UNIVERSITY OF
CAMBRIDGE

Efficient, robust and uncertainty aware mobile health

Lorena Qendro

Murray Edwards College

This dissertation is submitted for the degree of Doctor of Philosophy

September 2022

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Lorena Qendro
September 2022

Abstract

Efficient, robust and uncertainty aware mobile health

Lorena Qendro

Sensor-equipped smartphones and wearables are transforming various mobile applications, including health monitoring. As the difference between consumer health wearables and medical devices begins to soften, it is now common for a single wearable device to monitor a range of medical risk factors. Potentially, these devices could give patients direct access to personal analytics that can contribute to their health, facilitate preventive care, and aid in the management of ongoing illness. This data can be fed into machine learning algorithms aiming to help practitioners better assess the progress of the patient or other aspects of the disease evolution. Therefore, it is essential to have accurate model predictions and, equally significantly, a better understanding of what happens within the machine learning model suggesting a prediction. Deep learning is considered a key element in analyzing these new data types. However, traditional deep learning models cannot capture the predictive uncertainty leading to overconfident predictions, and, ultimately, they are less robust in real-world applications. Bayesian deep learning or other non-Bayesian probabilistic techniques can naturally quantify such uncertainty. Still, they do come with quite a few drawbacks, the biggest one being the fact that they tend to be computationally heavy. This dissertation attempts to address the challenges above by developing new efficient-by-design frameworks for robust and uncertainty aware mobile health.

Firstly, we introduce a framework that enables already trained deep learning models to generate uncertainty estimates on edge computing platforms with no need for fine-tuning or re-training strategies. This simple and system-efficient solution is built to provide predictive uncertainty for convolutional neural networks based only on one forward pass and a negligible number of

additional matrix multiplications. Next, we evaluate our approach on multiple mobile sensing datasets and show that it can provide reliable uncertainty estimates and accurate predictions with very little computation and memory overhead.

Next, we provide a new interpretation of early exit neural networks as an implicit ensemble of weight-sharing sub-networks from which predictive uncertainty can be estimated. Our approach can be applied to any feed-forward deep learning architecture. Empirical evaluation of several medical imaging and biosignal datasets and state-of-the-art architectures demonstrates strong performance in accuracy and uncertainty metrics as well as computation gain, highlighting the benefit of combining multiple structurally diverse models that can be jointly trained.

Finally, acknowledging that in real-world deployments, deep learning-based mobile health applications may encounter malign inputs that can have a catastrophic impact, we propose an attack-agnostic adversarial mitigation approach based on the previously described early exit ensemble framework. We show that this approach can guarantee robustness generalizable to various white box and universal adversarial inference time attacks, increasing the accuracy of vulnerable targeted deep learning models while providing mitigation comparable to adversarial training but without the computational burden.

The frameworks devised in this thesis suggest promising future directions for efficient uncertainty estimation in mobile health research. Furthermore, designing robust deep learning models has increasingly proved to be extremely relevant to health-critical mobile applications. By addressing several challenges in this problem space, these contributions take a few steps toward building machine learning systems that can be safely deployed in real-life settings.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Cecilia Mascolo, for her exceptional guidance, support, and invaluable mentorship throughout my Ph.D. journey. Her sharp eye for important research problems and boundless encouragement have been instrumental in shaping me into the researcher I am today. I will always be grateful for the immense freedom she gave me in pursuing my own research direction. I could not have completed this dissertation without her constant feedback, care, and immense trust in me and my research.

I am deeply grateful to my talented collaborators - Alberto Gil C. P. Ramos, Jagmohan Chauhan, Alexander Campbell, and Pietro Liò - who played a crucial role in making this dissertation a reality. I cannot thank them enough for their valuable contribution. I would also like to express my sincere appreciation to Alastair Beresford and Sean Holden for their insightful suggestions on my research. Their feedback was instrumental in refining my ideas and gave me the confidence to pursue this thesis. Furthermore, I am grateful to my examination committee, Katayoun Farrahi, and Carl Henrik Ek, for their patience in reading this thesis, the intellectual discussion during the viva, and their thoughtful suggestions that helped me improve the quality of my work.

