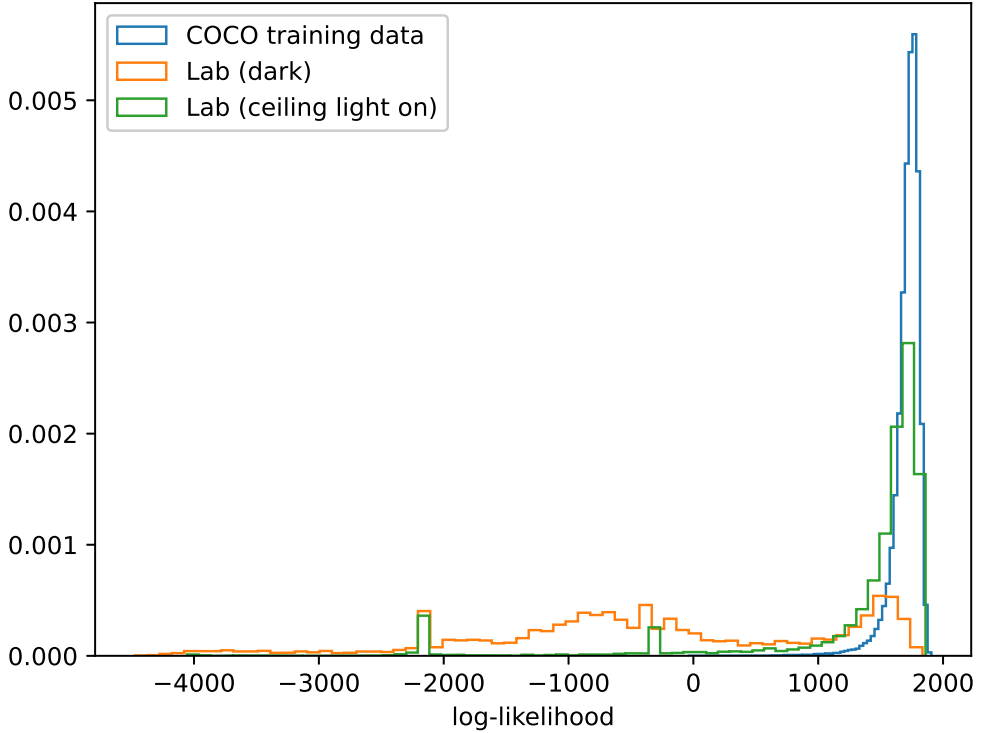


Figure 4: Normalized histogram plot of log-likelihood values. Blue: COCO training images. Orange: 10000 images in our lab with randomized camera parameters, without any lights on. Green: 10000 images in our lab with randomized camera parameters, with ceiling lights on.

5.4 Parameter Optimization

Here, we present results from a small-scale experiment conducted in one of our robot labs. In order to make the experiment as realistic as possible, the experiment was set up using one of our robots. Our robot has an industrial arm with an Intel Realsense D435 camera attached at the gripper.

For this experiment, we set up an optimization procedure using camera parameters as our input, and the log-likelihood from our normalizing flow model as output, as illustrated in Figure 5. More formally, let θ denote the camera parameters, and $C(\theta)$ denote the camera (a function that takes camera parameters and produces an image $\mathbf{x}$), then we solve:

$$\begin{aligned} & \arg \max_{\theta} \log p_x(C(\theta)) \\ &= \arg \max_{\theta} \log p_u(T^{-1}(C(\theta))) + \log |\det J_{T^{-1}}(C(\theta))|. \end{aligned}$$

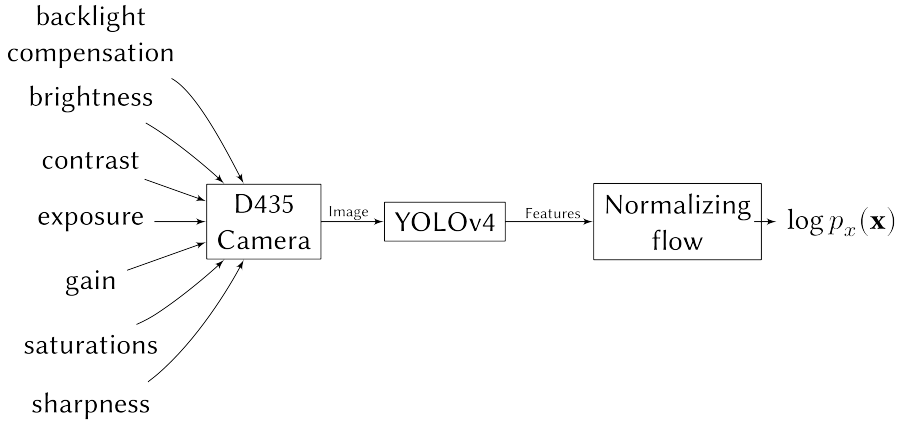


Figure 5: Parameter optimization setup.

For the optimization procedure, we used a very simple evolutionary optimization with elitism selection, a population of 50, and a mutation rate of 20%. No crossover was used. Optimization was carried out for a total of 200 evaluations. Population size, mutation rate, and number of evaluations were chosen based on prior experience with evolutionary optimization. A small number of trial runs confirmed that these values work well.

In our lab, we set up 13 different scenarios with varying COCO objects, viewing angles, and lighting conditions. We took care to make the scenarios as challenging as possible by introducing difficult lighting conditions, reflective surfaces, and background clutter. With the robot stationary, optimization was performed as described above. During optimization, the images with the highest and lowest log-likelihood scores were saved along with their corresponding parameters, and compared to images captured with the camera’s default parameter settings. Henceforth, these will be referred to as *best*, *worst* and *default* parameters, respectively.

After the optimization procedure, all images were annotated with bounding boxes for the COCO objects present. They were then fed through YOLOv4. Mean-Average-Precision (mAP), mean-Average-Recall (mAR), and F1 scores were computed and compared between the worst, default, and best camera parameters.

Table 1 shows average mAP, mAR, and F1 scores for best, default and worst camera parameters. In table 2 we show statistics for the improvement in mAP, mAR, and F1 scores of the best parameters compared to the default parameters. Similarly, Table 3 shows statistics for the improvement of the best compared to the worst parameters.

Figure 6 shows box plots of the improvements in mAP, mAR, and F1 scores for the best parameters compared with the default and worst parameters respectively.

Table 1: Average mAP, mAR, and F1 score for best, default, and worst parameters.

	mAP	mAR	F1
Best	0.7977	0.3283	0.4605
Default	0.7591	0.2977	0.4219
Worst	0.1731	0.0431	0.0687

Table 2: mAP, mAR, and F1 improvement. Best compared to default parameters.

	Min	Max	Average
mAP	-0.2889	+0.5778	+0.0386
mAR	-0.1083	+0.25	+0.0306
F1	-0.1056	+0.349	+0.0386

Table 3: mAP, mAR, and F1 improvement. Best compared to worst parameters.

	Min	Max	Average
mAP	-0.104	+0.95	+0.6247
mAR	+0.1375	+0.4278	+0.2852
F1	+0.1794	+0.5703	+0.3918

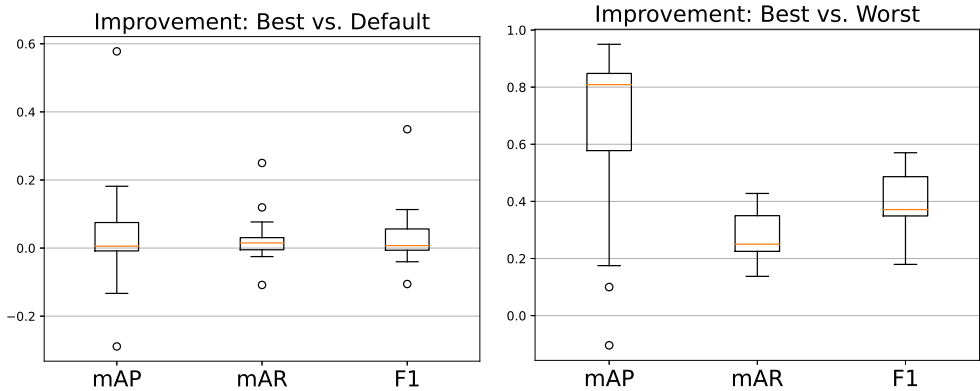


Figure 6: Boxplot of mAP, mAR, and F1 improvement. Left: Best compared to default parameters. Right: Best compared to worst parameters.

In Figures 7, 8, and 9 we show example scenarios where optimization led to varying degrees of success. Each figure includes images along with the respective F1 scores from YOLOv4. Figure 7 shows examples where parameter optimization resulted in better object detection than the default parameters. Figure 8 shows examples where optimization resulted in no significant difference in object detection. Finally, Figure 9 shows the only scenario where optimization resulted in significantly worse object detection.

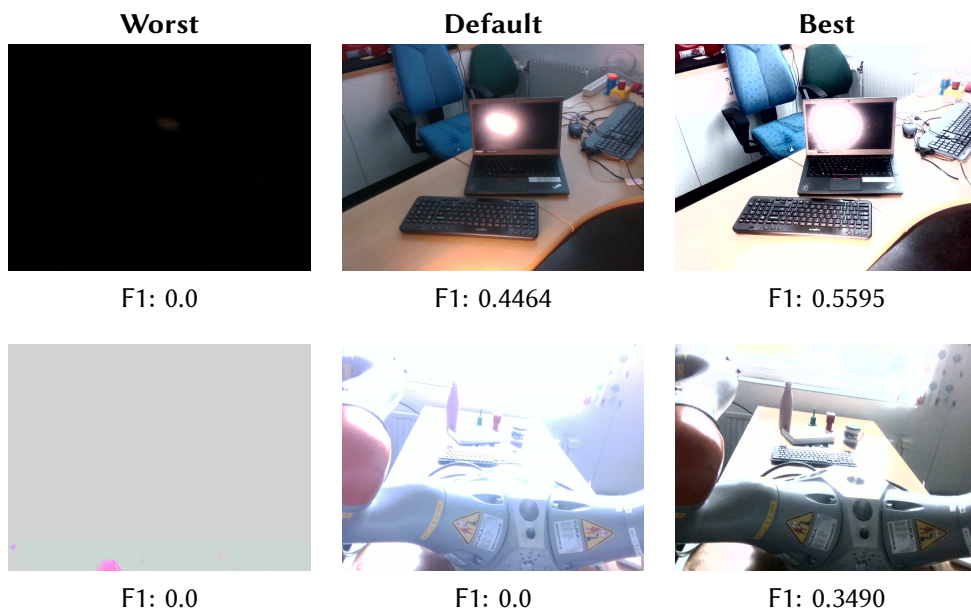


Figure 7: Examples where parameter optimization resulted in an improvement in object detection. Left: lowest log-likelihood. Center: default parameters. Right: highest log-likelihood.

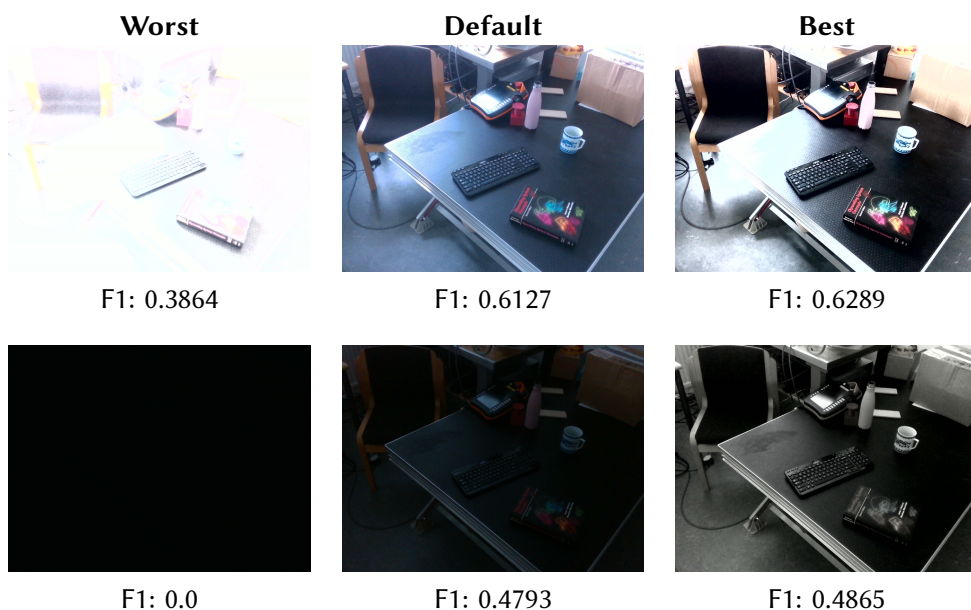


Figure 8: Examples where parameter optimization resulted in no significant difference in object detection. Left: lowest log-likelihood. Center: default parameters. Right: highest log-likelihood.



Figure 9: Example where parameter optimization resulted in worse performance in object detection. Left: lowest log-likelihood. Center: default parameters. Right: highest log-likelihood.

6 Discussion

6.1 Training on the COCO Dataset

When training on the full COCO dataset we see two important indicators that the model is learning properly. First, as seen in Figure 3, the output log-likelihoods span a wide range of values. It would be a red flag if all log-likelihoods were too similar, indicating that the learned probability distribution might be flat. Second, we observe that the validation and test datasets are distributed the same way as the training set, while the random images are given distinctly lower log-likelihood scores.

In figure 4, we see that the model is able to distinguish between real images from different distributions. As such, we have found no evidence to suggest that the size and diversity of the COCO dataset should pose any problems for OOD detection. This in turn suggests that normalizing flow models can be readily applied to diverse real-world scenarios.

Additionally, figure 4 shows that varying parameters in the D435 over the same scene can result in a wide range of log-likelihood values. This gives merit to the idea that camera parameters form a good basis for adapting to various imaging conditions.

6.2 Parameter Optimization Experiment

Table 1 tells us at a glance that our optimized camera parameters achieved the best average scores across the board. Looking at Table 2, we see an average improvement of 3 to 4 percentage points in terms of all metric scores. From the box plot in Figure 6, we see that most scenarios yield only a small difference in object detection compared to the default parameters. However, most importantly we see that the trend is a positive improvement. We also observe that the positive outliers are significantly larger than the negative outliers, again suggesting a positive trend.

Perhaps unsurprisingly, Table 3 shows a massive improvement (28 to 62 percentage points on average) between the best and worst parameters found. The box plot in Figure 6 further highlights this improvement.

In Figures 7, 8, and 9 we note that some of the parameter-optimized pictures look better to the human eye, while others look worse despite resulting in better object detection. This highlights the important fact that there is no equivalence between images that look good to the human eye, and images that are good for object detection.

Perhaps the most important result of this experiment is not the fact that parameter-optimization can lead to better object detection, but instead that it seems feasible to adapt and recover from a very poor visual scenario. Extrapolating from the fact that different camera parameters could result in images where YOLO performs well, and where it fails almost completely on the same scenario, it seems feasible that these parameters have the necessary capacity to adapt to a wide range of difficult visual conditions. This is given further merit by the fact that the varying camera parameters resulted in widely varying log-likelihood values in figure 4. As such, when the normalizing flow model detects something with extremely low log-likelihood value, it is most likely possible to adapt camera parameters to get an image more suitable for object detection. While these experiments have only been concerned with object detection, we expect that this behaviour generalises to other vision tasks as well.

7 Conclusion

In this paper we have shown that it is possible to train a normalizing flow model for OOD detection on a large and diverse dataset such as COCO. We have also conducted a real-world experiment which shows that a normalizing flow OOD detector can be used as a quality metric for a vision task. Optimizing camera parameters with respect to the output from our normalizing flow OOD model yielded, on average, a 3 to 4 percent unit improvement in the reliability of YOLOv4 in terms of object detection performance.

8 Future Work

This paper lays the groundwork for a larger framework of quality metrics for vision task. In order to further expand this framework, a larger benchmark dataset is beneficial. It is also an interesting research direction to explore whether image improvements can be *learned* based on this OOD metric, rather than going through the somewhat slow optimization process used in our experiments.

Acknowledgements

This research is funded by ELLIIT – Excellence Center at Linköping - Lund in Information Technology. Computations for this publication were enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

Bibliography

- [1] Intel realsense depth camera d435, <https://www.intelrealsense.com/depth-camera-d435/>
- [2] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection (2020). <https://doi.org/10.48550/ARXIV.2004.10934>, <https://arxiv.org/abs/2004.10934>
- [3] Bogachev, V.I., Ruas, M.A.S.: Measure theory, vol. 1. Springer (2007)
- [4] Deecke, L., Vandermeulen, R., Ruff, L., Mandt, S., Kloft, M.: Image anomaly detection with generative adversarial networks. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 3–17. Springer International Publishing, Cham (2019)
- [5] Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. (2016), <https://arxiv.org/abs/1605.08803>
- [6] Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., Zhu, X.X.: A survey of uncertainty in deep neural networks (2021). <https://doi.org/10.48550/ARXIV.2107.03342>, <https://arxiv.org/abs/2107.03342>
- [7] Hsu, Y.C., Shen, Y., Jin, H., Kira, Z.: Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
- [8] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014), <https://arxiv.org/abs/1412.6980>
- [9] Kirichenko, P., Izmailov, P., Wilson, A.G.: Why normalizing flows fail to detect out-of-distribution data. Advances in neural information processing systems **33**, 20578–20589 (2020)

- [10] Liang, S., Li, Y., Srikant, R.: Principled detection of out-of-distribution examples in neural networks. CoRR **abs/1706.02690** (2017), <http://arxiv.org/abs/1706.02690>
- [11] Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context (2014). <https://doi.org/10.48550/ARXIV.1405.0312>, <https://arxiv.org/abs/1405.0312>
- [12] McAllister, R., Kahn, G., Clune, J., Levine, S.: Robustness to out-of-distribution inputs via task-aware generative uncertainty. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 2083–2089 (2019). <https://doi.org/10.1109/ICRA.2019.8793552>
- [13] Mohseni, S., Pitale, M., Yadawa, J., Wang, Z.: Self-supervised learning for generalizable out-of-distribution detection. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 5216–5223 (Apr 2020). <https://doi.org/10.1609/aaai.v34i04.5966>, <https://ojs.aaai.org/index.php/AAAI/article/view/5966>
- [14] Papamakarios, G., Nalisnick, E.T., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. **22**(57), 1–64 (2021)
- [15] Ren, J., Liu, P.J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., Lakshminarayanan, B.: Likelihood ratios for out-of-distribution detection. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), <https://proceedings.neurips.cc/paper/2019/file/1e79596878b2320cac26dd792a6c51c9-Paper.pdf>
- [16] Rudin, W.: Real and Complex Analysis. Tata McGraw-Hill Education (2006)
- [17] Wellhausen, L., Ranftl, R., Hutter, M.: Safe robot navigation via multi-modal anomaly detection. IEEE Robotics and Automation Letters **5**(2), 1326–1333 (2020). <https://doi.org/10.1109/LRA.2020.2967706>
- [18] Winkens, J., Bunel, R., Roy, A.G., Stanforth, R., Natarajan, V., Ledsam, J.R., MacWilliams, P., Kohli, P., Karthikesalingam, A., Kohl, S., Cemgil, A.T., Es-lami, S.M.A., Ronneberger, O.: Contrastive training for improved out-of-distribution detection. CoRR **abs/2007.05566** (2020), <https://arxiv.org/abs/2007.05566>

- [19] Yang, J., Zhou, K., Li, Y., Liu, Z.: Generalized out-of-distribution detection: A survey (2021). <https://doi.org/10.48550/ARXIV.2110.11334>, <https://arxiv.org/abs/2110.11334>
- [20] Yuhas, M., Feng, Y., Ng, D.J.X., Rahiminasab, Z., Easwaran, A.: Embedded out-of-distribution detection on an autonomous robot platform. In: Proceedings of the Workshop on Design Automation for CPS and IoT. p. 13–18. Destion '21, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3445034.3460509>, <https://doi-org.ludwig.lub.lu.se/10.1145/3445034.3460509>

Paper II



Making the Flow Glow – Robot Perception under Severe Lighting Conditions using Normalizing Flow Gradients

Simon Kristoffersson Lind

Lund University LTH

`simon.kristoffersson_lind@cs.lth.se`

Rudolph Triebel

German Aerospace Center and

Karlsruhe Institute of Technology

`rudolph.triebel@dlr.de`

Volker Krüger

Lund University LTH

`volker.krueger@cs.lth.se`

Abstract

Modern robotic perception is highly dependent on neural networks. It is well known that neural network-based perception can be unreliable in real-world deployment, especially in difficult imaging conditions. Out-of-distribution detection is commonly proposed as a solution for ensuring reliability in real-world deployment. Previous work has shown that normalizing flow models can be used for out-of-distribution detection to improve reliability of robotic perception tasks. Specifically, camera parameters can be optimized with respect to the likelihood output from a normalizing flow, which allows a perception system to adapt to difficult vision scenarios. With this work we propose to use the absolute gradient values from a normalizing flow, which allows the perception system to optimize local regions rather than the whole image. By setting up a table top picking experiment with exceptionally difficult lighting conditions, we show that our method achieves a 60% higher success rate for an object detection task compared to previous methods.

1 Introduction

In robotics, neural networks have emerged as an indispensable tool for perception and computer vision tasks. However, they operate under the assumption that all inputs come from the same distribution as their training data. In other words, when confronted with unfamiliar inputs, their outputs can be unpredictable [1]. This unpredictability poses a significant risk for robots operating in dynamic real-world settings. Moreover, the accuracy of computer vision hinges on the camera’s ability to produce clear images for the neural network. A case in point is the 2015 Tesla crash, where the autopilot system failed to detect a truck’s side against a bright sky [2].

One common way to prevent unpredictable behaviour is out-of-distribution (OOD) detection [1]. Most OOD methods work by directly or indirectly approximating the probability density function (PDF) of the training data. The approximate PDF is then used as a score function to determine whether an input is in- or out-of-distribution. Using such a score, an autonomous system can react by performing a safety behaviour whenever an input is deemed too far from the training distribution. Most commonly, this safety behaviour simply discards any OOD input while waiting for better inputs.

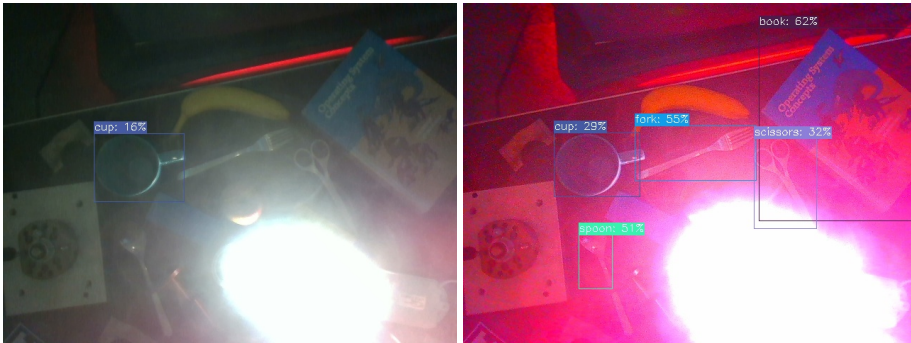


Figure 1: In the leftmost image, taken with default camera parameters and auto-exposure, only the cup can be detected by YOLOv4. In the rightmost image, where camera parameters have been optimized using our proposed method, the cup, spoon, fork, scissors, and book are successfully detected.

While discarding inputs may be feasible for many applications, it can be problematic for many autonomous systems. Consider an autonomous car that encounters OOD data. Should it keep driving while discarding inputs, and risk hitting someone? Or should it stop while waiting for better inputs, and risk an accident by blocking traffic? We argue that it is necessary for autonomous systems to detect and adapt to OOD data.