The time I spent at ARM Research during my internship was truly remarkable, and I attribute it to the extraordinary people I had the opportunity to work with. Working alongside Partha Maji's team provided me with extremely valuable brainstorming opportunities. Together with Sangwon Ha and René de Jong, they helped me consolidate my interests and develop a cohesive research vision that is reflected in my thesis. I am immensely grateful for their mentorship and guidance throughout this experience.

Throughout my career, I have been fortunate enough to have worked with exceptional mentors and colleagues who have contributed significantly to my growth. Enrico Denti and Michela Milano were instrumental in the early stages of my research at the University of Bologna, providing

invaluable guidance that helped lay the foundation for my career. At Telekom Innovation Laboratories, Sebastian Möller and Tim Polzehl gave me the amazing opportunity to work at my first research center. Furthermore, I owe a debt of gratitude to Nic Lane for introducing me to deep learning and mobile systems, which have become a significant focus of my research. Additionally, I would like to express my heartfelt appreciation to all my colleagues at Ellexus, Connected Places Catapult, and Geospock, who have made my journey incredibly rewarding. The professional and personal moments I shared with them are unforgettable, and I will always cherish them.

As a mentor myself, I have gained a wealth of knowledge and experience that I am deeply grateful for. I owe this gratitude to the many students I have had the privilege of supervising over the years. In particular, I would like to thank my research students Yu Chen Lim, Malachy O'Connor Brown, Oscar Hill, Terry Fawden, and Sotirios Vavaroutas, who have not only allowed me to explore new research directions but also taught me a great deal through our collaboration. I look forward to hearing about their future accomplishments and successes.

I am immensely grateful to the staff of the Department of Computer Science and Technology for providing graduate students with an intellectually stimulating environment. I would like to express a special thanks to Lise Gough, who has been helping me since the very first day I applied for the Ph.D. I am also thankful to Matthew Danish, who organized the happy hours where I had the opportunity to meet and connect with most of the people from the department. The Women@CL initiative was another amazing experience that allowed me to meet some of the most remarkable women in computer science. I owe my entire Ph.D. journey to Nokia Bell Labs for their generous financial support. Their contribution enabled me to pursue my research, present my work at international conferences, and engage with others in my field. Lastly, I would like to extend my gratitude to Murray Edwards College for their constant support and for making my time here in Cambridge both enjoyable and memorable. A special thanks goes to Elrika Erasmus for her immense help during the more challenging moments of my journey.

Being part of the fantastic Mobile Systems group has made these years not only enjoyable and engaging but also social, both within and outside office hours. Dionysis, Krittika, Xiao, Dimitris, Andreas, Andrea, Api, Dong, Ian, Erika, Kayla, Young, Tong, Ting, Jing, Hong, Abhirup, Yang, Yvonne, and Sotirios made my journey unforgettable. I am grateful for the friendships and collaborations that have grown out of our shared experiences. My dear Cambridge friends Zahra, Ali, Petko, Ellen, Desi, Lorena, Laia, Diana, Marco, Laura, Stefanos, Galinie, Kai, Flavia, Jamie, Ben, Pon, and the Italian crew Sabbia, Lucilla, Antonietta, Eleonora, thank you for being you

and for the amazing moments we've passed together.

As I approach the end of my academic journey, I cannot help but express my deepest appreciation to my beloved family whose support and encouragement have been instrumental in my success. My parents, Kristina and Ali, and my brother Kleo have been my pillars of strength, providing me with love, patience, and understanding throughout the ups and downs of this journey. I am indebted to them for the sacrifices they have made to help me achieve my goal. My beloved grandmother Aleksandra and lovely aunts Elvira and Majlinda have been my strongest supporters, consistently there for me whenever I needed their help.

I am incredibly grateful to the love of my life, Alessandro, who has been my constant source of inspiration and motivation. He has given me the strength to persevere in the face of adversity. I cannot express enough how much his support means to me, and I am forever grateful for his presence in my life. I know I am fortunate to have such a wonderful and supportive family and partner who have stood by me through thick and thin. I owe my success to their unending love. *I love you!*

Thank you all for your encouragement and inspiration. I could not have completed this dissertation without you.

Contents

Chapter 1

Introduction

The proliferation of smartphones, wearables, and edge devices has revolutionized our lives. They come equipped with various sensors, for instance, microphones, accelerometers, electroencephalogram (EEG), and electrocardiogram (ECG) sensors which enable the fine-grained and continuous collection of data carrying information about human behavior, health, and context. At the same time, machine learning (ML) algorithms have made unprecedented advancements in how computers extract information from the collected data. The combination of data availability and powerful ML techniques are the main drivers and enablers of a new set of sensing-based applications that were not feasible before.

Recently, deep learning models have become the method of choice for the analysis of many mobile sensing and health applications such as activity and context recognition (Wang et al., 2018; Hsu et al., 2018; Alharbi and Farrahi, 2018; Lane et al., 2015, 2016), health and well-being monitoring (De Pessemier and Martens, 2018; Ryder et al., 2009; Subasi et al., 2018; Wang et al., 2022), and location prediction (Ballinger et al., 2018; Lu et al., 2012; Ronao and Cho, 2016; Figo et al., 2010). Most importantly, deep learning achieves state-of-the-art performance on a variety of tasks within the medical field, such as classification (Esteva et al., 2017; Wang et al., 2022), segmentation (Perslev et al., 2019), and monitoring (Chan et al., 2019). The key to their success relies on their structure of multiple non-linear hidden layers, enabling them to learn complex, intricate relationships between the sensory inputs and predicted outputs. Many mobile sensing applications have a significant role in healthcare, for instance, in detecting cardiac arrest via smart devices (Stamate et al., 2017), sleep apnea (Nandakumar et al., 2015) or freezing of gaits

in Parkinson's Disease (PD) patients (Chan et al., 2019). In addition, mental health illnesses, such as depression (Valstar et al., 2013) and stress (Lu et al., 2012), can also be detected using speech samples collected with device microphones.

The crucial motivation of machine learning research is to eventually integrate automatic intelligent systems into humans' lives and enable them to co-operate with humans and the surrounding environment towards solving the aforementioned vital health tasks. However, as these systems become more pervasive, so too has the concern over how we can trust their underlying algorithms. Therefore, in April 2021, the European Commission became the first governmental body to propose an international regulation for artificial intelligence (AI) (Commission et al., 2021) highlighting the need for techniques necessary to build trust in AI.

Reasonably, as soon as the developed ML-based frameworks are deployed into the real world, they will face challenging requirements caused by introducing new devices, noisy and unpredictable settings, the physical world, and device constraints. Therefore, we need to equip them with capabilities to capture the inherent uncertainty stemming from the environment, the data, and the ML models. These frameworks are required to be efficient by design, too. They can guarantee deployability in all kinds of devices, even the ones with resource limitations. Smart devices do not provide only data. At the same time, they are computing platforms that can be exploited for on-device intelligence. These platforms promise a seamless experience tailored to each user's specific needs while maintaining the integrity of their personal health data.

Although the research literature offers crucial steps toward accurate machine learning models, designing efficient, robust, and uncertainty aware mobile health applications present several unresolved challenges. While the research community and policy-makers design new trust metrics and policies, many models have already been deployed in the real world. Many of these models, however, are not robust enough to handle difficult situations in the wild. For example, we can mention two disastrous incidents caused by unreliable ML models: an assisted driving car autopilot confusing the white side of a tractor-trailer for bright sky (NHTSA, Jan 2017) resulting in a life-threatening accident, and an image classifier mistakenly identifying two African Americans as gorillas (Guynn, 2015). Chapter 3 of the thesis helps with building more robust ML-based applications by providing a framework to enable already trained and deployed deep learning models to quantify their predictive uncertainty. Many of these wrong decisions that lead to severe consequences are caused by machine learning models sometimes encountering examples outside of the data distribution used during the training procedure. Chapter 4 delves into this problem space and proposes a new framework able to quantify in-distribution calibration

as well as out-of-distribution detection efficiently. Finally, a significant challenge is handling and mitigating adversarial attacks. While out-of-distribution inputs, noisy data, and distribution shifts can happen naturally in the wild, adversarial attacks are operated by malign agents whose aim is to inject malicious inputs intentionally or leak prediction model parameters or training data (Shokri et al., 2017; Mo et al., 2020). Based on the methodology proposed in Chapter 4, Chapter 5 presents an attack-agnostic-based defense framework designed to mitigate against prediction time injection attacks efficiently.