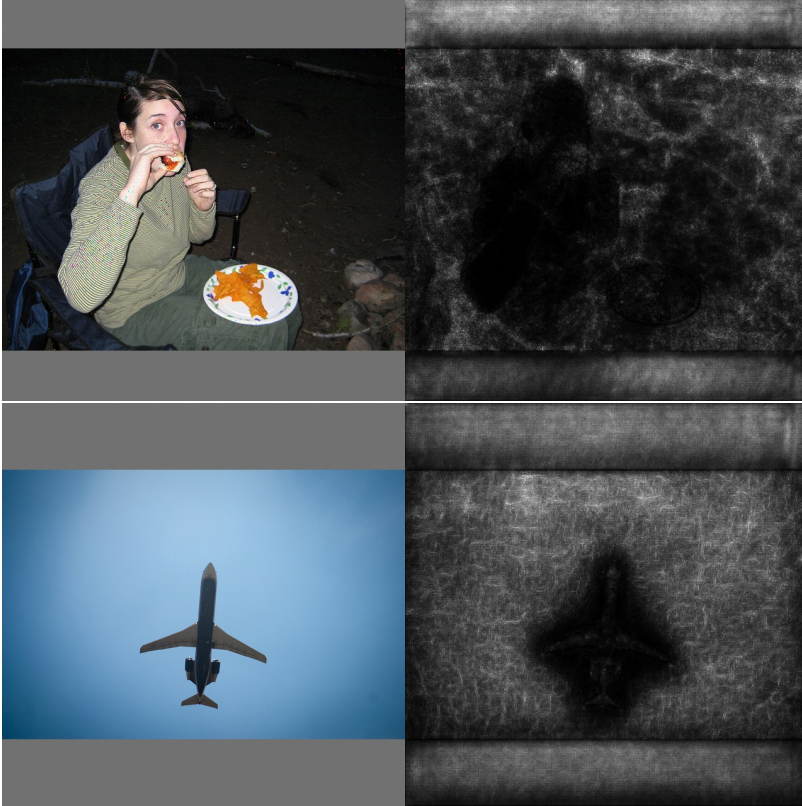


Figure 2: Left: Images from the COCO validation set.
right: Absolute gradient images from YOLOv4 + Normalizing Flow.

In our earlier work [3], we used a Normalizing Flow (NF) as an OOD detector, and we created an adaptable vision system by allowing our robot to change parameters in its camera on-the-fly. We chose to implement this behaviour as an optimization procedure, where the parameters *backlight compensation*, *brightness*, *exposure*, *gain*, *saturation*, and *sharpness* were optimized with respect to the OOD score from our NF. Effectively, our robot learns the best camera parameters for any given scene, by minimizing the OOD score. We were able to show that this type of adaptive behaviour improves the performance of a YOLOv4 object detector in difficult visual scenarios.

A major weakness of our earlier method is the fact that the NF evaluates an entire image as either in- or out-of-distribution. Consider the example in Fig. 1. Due to the bright light, it is not possible to find any single camera parameter setting that makes the entire image visible. Despite this, the objects around the bright light can be made almost perfectly visible by changing camera parameters. Therefore, we see a need to express OOD detection on regions rather than the full image.

When inspecting the pixel-wise gradient with regard to the OOD score from our NF model, it seems that OOD regions correspond to large absolute gradient values, and in-distribution regions correspond to small absolute gradient values. This effect can be seen in Fig. 2.

In this work, we propose to use the absolute pixel-wise NF gradient values as a local OOD score. We conduct two experiments to provide empirical evidence that the absolute NF gradient values carry OOD information at the pixel-level. Finally, we show that our proposed absolute NF gradient outperforms our previous method [3] on a real-world robotic picking experiment in a difficult visual scenario. To summarize, we make the following contributions:

- We propose to use the absolute gradient from an NF to provide pixel-level OOD information.
- We empirically show the efficacy of our proposed NF gradient for OOD detection.
- We show, across two common object detection architectures (YOLOv4[4], Faster-RCNN[5]) that object detectors tend to perform better in regions with small absolute NF gradient values.
- We experimentally establish the superior performance of our method over [3] in adapting camera parameters for challenging imaging scenarios.

2 Related Work

A substantial body of research focuses on OOD detection, such as [6, 7, 8, 9, 10, 11]. However, these studies predominantly validate their findings using benchmarks. The benchmarks employed generally fail to capture the diverse scenarios a robot might face in real-world operations. This limitation arises primarily because these works ensure that in-distribution and out-of-distribution inputs come from distinct sets.

A more niche segment of research delves into the application of OOD detection in robotics, for example [12, 13]. These studies, while applying OOD detection to robotics, typically resort to aborting operations upon encountering OOD data.

Another research avenue explores OOD techniques for visual anomaly detection, with studies like [14, 15, 16]. These investigations typically employ autoencoder methods to identify OOD elements within images, such as in medical imaging contexts. However, the OOD strategies in these studies are usually tailored to their specific applications, limiting their broader applicability.

Lastly, [3] leverages an NF to adapt to challenging imaging situations by fine-tuning camera parameters. Our contribution enhances this by using the NF’s gradient to incorporating pixel-level OOD information.

3 Background

Given a training distribution $p_{\text{train}}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^D$, the goal of OOD detection is to identify whether or not a new sample $\hat{\mathbf{x}}$ comes from $p_{\text{train}}(\mathbf{x})$. We assume that $p_{\text{train}}(\mathbf{x})$ is unknown or intractable to evaluate – otherwise OOD detection would be trivial. NFs enable OOD detection by approximating p_{train} [17].

Formally, an NF is a *diffeomorphism*, an invertible transformation $T(\mathbf{x})$ where both $T(\mathbf{x})$ and $T^{-1}(\mathbf{x})$ are differentiable [18]. In training an NF, the goal is to learn this diffeomorphism such that $\mathbf{u} = T(\mathbf{x})$, $\mathbf{u} \in \mathbb{R}^D$, follows a different, tractable distribution $p_u(\mathbf{u})$. The distribution $p_u(\mathbf{u})$ is typically chosen to be a unit Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{1})$, since a Gaussian is easy to evaluate and sample. T being a diffeomorphism allows $p_{\text{train}}(\mathbf{x})$ to be approximated using the following formula [18]:

$$\begin{aligned} p_{\text{train}}(\mathbf{x}) &= p_u(\mathbf{u}) |\det J_{T^{-1}}(\mathbf{u})|^{-1} \\ &= p_u(T(\mathbf{x})) |\det J_T(\mathbf{x})|^{-1}, \end{aligned}$$

where J_T denotes the Jacobian of T . As such, the NF allows us to approximate p_{train} by transforming it into a tractable distribution. In the standard OOD detection setting, an OOD threshold is set with respect to $p_{\text{train}}(\mathbf{x})$, typically based on a fixed false positive rate [19].

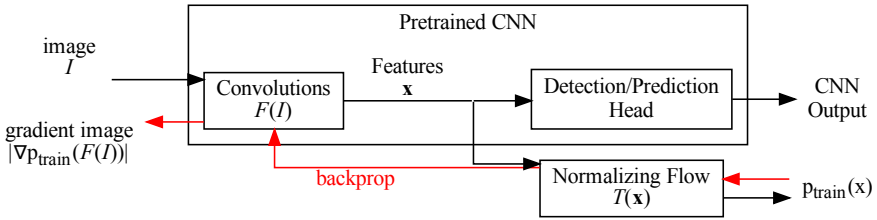


Figure 3: Illustration of how we apply NFs alongside a pretrained CNN for vision tasks.

The primary difficulty in constructing an NF is ensuring invertibility, along with a tractable $|\det J_T(\mathbf{x})|$. One common way to solve these problems is *affine coupling layers*, used in for example RealNVP[20] and GLOW[21]. Coupling layers divide their input $\mathbf{x}$ into two non-overlapping parts $\mathbf{x}_1, \mathbf{x}_2$. An affine transformation is then performed on one part, while leaving the other unchanged:

$$\mathbf{x}'_1 = \boldsymbol{\alpha} \cdot \mathbf{x}_1 + \boldsymbol{\beta} ,$$

where $\boldsymbol{\alpha} = \exp(F_1(\mathbf{x}_2))$, $\boldsymbol{\beta} = F_2(\mathbf{x}_2)$.

Here we use $\cdot$ to denote element-wise multiplication. F_1 and F_2 denote arbitrary functions, for example neural networks. The output of the coupling layer is then computed by simply reconstructing $\mathbf{x}$ using the new $\mathbf{x}'_1$ and the unchanged $\mathbf{x}_2$. Commonly, $\mathbf{x}_1$ and $\mathbf{x}_2$ are constructed by simply splitting $\mathbf{x}$ in half, in which case the output is constructed by simply concatenating: $\mathbf{x}' = [\mathbf{x}'_1 | \mathbf{x}_2]$ [18]. In this setting, the layers typically operate on alternating halves of $\mathbf{x}$, otherwise one half would be left permanently unchanged.

Kirichenko *et al.* [17] observe that applying NFs directly to images tends to work poorly for OOD detection, They show that NFs seem to learn pixel-to-pixel correspondences, rather than extracting any semantic information about the image distribution. As a solution, they propose to apply an NF to features extracted from another pretrained neural network.

In our previous work [3], we adopted the approach from [17], and applied it to a robotic perception setting. Instead of using a pretrained network only for feature extraction, we applied an NF alongside a pretrained network for a vision task. This approach allows a single network to perform a vision task, while also being used as a feature extractor for the NF as illustrated in Fig. 3. In turn, this allows the NF to provide an OOD score for each image with negligible runtime overhead. Using this NF construction, we applied an evolutionary optimization procedure [22] to minimize $p_{\text{train}}(\mathbf{x})$ by adjusting camera parameters (backlight compensation, brightness, contrast, exposure, gain, saturation, and sharpness). We implemented this optimization procedure in our robot, using YOLOv4[4] as the pretrained CNN. By deploying our robot in a number of difficult visual scenarios, we were able to show an increase in object detection performance using our optimization procedure (average increase of 3.86 percentage units in F1 score).

4 Our Approach

Instead of computing a global OOD score, we extend the NF approach from [3] and [17] by computing gradients to extract pixel-level OOD information. Formally, given an input image

$$I = \begin{bmatrix} i_{11} & i_{12} & \dots & i_{1W} \\ i_{21} & i_{22} & \dots & i_{2W} \\ \vdots & \vdots & \ddots & \vdots \\ i_{H1} & i_{H2} & \dots & i_{HW} \end{bmatrix}, \quad I \in \mathbb{R}^{H \times W},$$

the OOD detection method in [3, 17] computes:

$$\begin{aligned} \mathbf{x} &= F(I), \quad F: \mathbb{R}^{H \times W} \mapsto \mathbb{R}^D \\ p_{\text{train}}(\mathbf{x}) &= p_u(T(\mathbf{x})) |\det J_T(\mathbf{x})|, \end{aligned} \quad (1)$$

where F denotes the feature extraction layers from a pretrained neural network, and T denotes the learned NF.

In order to gain pixel-level OOD information, we propose to use the gradient of Eq. (1). Specifically, we propose to use the absolute value of each element in the gradient. With some notational abuse, we write

$$\begin{aligned} |\nabla p_{\text{train}}(F(I))| &\in \mathbb{R}^{H \times W} \\ |\nabla p_{\text{train}}(F(I))|_{yx} &= \left| \frac{\delta p_{\text{train}}}{\delta i_{yx}} \right|. \end{aligned} \quad (2)$$

In other words, we simply backpropagate through p_{train} and F , as indicated by the red edges in Fig. 3. Here we use 2-dimensional grayscale images purely for notational convenience. Our method extends directly to RGB images by averaging across the channel dimension.

Through initial testing we have observed that image regions that closely match the distribution p_{train} tend to have small gradient values compared to regions that are OOD. Intuitively, we reason that in-distribution data will likely lie close to a maximum of $p_{\text{train}}(\mathbf{x})$. Due to the continuity and differentiability of $p_{\text{train}}(\mathbf{x})$, these data points should exhibit small gradient values. This intuition also serves as our rationale for using the absolute value of gradients, as the direction of change is inconsequential.

Just as before, our goal is to construct an optimization procedure over camera parameters, in order to allow our robot to adaptively overcome difficult visual scenarios. In our robotics work, we mostly perform object detection tasks, for example object

picking. All object detection tasks involve some type of region proposal, whether it be bounding-boxes or image segmentations. For such tasks, we propose to optimize:

$$\min_{\theta} \frac{1}{|\text{ROI}|} \sum_{x,y \in \text{ROI}} |\nabla p_{\text{train}}(F(I(\theta)))|_{xy} ,$$

where ROI denotes the region of interest proposed by the object detector, and θ denotes camera parameters. We use $|\text{ROI}|$ to represent the number of pixels inside the region. Here we also write $I(\theta)$ to highlight the dependence of the image on the camera parameters. When the object detector fails to propose a suitable region of interest, we fall back to computing the average over the entire absolute NF gradient image $|\nabla p_{\text{train}}(F(I(\theta)))|$. We expect this fallback behaviour to perform similarly to optimizing p_{train} , as in [3]. Additionally, for multi-object detectors, we apply the sum over the union of all proposed regions:

$$\text{ROI} = \bigcup_{\text{ROI}_i} \text{ROI}_i .$$

For other types of visual tasks, it may be necessary to construct other ways to select suitable regions to optimize over.

5 Experiments and Results

In this section, we begin by performing two separate experiments to empirically establish evidence supporting our proposed optimization procedure.

First, we perform a conventional OOD detection experiment using the average absolute NF gradient as an OOD score:

$$\frac{1}{HW} \sum_{y=1}^H \sum_{x=1}^W \left| \frac{\delta p_{\text{train}}}{\delta i_{yx}} \right| . \quad (3)$$

Showing that the absolute NF gradient can be used successfully for OOD detection directly implies that it is useful for constructing an adaptive optimization procedure as in [3]. Additionally, showing success in OOD detection using the average absolute gradient over the entire image supports our decision to use this as a fallback when our object detection fails to propose a region of interest.

Second, we investigate the relationship between object detector performance, and the absolute NF gradient. Specifically, we look at the predicted bounding-boxes from two common object detectors (YOLOv4[4], Faster-RCNN[5]), and we show that

bounding-boxes that are predicted correctly tend to contain smaller absolute NF gradient values compared to incorrectly predicted bounding-boxes. This experiment serves directly to support our decision to optimize camera parameters over the predicted regions of interest from our object detector.

Finally, we conduct a real-world experiment using our robot, where we set up an object picking task in a cluttered scene. We manually construct an exceedingly difficult visual scenario by making our lab as dark as possible, while aiming a bright light at our robot. Our proposed absolute NF gradient optimization procedure is then compared to [3], and simple auto-exposure and auto-whitebalance.

5.1 NF Training

In all of our experiments we use an NF consisting of 10 affine coupling layers, operating on alternating halves of the input. Each coupling layer consists of two shared ReLU-activated linear layers followed by two separate prediction heads, one for α and one for β . Our NF is trained for 200 epochs, using the Adam[23] optimizer with a constant learning rate of 10^{-4} .

5.2 OOD Experiment

We run the same conventional OOD detection experiment as in [17], in order to compare performance of our proposed average absolute NF gradient to that of using p_{train} directly. For feature extraction, we use an EfficientNet [24] model pretrained on ImageNet [25]. We perform OOD detection on the CIFAR-10 [26], SVHN [27], and CelebA [28] datasets. An NF model is trained for each of these datasets, which results in the other two datasets being considered OOD. First, we compute OOD detection performance using $p_{\text{train}}(F(I))$ as an OOD score directly, exactly like [17]. Then, we compute OOD detection performance using the average absolute NF gradient (Eq. 3). We calculate an OOD threshold with a fixed 5% false-positive rate. The resulting true-positive rates are shown in Table 1 and 2 respectively.

Table 1: OOD detection true-positive rates using $p_{\text{train}}(F(I))$.

		OOD		
		Cifar10	SVHN	CelebA
Train	Cifar10	-	0.4148	0.9953
	SVHN	0.999	-	1.0
	CelebA	1.0	1.0	-

Table 2: OOD detection true-positive rates using average absolute NF gradient.

		OOD		
		Cifar10	SVHN	CelebA
Train	Cifar10	-	0.6688	0.9205
	SVHN	0.972	-	0.9999
	CelebA	0.9464	0.9747	-

From Table 2, it is evident that the NF gradient works well for OOD detection. While we do observe a minor decrease in overall OOD detection performance, our NF gradient is not far behind the plain NF approach, and even achieves improved performance on CIFAR10/SVHN. We would especially like to stress that we do not aim to achieve state-of-the-art OOD detection performance, but only highlight the usefulness of accessing this type of information at a pixel-level.

5.3 Correlation with Model Performance

In order to further establish our proposed method for adapting camera parameters, we wish to investigate whether the absolute NF gradient correlates with object detection performance. We perform an experiment using YOLOv4[4] and Faster-RCNN[5], both trained on the COCO[29] dataset. Both of these models output bounding-boxes corresponding to each predicted object. In order to investigate the absolute NF gradient with respect to object detection performance, we measure the average absolute NF gradients inside correctly classified bounding-boxes, and compare it to incorrectly classified bounding-boxes.

If a predicted bounding-box overlaps at least 50% (intersection-over-union) with a ground-truth box, and they both have the same class label, then we classify that predicted bounding-box as correct. Otherwise it is classified as incorrect. Similarly, if the model fails to predict a ground-truth box, we add that ground-truth box to our list of incorrect boxes. These results are shown in Table 3.

Table 3 shows average absolute NF gradients inside bounding-boxes from the COCO validation dataset. Since we operate on the absolute values of the NF gradients, the resulting distributions are not Gaussian, and standard deviations appear meaningless. Therefore, it is difficult to precisely gauge the nature of this correlation. But what is evident is that there is a correlation, as the correct bounding boxes have on average around 20% lower absolute gradient values.

Table 3: Average gradients for COCO object detection.

	Correct bounding-box	Incorrect bounding-box
YOLOv4	0.4447	0.5640
Faster-RCNN	0.7658	0.9510

5.4 Parameter Optimization

In order to show the benefits of our proposed absolute NF gradient for robotic perception, we construct a table-top object picking task with difficult imaging conditions. First we create a cluttered tabletop scene (Fig. 4). Then, we make efforts to block out natural light to make the room as dark as possible, and we shine a strong light facing straight up from the cluttered scene. With all that in place, we position a UR-5 robot arm with a camera at its wrist (Realsense D435 RGB-D camera), such that the camera is looking at our cluttered scene from above. Our experiment then proceeds in an automated fashion as follows:

- The arm is positioned such that the camera is looking down at the scene from a random position on a semi-sphere.
- Images are captured with:
 - The camera’s default image parameters.
 - The camera’s default image parameters, with auto-exposure and auto-whitebalance.
 - Parameters optimized using $p_{\text{train}}(\mathbf{x})$, as in [3].
 - Parameters optimized using our proposed absolute NF gradient image.
- For each image, a YOLOv4 model detects COCO objects in the scene.

Example images captured from one arm position can be seen in Fig. 5. In our scene we include the following COCO objects: apple, banana, book, cup, fork, mouse, scissors, and spoon (Fig. 4). We ensure that all COCO objects are visible to a human under some camera parameter settings. Before starting our experiment, we compute the true position of each COCO object, and convert these positions into the robot’s coordinate frame. For each trial in our experiment, we compare the outputs from YOLOv4 to these positions in order to determine whether an object was correctly classified at the right location.

Given that we use the same camera model as in [3], we choose to optimize the same parameters: backlight compensation, brightness, contrast, exposure, gain, saturation, and sharpness.

For both parameter optimization procedures, we employ an evolutionary optimization algorithm with elitism selection, population size=50, mutation rate=20%, and a total of 200 iterations [22]. All of these hyperparameters are the same as in [3].

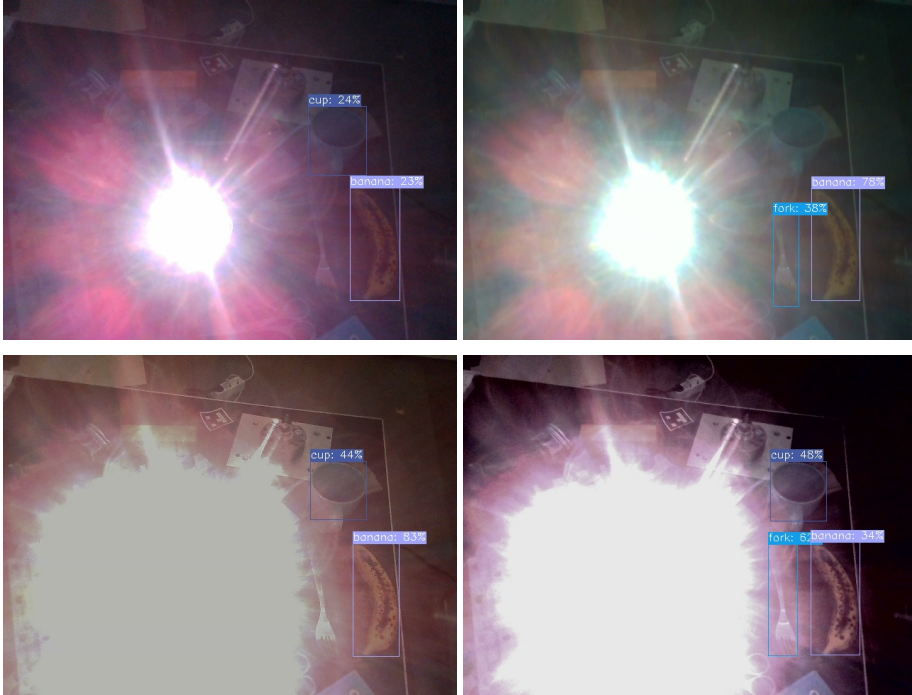


Figure 5: Example images captured during experiments. Top left: default parameters. Top right: Default parameters with auto-exposure and auto-whitebalance. Bottom left: parameters optimized according to [3]. Bottom right: parameters optimized using our NF gradient image.

While we ensured that all objects were visible under some parameter settings, we never found a single parameter setting that made all objects clearly visible. Most likely, this explains why neither method could successfully detect the *apple* and *mouse*. Since they were placed close to the light source, they were typically more difficult to discern. This further highlights the importance of local OOD information compared to a single global OOD score. By computing a local OOD score, our NF gradient method allows the perception pipeline to focus only on the regions that matter, and make sure those are visible.

We would like to note that it is likely possible to further improve the object detection performance. In our experiment, camera parameters were jointly optimized for all detected objects. However, we believe that individual optimization for each object region might yield even better performance.

6 Conclusions and Future Work

In this work, we introduce a technique that employs gradients from a Normalizing Flow (NF) to obtain pixel-level Out-of-Distribution (OOD) information for an image. Experimental results validate the efficacy of our proposed NF gradient for general OOD detection. Moreover, this approach significantly surpasses our prior methods for adapting to challenging imaging scenarios. Consequently, our robotic perception framework demonstrates enhanced object detection capabilities, even in the face of adverse lighting conditions.

We will make our code publicly available at <https://github.com/SimonKLind/NF-Gradients>. Additionally, we will compile the images collected from our experiment into a dataset, which we will also make publicly available for future research.

A notable limitation of our current method is its dependency on the camera to sample parameter configurations during the optimization procedure. The process of acquiring multiple image samples is inherently time-consuming, resulting in a runtime that is slower than desired, which may prohibit its use in some robot applications. To broaden the applicability of our techniques across diverse robotic tasks, we hope to improve the runtime by removing the dependency on the camera for sampling.