The rest of this chapter motivates and outlines a series of methods we have designed and presented to the mobile systems and machine learning communities. They each contribute toward providing robust and uncertainty aware predictions for mobile sensing and health applications–by either improving upon previous approaches or achieving comparable performance while reducing the computational burden. The decisions made throughout this dissertation have been guided by both the application settings mentioned earlier and the computational limitations of modern-day mobile and edge devices. Our research, therefore, is not solely focused on advancing the field from a scientific perspective, but rather on addressing real-world situations. We have aimed to produce a line of work that is both theoretically sound and practically significant, offering the potential to enhance and streamline real-world practices.

1.1 Motivation

Deep neural networks are deployed in increasingly safety-critical and health-impactful applications, motivating the need for novel techniques that guarantee robustness by preserving model performance under both naturally-induced corruptions or alterations and adversarial perturbations. Despite the great success and predictive capacity, DNNs lack interpretability which does not allow a proper understanding of how the final predictions were deduced. When encountered with examples outside its data distribution, a DNN model deployed in high-risk scenarios can make inaccurate decisions or suggestions that can lead to severe consequences.

Understanding the model's uncertainty gives a helpful indication of when the system is guessing at random or if it is confident about the prediction. In the medical field, uncertainty quantification is critical since the input distributions are often shifted from the training distribution due to different hardware, data collection protocols, change in demographics, or country where the data is collected from (Mårtensson et al., 2020; Amodei et al., 2016). In such scenarios, a model with well-calibrated uncertainty can indicate if a prediction should be trusted (Ovadia et al.,

2019). Furthermore, such a model could inform clinicians on how its performance may degrade in different deployment settings and when a human-in-the-loop is required to analyze uncertain samples (García Rodríguez et al., 2020; Xia et al., 2021). While noisy conditions might occur naturally, an adversary entity can exploit the vulnerabilities above of the deep learning model and compromise its prediction by injecting malign perturbations. For instance, simple manipulation of the peripherals of brain-computer interfaces can alter the model output (e.g., from drowsy to awake), causing severe traffic accidents in EEG-based driving assistance for self-driving cars (Zhu et al., 2021). Additionally, automatic medical systems are notoriously difficult to update, so new defenses could be challenging to roll out (Finlayson et al., 2019) especially given the speed new attacks are produced. Therefore, to keep up with the fast pace of new attacks, there is a need for attack-agnostic and generalizable solutions.

Adopting a Bayesian framework offers uncertainty estimations on predictions made by DNNs (Welling and Teh, 2011; Graves, 2011; Ustyuzhaninov et al., 2020; Dutordoir et al., 2021), which helps capture the erroneous overconfident decisions in case of out-of-distribution or noisy data. However, their training and inference [1] are computationally very expensive (Dusenberry et al., 2020). Hence running them on devices with limited resources, such as smartphones and wearables, is a big challenge (Yao et al., 2018a). Even if the training is done on clusters of powerful GPUs, just running inference remains a computational burden for these platforms. The latest studies by the machine learning community offer enlightening directions on getting robust uncertainty estimations for DNNs. Although recent efforts have been devoted to making them more efficient, their improvements are still not a good fit for mobile or edge devices since they are based either on *sampling* (Gal and Ghahramani, 2015a, 2016a,b) or *ensembles* of models (Lakshminarayanan et al., 2017; Wenzel et al., 2020). While sampling demands running a single stochastic neural network multiple times, ensemble methods require training and running various neural networks, which linearly increases latency if run in sequence or memory if run in parallel. Indeed, these solutions are resource agnostic and would incur unfeasible increases in power consumption, latency, and memory requirements on many devices with limited resources.