Acknowledgments

This research is funded by the Excellence Center at Linköping-Lund in Information Technology (ELLIIT), and the Wallenberg AI, Autonomous Systems and Software Program (WASP). Computations for this publication were enabled by the supercomputing resource Berzelius provided by the National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

Bibliography

- [1] J. Gawlikowski, C. Rovile Njjeutcheu Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu, “A survey of uncertainty in deep neural networks,” *arXiv e-prints*, Jul 2021, aDS Bibcode: 2021arXiv210703342G.
- [2] Tesla, “A tragic loss,” <https://www.tesla.com/blog/tragic-loss>, accessed 2023-09-15.
- [3] S. Kristoffersson Lind, R. Triebel, L. Nardi, and V. Krueger, “Out-of-distribution detection for adaptive computer vision,” in *Image Analysis*, ser. Lecture Notes in Computer Science, R. Gade, M. Felsberg, and J.-K. Kämäräinen, Eds. Cham: Springer Nature Switzerland, 2023, pp. 311–325.
- [4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” Apr 2020, arXiv:2004.10934 [cs, eess].
- [5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” Jan 2016, arXiv:1506.01497 [cs].
- [6] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft, “Image anomaly detection with generative adversarial networks,” in *Machine Learning and Knowledge Discovery in Databases*, M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, Eds. Cham: Springer International Publishing, 2019, pp. 3–17.
- [7] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [8] S. Liang, Y. Li, and R. Srikant, “Principled detection of out-of-distribution examples in neural networks,” *CoRR*, vol. abs/1706.02690, 2017.

- [9] J. Winkens, R. Bunel, A. G. Roy, R. Stanforth, V. Natarajan, J. R. Ledsam, P. MacWilliams, P. Kohli, A. Karthikesalingam, S. Kohl, A. T. Cemgil, S. M. A. Eslami, and O. Ronneberger, “Contrastive training for improved out-of-distribution detection,” *CoRR*, vol. abs/2007.05566, 2020.
- [10] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan, “Likelihood ratios for out-of-distribution detection,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.
- [11] S. Mohseni, M. Pitale, J. Yadawa, and Z. Wang, “Self-supervised learning for generalizable out-of-distribution detection,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5216–5223, April 2020.
- [12] M. Yuhas, Y. Feng, D. J. X. Ng, Z. Rahiminasab, and A. Easwaran, “Embedded out-of-distribution detection on an autonomous robot platform,” in *Proceedings of the Workshop on Design Automation for CPS and IoT*, ser. Destion '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 13–18.
- [13] R. McAllister, G. Kahn, J. Clune, and S. Levine, “Robustness to out-of-distribution inputs via task-aware generative uncertainty,” in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2083–2089.
- [14] Y. Du, “An anomaly detection method using deep convolution neural network for vision image of robot,” *Multimedia Tools and Applications*, vol. 79, no. 13, pp. 9629–9642, Apr 2020.
- [15] D. J. Samuel and F. Cuzzolin, “Unsupervised anomaly detection for a smart autonomous robotic assistant surgeon (saras) using a deep residual autoencoder,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7256–7261, Oct 2021.
- [16] S. Salimpour, J. P. Queralta, and T. Westerlund, “Self-calibrating anomaly and change detection for autonomous inspection robots,” in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, Dec 2022, pp. 207–214.
- [17] P. Kirichenko, P. Izmailov, and A. G. Wilson, “Why Normalizing Flows Fail to Detect Out-of-Distribution Data,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 20 578–20 589.
- [18] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing Flows for Probabilistic Modeling and Inference,” *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021.

- [19] J. Yang, K. Zhou, Y. Li, and Z. Liu, “Generalized out-of-distribution detection: A survey,” Aug 2022, arXiv:2110.11334 [cs].
- [20] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” Feb 2017, arXiv:1605.08803 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1605.08803>
- [21] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” Jul 2018, arXiv:1807.03039 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1807.03039>
- [22] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*. Springer, 2017.
- [23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [24] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” Sep 2020, arXiv:1905.11946 [cs, stat].
- [25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [26] A. Krizhevsky, G. Hinton, *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [27] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.
- [28] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
- [29] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, “Microsoft coco: Common objects in context,” Feb 2015, arXiv:1405.0312 [cs].

Paper III



Uncertainty Quantification Metrics for Deep regression

Simon Kristoffersson Lind*

Lund University LTH

`simon.kristoffersson_lind@cs.lth.se`

Ziliang Xiong*

Linköping University

`ziliang.xiong@liu.se`

Per-Erik Forssén

Linköping University

`per-erik.forssen@liu.se`

Volker Krüger

Lund University LTH

`volker.krueger@cs.lth.se`

*: Equal contribution

Abstract

When deploying deep neural networks on robots or other physical systems, the learned model should reliably quantify predictive uncertainty. A reliable uncertainty allows downstream modules to reason about the safety of its actions. In this work, we address metrics for uncertainty quantification. Specifically, we focus on regression tasks, and investigate Area Under Sparsification Error (AUSE), Calibration Error (CE), Spearman’s Rank Correlation, and Negative Log-Likelihood (NLL). Using multiple datasets, we look into how those metrics behave under four typical types of uncertainty, their stability regarding the size of the test set, and reveal their strengths and weaknesses. Our results indicate that Calibration Error is the most stable and interpretable metric, but AUSE and NLL also have their respective use cases. We discourage the usage of Spearman’s Rank Correlation for evaluating uncertainties and recommend replacing it with AUSE.

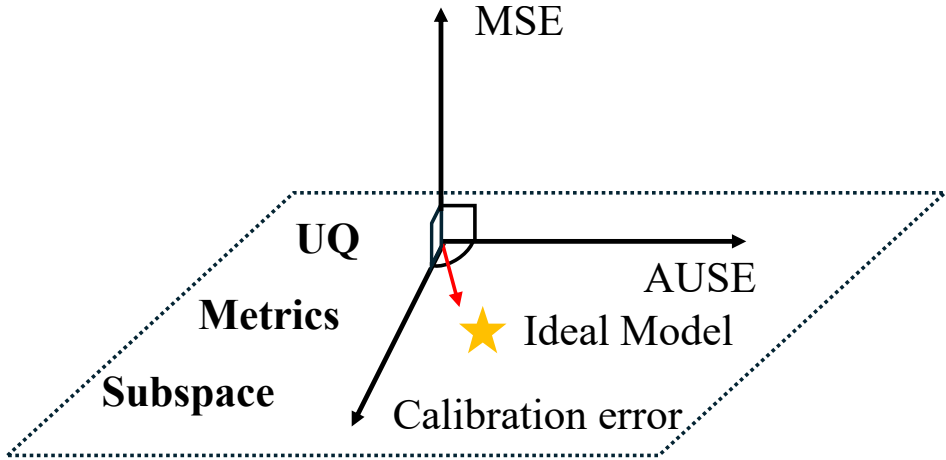


Figure 1: An illustration for UQ metrics and regression metrics. Note: the axes of CE and AUSE are distinct, but not orthogonal.

1 Introduction

In recent years, there has been a rapid advance in the adoption of neural network-based methods in many real-world applications, for example robotics. Following this adoption, increasing scrutiny has been directed towards neural network-based methods for their lack of reliability and interpretability. While neural networks have achieved impressive performance for many different tasks, the fact remains that they can be unreliable in real-world deployment (Grimmett et al., 2016). Additionally, their lack of interpretability makes it difficult to know how and when they may perform unreliably. For these reasons, increasing attention has been directed at the uncertainty output from neural networks, and the importance of introspective qualities (Grimmett et al., 2016). Arguably the most important introspective quality is a reliable uncertainty estimate.

Despite increasing attention being directed toward *uncertainty quantification* (UQ), much of this work focuses on uncertainty in classification tasks where metrics for assessment are well understood (Ovadia et al., 2019). Many real-world applications, however, rely instead on regression, and there is a lack of common understanding surrounding the available metrics for regression. In this work, we have identified four metrics that are commonly used to measure various qualities in a predicted uncertainty in regression. Specifically, we investigate the *Area Under Sparsification Error* (AUSE) (Ilg et al., 2018), *Spearman’s Rank Correlation* (Spearman, 1904), *Negative Log Likelihood* (NLL) (Lakshminarayanan et al., 2017), and *Calibration Error* (CE) (Pakdaman Naeini et al., 2015). These UQ metrics measure different aspects of uncertainty

that are all orthogonal to the regression task performance (typically *mean squared error* (MSE)) as indicated in Fig. 1.

With the help of synthetic datasets, we explore the usage of these metrics with the aim of better understanding their differences, strengths and limitations.

Our contributions are as follows:

- We create simple synthetic datasets that highlight different types of uncertainty, and use these to explore how the metrics behave, compared to the generating data distribution.
- We compare each metric in terms of their stability to different dataset sizes.
- We reason about the strengths and limitations of each metric.
- Finally, we experiment on a real stereo disparity task, where we add synthetic noise to the input, to further illustrate the behaviour of the tested measures.

We also provide a mathematical formulation for the AUSE metric. To the best of our knowledge, AUSE has previously only been described informally, using natural language.

2 Related Work

Historically, methods that deal with any type of uncertainty in regression have been formulated using parameterized distributions. As a classic example, consider Kalman filters (Kalman, 1960), which are built on a parameterized distribution with well-defined uncertainty. Modern machine learning methods are built upon a solid statistical foundation and most methods output some form of uncertainty (Bishop, 2006). However, it has been observed that there is a discrepancy between the predicted uncertainty from modern neural networks, and the observed empirical accuracy (Guo et al., 2017). Additionally, lacking interpretability in these uncertainties have led to the development of methods to formulate more statistically grounded uncertainties (Gawlikowski et al., 2023).

Along with the research on uncertainty estimation, a number of methods have been developed to assess how trustworthy a predicted uncertainty is. Perhaps the most commonly used (for example in Guo et al. (2017); Kuleshov et al. (2018)) is the *expected calibration error* (ECE) (Pakdaman Naeini et al., 2015). While not as common as ECE, NLL (*negative log likelihood*) is also used, for example in Heiss et al. (2022); Loquercio et al. (2020). Spearman’s rank correlation coefficient (Spearman, 1904) has also found new use in this area (for example in Tan et al. (2023); Ng et al. (2023)). Finally, the most

recent method we examine is AUSE (Ilg et al., 2018), used in for example Eldesokey et al. (2020); Xiong et al. (2024). While other uncertainty assessment methods exist, we have chosen to focus on these four as they are most commonly used.

3 Theory

First, we introduce different types of uncertainty in Sec. 3.1. Then, in Sec. 3.2, we introduce the four common uncertainty evaluation metrics to be analyzed.

3.1 Different types of Uncertainty

In the realm of predictive uncertainty estimation, we encounter two fundamental types of uncertainty (Kendall & Gal, 2017). The first, known as *aleatoric uncertainty*, encompasses the intrinsic noise and ambiguity present in observations. This noise, stemming from sources like sensor or motion irregularities, persists even with an increase in data collection and cannot be mitigated. The second type, *epistemic uncertainty* constitutes the uncertainty surrounding model parameters, reflecting our lack of knowledge about the precise model generating the observed data. This form of uncertainty tends to diminish as more data is acquired, and is thus often termed *model uncertainty*. The most common cause of epistemic uncertainty is *out-of-distribution* (OOD) data. In other words, data that comes from a distribution that is different from the training data. For example, autonomous driving models that are trained on synthetic data usually face domain gap to real-world data.

Aleatoric uncertainty can be subdivided into *homoscedastic uncertainty*, which remains constant across various inputs, and *heteroscedastic uncertainty*, which varies depending on the model inputs. Heteroscedastic uncertainty plays a crucial role in computer vision tasks. For example, in depth regression, images with intricate textures and prominent vanishing lines typically yield confident predictions, whereas images of featureless surfaces are expected to produce higher uncertainty.

3.2 Uncertainty Evaluation Metrics

Here, we introduce the four selected uncertainty metrics for the deep regression tasks. We define the metrics as functions of a dataset $S = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \dots, N\}$, consisting of input-output pairs $(\mathbf{x}_i, y_i)$. For any learned model, we further define $F(\mathbf{x}; \theta) \mapsto \mathbb{R}$, and $U(\mathbf{x}; \theta) \mapsto \mathbb{R}$ to be functions based on the model parameters θ . $F(\mathbf{x}; \theta)$ produces a prediction of y , and $U(\mathbf{x}; \theta)$ produces an uncertainty estimate.

AUSE: *Sparsification plots* are widely used in optical flow (Mac Aodha et al., 2013; Wannewetsch et al., 2017) and stereo disparity (Häger et al., 2021) to assess how well the predicted uncertainty coincides with the prediction error on a test dataset. A sparsification plot is created by plotting the mean absolute error for a dataset while iteratively removing the data point with the highest predicted uncertainty. If the predicted uncertainty coincides well with the error, the sparsification plot should form a monotonically decreasing curve. The best possible sparsification plot, commonly called *oracle sparsification*, is generated when the predicted uncertainty corresponds exactly to the absolute error. Ilg et al. (2018) define *sparsification error* as the difference between a models' sparsification and the corresponding oracle. AUSE is then defined as the area under the sparsification error curve. However, Ilg et al. (2018) never provide a formal mathematical definition for AUSE, and therefore we provide one here.

AUSE is computed as an aggregate metric over a dataset S consisting of input-output $(\mathbf{x}_i, y_i)$ pairs. We define the *mean absolute error* (MAE) on S as:

$$\text{MAE}(S) = \frac{1}{|S|} \sum_{(\mathbf{x}_i, y_i) \in S} |y_i - F(\mathbf{x}_i; \theta)| . \quad (1)$$

Based on a parameter $\alpha \in [0, 1]$, we partition S into disjoint subsets $S_{\wedge}^{\text{AE}}(\alpha)$ and $S_{\vee}^{\text{AE}}(\alpha)$ such that $|S_{\wedge}^{\text{AE}}(\alpha)| = \alpha|S|$, and

$$\begin{aligned} & |y_i - F(\mathbf{x}_i; \theta)| \geq |y_j - F(\mathbf{x}_j; \theta)| \\ \forall (\mathbf{x}_i, y_i) \in S_{\wedge}^{\text{AE}}(\alpha), (\mathbf{x}_j, y_j) \in S_{\vee}^{\text{AE}}(\alpha) . \end{aligned} \quad (2)$$

In other words, the size of $S_{\wedge}^{\text{AE}}(\alpha)$ is a fraction α of the entire set S , and all absolute errors in $S_{\wedge}^{\text{AE}}(\alpha)$ are larger than those in $S_{\vee}^{\text{AE}}(\alpha)$. We also define analogous partitions $S_{\wedge}^U(\alpha)$ and $S_{\vee}^U(\alpha)$ with respect to the uncertainty:

$$\begin{aligned} & U(\mathbf{x}_i; \theta) \geq U(\mathbf{x}_j; \theta) \\ \forall (\mathbf{x}_i, y_i) \in S_{\wedge}^U(\alpha), (\mathbf{x}_j, y_j) \in S_{\vee}^U(\alpha) . \end{aligned} \quad (3)$$

With these partitions in place, we define:

$$\text{AUSE}(S) = \int_0^1 \frac{\text{MAE}(S_V^U(\alpha))}{\text{MAE}(S)} - \frac{\text{MAE}(S_V^{\text{AE}}(\alpha))}{\text{MAE}(S)} d\alpha. \quad (4)$$

AUSE computes the normalized area between the two curves $\text{MAE}(S_V^U(\alpha))$ and $\text{MAE}(S_V^{\text{AE}}(\alpha))$ formed by varying α from 0 to 1. The curve $\text{MAE}(S_V^{\text{AE}}(\alpha))$ is called the *oracle*, which represents a lower bound on sparsification. In practice, the integral (4) is replaced with summing over finite test set samples. See Fig. 5 for an example of an AUSE sparsification plot.

Since AUSE is computed as the difference between two curve, the definitive lower bound is zero, but there is no clear upper bound.

Calibration Error The reliability diagram (Niculescu-Mizil & Caruana, 2005) and the expected calibration error (Pakdaman Naeini et al., 2015) are originally diagnostic tools for classification models that compare sample accuracy against predicted confidence. However, this approach does not apply to regression tasks. Therefore, Kuleshov et al. (2018) introduce a calibration plot for regression tasks in terms of the cumulative distribution function from the model, and summarize it with *CE* as a numerical score. Formally, let the predicted probability distribution of an input $\mathbf{x}_i$ be $P(y_i|\theta) = F(\mathbf{x}_i; \theta)$. The empirical frequency is then defined as

$$\hat{p}_j = \frac{|\{y_i \mid P(y < y_i|\theta) \leq p_j, (\mathbf{x}_i, y_i) \in S\}|}{N}. \quad (5)$$

Here $p_j \in [0, 1]$ represents an arbitrary threshold value. Given M distinct thresholds $p_1, p_2, \dots, p_M$ (typically evenly spaced from 0 to 1), the calibration error is defined for regression as

$$\text{cal}(\hat{p}_1, \dots, \hat{p}_M) = \sum_{j=1}^M w_j (p_j - \hat{p}_j)^2, \quad (6)$$

where w_j are arbitrary scaling weights, usually $w_j = \frac{1}{N/\hat{p}_j}$.

Spearman correlation In 1904, psychologist Charles Spearman defined what he called the *rank method* of correlation (Spearman, 1904). He observed that there may exist correlations that would not be adequately captured by simple linear correlation. Instead, he argued that more complex correlations may be adequately captured by comparing *ranks* of elements.

Given a set of samples, the rank of a sample is simply the index that number would have in a list. More formally, let $L = [u_1, u_2, \dots, u_N]$ be a sequence of samples, then the rank of a sample is defined as:

$$r(u_i) = 1 + |\{u_j \mid u_j < u_i, u_j \in L\}|. \quad (7)$$

We then define the rank operation for a sequence:

$$R(L) = [r(u_1), r(u_2), \dots, r(u_N)] . \quad (8)$$

Spearman’s rank method is then the correlation between two rank sequences:

$$\rho_{R(L_1), R(L_2)} = \frac{\text{cov}(R(L_1), R(L_2))}{\sigma_{R(L_1)} \sigma_{R(L_2)}} . \quad (9)$$

When used as a metric for uncertainty, this correlation is computed between the predictive uncertainty, and the absolute prediction errors (Tan et al., 2023).

NLL The popular negative log-likelihood is proved to be a *strictly proper scoring rule* (Lakshminarayanan et al., 2017). Given a dataset $S = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \dots, N\}$ and a probability density function $p(y|\mathbf{x})$ parameterised by learnable parameters θ , the NLL is defined as

$$\text{NLL}(S) = - \sum_{i=1}^N \log p(y_i | \mathbf{x}_i; \theta) . \quad (10)$$

In expectation, the NLL is minimized if and only if $p(y_i | \mathbf{x}_i; \theta)$ is equal to the true underlying data distribution (Hastie et al., 2009). As such, the NLL can also be used as a metric for uncertainty predictions, since a model with a lower NLL does a better job (in expectation) of fitting the true data distribution.

3.3 Regression Models with Uncertainty Predictions

There exist many different model architectures that incorporate uncertainty predictions. Note that our goal is not to investigate properties in the models. We choose to only use two different models, namely an ensemble and an energy-based model. While there are many other popular models for UQ (e.g. the recently introduced SNGP by Liu et al. (2024)) the purpose of this paper is to evaluate the metrics, not the models.

Deep Ensemble (DE)

Ensemble learning combines the predictions from multiple individual models to achieve better predictive performance than any single model. For estimating predictive uncertainty, *Deep Ensemble* (DE) (Lakshminarayanan et al., 2017) is a simple yet effective method that trains multiple models in the same architecture with different random initialization and data shuffling. The ensemble is treated as a uniformly weighted mixture model. In practice, for the regression task, each individual model outputs

two scalars, which are interpreted as the mean and variance of a Gaussian distribution, and then it is trained to minimize the Gaussian NLL loss as in (11) on the training set.

$$-\log p_{\theta}(y_n | \mathbf{x}_n) = \frac{\log \sigma_{\theta}^2(\mathbf{x})}{2} + \frac{(y - \mu_{\theta}(\mathbf{x}))^2}{2\sigma_{\theta}^2(\mathbf{x})} + C \quad (11)$$

We follow the suggestion by Lakshminarayanan et al. (2017) to train 5 models and thus get a mixture of Gaussians. This mixture is further approximated by a Gaussian whose mean and variance are the mean and variance of the mixture respectively.

Energy Based Regression (EBR)

Energy-based learning involves learning an energy function $\mathcal{E}(x)$. The goal when learning $\mathcal{E}(x)$ is to assign low energy to observed samples (Gustafsson et al., 2019). Learning $\mathcal{E}(x)$ is in many ways analogous to learning a probability density function, and as such, it has commonly been used for unsupervised learning tasks (Gustafsson et al., 2019). Gustafsson et al. (2019) construct a supervised regression model based on an energy function. First, they define $f_{\theta}(x, y) \mapsto \mathbb{R}$ to be their learned energy function. Then, they construct a probability density function:

$$p(y|x; \theta) = \frac{e^{f(x, y; \theta)}}{Z(x; \theta)}, \quad Z(x; \theta) = \int e^{f(x, \tilde{y}; \theta)} d\tilde{y} . \quad (12)$$

Parameters θ are trained by minimizing the NLL with respect to the training data:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \log Z(x_i; \theta) - f(x_i, y_i; \theta) . \quad (13)$$

$Z(x_i; \theta)$ is approximated by Monte Carlo sampling. Predictions are generated by performing multi-start gradient ascent to maximize $f(x_i, \tilde{y}; \theta)$ with respect to $\tilde{y}$. For uncertainty quantification, the raw value of the energy function is used.

4 Experiments and Results

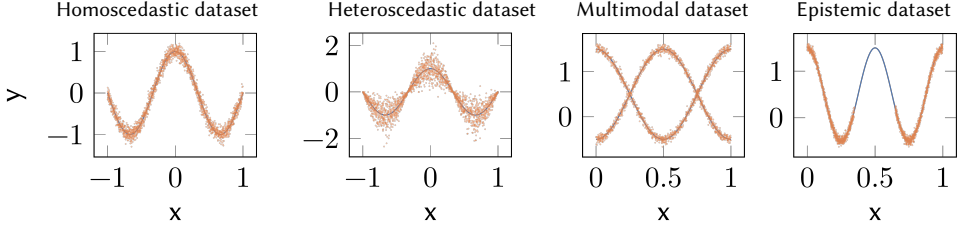


Figure 2: The four synthetic regression datasets. Data points are orange, and the solid blue lines represent the expectation of the generating function.

4.1 Synthetic Regression Datasets

In order to gauge the behavior of the different uncertainty metrics, we construct four simple synthetic regression datasets, each with a different source of uncertainty. These can be seen in Fig. 2. We will henceforth refer to these datasets by their names: *homoscedastic*, *heteroscedastic*, *multimodal*, and *epistemic*. The names homoscedastic and heteroscedastic are in reference to the type of Gaussian noise applied to the generating function. Multimodal refers to the fact that data is generated from two separate generating functions. Finally, epistemic, refers to the epistemic uncertainty arising from a gap in the training data. Fundamentally, each dataset is a collection of input-output pairs (x, y) . Outputs y are generated as follows:

- Homoscedastic:

$$y = \cos(1.5\pi x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1) .$$

- Heteroscedastic:

$$y = \cos(1.5\pi x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.4 \cdot |\cos(1.5\pi x)|) .$$

- Multimodal:

$$y = 0.5 \pm \cos(2\pi x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.05) .$$

- Epistemic:

$$y = 0.5 + \cos(4\pi x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.05) .$$

Here, $\mathcal{N}(0, \sigma)$ refers to a Gaussian distribution with mean 0, and standard deviation σ . $\pm$ in the definition of the multimodal dataset refers to an equal chance of being + or -, which represents the two different modes. In the epistemic dataset, there is a gap in the training data for $x \in [0.35, 0.65]$, which is not present in the test data. This means test data is OOD, which causes epistemic uncertainty. Domains for the inputs are $x \in [-1, 1]$ for the homoscedastic and heteroscedastic datasets, and $x \in [0, 1]$ for the multimodal and epistemic datasets.

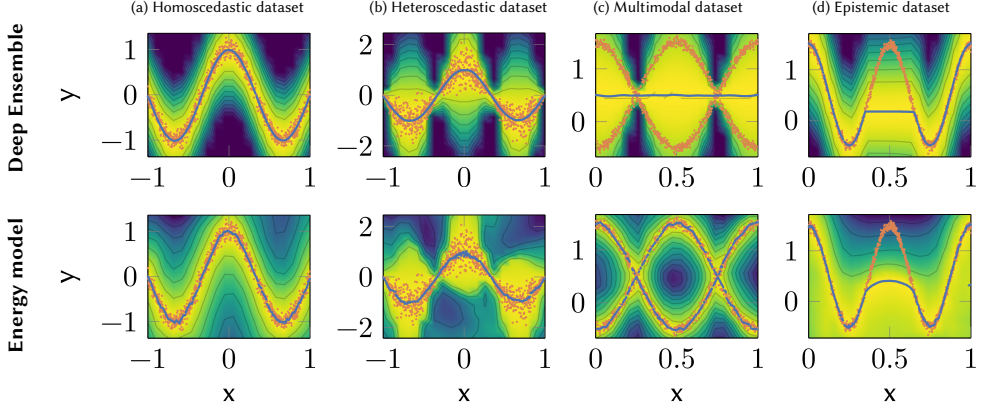


Figure 3: Visualization of the predicted density on the test set for trained models. Contour plots: Log-likelihood output from each model. Yellow: the high-density region; Blue: the low-density region. Blue points: Predicted mean. Orange points: Test set.

4.2 Implementation Details

For each synthetic dataset, we train one EBR model, and one DE model. Both types of models were trained with a batch size of 128 using the Adam (Kingma & Ba, 2015) optimizer. The sizes of the training, validation, and testing datasets were respectively, 100000, 1000, and 1000 samples. Fig. 3 shows the resulting log-likelihood functions from the models, along with their predictions on our test sets.

Energy Based Regression We implement EBR as a 9-layer perceptron with ReLU activations and a hidden size of 256, which we train for 20 epochs with a fixed learning rate 10^{-4} . While 9 layers may sound too deep for our simple toy datasets, our initial experiments showed clear signs of underfitting with shallower models.

Deep Ensemble All models in our ensemble are identical 5-layer perceptrons, with ReLU activations, and a hidden size of 256. In this case 5 layers is more than enough for the model to learn our toy datasets, and we choose this size simply to avoid a comparison between models of drastically different depth. Overfitting is not a concern since we can control the amount of data we generate. We train five models for 20 epochs each with a fixed learning rate of 10^{-3} . In our experiments we use the predicted variance as $U(x)$, more choices are compared by Xiong et al. (2024).

4.3 Stability on varying test set sizes

All metrics are estimates that are computed on a test set of finite size. It is thus important to investigate how quickly each metric converges to its expected value, and whether the estimates are biased for small test set sizes.

First, in order to test convergence, we emulate a process of iteratively collecting points. We begin with generating a single large test dataset of size 2^{16} . From this dataset, we randomly sample data points without replacement, which allows us to investigate how quickly each metric converges to its “true” value for a given dataset. We evaluate each metric at test set sizes $2^3, 2^4, \dots, 2^{16}$. The resulting metrics from each subset are reported in Fig. 4 (a).

Second, in order to investigate any bias in the approximation of each metric, we sample 100 i.i.d. datasets at each size $2^3, 2^4, \dots, 2^{16}$. We then report the average of each metric over these 100 datasets in Fig. 4 (b). If a metric is truly unbiased, the average at a small dataset size should converge to the same value as the average at larger dataset sizes.

For both experiments, we use our heteroscedastic dataset because we believe it is the best candidate to highlight any stability issues, due to its varying noise levels. We apply the DE model in both experiments since its performance is similar to EBR model on the heteroscedastic dataset as shown in Fig. 3 (b) and the choice of model is agnostic.

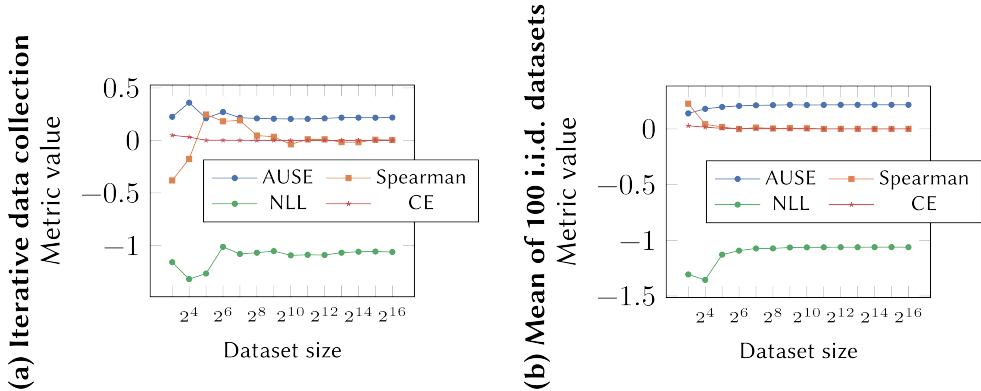


Figure 4: Experiments to test two types of stability of metrics under different test dataset sizes.

4.4 Metrics under different types of uncertainty

We compute all metrics for both our models on the test sets of each of the four datasets. As a reference, we compare the results to those computed on the data-generating distribution. Using the data generating distribution will, in expectation, minimize NLL and CE, and should provide a good point of comparison for all metrics. We summarize metrics of both models for each dataset in Tab. 1, 2, 3 and 4 respectively.

Table 1: Metrics for all models on Homoscedastic dataset.

	AUSE↓	CE↓	NLL↓	Spearman↑
DE	0.5915	0.0023	-0.8819	0.0444
EBR	0.5707	0.0032	-0.8568	-0.0264
True dist.	0.5917	0.0003	-0.8965	0.0076

Table 2: Metrics for both models on Heteroscedastic dataset.

	AUSE↓	CE↓	NLL↓	Spearman↑
DE	0.2334	0.0001	-0.1091	0.0005
EBR	0.2422	0.0001	-0.0990	-0.0419
True dist.	0.2305	0.0001	-0.1472	0.0280

Table 3: Metrics for both models on multimodal dataset.

	AUSE↓	CE↓	NLL↓	Spearman↑
DE	0.0071	0.0229	0.8098	-0.0420
EBR	0.5821	0.0018	-0.6535	0.0093
True dist.	0.5180	0.0001	-0.7935	0.0073

Table 4: Metrics for both models on epistemic dataset.

	AUSE↓	CE↓	NLL↓	Spearman↑
DE	0.6016	0.0145	36.4332	0.0298
EBR	1.3888	0.0298	31.2425	0.0067
True dist.	0.5454	0.0001	-1.5871	-0.0194

4.5 Metrics for Real-world Applications

To further demonstrate the behaviour of the metrics, we apply them to the real-world problem of stereo disparity prediction. This is a regression task where the disparity of X-coordinates are to be predicted for corresponding pixels in the left and right frames of a stereo camera. We adopt the continuous disparity network(CDN) from Garg et al. (2020), and we use the Hinge-Wasserstein training loss (Xiong et al., 2024) (hinge value 0.005) as it improves both the regression and UQ performance. The model is trained on the large dataset Sceneflow (Mayer et al. (2016)), where dense ground truth disparity maps are provided. As this is real image data, there is already an unknown amount of aleatoric uncertainty in the test set. To this we add controllable amounts of uniform Gaussian noise, to both left and right frames during testing. Specifically, Gaussian noise with a set of different variances is added to the RGB image normalized by ImageNet mean and variance. (This means that the image variance can be expected to be around 1 in the noise free case.) We use End-Point-Error (EPE) and 1-Pixel Threshold Error (1PE), as in Garg et al. (2020) to quantify for regression performance. All relevant performance and uncertainty metrics are provided in Tab. 5.

Table 5: All metrics for stereo disparity prediction with CDN supervised by hinge-Wasserstein (hinge value 0.005). Variances are reported with 5 runs.

Noise scale	EPE	1px	NLL	AUSE	CE	Spearman
0	1.01 $\pm$ 0	9.44 $\pm$ 0	0.000521 $\pm$ 3.6×10^{-6}	0.159 $\pm$ 0.0006	0.2663 $\pm$ 0	0.0012 $\pm$ 0.0004
0.1	1.13 $\pm$ 0.00	10.97 $\pm$ 0.00	0.000590 $\pm$ 4.2×10^{-6}	0.161 $\pm$ 0.0014	0.2629 $\pm$ 0.0001	0.0010 $\pm$ 0.0006
0.2	1.85 $\pm$ 0.00	15.91 $\pm$ 0.00	0.000735 $\pm$ 2.8×10^{-6}	0.128 $\pm$ 0.0013	0.2457 $\pm$ 0	0.0006 $\pm$ 0.0006
0.3	2.81 $\pm$ 0.01	22.54 $\pm$ 0.01	0.000904 $\pm$ 1.2×10^{-6}	0.113 $\pm$ 0.0008	0.2210 $\pm$ 0	0.0024 $\pm$ 0.0005
0.4	4.01 $\pm$ 0.01	29.81 $\pm$ 0.01	0.001061 $\pm$ 1.9×10^{-6}	0.103 $\pm$ 0.0008	0.1947 $\pm$ 0.0002	0.0026 $\pm$ 0.0006
0.5	5.50 $\pm$ 0.01	37.32 $\pm$ 0.01	0.001212 $\pm$ 2.0×10^{-6}	0.099 $\pm$ 0.0008	0.1686 $\pm$ 0.0001	0.0040 $\pm$ 0.0757

5 Discussion

Here we analyze the experiment results, and reason about various strengths and limitations of each metric.

5.1 Stability on varying test set sizes

From Fig. 4 (a) we can see that all metrics vary as the test set sizes increase and finally converge to the expectation. We can conclude that all metrics converge beyond a dataset size of 2^{10} . Hence, for most modern regression applications, where test datasets are larger than ~ 1000 samples, stability should not be a cause for concern. While our analysis is based only on a single dataset, we expect the convergence speed to be reasonably consistent across different datasets. Though the amplitudes of metrics vary from each other, arguably we can also conclude that CE is the most stable and AUSE converges second fastest.

From Fig. 4 (b) we can see that the averages change for different dataset sizes. We will henceforth refer to this behavior as *estimation bias of mean*. Arguably, estimation bias is undesirable, since we expect the average of a metric to converge to the same value regardless of the dataset size. As such, a metric with significant bias of the mean should be considered unstable and its use discouraged. Overall, CE is the most stable and AUSE the second most in terms of bias of the mean. We can also conclude that no metric seems to exhibit any meaningful bias beyond a dataset size of 2^6 .

Together, these two plots suggest that the most stable metric is CE, followed by AUSE, then NLL, and finally Spearman correlation. Surprisingly, in both Fig. 4 (a) and (b), Spearman correlation has large (absolute) values with small test set sizes and gradually converges to zero as the test set size grows. This can be explained by a closer look at Fig. 3 (b): All points close to $x = 0$ will have similar predicted uncertainty values, but their errors may be vastly different, which counteracts the correlation between error and uncertainty. Naturally, this counteracting effect becomes less impactful when samples are sparse, and Spearman correlation has been used successfully on for example the rMD17 dataset (Christensen, 2020).

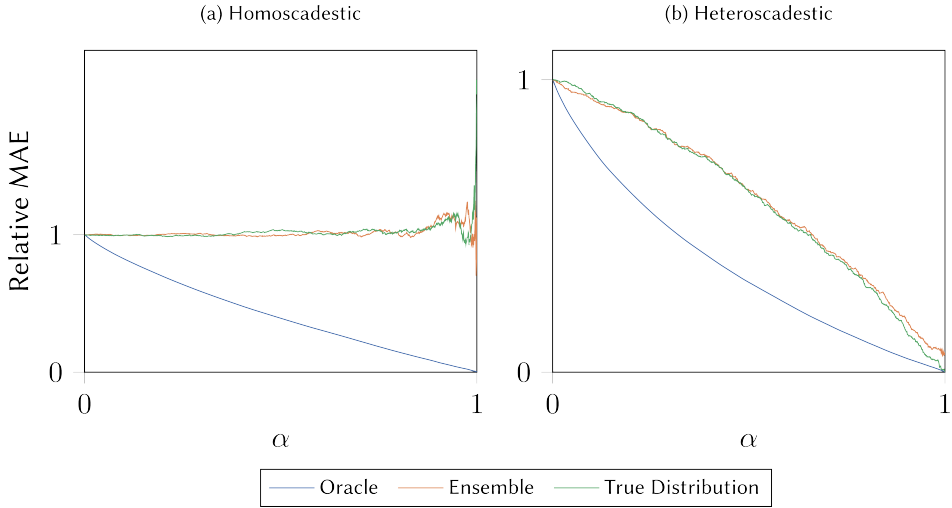


Figure 5: Sparsification plot from Deep Ensemble and True Distribution for the homoscedastic and heteroscedastic datasets. α is the fraction of removed samples.

5.2 Metrics under different types of uncertainty

Homoscedastic and Heteroscedastic datasets: In Fig. 3 (a) and (b) we can see that both models learn predictive distributions close to the generating functions (defined in Sec. 4.1).

In Tab. 1 and 2, all metrics except Spearman suggest that both models learn the homoscedastic and heteroscedastic distributions approximately equally well. In Fig. 5 (a), the sparsification curve is nearly horizontal, indicating that it is impossible to learn a meaningful correlation between errors and uncertainty. The predicted variance can only partially explain the error, the remaining part is caused by the difference between the annotation and the predicted mean. Intuitively, we expect this scenario of a perfectly uniform variance to be rare in real-world applications. This also indicates that generating uncertainty measures from the predicted parametric distribution may not be the optimal approach for sorting samples. Fig. 3 (a) and Fig. 5 (b) together further show this: even when the predictive and generating distributions are very well aligned, the AUSE value is not 0. For the heteroscedastic dataset, the true distribution achieves the best NLL, and the DE model is slightly better than EBR model. This is because the DE model learns a sharper distribution than EBR, as can be seen in the contour plots of Fig. 3 (a). NLL, as a proper scoring rule, addresses calibration and sharpness simultaneously (Gneiting & Raftery, 2007).

Multimodal dataset: In Fig. 3 (c) we can see that energy-based model successfully learns to predict both modes, but as the ensemble is not designed to output multimodal predictions, it predicts the average of the modes, and a wide distribution.

Tab. 3 shows larger discrepancy between metrics. All metrics suggest that the energy-based model has learned a good distribution that is close to the generating distribution. AUSE for DE is near perfect, which indicates good detection of when DE fails to predict multiple modes. CE and NLL instead indicate the ensemble’s failure to learn both modes with high values. This highlights the main difference between AUSE (and Spearman correlation) and CE/NLL: AUSE measures a correlation between uncertainty and the absolute error, while CE and NLL measure how well the learned distribution captures the data. This raises the question of which type of measurement is most useful, which we will discuss in Sec. 5.5.

Epistemic dataset: In Fig. 3 (d), both models exhibit somewhat larger uncertainty in the OOD region. However, neither model predicts a distribution that assigns high probability to the test data. Visually, the predicted distributions of both models look similar. The only obvious perceptible difference is the slight bend in the center of the distribution of the EBR model.

In Tab. 4, AUSE indicates that the ensemble predicts distribution that can find OOD samples almost as well as the generating distribution, while the distribution from the EBR model is much worse. Both CE and NLL agree that both models failed to learn a good distribution, but they disagree on which model is worse. The contours in Fig. 3 (d) also shows the DE model gives sharper distribution. It is worth noting that there is an entire field of study of OOD detection, which offers better methods for handling epistemic uncertainty. Here, we are instead interested in whether or not each metric can detect a failure to learn the underlying distribution.

Looking at the AUSE scores of 0.6016 and 1.3888 for the ensemble and EBR models respectively, we can infer that this small bend reduces the correlation between error and uncertainty enough to (more than) double the AUSE score. Intuitively, we would expect these two similar distributions to get similar scores, which is exactly the case with both CE and NLL. This raises questions regarding the stability of AUSE under small model variations.

5.3 Real-world Applications

Tab. 5 summarize the results of the real-world stereo disparity experiment. As can be expected, regression performances drop as the noise scale increases. It is immediately clear that the variance of Spearman correlation is too large to be a reliable metric. Both AUSE and CE keeps decreasing as the noise scale increases, which indicates that the used Hinge–Wasserstein loss is able to quantify uncertainty under adversarial noise. Meanwhile, NLL keeps increasing, which means the sharpness of predictions get worse with higher noise levels. This is consistent with Xiong et al. (2024) which illustrates that a lack of sharpness is a side effect of the Hinge–Wasserstein loss.

5.4 Interpretability

CE has well defined bounds which makes it easy to interpret. With NLL and AUSE, this is more difficult. We can compare them across models, however, the lower and upper bounds depend on both model and data.

CE: CE has a lower bound 0 when the predicted and generating distributions are equal. The upper bound, however, is dependent on the number of intervals and the weights w_j .

NLL: The lower bound of NLL is given by the generating distribution, which is typically unknown when training for real-world deployment. The upper bound is positive infinity.

AUSE: From Sec. 5.2 we know that the generating distribution may not even give the optimal lower bound 0. We argue that predicted variance can only partially explain the error. There is no clear upper bound for AUSE.

In summary, NLL and AUSE can only tell us which model is better at UQ but cannot tell us how good it is. This lack of interpretability constitutes a severe limitation of applying AUSE and NLL into real-world autonomous systems.

5.5 Concluding Remarks

We have tested four common uncertainty assessment metrics on four synthetic datasets that are designed to illustrate properties of these metrics. We also test all the metrics on a real-world regression task, Stereo disparity. Below we summarize our conclusions:

Spearman Correlation This metric is in general unsuitable. It has stability issues for small test sets, and converges to zero as the test set size grows, see Sec. 5.1.

AUSE This metric offers a robust quantized measure on the correlation between predictive uncertainty and errors compared with Pearson correlation and Spearman correlation. It tells us how wrong a model prediction is likely to be, which is vital for trust-worthy autonomous systems. On the down side, its value is unbounded and thus it lacks interpretability, and it may fail in tasks where homoscedastic uncertainty dominates (but this is rare in real-world applications). It may be more appropriate for loss prediction (Yoo & Kweon, 2019; Cui et al., 2024), which is another UQ approach compared to predicting full distributions.

Calibration Error This metric has a clear lower bound of 0, and is highly interpretable; It requires the least amount of samples to be stable, and can be extended to general confidence intervals Kuleshov et al. (2018). Limitations are that it requires predicting a distribution, and it does not capture how wrong the predictions can be.

NLL This measures the shape of the predictive distributions, including calibration and sharpness. It fails when the generating distribution is multimodal but the training target is unimodal (Xiong et al., 2024), and it requires the largest amount of test samples to be stable among those tested here.

Essentially CE and NLL measure how well the predicted probability distribution corresponds to the true data generating distribution, but CE is the more interpretable of the two. In contrast, AUSE and Spearman measure how well the predicted uncertainty can tell the magnitude of errors. We discourage the use of Spearman correlation for uncertainty quantification, as AUSE has consistently proved to be more robust. Which metric is most useful will naturally differ between applications.

Acknowledgement: This work was funded by Swedish national strategic research environment ELLIIT, grant C08. Computations for this work were enabled by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Berzerlius resource provided by National Supercomputing Centre at Linköping University and the Knut and Alice Wallenberg foundation, partially funded by the Swedish Research Council through grant agreement no. 2022-06725. This project is also supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

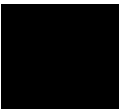
Bibliography

- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information science and statistics. New York: Springer.
- Christensen, A., Anders S.; Von Lilienfeld (2020). Revised md17 dataset (rmd17). <https://doi.org/10.6084/m9.figshare.12672038.v3>, .
- Cui, P., Zhang, D., Deng, Z., Dong, Y., & Zhu, J. (2024). Learning sample difficulty from pre-trained models for reliable prediction. *Advances in Neural Information Processing Systems*, 36.
- Eldesokey, A., Felsberg, M., Holmquist, K., & Persson, M. (2020). Uncertainty-aware CNNs for depth completion: Uncertainty from beginning to end. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Garg, D., Wang, Y., Hariharan, B., Campbell, M., Weinberger, K. Q., & Chao, W.-L. (2020). Wasserstein distances for stereo disparity estimation. *Advances in Neural Information Processing Systems*, 33, 22517–22529.
- Gawlikowski, J., Rovile Njietcheu Tassi, C., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., & Zhu, X. X. (2023). A Survey of Uncertainty in Deep Neural Networks. *Artificial Intelligence Review*, 56, 1513–1589. doi:10.1007/s10462-023-10562-9.
- Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102, 359–378.
- Grimmett, H., Triebel, R., Paul, R., & Posner, I. (2016). Introspective classification for robot perception. *The International Journal of Robotics Research*, 35, 743–762. doi:10.1177/0278364915587924.
- Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321–1330). PMLR.
- Gustafsson, F. K., Danelljan, M., Bhat, G., & Schön, T. B. (2019). DCTD: deep conditional target densities for accurate regression. *CoRR, arXiv:1909.12297*.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer. doi:10.1007/978-0-387-84858-7.
- Heiss, J., Weissteiner, J., Wutte, H., Seuken, S., & Teichmann, J. (2022). NOMU: Neural optimization-based model uncertainty. In *39th International Conference on Machine Learning* PLMR 162. doi:10.48550/arXiv:2102.13640.

- Häger, G., Persson, M., & Felsberg, M. (2021). Predicting disparity distributions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4363–4369). doi:10.1109/ICRA48506.2021.9561617.
- Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., & Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 652–667).
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82, 35–45.
- Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.
- Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*. San Diego, CA, USA.
- Kuleshov, V., Fenner, N., & Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *Proceedings of the 35th International Conference on Machine Learning* (p. 2796–2804). PMLR.
- Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.
- Liu, J. Z., Padhy, S., Ren, J., Lin, Z., Wen, Y., Jerfel, G., Nado, Z., Snoek, J., Tran, D., & Lakshminarayanan, B. (2024). A simple approach to improve single-model deep uncertainty via distance-awareness. *J. Mach. Learn. Res.*, 24.
- Loquercio, A., Segu, M., & Scaramuzza, D. (2020). A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5, 3153–3160. doi:10.1109/LRA.2020.2974682.
- Mac Aodha, O., Humayun, A., Pollefeys, M., & Brostow, G. J. (2013). Learning a confidence measure for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 1107–1120. doi:10.1109/TPAMI.2012.171.
- Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4040–4048).
- Ng, M., Guo, F., Biswas, L., Petersen, S. E., Piechnik, S. K., Neubauer, S., & Wright, G. (2023). Estimating uncertainty in neural networks for cardiac mri segmentation: A benchmark study. *IEEE Transactions on Biomedical Engineering*, 70, 1955–1966. doi:10.1109/TBME.2022.3232730.

- Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning* ICML '05 (p. 625–632). New York, NY, USA: Association for Computing Machinery. doi:10.1145/1102351.1102430.
- Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., & Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32.
- Pakdaman Naeini, M., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29. doi:10.1609/aaai.v29i1.9602.
- Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15, 72–101. doi:10.2307/1412159.
- Tan, A. R., Urata, S., Goldman, S., Dietschreit, J. C. B., & Gómez-Bombarelli, R. (2023). Single-model uncertainty quantification in neural network potentials does not consistently outperform model ensembles. *npj Computational Materials*, 9, 1–11. doi:10.1038/s41524-023-01180-8.
- Wannenwetsch, A. S., Keuper, M., & Roth, S. (2017). Probflow: Joint optical flow and uncertainty estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 1182–1191). doi:10.1109/ICCV.2017.133.
- Xiong, Z., Eldesokey, A., Johnander, J., Wandt, B., & Forssén, P.-E. (2024). Hingewasserstein: Mitigating overconfidence in regression by classification. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2024)*.
- Yoo, D., & Kweon, I. S. (2019). Learning loss for active learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 93–102). doi:10.1109/CVPR.2019.00018.

Paper IV



GPify: Leveraging the combined strength of Normalizing Flow and Softmax for an out-of-distribution aware confidence score

Simon Kristoffersson Lind

Lund University LTH

`simon.kristoffersson_lind@cs.lth.se`

Rudolph Triebel

German Aerospace Center and

Karlsruhe Institute of Technology

`rudolph.triebel@dlr.de`

Volker Krüger

Lund University LTH

`volker.krueger@cs.lth.se`

Abstract

In order for any learning-based model to be considered reliable, it needs a well-behaved uncertainty or confidence estimate. Most modern neural networks do produce a confidence estimate in the form of their softmax output probability. However, the softmax probability is invalid for out-of-distribution data. Gaussian processes are known to produce a well-behaved confidence estimate that is aware of out-of-distribution samples. Inspired by Gaussian processes, we propose GPify, which combines the softmax probability with a Normalizing Flow in order to add out-of-distribution awareness to the confidence estimate from a neural network. The resulting confidence from GPify is an uncertainty measure that is interpretable and intuitive, while also being probabilistically sound. We evaluate GPify in a selective classification framework, and conclude that it achieves comparable performance to state-of-the-art methods. In addition, we show that GPify has capabilities for detecting adversarial examples, which is a direct improvement over softmax confidence.

1 Introduction

It is well known that neural networks perform unreliably when an input is too dissimilar from the data it was trained on (Yang et al., 2024). In order to reason about reliability a model needs introspective capabilities, specifically a well-behaved confidence or uncertainty estimate (Grimmett et al., 2016). Confidence and uncertainty are fundamentally equivalent in this context, and therefore we discuss them interchangeably throughout this work.

Gaussian Processes (GP) (Rasmussen and Williams, 2005) are considered to be one of the most reliable models, in part due to their ability to produce a well-behaved and probabilistically sound confidence estimate. However, due to their non-parametric nature, GPs fail to scale to the data quantities in modern learning applications. With this work, we propose *GPify*: a simple method of combining the softmax confidence from a neural network with probability estimates from a Normalizing Flow (NF) in a way that mimics the confidence estimates from a GP. The resulting confidence score is probabilistically sound and interpretable like a GP, but it *a)* successfully scales to larger applications, and *b)* is easy to apply with any neural network.

Uncertainty is commonly discussed in the form of aleatoric and epistemic uncertainty (Kristofferson Lind et al., 2024). In the context of classification problems, aleatoric uncertainty refers to the inherent ambiguity between classes. As such, the output confidence from a softmax operation reflects a model’s aleatoric uncertainty. Epistemic uncertainty, however, refers to gaps in the training data, where a model cannot make accurate predictions because it was never trained on similar samples. Since most neural network architectures use a simple classifier with linear decision boundaries, they do not inherently reflect epistemic uncertainty. These two types of uncertainty are typically measured and evaluated in entirely separate problem settings. Aleatoric uncertainty is commonly evaluated in the context of calibration (Guo et al., 2017), and epistemic uncertainty is typically evaluated in the context of out-of-distribution (OOD) detection (Yang et al., 2024).

Selective classification is a problem that extends classification with an option to reject a data sample (El-Yaniv and Wiener, 2010). Here, classification is the primary task, and rejection is presented as a mechanism to improve accuracy on the remaining samples. Kim et al. (2023) propose *unknown detection* as a selective classification problem where out-of-distribution samples are included among the inputs. The inclusion of both in- and out-of-distribution samples implies the presence of both aleatoric and epistemic uncertainty. *Softmax Information Retaining Combination* (SIRC) (Xia and Bouganis, 2024) is a recent approach that combines an aleatoric and epistemic confidence into a single score. SIRC is shown to perform well on the unknown

detection problem. Xia and Bouganis (2024) coin the acronym SCOD for *Selective Classification in the presence of Out-of-distribution Data*, as a synonym to unknown classification. In this work we also adopt the term SCOD, since it better emphasizes classification as the primary task.

Our work is undoubtedly similar to SIRC (Xia and Bouganis, 2024). However, instead of focusing directly on maximizing performance within the SCOD framework, we focus primarily on designing GPify to be intuitive and probabilistically sound, which makes it more desirable for end-users in real-world applications. Nonetheless, we compare GPify against SIRC within the SCOD problem setting, to evaluate its performance.

In addition to our SCOD evaluation, we conduct a small-scale experiment to evaluate whether GPify has the capacity to detect adversarial examples.

To summarize, our contributions are as follows:

- We propose GPify, which combines the confidence scores from a softmax classifier with the likelihood from a Normalizing Flow, to create a single confidence score that captures both aleatoric and epistemic uncertainty.
- We evaluate GPify in the SCOD problem setting, and show that GPify offers highly competitive performance.
- We evaluate GPify with different neural network architectures, to verify that it generalizes to different architectures.
- In an additional, minor experiment, we show that GPify is capable of detecting adversarial examples, though a more thorough experiment is needed to evaluate how GPify compares to dedicated adversarial detection methods.

2 Related Work

Out-of-distribution detection is an active research area with the aim of classifying data samples as in- or out-of-distribution (Yang et al., 2024; Huang et al., 2024; Wang et al., 2022). The general problem of OOD detection is not necessarily concerned with the reliability of neural networks, but the task of discriminating between different data distributions is fundamentally the same. If we place OOD detection in the context of uncertainty estimation, it is only capable of representing aleatoric uncertainty. As such, OOD detection methods tend to underperform in the SCOD problem setting, as shown in (Kim et al., 2023).

Complementary to OOD detection, there exists a body of work focused on improving *calibration* (Guo et al., 2017; Wengert et al., 2020), which is a term used to describe how accurate a model’s confidence score is. In this context, the accuracy of a confidence score is determined by how well it reflects the probability of a sample being correctly classified (Guo et al., 2017). As such it is only defined on in-distribution data, and only reflects epistemic uncertainty.

To the best of our knowledge, SIRC (Xia and Bouganis, 2024) is the first work to explicitly construct a combined confidence score that incorporates both aleatoric and epistemic uncertainty. However, as pointed out in (Xia and Bouganis, 2024), there exists other methods that can be interpreted as a combination score, most notably GradNorm (Huang et al., 2024) and ViM (Wang et al., 2022). While both of these methods are framed as OOD detection methods, GradNorm can be interpreted as a combination of softmax and the norm of a feature vector (Huang et al., 2024), and ViM can be interpreted as a combination of logits and the residual of a feature vector (Wang et al., 2022). Neither of these combination scores produce values that are easily interpretable or probabilistically sound.

There has been some work focused on modifying Gaussian Processes to improve their performance, and reduce their computational load on larger learning problems (van der Wilk et al., 2017; Blomqvist et al., 2020). Despite these efforts, GPs are still significantly more computationally demanding, and perform significantly worse than neural network-based solutions. While we use GPs as a reference point for their probabilistically sound confidence, we refrain from comparing against them empirically as we aim to construct a well-behaved confidence without degrading the performance of neural networks.

3 Background

3.1 Uncertainty, Calibration, and OOD Detection

There exists two, largely separate bodies of research in the domain of uncertainty estimation, namely those that focus on aleatoric and epistemic uncertainty. Aleatoric uncertainty refers to uncertainty that is inherent in the data, for example class ambiguities or measurement noise (Kristoffersson Lind et al., 2024). As such, aleatoric uncertainty cannot be reduced or removed by adding more data. Epistemic uncertainty, on the other hand, refers to uncertainty stemming from missing data. As an example, consider an animal classifier that is trained on pictures of japanese spitz’s, and arctic foxes. For such a classifier, aleatoric uncertainty stems from the fact that many Japanese spitz’s look similar to arctic foxes, and vice versa. Conversely, a pic-

ture of a penguin would constitute epistemic uncertainty.

Uncertainty is divided into these two categories because they answer fundamentally different questions (Kristoffersson Lind et al., 2024). One way to view these two uncertainties is as probability distributions, where aleatoric uncertainty has the goal of modelling $p(class | \mathbf{x})$, as is a common interpretation of a softmax classifier (Xia and Bouganis, 2024). In this view, epistemic uncertainty aims to model $p(\mathbf{x})$, either directly or indirectly (Yang et al., 2024).

Aleatoric uncertainty is typically evaluated using the Expected Calibration Error (ECE) metric (Guo et al., 2017), which aims to directly evaluate the error in $p(class | \mathbf{x})$:

$$ECE = \sum_m \frac{|B_m|}{|n|} |\text{accuracy}(B_m) - \text{confidence}(B_m)| .$$

Here, B_m represents equidistant bins created from the testing data based on the predicted confidence for each sample, and n represents total number of testing samples.

We argue that ECE is entirely invalid in the presence of OOD samples. Since any prediction made on OOD data is invalid, the accuracy is 0, and any non-zero confidence contributes to increasing the ECE metric.

Epistemic uncertainty is commonly evaluated in the OOD detection setting, using an AUROC score based on the detector’s ability to discriminate between in- and out-of-distribution data samples (Yang et al., 2024). In addition to AUROC, OOD detectors are often evaluated based on their successful OOD detection at a fixed false-negative-rate, where a detection threshold is set such that a fixed percentage of in-distribution samples are rejected as OOD. As OOD detection is a classification problem itself, which treats the in-distribution as a singular class, it fundamentally cannot evaluate aleatoric uncertainty.

3.2 Adversarial Examples

First coined by Szegedy et al. (2014), the term *Adversarial Example* refers to images that have been perturbed at a pixel-level to cause a misclassification. By constructing an optimization problem

$$\min \|\delta\|_2, \text{ s.t. } f(\mathbf{x} + \delta) \neq y_{\text{true}}$$

they are able to generate images that look unchanged to the human eye, but are misclassified by a neural network. Here, we use $f(x)$ to denote the predicted label from a classifier f , and y_{true} to denote the true class label. Since these examples cause a

misclassification in the softmax output, they cannot be detected using softmax probabilities alone.

Goodfellow et al. (2015) improve the process of creating adversarial examples, resulting in the *Fast Gradient Sign Method* (FGSM):

$$\mathbf{x}_{\text{adversarial}} = \mathbf{x} + \epsilon \text{sign}(\nabla_x \mathcal{L}(f(\mathbf{x}), y_{\text{true}})) .$$

Here $\mathbf{x}$ is directly modified in the direction of the gradient in order to maximize the loss function $\mathcal{L}$. FGSM is further modified by Kurakin et al. (2017), who perform the same update on $\mathbf{x}$ in several steps:

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{x}_t + \alpha \text{sign}(\nabla_x \mathcal{L}(f(\mathbf{x}_t), y_{\text{true}})) \\ \mathbf{x}_0 &= \mathbf{x} , \end{aligned}$$

which they call the *Basic Iterative Method*. In this method α refers to some small constant, similar to the learning rate in gradient descent.

3.3 Gaussian Processes

We derive GPify by imitating GPs, and therefore we cover the necessary prerequisites to understand their uncertainty estimation, and why GPs are intractable for learning at a larger scale. For a complete treatment, we refer to (Rasmussen and Williams, 2005).

A GP is a non-parametric learning method that can approximate arbitrary functions, given some basic assumptions of smoothness. Contrary to neural networks, and most other parametric methods, the function approximation learned by a GP is probabilistic, and thus non-deterministic (Rasmussen and Williams, 2005).

Predictions from a GP take the form of a Gaussian distribution:

$$y_{\text{pred}} \sim \mathcal{N}(\mu, \sigma) .$$

where the parameters μ and σ are defined using a mean function $m(\mathbf{x})$, and a covariance function $k(\mathbf{x}, \mathbf{x}')$. The mean function is most commonly defined to be a constant $\mathbf{0}$. While there are many different forms for the covariance function, they are all constructed around the idea that the covariance should be tied to the distance between points such that points that are closely clustered should have high covariance. To illustrate, one of the most common choices for k is the squared exponential:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\frac{1}{2}|\mathbf{x}-\mathbf{x}'|^2} .$$

Let X and Y form the GP’s training data, where $X = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ and $Y = [y_1, \dots, y_n]^\top$. In order to model aleatoric uncertainty, Y is assumed to be subject to Gaussian noise:

$$y_i \sim \mathcal{N}(y_i^{\text{true}}, \epsilon) ,$$

where ϵ is usually found by maximum likelihood estimation. Then the mean and variance for a prediction at some unseen point $\mathbf{x}^*$ is:

$$\begin{aligned} \mu &= m(\mathbf{x}^*) + k(\mathbf{x}^*, X) [k(X, X) + \epsilon^2 \mathbf{I}]^{-1} (Y - m(X)), \\ \sigma &= k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, X) [k(X, X) + \epsilon^2 \mathbf{I}]^{-1} k(X, \mathbf{x}^*) . \end{aligned}$$

Where we take $m(X)$ and $k(X, X)$ to denote element-wise and pair-wise applications of the functions $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$.

Usually, the predicted function value is taken to be μ , and the uncertainty is taken to be σ . Importantly, σ explicitly models both epistemic and aleatoric uncertainty.

Here we also want to direct the reader’s attention to the term $[k(X, X) + \epsilon^2 \mathbf{I}]^{-1}$, which is the inverse of an $n \times n$ covariance matrix. Even with algorithmic optimizations, this inversion scales in the order of $O(n^2)$ (Gardner et al., 2018), which makes GPs intractable for the quantities of data used in modern learning problems.

3.4 SIRC and the SCOD problem setting

In order to unify the evaluation of aleatoric and epistemic uncertainty measures, Kim et al. (2023) propose the unknown detection (or SCOD, in (Xia and Bouganis, 2024)) problem setting. Specifically, they do this by extending the selective classification setting (El-Yaniv and Wiener, 2010) with OOD data. For any trained classifier, the distribution made up by its training dataset constitutes its *in-distribution* (ID). Any data sample that comes from a different distribution is by definition out-of-distribution (OOD). Xia and Bouganis (2024) formalize the SCOD problem setting by constructing a two-way discrimination problem. They subdivide the ID dataset into disjoint sets $ID_\checkmark$ and $ID_\times$ containing the correctly and incorrectly classified examples respectively. AUROC scores are then calculated separately for the discriminative tasks $ID_\checkmark$ vs. $ID_\times$, and $ID_\checkmark$ vs. OOD. They argue that, in practical applications, an incorrectly classified sample incurs a similar cost as an OOD sample, and therefore it makes sense to detect both separately.

SIRC

Within the SCOD problem setting, Xia and Bouganis (2024) propose Softmax Information Retaining Combination (SIRC). SIRC is based on the idea that a softmax classifier is able to reflect aleatoric uncertainty, but not epistemic uncertainty. In other words, softmax is expected to perform well on $ID_{\checkmark}$ vs. $ID_{\times}$ discrimination, but not on $ID_{\checkmark}$ vs. OOD discrimination. Their idea with SIRC is then to combine softmax with another confidence score that performs better on $ID_{\checkmark}$ vs. OOD discrimination.

Mathematically, SIRC is defined on two scores s_1 and s_2 , where s_1 is expected to perform well on $ID_{\checkmark}$ vs. $ID_{\times}$ discrimination, and s_2 is expected to perform well on $ID_{\checkmark}$ vs. OOD detection:

$$SIRC(s_1, s_2) = -(s_1^{max} - s_1) (1 + e^{-b(s_2 - a)}) .$$

Here, s_1^{max} denotes a known upper bound for the s_1 score. a and b are parameters that change the shape and location of the function surface. Default values are

$$a = \mu_{s_2} - 3\sigma_{s_2}, \text{ and } b = \frac{1}{\sigma_{s_2}},$$

where μ_{s_2} and σ_{s_2} are the empirical mean and standard deviation of s_2 measured on ID data, either from training or validation data.

Confidence Scores used in SIRC

In their experiments, Xia and Bouganis (2024) evaluate different score combinations for s_1 and s_2 . For s_1 , they use Maximum Softmax Probability (MSP) and negative softmax entropy ($\mathcal{H}$):

$$MSP = \max_i \pi_i, \text{ and } \mathcal{H} = -\sum_i \pi_i \log \pi_i,$$

where π_i denotes the output softmax probability for class i .

All s_2 scores used in SIRC operate on feature vectors $\mathbf{z}$, where $\mathbf{z}$ is taken to be the input vector to the final hidden layer of the model. The specific scores used in SIRC are the L_1 -norm $\|\mathbf{z}\|_1$, the Residual score $\|\mathbf{z}^{P^\perp}\|_2$ (Wang et al., 2022), and the L_2 -normalized distance to the k th nearest neighbor of $\mathbf{z}$ from the training data.

3.5 Normalizing Flows

In our work we use NFs to supplement the softmax with epistemic uncertainty. NFs are a class of models, and we use a specific instance in this class, called *coupling layers* (Papamakarios et al., 2021). Here we give an introduction to NFs, focusing specifically on coupling layers. For a complete treatment of NFs, we refer to (Papamakarios et al., 2021).

When reasoning about epistemic uncertainty, we are interested in a model’s in-distribution. Specifically, this in-distribution is made up of the model’s training data $p_{\text{train}}(\mathbf{x})$. Normalizing Flows allow us to directly learn $p_{\text{train}}(\mathbf{x})$ and provides us with ways to sample and evaluate it.

A Normalizing Flow is an invertible and differentiable bijective function, also called a *diffeomorphism* (Papamakarios et al., 2021). It is often interpreted as a transformation between two vector-spaces. Given a sample $\mathbf{x} \sim p_{\text{train}}(\mathbf{x})$, a Normalizing Flow is a transformation T such that:

$$\mathbf{u} = T^{-1}(\mathbf{x}) \sim p_u(\mathbf{u}) .$$

T being a diffeomorphism allows us to directly compute the probability density at $\mathbf{x}$ (Papamakarios et al., 2021):

$$\begin{aligned} p_{\text{train}}(\mathbf{x}) &= p_u(\mathbf{u}) |\det J_T(\mathbf{u})|^{-1} \\ &= p_u(T^{-1}(\mathbf{x})) |\det J_{T^{-1}}(\mathbf{x})| \end{aligned}$$

where J_T and $J_{T^{-1}}$ denote the Jacobians of T and T^{-1} respectively. Finally, making T a learnable function allows us to define $p_u(\mathbf{u})$ freely. The most common choice is a standard Gaussian:

$$p_u(\mathbf{u}) = \mathcal{N}(0, 1) .$$

While it may be tempting to construct T as a regular neural network, such models typically do not fulfill the requirements of being both invertible and differentiable. Even in models that do, it is generally intractable to calculate the determinant of the Jacobian matrix. As such, some care must be taken to ensure that T is invertible, differentiable, and has a tractable Jacobian determinant. Coupling layers solve all three problems by splitting the input vector $\mathbf{x}$ into two disjoint parts $\mathbf{x}_1$, $\mathbf{x}_2$. This split can be constructed arbitrarily, but is commonly performed by taking $\mathbf{x}_1$ and $\mathbf{x}_2$ to be the first and second halves of $\mathbf{x}$. One part of $\mathbf{x}$ is then transformed while leaving the other fixed:

$$\mathbf{x}'_1 = \mathbf{x}_1 \circ \alpha(\mathbf{x}_2) + \beta(\mathbf{x}_2)$$

where we use $\circ$ to denote element-wise multiplication. The final output $\mathbf{x}'$ is constructed by concatenating $\mathbf{x}'_1$ and the unmodified $\mathbf{x}_2$. Constructing T in this way

results in a triangular Jacobian, for which the determinant is just a product along the diagonal:

$$|\det J_T(\mathbf{x})| = \prod_i \alpha_i$$

where α_i are the elements of $\alpha(\mathbf{x}_2)$. Powerful flow models can be constructed by applying several coupling layers in sequence, while alternating which partition of $\mathbf{x}$ is being modified.

Normalizing Flows Applied to Features

Kirichenko *et al.* (Kirichenko et al., 2020) observe that NFs underperform in terms of OOD detection when applied directly to images. Their results indicate that NFs tend to learn simple pixel-correspondences. To prevent such behaviour they propose to instead apply NFs to features extracted from a pretrained neural network as illustrated in Fig. 1 (Kirichenko et al., 2020).

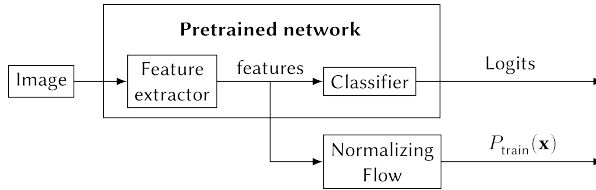


Figure 1: Illustration of how we apply NFs to features from a pretrained network. When working with images, the feature extractor will typically be a Convolutional network or a Transformer architecture.

As an added benefit, applying NFs to pretrained features allows the NF itself to be significantly smaller, which reduces both training and inference times. This structure also makes NFs a prime candidate for adding epistemic uncertainty to an existing neural network.

4 GPify: Derivation

Here we discuss the idea behind GPify through intuitive arguments with a series of increasingly complex examples. Since GPs compute an intuitive and probabilistically grounded confidence estimate, we aim to design GPify in a way that mimics the confidence estimate from a GP. We begin with a 1-dimensional 2-class toy problem, where we derive our GPify formula by comparison with a GP. Afterwards, we continue by comparing GPify to a GP on a 2-dimensional 3-class toy problem, and finally on MNIST (Lecun et al., 1998).

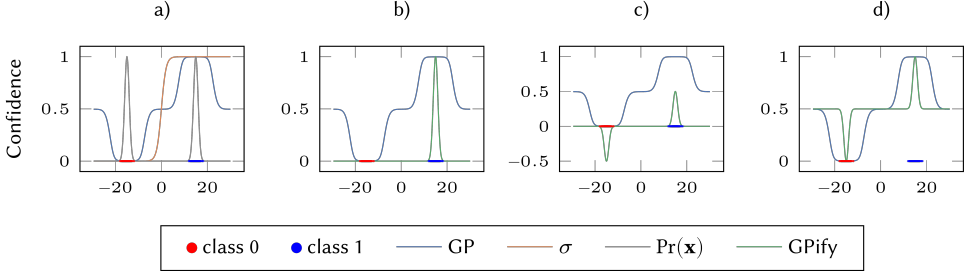


Figure 2: Various stages of our derivation of GPify applied to our 1-dimensional 2-class toy problem. These plots display each respective model’s confidence in class=1. **a)** Shows confidences from a GP along with an unmodified sigmoid, $\sigma(\mathbf{x})$, and unmodified $\Pr(\mathbf{x})$ from our NF. **b)** Shows how we can select the mode representing class=1 by $\sigma(\mathbf{x}) \cdot \Pr(\mathbf{x})$. **c)** Shows how we can represent lower confidence for class=0 by $(\sigma(\mathbf{x}) - \frac{1}{2}) \cdot \Pr(\mathbf{x})$. **d)** Shows our final GPify formula $(\sigma(\mathbf{x}) - \frac{1}{2}) \cdot \Pr(\mathbf{x}) + \frac{1}{2}$.

4.1 Turning NF densities into probabilities

By using a Normalizing Flow as our model for GPify, we already have a strong probabilistic foundation. However, by default our NF outputs a log-likelihood $\ln p_{\text{train}}(\mathbf{x})$, which is unbounded and difficult to interpret for an end-user. Even if we remove the logarithm, $p_{\text{train}}(\mathbf{x})$ represents a probability density, which is still unintuitive. Instead, we aim to convert the log-likelihood directly into a probability by the means of an empirical distribution. Given a set of ID samples $S = \{\mathbf{x}_i \mid i = 1 \dots n\}$ we compute:

$$\Pr(\mathbf{x}) = \frac{|\{\mathbf{x}_i \mid \ln p_{\text{train}}(\mathbf{x}_i) \leq \ln p_{\text{train}}(\mathbf{x}), \mathbf{x}_i \in S\}|}{|S|}$$

where $|S|$ denotes the size of the dataset. While we use the notation $\Pr(\mathbf{x})$, we should note that this is not strictly the probability of the element $\mathbf{x}$, but rather the probability that $\mathbf{x} \sim p_{\text{train}}$. It is also noteworthy that $\Pr(\mathbf{x})$ exactly corresponds to the false-negative-rate as used in OOD detection literature (Yang et al., 2024). When constructing this empirical distribution, it is valid to choose S to be the training data for the NF. However, since the NF learns the data distribution directly, it is also possible to construct S by sampling from the NF, which is especially useful when training data is sparse. In our experiments we evaluate both methods of constructing S .

4.2 1D Toy Problem

Fig. 2 illustrates the stages in our intuitive process when designing GPify. All 4 plots display each respective model’s confidence $P(\text{class} = 1)$. Since the NF alone has no notion of classes, we plot the empirical probability $\Pr(\mathbf{x})$ in a). We also use a sigmoid here instead of softmax, since it is a 2-class problem.

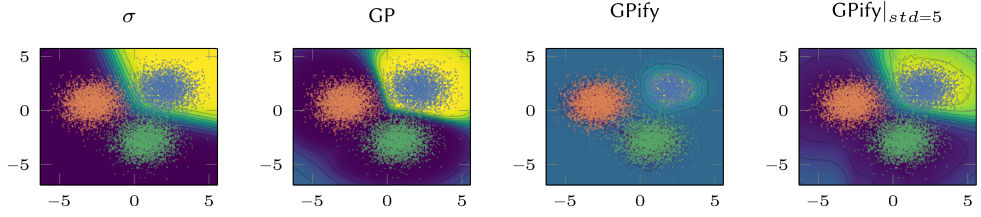


Figure 3: Our 2-dimensional 3-class problem. Each contour plot shows the confidence of $class = 0$ for its respective model. The scatter points represent the 3 classes: blue=0, red=1, green=2. $GPify|_{std=5}$ refers to GPify where the empirical distribution is sampled from $\mathcal{N}(0, 5)$.

Fig. 2 a) leads to the observation that we can effectively select a mode in $\Pr(\mathbf{x})$ by multiplying it with the sigmoid, as illustrated in Fig. 2 b). By shifting the sigmoid, we recover the $class = 0$ mode with negative confidence:

$$\left(\sigma(\mathbf{x}) - \frac{1}{2} \right) \cdot \Pr(\mathbf{x}) ,$$

as illustrated in Fig. 2 c). Shifting the resulting function by $\frac{1}{2}$ removes any negative confidence values:

$$GPify(\mathbf{x}) = \left(\sigma(\mathbf{x}) - \frac{1}{2} \right) \cdot \Pr(\mathbf{x}) + \frac{1}{2} .$$

As can be seen in Fig. 2 d), this construction gives us a result that is similar to the GP. The primary difference between GPify and the GP is that GPify produces a narrower probability mass. GPify successfully reproduces both the base uncertainty-level of $\frac{1}{2}$, and the probability of $< \frac{1}{2}$ for the mode representing $class = 0$.

4.3 2D toy problem

In this section we extend GPify to multi-class problems by a simple change in the formula. For a problem with C classes, the base uncertainty-level should be $\frac{1}{C}$. As such we adjust GPify to the following:

$$GPify(\mathbf{x}) = \left(MSP(\mathbf{x}) - \frac{1}{C} \right) \cdot \Pr(\mathbf{x}) + \frac{1}{C} ,$$

Where MSP refers to the maximum softmax probability as defined in section 3.4. Fig. 3 shows contour plots of each model's confidence for $class = 0$. The leftmost plot shows softmax, and we observe that it can be used to select a mode in the NF, just like the sigmoid in a single dimension. It is immediately clear that the general structure of GPify is identical to the GP. However, as in the 1-dimensional case, GPify produces a narrower probability mass.

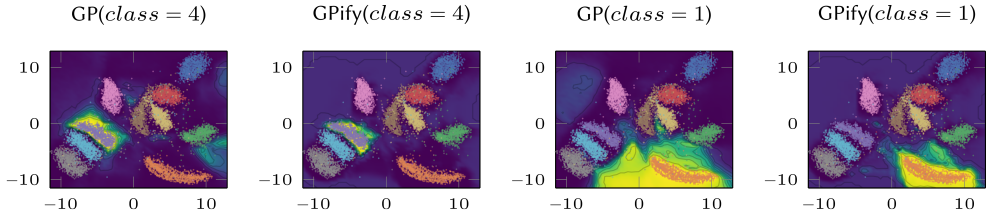


Figure 4: Contour plots from MNIST, mapped into the 2D plane using UMAP. Each color of the scatter points indicate a digit class. The two leftmost plots display their respective model’s confidence in $class = 4$, and the two rightmost display confidence for $class = 1$.

When sampling S from the NF in order to create our empirical distribution $\Pr(\mathbf{x})$, we sample from a standard normal $\mathcal{N}(0, 1)$, since that is the distribution we choose for $p_u(\mathbf{u})$ when we train our NF. If we instead sample S using a larger standard deviation, it will contain more low-probability samples, and thus produce a less narrow empirical probability mass. In the rightmost plot of Fig. 3 we plot GPify with an empirical distribution sampled from $\mathcal{N}(0, 5)$. We include this example to further highlight the similarity to the GP.

4.4 MNIST

Our purpose with this section is to apply GPify to a larger classification problem to investigate if its properties scale to larger problems. Since MNIST is a significantly larger problem compared to our toy problems, it is not feasible to use the entire training set for the GP. As such, we randomly sample 8192 images from the training set to use for the GP. Using this sub-sampled training set, our GP gets $\sim 95\%$ test accuracy, and as such we consider it a fair comparison. However, for larger and more complex datasets GPs do not scale. Even for CIFAR-10 (Krizhevsky et al., 2009), which is a small dataset by modern standards, GPs perform significantly worse than neural network-based models (van der Wilk et al., 2017; Blomqvist et al., 2020).

Another reason we choose MNIST for this section is because it is small and simple enough to be visualized in 2D using UMAP (McInnes et al., 2020). In order to produce 2D contour plots, like Fig. 3, we train a parametric UMAP (Sainburg et al., 2021) autoencoder, which allows us to map MNIST to and from the 2D plane.

Fig. 4 shows contour plots representing confidences in two different MNIST digit classes. We choose to plot confidences for $class = 4$ and $class = 1$ to highlight the fact that both models appear to produce sharp decision boundaries where data is tightly clustered. Again, we observe for $class = 1$ that GPify produces a narrower probability mass compared to the GP. However, in contrast to the previous two ex-

amples, here it looks like the GP’s probability mass includes areas outside the data regions, which makes GPify’s narrow distribution appear more conservative.

We would like to stress that UMAP is an approximation, and therefore it is impossible to draw definitive conclusions from it. In spite of that, we believe this UMAP projection serves as a good indication that our GPify confidence formula retains its desirable properties, and compares well to the GP even for larger, higher-dimensional problems.

4.5 Interpretation

In accordance with the view of uncertainties representing probability distributions, we interpret our resulting GPify score as a joint probability. While the softmax computes its confidence in the form of a probability $P(class = i \mid \mathbf{x})$, that probability is only valid if $\mathbf{x} \sim p_{\text{train}}(\mathbf{x})$. As such, we interpret GPify as a joint probability $P(class = 1, \mathbf{x} \sim p_{\text{train}}(\mathbf{x}) \mid \mathbf{x})$.

Our main field of study is robotics, where we tend to view behaviours as a sequence of operations. In this view, classifying an image is made up of two operations: capturing an image, and then classifying it. Thus, softmax gives us the probability that its classification is correct, and the NF gives us the probability that the captured image is valid. Finally GPify gives us the joint probability that both operations were executed successfully.

5 SCOD Evaluation

As we have seen in previous sections, GPify is an intuitive confidence score that encompasses both aleatoric and epistemic uncertainty in a way that is similar to a GP. We have also seen that GPify appears to scale to larger problems beyond our toy examples. In this section, we complement our qualitative results with a quantitative evaluation within the SCOD (Xia and Bouganis, 2024) framework.

For our experiments we construct ID and OOD data by collecting subsets of ImageNet1K (Russakovsky et al., 2015). We select 150 random classes without replacement from ImageNet1K, and use 100 classes (127,478 train and 5000 val images) for our ID data, and the remaining 50 classes (2500 val images) as our OOD data. This partitioning follows Xia and Bouganis (2024) in that OOD data is disjoint from ID data, and as such any prediction on an OOD sample should be considered invalid by default.

In addition to evaluating $ID_{\checkmark}$ vs. $ID_{\times}$, and $ID_{\checkmark}$ vs. OOD discrimination, we also evaluate ID vs. OOD discrimination. We agree with Xia and Bouganis (2024) that it is desirable for the combined score to retain the softmax’s capabilities in $ID_{\checkmark}$ vs. $ID_{\times}$ discrimination, and by the same reasoning we believe it is also important for the combined score to retain the NF’s capabilities for ID vs. OOD discrimination.

5.1 Backbone Networks

Since the NF in our GPify score is applied to features extracted from a trained network, we are interested in its generalizability across different network architectures. Kirichenko et al. (2020), as well as our previous work (Kristoffersson Lind et al., 2023; Lind et al., 2024), has shown that NFs work well for epistemic uncertainty when paired with convolutional architectures. However, while there are works to integrate attention-mechanisms into NFs (Ho et al., 2019), we are not aware of any work that applies an NF to features from an attention-based architecture. Therefore, we conduct our SCOD experiment using 4 different backbone networks, both convolutional and attention-based: ResNet34, EfficientNet-B0, ViT-B16, and Swin-B. We refrain from evaluating other backbone architectures, as these are the most commonly used in image processing.

Each backbone is trained for 300 epochs with MixUp ($\alpha = 0.2$) using the Adam (Kingma and Ba, 2014) optimizer. Both convolutional architectures, ResNet34 and EfficientNet-B0, are trained with a cosine annealing scheduler (Loshchilov and Hutter, 2017), a starting learning rate of 0.008, and a batch size of 256. For the transformer architectures, ViT-B16 and Swin-B, we found better results using a fixed learn rate of 0.0001, and a batch size of 128.

5.2 Normalizing Flow

In our preliminary testing, the NF consistently produced its best results when trained on the validation set for its corresponding backbone model. We believe that applying the backbone to its own training data produces a skewed distribution of features compared to examples that were not seen during training, perhaps due to some amount of overfitting. More tests are required to verify this hypothesis. Nonetheless, we exclude the backbone’s training set when training our NF, and only use it for evaluation, to avoid overfitting.

Ideally, we would run our SCOD evaluation using the ImageNet1K test set while reserving the entire validation set for our NF training. However, we do not have

access to labels for the ImageNet1K test set. Therefore, we further subdivide the ID validation set and use 4000 samples to train our NF, and reserve 1000 samples for the final SCOD evaluation.

Our NF architecture is made up of 10 sequential coupling layers, operating on alternating halves of the input vector. $\alpha(\mathbf{x})$ and $\beta(\mathbf{x})$ are each made up of a 2-layer neural network, with a hidden layer size of 512. The same architecture is used for all 4 backbones.

We train our NFs for 300 epochs using the Adam (Kingma and Ba, 2014) optimizer and a cosine annealing scheduler (Loshchilov and Hutter, 2017), with a starting learning rate of 0.001, a batch size of 256, and an L_2 weight decay factor of 0.01.

5.3 Constructing the Empirical Distribution

As discussed in section 4.1, the set S used when creating our empirical distribution can either be taken directly from the NF’s training data, or it can be sampled from the NF itself. In section 4.3, we briefly mentioned that changing the standard deviation when sampling the NF changes the resulting empirical distribution. The first method we evaluate sets S to be equal to the NF’s training set, and we denote this by $\text{GPify}|_{\text{train}}$. We also evaluate two strategies of sampling S from the NF itself, one where we use a fixed standard deviation of 1, and one where we optimize the standard deviation based on the $\text{ID}_\sqrt{\cdot}$ vs. $\text{ID}_\times$ discrimination performance on the NF’s training set. These are denoted by $\text{GPify}|_{\text{std}=\dots}$.

5.4 Other Scores

For comparison with GPify, we evaluate all the originally proposed variants of SIRC (Xia and Bouganis, 2024), along with its constituent components: Maximum Softmax Probability (MSP), entropy ($\mathcal{H}$), $\|\mathbf{z}\|_1$ (L_1), Residual $\|\mathbf{z}^{P^\perp}\|_2$ (Wang et al., 2022) (Res), and L_2 -normalized distance to the k th nearest ID neighbor (KNN). Like Xia and Bouganis (2024), we set $k = 10$ for KNN. With the Residual score, we compute the principal subspace based on $\frac{1}{4}$ of the dimensions in the feature vector.

We also compare with GradNorm and ViM, as well as the Energy score over logits (Liu et al., 2020): $-\log \sum_i \exp(l_i)$, since it is a component in ViM (Wang et al., 2022).

Additionally, we compare with the plain likelihood from our NFs, to verify that GPify does not significantly reduce the efficacy for ID vs. OOD discrimination. We also evaluate a SIRC combination using the same components as GPify (s_1 =MSP and s_2 =NF).

5.5 Results and Discussion

Tables 1, 2, 3, and 4 show AUROC scores for each test in the SCOD framework. Each table corresponds to one backbone model.

Table 1: AUROC scores for the ResNet34 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	ID _✓ vs. ID _×	ID _✓ vs. OOD	ID vs. OOD
MSP	89.73%	86.53%	79.89%
$\mathcal{H}$	87.98%	85.50%	79.35%
L_1	83.67%	83.97%	78.46%
Res	81.76%	82.37%	77.30%
KNN	25.00%	21.19%	24.89%
NF	80.94%	86.99%	82.65%
Energy	86.88%	86.34%	80.39%
ViM	85.89%	85.71%	79.94%
GradNorm	26.07%	24.88%	28.89%
SIRC(MSP, L_1)	90.16%	87.19%	80.43%
SIRC(MSP, Res)	90.00%	87.12%	80.47%
SIRC(MSP, KNN)	89.61%	86.11%	79.47%
SIRC($-\mathcal{H}$, L_1)	88.95%	86.63%	80.21%
SIRC($-\mathcal{H}$, Res)	88.76%	86.58%	80.31%
SIRC($-\mathcal{H}$, KNN)	87.64%	84.84%	78.72%
SIRC(MSP, NF)	89.92%	87.47%	81.04%
GPify_{train}	86.45%	89.11%	83.90%
GPify_{std=1.0}	<u>90.37%</u>	<u>94.39%</u>	<u>93.66%</u>
GPify_{std=0.96}	90.49%	94.54%	93.87%

Table 2: AUROC scores for the EfficientNet-B0 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	ID _✓ vs. ID _×	ID _✓ vs. OOD	ID vs. OOD
MSP	<u>87.58%</u>	<u>83.82%</u>	75.87%
$\mathcal{H}$	85.04%	83.27%	76.30%
L_1	67.61%	66.37%	66.70%
Res	45.65%	41.92%	42.76%
KNN	62.54%	67.27%	64.90%
NF	82.15%	82.21%	75.83%
Energy	72.76%	74.26%	69.62%
ViM	67.34%	68.91%	65.36%
GradNorm	81.71%	81.77%	75.61%
SIRC(MSP, L_1)	87.89%	<u>83.88%</u>	75.72%
SIRC(MSP, Res)	<u>87.70%</u>	<u>83.82%</u>	75.72%
SIRC(MSP, KNN)	<u>87.48%</u>	84.30%	<u>76.67%</u>
SIRC($-\mathcal{H}$, L_1)	85.46%	83.42%	76.27%
SIRC($-\mathcal{H}$, Res)	85.15%	83.27%	76.20%
SIRC($-\mathcal{H}$, KNN)	85.10%	<u>83.93%</u>	77.22%
SIRC(MSP, NF)	<u>87.83%</u>	<u>83.67%</u>	75.53%
GPify_{train}	84.79%	83.24%	76.25%
GPify_{std=1.0}	<u>87.71%</u>	<u>83.78%</u>	75.80%
GPify_{std=1.11}	<u>87.80%</u>	<u>83.80%</u>	75.78%

Table 3: AUROC scores for the ViT-B16 model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	ID _✓ vs. ID _×	ID _✓ vs. OOD	ID vs. OOD
MSP	83.27%	80.76%	72.53%
$\mathcal{H}$	83.01%	80.56%	72.53%
L_1	81.49%	81.23%	73.30%
Res	86.03%	85.38%	76.20%
KNN	72.66%	69.35%	63.75%
NF	85.90%	85.43%	76.41%
Energy	80.94%	77.87%	70.36%
ViM	<u>86.58%</u>	<u>85.65%</u>	76.35%
GradNorm	70.91%	67.73%	62.90%
SIRC(MSP, L_1)	83.94%	81.55%	73.12%
SIRC(MSP, Res)	83.87%	81.52%	73.08%
SIRC(MSP, KNN)	83.70%	81.05%	72.70%
SIRC($-\mathcal{H}$, L_1)	83.79%	81.58%	73.26%
SIRC($-\mathcal{H}$, Res)	83.79%	81.52%	73.17%
SIRC($-\mathcal{H}$, KNN)	83.47%	80.88%	72.65%
SIRC(MSP, NF)	84.41%	82.32%	73.76%
GPify_{train}	87.08%	86.35%	78.63%
GPify_{std=1.0}	86.16%	84.85%	75.90%
GPify_{std=1.01}	86.16%	84.85%	75.90%

Table 4: AUROC scores for the Swin-B model. **Bold** indicates the best score, and underline indicates scores that are less than 1% below the best.

	ID _✓ vs. ID _×	ID _✓ vs. OOD	ID vs. OOD
MSP	<u>90.90%</u>	86.96%	81.82%
$\mathcal{H}$	90.29%	86.68%	81.72%
L_1	80.61%	80.94%	76.96%
Res	86.95%	86.78%	82.16%
KNN	78.54%	75.38%	71.91%
NF	86.42%	86.05%	81.53%
Energy	88.00%	84.54%	79.92%
ViM	88.26%	<u>87.76%</u>	82.95%
GradNorm	65.44%	64.07%	62.24%
SIRC(MSP, L_1)	91.29%	<u>87.63%</u>	82.35%
SIRC(MSP, Res)	<u>91.26%</u>	<u>87.73%</u>	82.51%
SIRC(MSP, KNN)	<u>91.16%</u>	87.20%	82.02%
SIRC($-\mathcal{H}$, L_1)	<u>90.61%</u>	<u>87.47%</u>	82.34%
SIRC($-\mathcal{H}$, Res)	<u>90.65%</u>	<u>87.63%</u>	82.55%
SIRC($-\mathcal{H}$, KNN)	<u>90.58%</u>	87.00%	81.96%
SIRC(MSP, NF)	<u>91.27%</u>	<u>87.94%</u>	82.72%
GPify_{train}	88.10%	87.10%	82.54%
GPify_{std=1.0}	88.47%	<u>87.85%</u>	84.63%
GPify_{std=2.33}	<u>90.52%</u>	88.34%	83.26%

Our first and most important observation is that GPify performs competitively with SIRC, even outperforming SIRC in 8 out of the 12 AUROC tests. Additionally, GPify appears to retain the discriminative performance of both the softmax and the NF. We observe no cases where ID_✓ vs. OOD or ID vs. OOD detection performance is reduced by $> 1\%$ compared to the standalone softmax or NF. The only cases where we do observe a decrease in performance of $> 1\%$ are for ID_✓ vs. ID_×, where GPify_{|train} reduces performance in 3 out of the 4 backbones, and GPify_{|std=1} reduces performance for Swin. As such we recommend to optimize the sample standard deviation to maximize ID_✓ vs. ID_× on the NF’s training set, since that appears to result in the best performance overall. However, with 3 of our 4 backbones, the optimized standard deviation is close to 1, and thus it appears that sampling with $std = 1$ is a good default choice.

6 Adversarial Detection

Here we reuse the same models, and datasets from our SCOD evaluation, and we create adversarial examples using the basic iterative method.

For each classifier, we create adversarial examples based on all the ID images in our test set. In order to evaluate the performance at different levels of severity, we save multiple adversarial examples throughout the iterative method. First, we save one adversarial example as soon as the image is misclassified, which is similar to the original idea of finding the minimum perturbation required for misclassification. We also save adversarial examples when the misclassification confidence (MSP) reaches 90%, 95%, and 99%.

Table 5: AUROC scores for adversarial detection.

Min Diff refers to the adversarial image created using the minimum number of iterations for misclassification.
> x% refers to images where the Softmax confidence is $> x\%$ for the adversarial class.

	ResNet	EffNet	ViT	Swin
Min Diff	96.04%	95.79%	89.45%	79.01%
> 90%	97.75%	23.59%	79.64%	80.26%
> 95%	97.81%	13.65%	74.84%	83.61%
> 99%	98.81%	1.89%	80.43%	92.63%

Table 5 summarizes AUROC scores for each model, and each misclassification confidence threshold. Based on these results, we can quickly conclude that GPify does have the capability of detecting adversarial examples. Since these misclassifications are constructed based on the classifier’s MSP, it follows that they cannot be detected using MSP alone. Hence, these results show that our $\text{Pr}(\mathbf{x})$ from the NF is sensitive enough for GPify to detect these adversarial examples in spite of their high MSP values.

It should be noted, however, from this simple experiment, we cannot draw any conclusions as to how well GPify compares against dedicated adversarial detectors (Aldahdooh et al., 2022). Additionally, the efficacy seems to differ between the different backbone models, especially EfficientNet. As such, we call for future work to more closely investigate differences between architectures, and a thorough comparison to dedicated adversarial detection methods.

7 Future Work

As indicated in section 3.1, the notion of epistemic uncertainty is closely tied to the problem of OOD detection. In our work, we have assumed only the availability of in-distribution data, which is a common assumption in OOD detection literature (Yang et al., 2024), mainly due to the inherent difficulty of data collection within some domains. Consider, for example, human intent recognition (Jain and Argall, 2019), where the inherent inconsistency in human behavior makes OOD data sampling impractical, or terrain traversability classification (Sevastopoulos and Konstantopoulos, 2022), where the sheer variability of outdoor terrain makes it difficult to define OOD. For domains where OOD data is available, however, supervised learning methods can be used to explicitly reject OOD samples, which often leads to improved performance (Yang et al., 2024). It would thus be an interesting research direction to evaluate supervised epistemic uncertainties within the SCOD setting, for example using a supervised NF (Izmailov et al., 2020).

Additionally, Since both SIRC and GPify rely heavily on the backbone model for estimating aleatoric uncertainty, it would be interesting to evaluate their performance on long-tailed datasets. For such datasets, the aleatoric uncertainty from softmax might be poorly calibrated due to data sparsity among tail classes. We hypothesize that the NF in our GPify score would recognize tail classes as low-density regions and thus lower the final confidence estimate, but this remains to be empirically confirmed.

Finally, in section 5.2 we posed a hypothesis that features extracted from the backbone’s training set is not representative of the broader dataset. First and foremost, more experiments are needed to thoroughly investigate this hypothesis. Second, it is commonly accepted within learning literature that more training data leads to better results. As such, the backbone’s training set is a source of untapped potential for improving the NF, and it would therefore be interesting to see if it can be effectively integrated into the NF’s training.

8 Conclusion

In this paper, we have proposed GPify, which is a combined confidence score based on softmax and normalizing flows. We have derived GPify mostly based on intuitive reasoning, with the goal making it an intuitive and probabilistically founded confidence score similar to that of a Gaussian process. As a direct improvement over softmax, we show that GPify has capabilities for detecting adversarial examples. Our quantitative experiments show that GPify performs competitively with SIRC, and preserves the properties of both the softmax and normalizing flow. We believe that the main advantage to GPify over a Gaussian process is its scalability, where it can scale to arbitrarily large problems thanks to its neural network-based implementation. Its main advantage over SIRC is its probabilistic foundation, from the normalizing flow, as well as the fact that it is intuitive and interpretable by design.

Acknowledgements

This research is funded by the Excellence Center at Linköping-Lund in Information Technology (ELLIIT), and the Wallenberg AI, Autonomous Systems and Software Program (WASP). Computations for this publication were enabled by the supercomputing resource Berzelius provided by the National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

Competing interests

The authors declare no competing interests.

Data Availability

All data used in this work is publicly available under the ImageNet dataset (Russakovsky et al., 2015).

Bibliography

- Aldahdooh, A., Hamidouche, W., Fezza, S.A., Déforges, O.: Adversarial example detection for dnn models: a review and experimental comparison. *Artificial Intelligence Review* 55(6), 4403–4462 (2022) <https://doi.org/10.1007/s10462-021-10125-w>
- Blomqvist, K., Kaski, S., Heinonen, M.: Deep convolutional gaussian processes. In: Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., Robardet, C. (eds.) *Machine Learning and Knowledge Discovery in Databases*, pp. 582–597. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46147-8_35
- El-Yaniv, R., Wiener, Y.: On the foundations of noise-free selective classification. *Journal of Machine Learning Research* 11(53), 1605–1641 (2010)
- Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17*, pp. 1321–1330. JMLR.org, Sydney, NSW, Australia (2017)
- Gardner, J., Pleiss, G., Weinberger, K.Q., Bindel, D., Wilson, A.G.: Gpytorch: Black-box matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems* 31 (2018)
- Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (arXiv:1412.6572) (2015) <https://doi.org/10.48550/arXiv.1412.6572>. arXiv:1412.6572 [stat]
- Grimmett, H., Triebel, R., Paul, R., Posner, I.: Introspective classification for robot perception. *The International Journal of Robotics Research* 35(7), 743–762 (2016) <https://doi.org/10.1177/0278364915587924>
- Ho, J., Chen, X., Srinivas, A., Duan, Y., Abbeel, P.: Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design (2019). <https://arxiv.org/abs/1902.00275>
- Huang, R., Geng, A., Li, Y.: On the importance of gradients for detecting distributional shifts in the wild. In: *Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21*, pp. 677–689. Curran Associates Inc., Red Hook, NY, USA (2024)
- Izmailov, P., Kirichenko, P., Finzi, M., Wilson, A.G.: Semi-supervised learning with normalizing flows. In: Ill, H.D., Singh, A. (eds.) *Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine*

- Learning Research, vol. 119, pp. 4615–4630. PMLR, (2020). <https://proceedings.mlr.press/v119/izmailov20a.html>
- Jain, S., Argall, B.: Probabilistic human intent recognition for shared autonomy in assistive robotics. *J. Hum.-Robot Interact.* **9**(1), 2–1223 (2019) <https://doi.org/10.1145/3359614>
- Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. *arXiv* (2014). <https://arxiv.org/abs/1412.6980>
- Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world (arXiv:1607.02533) (2017) <https://doi.org/10.48550/arXiv.1607.02533> . arXiv:1607.02533 [cs]
- Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
- Kirichenko, P., Izmailov, P., Wilson, A.G.: Why Normalizing Flows Fail to Detect Out-of-Distribution Data. In: *Advances in Neural Information Processing Systems*, vol. 33, pp. 20578–20589. Curran Associates, Inc., (2020)
- Kim, J., Koo, J., Hwang, S.: A unified benchmark for the unknown detection capability of deep neural networks. *Expert Systems with Applications* **229**, 120461 (2023) <https://doi.org/10.1016/j.eswa.2023.120461> . arXiv:2112.00337 [cs]
- Kristoffersson Lind, S., Triebel, R., Nardi, L., Krueger, V.: Out-of-distribution detection for adaptive computer vision. In: Gade, R., Felsberg, M., Kämäräinen, J.-K. (eds.) *Image Analysis. Lecture Notes in Computer Science*, pp. 311–325. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-31438-4_21
- Kristoffersson Lind, S., Xiong, Z., Forssén, P.-E., Krüger, V.: Uncertainty quantification metrics for deep regression. *Pattern Recognition Letters* **186**, 91–97 (2024) <https://doi.org/10.1016/j.patrec.2024.09.011>
- Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324 (1998) <https://doi.org/10.1109/5.726791>
- Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts (arXiv:1608.03983) (2017) <https://doi.org/10.48550/arXiv.1608.03983> . arXiv:1608.03983 [cs]
- Lind, S.K., Triebel, R., Krüger, V.: Making the flow glow – robot perception under severe lighting conditions using normalizing flow gradients

- (arXiv:2412.07565) (2024) <https://doi.org/10.48550/arXiv.2412.07565> . arXiv:2412.07565 [cs]
- Liu, W., Wang, X., Owens, J., Li, Y.: Energy-based out-of-distribution detection. In: *Advances in Neural Information Processing Systems*, vol. 33, pp. 21464–21475. Curran Associates, Inc., (2020). <https://proceedings.neurips.cc/paper/2020/hash/f5496252609c43eb8a3d147ab9b9c006-Abstract.html>
- McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (arXiv:1802.03426) (2020) <https://doi.org/10.48550/arXiv.1802.03426> . arXiv:1802.03426 [stat]
- Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research* **22**(57), 1–64 (2021)
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115**(3), 211–252 (2015) <https://doi.org/10.1007/s11263-015-0816-y>
- Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge (2005). <https://doi.org/10.7551/mitpress/3206.001.0001>
- Sevastopoulos, C., Konstantopoulos, S.: A survey of traversability estimation for mobile robots. *IEEE Access* **10**, 96331–96347 (2022) <https://doi.org/10.1109/ACCESS.2022.3202545>
- Sainburg, T., McInnes, L., Gentner, T.Q.: Parametric umap embeddings for representation and semisupervised learning. *Neural Computation* **33**(11), 2881–2907 (2021)
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (arXiv:1312.6199) (2014) <https://doi.org/10.48550/arXiv.1312.6199> . arXiv:1312.6199 [cs]
- Wilk, M., Rasmussen, C.E., Hensman, J.: Convolutional gaussian processes. In: *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., New York (2017). https://papers.nips.cc/paper_files/paper/2017/hash/1c54985e4f95b7819ca0357c0cb9a09f-Abstract.html
- Wenger, J., Kjellström, H., Triebel, R.: Non-parametric calibration for classification. In: Chiappa, S., Calandra, R. (eds.) *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, vol. 108, pp. 178–190. PMLR, (2020). <https://proceedings.mlr.press/v108/wenger20a.html>

- Wang, H., Li, Z., Feng, L., Zhang, W.: Vim: Out-of-distribution with virtual-logit matching. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4911–4920 (2022). <https://doi.org/10.1109/CVPR52688.2022.00487>. <https://ieeexplore.ieee.org/document/9879414>
- Xia, G., Bouganis, C.-S.: Augmenting the softmax with additional confidence scores for improved selective classification with out-of-distribution data. *International Journal of Computer Vision* **132**(9), 3714–3752 (2024) <https://doi.org/10.1007/s11263-024-02029-3>
- Yang, J., Zhou, K., Li, Y., Liu, Z.: Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision* **132**(12), 5635–5662 (2024) <https://doi.org/10.1007/s11263-024-02117-4>

Paper V



Normalizing Flows are Capable Bi-manual Visuomotor Policy Models

Jialong Li*

Lund University LTH

jialong.li@cs.lth.se

Simon Kristoffersson Lind*

Lund University LTH

simon.kristoffersson_lind@cs.lth.se

Wenrui Xie

Lund University LTH

we5415xi-s@student.lu.se

Maj Stenmark

Lund University LTH

maj.stenmark@dlr.de

Volker Krüger

Lund University LTH

volker.krueger@cs.lth.se

*: Equal contribution

Abstract

The field of general-purpose robotics has recently embraced powerful probabilistic diffusion-based models to learn the complex embodiment behaviours. However, existing models often come with significant trade-offs, namely high computational costs for inference and a fundamental inability to quantify output uncertainty.

We introduce *Normalizing Flows Policy* (NF-P), a conditional normalizing flow-based visuomotor policy for bi-manual manipulation. NF-P learns a conditional density over action sequences and enables single-pass generative sampling with tractable likelihood computation. Using this property, we propose two inference-time optimization strategies: Stochastic Batch Selection, which selects the highest-likelihood trajectory among sampled candidates, and Gradient Refinement, which directly ascends the log-likelihood to improve action quality.

In both simulation and real robot experiments, NF-P achieves promising success rates compared to the baseline. In addition to improved task performance, NF-P demonstrates faster training and lower inference latency. These results establish normalizing flows as a competitive and computationally efficient visuomotor policy, particularly for real-time, uncertainty-aware robotic control.

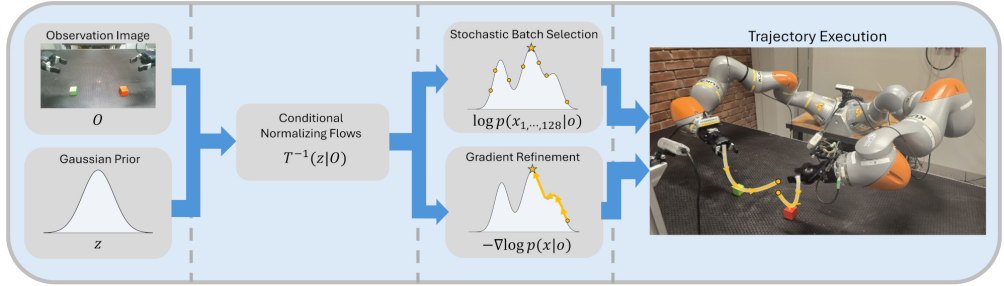


Figure 1: Overview of the Normalizing Flow Policy. During training, the conditional normalizing flow learns a bijective mapping from the complex manifold of robot actions to a simple Gaussian prior $\mathbf{z}$, conditioned on visual observations $\mathbf{o}$. At inference, the learned transformation is used inversely as $T^{-1}(\mathbf{z}|\mathbf{o})$ to generate actions. A key advantage of normalizing flows is the ability to query the exact likelihood of generated actions, enabling two efficient inference optimisation strategies: 1) *Stochastic Batch Selection*: generate a batch of candidates ($\mathbf{x}_{1...N}$) and select the sample with the highest likelihood, and 2) *Gradient Refinement*: refine a proposal trajectory by ascending the exact log-likelihood gradient $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{o})$ for a small number of steps. The optimised trajectory is then executed by the robot.

1 Introduction

Learning from demonstrations has emerged as a promising paradigm for enabling robots to perform diverse manipulation tasks without extensive programming. Recent approaches rely on powerful generative models such as diffusion models [1, 2] to represent multi-modal distributions over action sequences conditioned on visual observations. While diffusion models offer impressive expressiveness, they require expensive multi-step denoising during inference and do not provide tractable estimates of output likelihood. Both limitations hamper their deployment on resource-constrained hardware and preclude a principled measure of confidence.

In this paper we argue that *normalizing flows* (NFs) [3] offer an attractive alternative to existing RGB-based visuomotor policy learning models. NFs learn a bijective, differentiable transformation that maps complex action distributions to a simple latent prior and vice versa. This bijection enables efficient sampling via a single forward pass and allows exact likelihood computation, enabling the policy to quantify uncertainty and optimise generated trajectories. We present *Normalizing Flow Policy* (NF-P), and demonstrate that it can match or exceed the performance of diffusion-based visuomotor policies while offering faster training, inference, and explicit confidence measures. NF-P brings together three advantages that, to the best of our knowledge, have not previously been demonstrated in previous robotic policy: (i) exact likelihood computation for principled uncertainty estimation, (ii) single-pass generative sampling that enables real-time inference, and (iii) the capacity to model complex distributions over high-dimensional bimanual action spaces. This combination allows NF-P to provide qualitatively new capabilities such as uncertainty-guided action optimisation and safe exploration on real hardware.

Our contributions are threefold:

- We propose NF-P, a conditional normalizing flow architecture tailored for RGB-based visuomotor policy learning, including two novel design choices: stride data sampling during training, and likelihood optimization over inference.
- We evaluate NF-P across a wide range of simulated bimanual tasks in the RoboTwin 2.0 framework [4] as well as on real robot demonstrations. NF-P consistently achieves competitive or superior success rates compared to the baseline diffusion policy and exhibits strong data efficiency.
- We demonstrate how NF-P scales with the size of the dataset, demonstrating that NF-P has good scaling properties.
- We conduct an ablation study investigating factors that improve the performance of NF-P: stride-based sequencing, likelihood-based output optimisation, and sampling variance.

Taken together, we show that normalizing flows are highly capable for visuomotor policy learning, while offering a substantial advantage over diffusion-based policies in the form of efficient training and inference, and an explicit uncertainty estimate.

2 Related Work

2.1 Vision-based Imitation Learning

Vision-based imitation learning aims to leverage the ubiquitous nature and high information density of visual data to map observations to robotic actions. Within this paradigm, Transformer-based regression architectures [5, 6, 7, 8] represent a dominant methodology, utilizing self-attention mechanisms to perform direct or autoregressive action prediction. These models benefit from the Transformer’s ability to capture long-range temporal dependencies across sequences of visual tokens, making them highly effective for structured, predictable tasks. However, these deterministic or quasi-deterministic frameworks often collapse when encountering multi-modal data distributions [9]. Because they typically minimize a mean squared error (MSE) loss, they tend to regress toward the mean of conflicting demonstrations, rather than capturing distinct, valid behavioral modes. Furthermore, they lack inherent probabilistic mechanisms for failure recovery or uncertainty quantification, which limits their adaptability in non-deterministic environments. This fundamental inability to represent complex, non-Gaussian action distributions necessitates a shift from point-estimation models toward more expressive generative frameworks.

2.2 Diffusion-based Policy

To resolve the limitations of traditional regression, the Diffusion Policy framework [1] and broader diffusion-based generative models have been integrated into the control loop. These approaches shift the objective toward modeling a comprehensive conditional distribution over the action space, allowing the agent to represent multiple valid trajectories simultaneously. This framework has inspired numerous extensions. For instance, Hierarchical Diffusion Policy [10] combines high-level diffusion planning with low-level control, while Causal Diffusion Policy [11] incorporates causal transformers to refine temporal modeling. Despite their expressive power, these models face a significant trade-off in computational efficiency. While several works aim to reduce inference costs through one-step distillation [12], real-time iterative diffusion [13], or point-cloud-based optimizations [14], they inherit the underlying challenge of unquantified uncertainty and lack exact likelihood estimations.

Notably, recent advancements have introduced flow-matching vision-language-action (VLA) models such as π_0 and $\pi_{0.5}$ [15, 16]. Although these models utilize flow-matching to achieve state-of-the-art performance in robotic manipulation, they are fundamentally designed as large-scale VLAs that rely on massive pre-trained foundation models. Given that this work prioritizes efficiency, such large-scale VLA architectures fall outside our primary scope.

2.3 Normalizing Flows

Normalizing flows have established a robust theoretical foundation in density estimation and generative modeling [17] by leveraging the change-of-variables formula to map simple base distributions to complex target manifolds. Seminal architectures such as RealNVP [17] and GLOW [18] utilize affine coupling layers to construct expressive, bijective transformations that maintain computational tractability. Unlike diffusion-based models, which require iterative stochastic refinement, normalizing flows facilitate exact likelihood evaluation and single-pass, deterministic sampling.

Recent research has successfully demonstrated the versatility of normalizing flows across diverse domains, including high-fidelity image generation [19], behavioral cloning within reinforcement learning frameworks [20], and as specialized action-decoding heads for Vision-Language-Action models [21]. Building upon prior research [22, 23], which utilized normalizing flows as an auxiliary mechanism for out-of-distribution (OOD) detection to enhance robustness in visual processing, we extend these insights to visuomotor policy learning and show that normalizing flows can serve as the primary end-to-end policy model.

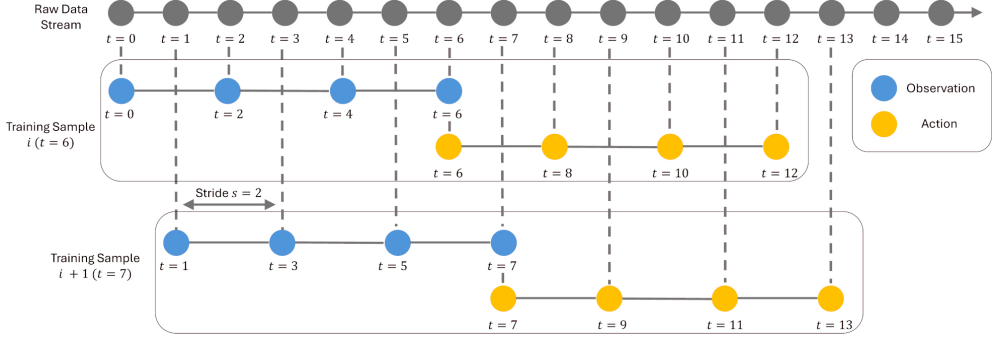


Figure 2: For a training sample i anchored at current sample $t = 6$, we extract a history of observations (blue) and a future trajectory of actions (yellow) using a temporal stride s (here $s = 2$, with both the observation and prediction horizon as 4). The sampling window slides by a single frame (shift = 1) for the next sample ($t = 7$). This stride sampling approach ensures that while the internal structure of each sample is sparse (low frequency), the dataset coverage remains dense, allowing the model to initiate plans from any state.

3 Preliminaries

3.1 RGB-based Visuomotor Policy

In its simplest form, RGB-based visuomotor policy learning can be expressed as a supervised learning problem consisting of observation–action pairs $(\mathbf{o}_t, \mathbf{a}_t)$, in which the goal is to predict the action $\mathbf{a}_t$ given the input image observation $\mathbf{o}_t$ at time t . Robotic tasks are often multi-modal with multiple, equally valid action sequences from a given state. Accordingly, modern policy models aim to learn a distribution $p(\mathbf{a}_t | \mathbf{o}_t)$ rather than a deterministic mapping.

3.2 Normalizing Flows

Similar to diffusion models, normalizing flows learn to model a distribution $p(\mathbf{x})$ by transforming data samples into Gaussian noise. Instead of modeling a diffusion process, normalizing flows represent the entire transformation as a single function [3]:

$$\mathbf{u} = T(\mathbf{x}), \quad \mathbf{x} \sim p(\mathbf{x}), \quad \mathbf{u} \sim p_u(\mathbf{u}) . \quad (1)$$

It should be noted that p_u can be chosen freely, but the most common choice is Gaussian noise $p_u = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Importantly, T is a *diffeomorphism*, which means that it is bijective with both T and T^{-1} being differentiable. Naturally, the bijective property allows sampling by:

$$\mathbf{x} = T^{-1}(\mathbf{u}), \quad \mathbf{u} \sim p_u(\mathbf{u}) .$$

Additionally, both T and T^{-1} being differentiable enables direct computation of $p(\mathbf{x})$:

$$p(\mathbf{x}) = p_u(T(\mathbf{x})) |\det J_T(\mathbf{x})| . \quad (2)$$

Here, $J_T(\mathbf{x})$ refers to the Jacobian matrix of $T(\mathbf{x})$.

Constructing a T that fulfills (1) for any distribution $p(\mathbf{x})$ is non-trivial, and as such learning-based methods, usually neural networks, are the only viable option. The primary challenge in normalizing flows is to design a neural network-based T that is sufficiently expressive, while fulfilling the requirements for a diffeomorphism in addition to having a tractable Jacobian determinant.

A common way to design T , pioneered by RealNVP [17], is *coupling layers*. Coupling layers first divide their input $\mathbf{x} = [x_1, x_2, \dots, x_D]$ into two parts, commonly:

$$\mathbf{x}_1 = [x_1, \dots, x_{D/2}], \quad \mathbf{x}_2 = [x_{D/2+1}, \dots, x_D] .$$

$\mathbf{x}_1$ is then transformed using an invertible and differentiable function $f_\theta(\mathbf{x}_1)$, with parameters θ computed from $\mathbf{x}_2$. Formally:

$$\begin{aligned} \mathbf{x}'_1 &= f_\theta(\mathbf{x}_1), \quad \theta = \text{NN}(\mathbf{x}_2) \\ \mathbf{x}' &= \text{concat}(\mathbf{x}'_1, \mathbf{x}_2) . \end{aligned} \quad (3)$$

In this formulation, NN has no restrictions, and is usually a regular feed-forward neural network. Various choices exist for the design of f_θ , arguably the most common being an element-wise linear transform:

$$f_\theta(\mathbf{x}_1) = \mathbf{x}_1 \cdot \alpha + \beta, \quad \text{with } (\alpha, \beta) = \theta .$$

In this work, however, we adopt Neural Spline Flows [24], which constructs f_θ as a rational-quadratic spline.

4 Methodology

Aiming to apply normalized flow to real-world dual-arm tasks, we propose several methods on top of the basic architecture to address the relatively higher noise levels in real-world data compared to simulation data, and the higher modalities in dual-arm tasks compared to single-arm tasks.

4.1 Observation Encoding and Conditioning

A pre-trained ResNet18 [25] is used to extract an embedding vector from each image before it is passed to the normalizing flow. This image embedding is supplemented with a vector representing the latest action, and these together form our observation $\mathbf{o}_t$. A conditional distribution can be modeled by adding a fixed conditioning vector to the NN function in the formulation (3) as $\theta = \text{NN}(\mathbf{x}, \mathbf{o}_t)$.

4.2 Strided Sampling

A persistent challenge in imitation learning is the modeling of unwanted motions, which often arise from the high-frequency noise inherent in human demonstrations. While adopting *observation horizons* and *action chunking* [1] [26] promotes temporal smoothness, these methods often fail to filter out the unconscious jitters and pauses characteristic of teleoperated data collection. Although post-processing techniques like low-pass filtering or downsampling can mitigate these artifacts, they risk discarding valuable behavioral patterns in real-world data.

To increase focus on meaningful actions during training, without reducing training data density, we introduce a *strided sampling* method. Given an observation history $\mathbf{o}_{t-m:t}$, we model a sliding window of n future actions $\mathbf{a}_{t:t+n}$. By applying a temporal stride s to both input and output sequences, as illustrated in Fig. 2, the model is forced to focus on more larger transitions between successive states, effectively bypassing high-frequency motions and minor stalling. Under this regime, the Normalizing Flow Policy learns the conditional distribution:

$$p(\mathbf{a}_t, \mathbf{a}_{t+s}, \dots, \mathbf{a}_{t+ns} \mid \mathbf{o}_t, \mathbf{o}_{t-s}, \dots, \mathbf{o}_{t-ms}).$$

In our later experiments we show that this technique significantly improves performance in both simulated environments and real-world robotic experiments. Additionally, different values for the stride s are evaluated in our ablation study. Note that, since actions during inference are produced by sampling from the NF, the stride s is not required, and thus we only apply strided sampling during training.

4.3 Likelihood-based Output Optimization

An advantage of normalizing flows is the ability to evaluate $p(\mathbf{a}|\mathbf{o})$ exactly. We demonstrate that likelihood estimate can be further leveraged at inference time to refine action selection.

To this end, we propose two optimization strategies that guide the model toward high-confidence trajectories without discarding the underlying stochasticity of the learned distribution.

Strategy 1: Stochastic Batch Selection (SBS): In this approach, we generate a parallel batch of $N = 128$ candidate action sequences from the latent distribution. By evaluating the log-likelihood for each candidate, we select the sequence that yields the highest probability. This "best-of- N " sampling effectively filters out low-density outliers that may arise from the tails of the distribution, ensuring the robot executes the most representative mode of the demonstrated behavior.

Strategy 2: Gradient Refinement (GR): Alternatively, we perform a localized optimization of a sampled trajectory. Starting from an initial sample, we execute 10 iterations of gradient ascent on the log-likelihood $\log p(\mathbf{a}|\mathbf{o})$ using the Adam optimizer [27]. This process "climbs" the manifold of the action space toward the nearest local maximum. Unlike diffusion models, which require a fixed number of denoising steps to reach a valid sample, this method treats the initial sample as a warm start, using the flow's gradient information to improve the policy's output. As shown in Fig. 1, both methods can be used directly at run time.

5 Simulation Experiments

This section describes the evaluation protocol and results for NF-P on a diverse set of simulated bimanual tasks. Unless otherwise noted, we report the *success rate*, defined as the percentage of trials that achieve the task goal within a fixed horizon, and the *inference time* per action sequence. All reported metrics are averaged over 100 randomised test trials. Our evaluation covers 50 tasks from the RoboTwin 2.0 suite [4], which spans a wide range of bi-manual manipulation skills including grasping, lifting, tool use, handover, and stacking. Some example simulation tasks are shown in Fig. 3.

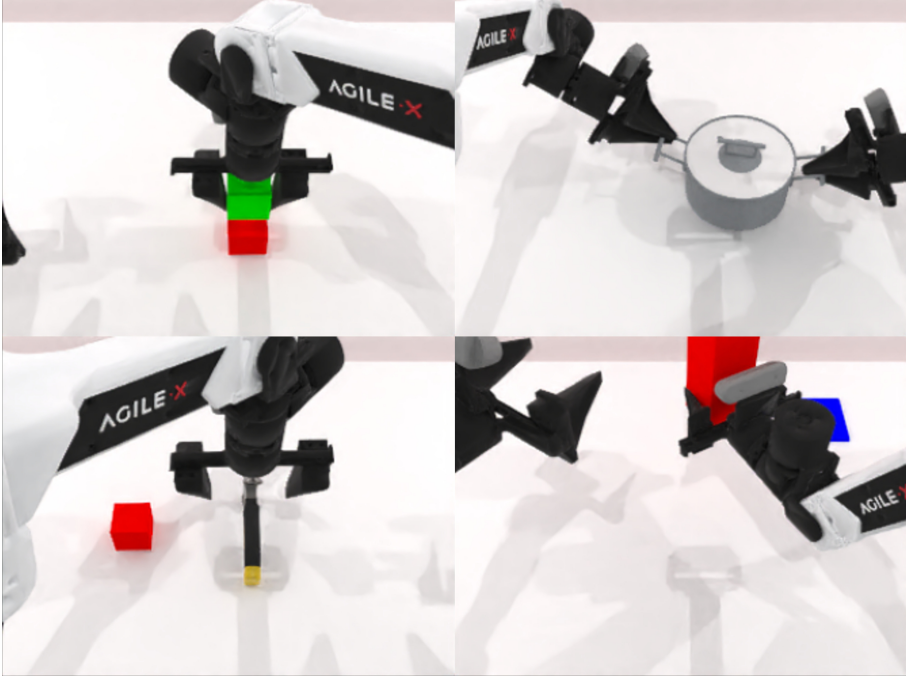


Figure 3: Visualization of four example simulation tasks: Stack Block Two, Lift Pot, Beat Block Hammer, and Handover Block.

5.1 Experimental Setup

We evaluate the performance of NF-P against the established Diffusion Policy (DP) baseline provided by RoboTwin-2.0, with scores taken from the official RoboTwin 2.0 leaderboard [28]. For each task, NF-P is trained on 50 clean scripted demonstrations, using only the head camera as visual input. Our NF consists of 10 sequential coupling spline layers, which results in approximately 100 million total trainable parameters. Each NF is trained for 300 epochs with a fixed learning rate $= 10^{-4}$ using the Adam optimizer[27]. We use a default value of $s = 4$ for the strided sampling, justified by our ablation study.

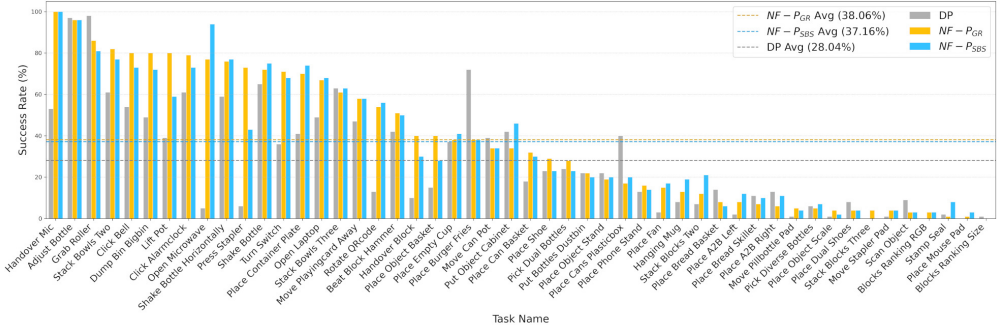


Figure 4: Summary of success rates on the full RoboTwin 2.0 evaluation suite. Bars show the success rate for 100 trials across all tasks for each method (DP, NF- P_{GR} and NF- P_{SBS}). Normalizing flow policies consistently outperform the diffusion-policy baseline.

5.2 Comprehensive RoboTwin Evaluation

Evaluation results are summarised in Figure 4. The comprehensive evaluation across the RoboTwin-2.0 benchmark suite reveals that NF-P outperforms the DP baseline in terms of overall success rates. As outlined in Fig.4, NF-P with gradient refinement (NF- P_{GR}) achieves an average success rate of 38.06%, representing a substantial 10 percentage-point improvement over the DP baseline (28.04%). Notably, the performance gains are most pronounced in tasks requiring high precision and contact-rich interactions, such as Open Microwave (77% vs. 5%) and Handover Mic (100% vs. 53%). These results suggest that the exact density estimation provided by NFs allows the policy to better exploit the underlying demonstration manifold compared to DP.

Furthermore, the comparison between our two proposed inference strategies highlights the utility of likelihood-based refinement. While NF- P_{SBS} provides a strong stochastic baseline, the GR variant frequently achieves better success rates in tasks like Press Stapler (73% vs. 43%) and Lift Pot (80% vs. 59%). This indicates that gradient ascent in log-likelihood is an effective mechanism for suppressing low-probability, sub-optimal action sequences. Interestingly, in highly complex tasks like Hanging Mug or Open Microwave, NF- P_{SBS} occasionally outperforms NF- P_{GR} , suggesting that a broad stochastic search may sometimes be more robust. Overall, both methods show promising results.

6 Real Robot Experiments and Results

This section details the physical hardware configuration, the selected manipulation tasks, and the subsequent performance results of NF-P in a bi-manual real-world environment

6.1 Experiment Setup

We evaluate NF-P on a bi-manual Kuka setup, which contains two Kuka iiwa 7 robots. Similar to the simulated experiments, only a single Realsense D435I is used, which captures the whole scene, as highlighted in Fig. 5. Task data are collected using Quest2ROS2 [29], a VR based teleoperation system, and the data types are the same as the RoboTwin 2.0 simulations detailed above.

6.2 Task Definitions and Policy Training

To assess the robustness and coordination of NF-P, we selected two distinct tasks that challenge different aspects of robotic control.

Stack Blocks Two: This task requires the robot to sequentially pick up two colored blocks and stack them in a predefined order. Beyond simple reaching and grasping, this task evaluates the model’s long-horizon temporal reasoning and the precision required to ensure the resulting stack remains physically stable.

Towel Folding: Unlike rigid-body manipulation, this task involves a deformable object that requires precise bi-manual coordination. The two robotic arms must execute a synchronized grasp, move at a uniform velocity to maintain tension, and perform a simultaneous release.

For the training phase, we collected 150 demonstrations for Stack Block Two and 100 for Towel Folding. To compensate for the higher noise levels in real robot data, the NF-P architecture was configured with an observation and prediction horizon of 4 and a temporal stride $s = 6$, and uses SBS optimization in inference. Similarly, the DP baseline employed the standard DP-C implementation [1], which has a longer horizon than the simulation setup.



Figure 5: Real-world experimental setup and task progression. (Left) The dual-arm manipulation environment for the Stack Block Two (top row) and Towel Folding (bottom row) tasks. The wrist and head cameras are not used for both task, the red circle indicates the single scene camera which provides visual observation input for the models. (Right) Key terminal stages of the tasks viewed from the perspective of the scene camera. These stages correspond to the bottleneck evaluation milestones (e.g., picking, placing, and folding).

6.3 Results and Discussion

To highlight the performance differences observed during real-world execution, we conducted a rigorous bottleneck analysis focused on the terminal stages of each multi-stage task. As detailed in Table 1, this analysis identifies specific points of failure, revealing how each policy handles sequential dependencies and compounding errors across long-horizon trajectories.

A critical behavioral trend emerges when examining the DP baseline: it frequently fails to initialize execution, and it often gets stuck between sub-goals. In the Stack Block Two task, DP suffered from an initial failure rate of 35%, often failing to trigger the first picking motion. Furthermore, 50% of its trials terminated prematurely after the "Arm One Placing" stage. Our observations indicate that DP commonly fails when transitioning between sub-goals. For example, after the first block has been placed, instead of engaging the second arm to place the other block, DP would often get stuck in a cycle repeating the first arm's trajectory. Similarly, in the Towel Folding task, DP exhibited initiation bottlenecks, with 45% of trials failing at the onset. In these instances, the model is typically able to steer the robot to target object but fails to give the gripper-close command, stalling indefinitely.

While NF-P also exhibits some amount of stalling when transitioning between sub-goals, the average transition time is significantly lower. Throughout our experiments, DP was given an 8-minute window for each sub-goal transition, whereas NF-P achieved a higher success rate with only a 30-second limit.

Table 1: Bottleneck Analysis and Success Rates for Real-World Rollouts. The table records the terminal stage of each of 20 trials alongside the overall success rate. While DP frequently succumbs to early-stage bottlenecks (e.g., failing at initial grasp or prior to execution), our method reliably completes early sub-tasks, pushing failures to later stages and achieving higher overall success rates (60% and 75%, respectively).

Stack Block Two						
Method	Initial Fail	Arm One Picking	Arm One Placing	Arm Two Picking	Arm Two Placing	Success Rate(%)
DP	7	0	10	0	3	15
NF-P _{multi}	0	0	1	7	12	60
Towel Folding						
Method	Initial Fail	Towel Picking		Towel Folding		Success Rate(%)
DP	9	0		11		55
NF-P _{multi}	3	2		15		75

In the Stack Block Two task, NF-P successfully picked and placed the first block in 95% of trials, and achieved a 0% initial failure rate. The sparse failures encountered were significantly deeper into the execution horizon compared to DP, primarily occurring at the "Arm Two Picking" stage, due to missing the placement of the block at the last step. For Towel Folding, NF-P initiated the trajectory successfully in 85% of cases, and only failed during towel picking in 10% of the cases.

These results suggest that the exact likelihood estimation inherent in Normalizing Flows provides a more stable density manifold for the agent to follow. By effectively mitigating early-stage stochastic drift and facilitating rapid state transitions, NF-P ensures that the policy remains within distribution even during complex handovers between sub-goals. This robustness leads to markedly higher overall success rates compared to DP, validating the efficacy of normalizing flows based architectures for extended bi-manual robotic manipulations.

7 Data Scaling Experiment

One concern regarding NFs, is that they might not be as expressive as other models, as they are constrained by requiring a tractable inverse and Jacobian determinant [3]. In order to verify that NFs are capable of scaling to larger datasets, we conduct an additional experiment using the RoboTwin 2.0 simulation framework. Here, we select 4 different tasks: Beat Block Hammer, Lift Pot, Handover Block, and Stack Blocks Two. For each task, we train and evaluate both NF-P and DP (configuration DP-C [1]) using a varying number of training episodes from 10 up to 500. Our results, in Fig. 6 show that NF-P exhibit similar scaling properties as DP across all 4 tasks, alleviating any initial concerns.

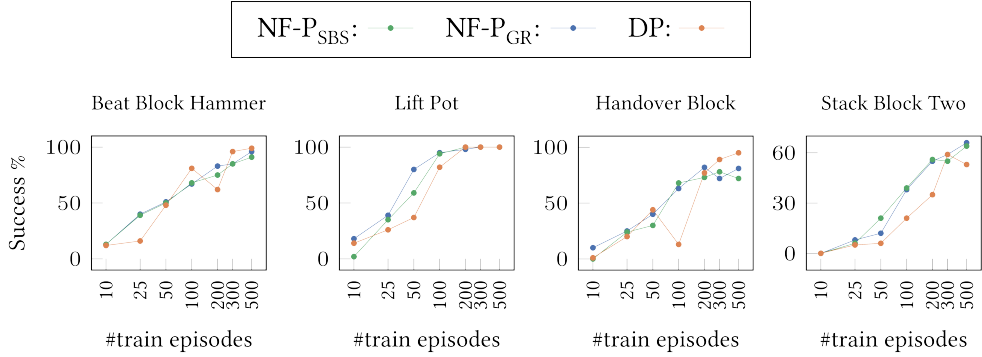


Figure 6: Results for scaling simulated experiments with varying amounts of training data.

8 Model Efficiency

Computational overhead during both training and inference is a critical constraint for real-world robotic deployment. NF-P demonstrates significant advantages in this regard, achieving high-fidelity results with relatively minimal resource expenditure. Specifically, on a single NVIDIA RTX 4090, the model reaches convergence in as few as 300 epochs on a dataset of 100 demonstrations, which takes roughly 50 mins to train. Also, due to its single-pass, deterministic sampling architecture, NF-P_{SBS} achieves inference times under 20ms. This represents a substantial improvement over diffusion models, enabling high-frequency control loops that are essential for reactive manipulation. Even the GR variant (NF-P_{GR}), which performs 10 steps of gradient ascent, maintains a latency of 455ms, which is faster than the standard DP baseline. Notably, NF-P_{SBS} offers an ideal middle ground, retaining the real-time advantages of the flow backbone while delivering the performance gains associated with likelihood-based selection. This efficiency ensures that NF-P can be deployed on standard robotic hardware without the need for specialized high-latency compensators.

9 Ablation Study

To isolate the contributions of specific design choices within the NF-P framework, we conducted a systematic ablation study using the Beat Block Hammer simulation task. We evaluated the sensitivity of the model to the temporal stride s , the sampling variance σ , and the choice of output optimization strategy. The results, summarized in Table 2, identify the configuration of $s = 4$, $\sigma = 0.5$ with gradient-ascent optimization as the optimal parameter set, achieving a peak success rate of 51% on the RoboTwin 2.0 Beat Block Hammer task.

Table 2: NF-P ablations on the RoboTwin 2.0 *Beat Block Hammer* task. Each entry reports the success rate (%) from 100 trials when varying one component of the NF-P design. The default configuration is $s = 4$, $\sigma = 0.5$, and gradient refinement.

Ablation			Success rate
NF-P _{GR}	$s = 4$	$\sigma = 0.5$	51
NF-P _{SBS}	$s = 4$	$\sigma = 0.5$	50
NF-P	$s = 4$	$\sigma = 0.5$	37
NF-P _{GR}	$s = 8$	$\sigma = 0.5$	46
NF-P _{GR}	$s = 2$	$\sigma = 0.5$	48
NF-P _{GR}	$s = 1$	$\sigma = 0.5$	37
NF-P _{GR}	$s = 4$	$\sigma = 1.0$	41
NF-P _{GR}	$s = 4$	$\sigma = 0.75$	47
NF-P _{GR}	$s = 4$	$\sigma = 0.25$	42
NF-P	No action chunking		28

Both NF-P_{SBS} and NF-P_{GR} yield substantial performance gains over direct sampling NF-P. This underscores the benefits of guiding the stochastic output toward high-probability regions of the learned manifold to ensure precision.

Temporal stride appears to play a critical role in balancing the generated motions. A stride of $s = 1$ resulted in a lower success rate of 37%. Conversely, excessively large strides ($s = 8$) also degraded performance to 46%. The value $s = 4$ appears to be the optimal for filtering demonstration noise while maintaining behavioral fidelity.

A moderate sampling variance appears to aid task execution, with $\sigma = 0.5$ delivering the highest success rate. Standard sampling, with $\sigma = 1$ yields reduced performance. Similarly, decreasing it to $\sigma = 0.25$ also led to worse success rates, indicating that the policy requires a moderate level of stochastic exploration during the inference process to remain robust to environmental perturbations.

Predicting a single action instead of an action chunk reduced the success rate to 28%. This result reinforces the importance of action chunking with regard to temporal consistency.

10 Conclusion

In this work, we introduced Normalizing Flow Policy (NF-P), a visuomotor policy framework based on normalizing flows designed for complex, bi-manual manipulation. By utilizing normalizing flows, NF-P provides a unique combination of exact likelihood estimation and efficient single-pass inference.

Our comprehensive evaluation across the RoboTwin-2.0 benchmark suite demonstrates that NF-P consistently matches or exceeds the performance of Diffusion Policy (DP),

achieving a higher average success rate while requiring significantly fewer training epochs. Real-world experiments on a bi-manual Kuka setup further validate the model’s robustness, specifically its ability to overcome stalling and transition bottlenecks.

Through detailed ablation studies, we identified that stride-based sampling and likelihood-based optimization lead to improved performance. Ultimately, NF-P represents a mathematically sound and computationally efficient alternative for scaling robotic learning to multi-modal settings where real-time responsiveness and reliable uncertainty quantification are required.

11 Future Work

While we have explored several different techniques for making NFs viable and even competitive in the domain of visuomotor policy learning, our work is by no means exhaustive. One such method that we left unexplored is the addition of noise during training [19]. It would be an interesting direction to investigate how adding noise behaves in conjunction with our gradient ascent likelihood-optimization, since the noise removal is effectively equivalent to one step of gradient ascent.

There are also works, for example [13], that attempt to improve inference times in diffusion models by updating an existing action sequence using fewer denoising steps compared to generating actions from scratch. Similar approaches could be explored for NFs. However, in the case of NFs, we believe such approaches would rather be applicable for improving consistency and smoothness in the action sequences, since the NF’s single-pass inference already provides low latency.

Another potential research direction could be to evaluate different NFs architectures. In this work, we adopted coupling layers due to their fast inference and ability to directly evaluate their probability density. There are, however, other NFs architectures that have shown competitive performance in various generative tasks [24].

Finally, our previous research has shown how NFs can be used as a complement to improve the performance of other models by providing a robust measure of confidence [22, 23]. It is possible that NFs could be applied in conjunction with for example DP or a transformer-based model to further improve performance.

Acknowledgment

This research is funded by the Excellence Center at Linköping-Lund in Information Technology (ELLIIT), and the Wallenberg AI, Autonomous Systems and Software Program (WASP). We thank the computations enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre. We thank Hashim Ismail and Davide Tateo for their discussion and feedback, and Marcus Klang for continued support on robotic setup.

Bibliography

- [1] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2024.
- [2] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *arXiv preprint arxiv:2006.11239*, 2020.
- [3] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, “Normalizing flows for probabilistic modeling and inference,” *Journal of Machine Learning Research*, vol. 22, no. 57, pp. 1–64, 2021. [Online]. Available: <http://jmlr.org/papers/v22/19-1028.html>
- [4] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Q. Liang, Z. Li, X. Lin, Y. Ge, Z. Gu, *et al.*, “Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation,” *arXiv preprint arXiv:2506.18088*, 2025.
- [5] Z. Gong, P. Ding, S. Lyu, S. Huang, M. Sun, W. Zhao, Z. Fan, and D. Wang, “Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction,” *arXiv preprint arXiv:2412.06782*, 2024.
- [6] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022.
- [7] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto, “Behavior generation with latent actions,” *arXiv preprint arXiv:2403.03181*, 2024.
- [8] S. Haldar, Z. Peng, and L. Pinto, “Baku: An efficient transformer for multi-task policy learning, 2024,” URL <https://arxiv.org/abs/2406.07539>.
- [9] 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 *Conference on robot learning*. PMLR, 2022, pp. 158–168.

- [10] X. Ma, S. Patidar, I. Haughton, and S. James, “Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 081–18 090.
- [11] J. Ma, Y. Qin, Y. Li, X. Liao, Y. Guo, and R. Zhang, “Cdp: Towards robust autoregressive visuomotor policy learning via causal diffusion,” *arXiv preprint arXiv:2506.14769*, 2025.
- [12] Z. Wang, Z. Li, A. Mandlekar, Z. Xu, J. Fan, Y. Narang, L. Fan, Y. Zhu, Y. Balaji, M. Zhou, *et al.*, “One-step diffusion policy: Fast visuomotor policies via diffusion distillation,” *arXiv preprint arXiv:2410.21257*, 2024.
- [13] Y. Duan, H. Yin, and D. Kragic, “Real-time iteration scheme for diffusion policy,” *arXiv preprint arXiv:2508.05396*, 2025.
- [14] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, “3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations,” *arXiv preprint arXiv:2403.03954*, 2024.
- [15] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, “ π_0 : A vision-language-action flow model for general robot control,” 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [16] K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, *et al.*, “ π_0 . 5: a vision-language-action model with open-world generalization,” *arXiv preprint arXiv:2504.16054*, 2025.
- [17] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using real nvp,” 2017. [Online]. Available: <https://arxiv.org/abs/1605.08803>
- [18] D. P. Kingma and P. Dhariwal, “Glow: Generative flow with invertible 1x1 convolutions,” 2018. [Online]. Available: <https://arxiv.org/abs/1807.03039>
- [19] S. Zhai, R. Zhang, P. Nakkiran, D. Berthelot, J. Gu, H. Zheng, T. Chen, M. A. Bautista, N. Jaitly, and J. Susskind, “Normalizing flows are capable generative models,” *arXiv preprint arXiv:2412.06329*, 2024.
- [20] R. Ghugare and B. Eysenbach, “Normalizing flows are capable models for continuous control,” in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

- [21] D. Tarasov, A. Nikulin, I. Zisman, A. Klepach, N. Lyubaykin, A. Polubarov, A. Derevyagin, and V. Kurenkov, “Nina: Normalizing flows in action. training vla models with normalizing flows,” *arXiv preprint arXiv:2508.16845*, 2025.
- [22] S. Kristoffersson Lind, R. Triebel, L. Nardi, and V. Krueger, “Out-of-distribution detection for adaptive computer vision,” in *Image Analysis*, ser. Lecture Notes in Computer Science, R. Gade, M. Felsberg, and J.-K. Kämäräinen, Eds. Cham: Springer Nature Switzerland, 2023, p. 311–325.
- [23] S. K. Lind, R. Triebel, and V. Krüger, “Making the flow glow – robot perception under severe lighting conditions using normalizing flow gradients,” no. arXiv:2412.07565, Dec. 2024, arXiv:2412.07565 [cs]. [Online]. Available: <http://arxiv.org/abs/2412.07565>
- [24] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, *Neural spline flows*. Red Hook, NY, USA: Curran Associates Inc., 2019.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. [Online]. Available: <https://arxiv.org/abs/1512.03385>
- [26] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *arXiv preprint arXiv:2304.13705*, 2023.
- [27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [28] T. Chen, Z. Chen, B. Chen, Z. Cai, Y. Liu, Q. Liang, Z. Li, X. Lin, Y. Ge, Z. Gu, *et al.*, “Robotwin 2.0 official leaderboard,” fetched 2026-02-23. [Online]. Available: <https://robotwin-platform.github.io/leaderboard>
- [29] J. Li, Z. Wang, T. Wang, M. Stenmark, and V. Krueger, “Quest2ros2: A ros 2 framework for bi-manual vr teleoperation,” *arXiv preprint arXiv:2601.18289*, 2026.