1.2 Research questions

We have reviewed the benefits of using deep learning for mobile health, what risks arise when traditional methods are employed, and what computational challenges are involved when adopting

[1] Throughout this dissertation by *inference* we mean the prediction of the model, not the statistical meaning of inference in variable marginalization.

existing state-of-the-art uncertainty aware approaches. This thesis proposes a set of contributions in the form of deep learning-based frameworks with the ultimate goal of *providing techniques to efficiently and accurately yield uncertainty aware mobile sensing and health applications and apply them to achieve real-world robustness against distributional shifts and adversarial attacks*. These frameworks are evaluated via several classification tasks, ranging from mobile sensing with inertial measurement units to audio recognition and real-time biosignal applications.

The central questions guiding the thesis are summarized below:

- ***Research Question 1:*** *How can we provide uncertainty awareness on deep learning models which have already been trained and deployed?*
- ***Research Question 2:*** *How can existing efficient paradigms be exploited under novel interpretation for uncertainty quantification?*
- ***Research Question 3:*** *To what extent can the introduced ensemble-based framework mitigate against inference time adversarial attacks? How can its orthogonality to existing mitigation techniques be exploited for better robustness?*

The deep learning-based frameworks proposed in this thesis aim to address these questions by, firstly, putting forward an efficient solution for predictive uncertainty estimation in NNs deployed on edge computing platforms with no need for fine-tuning or re-training strategies. Further, we design a framework that leverages existing efficient paradigms (i.e., early exit neural networks) under a novel interpretation as an ensemble technique for uncertainty quantification. Finally, we exploit the latter framework for mitigation against adversarial attacks and combine it with traditional defense approaches to increase robustness further.

1.3 Contributions and thesis outline

Chapter 2—Background This chapter starts with a definition, sources, and potential applications of predictive uncertainty in mobile health. It continues with the necessary theoretical background required to carry out the research presented in the contribution chapters—machine learning and neural network fundamentals, Bayesian deep learning literature, and the fundamentals of more pragmatic uncertainty-aware approaches (Monte Carlo Dropout and deep ensembles). We then discuss the characteristics of the edge platforms used in this thesis.

Chapter 3—Uncertainty aware sensing for edge computing platforms In Chapter 3, we propose an efficient framework for predictive uncertainty estimation in NNs deployed on edge computing platforms with no need for fine-tuning or re-training strategies. To meet the energy and latency requirements of these systems, the framework is built from the ground up to provide predictive uncertainty based only on one forward pass and a negligible amount of additional matrix multiplications. We aim to enable already trained deep learning models to generate uncertainty estimates on resource-limited devices at inference time, focusing on classification tasks.

This framework is founded on theoretical developments casting dropout training as approximate inference in Bayesian NNs. Our layerwise distribution approximation to the convolution layer cascades through the network, providing uncertainty estimates in one single run, which ensures minimal overhead, especially compared with uncertainty techniques that require multiple forward passes and an equal linear rise in energy and latency requirements, making them unsuitable in practice. We demonstrate that it yields better performance and flexibility over previous work based on multilayer perceptrons to obtain uncertainty estimates. Although our approach draws on the theoretical underpinnings of Monte Carlo Dropout (Gal and Ghahramani, 2015a), it significantly diverges from it. Unlike Monte Carlo Dropout, our technique embeds the distributions within the network, eliminating the need for multiple runs to produce them. Moreover, we introduce a novel method for adapting the distribution output to make accurate predictions in classification tasks while simultaneously providing the desired predictive uncertainty estimates.

Our evaluation with mobile applications datasets on Nvidia Jetson TX2 and Nano shows that our approach obtains not only robust and accurate uncertainty estimations but also outperforms state-of-the-art methods in terms of systems performance, reducing energy consumption (up to 28–folds), keeping the memory overhead at a minimum while still improving accuracy (up to 16%).

Chapter 4—Early exit ensembles for uncertainty quantification Chapter 4 introduces early exit ensembles, a novel framework capable of capturing predictive uncertainty via an implicit ensemble of early exits. In particular, early exit ensembles are a collection of weight-sharing sub-networks created by adding exit branches to any backbone neural network architecture. We evaluate our approach to the task of classification using four state-of-the-art deep learning architectures applied to four medical imaging datasets. Empirical evaluation of the proposed solution demonstrates strong performance in accuracy and uncertainty metrics as well as computation gain highlighting the benefit of combining multiple structurally diverse models that can be jointly trained.

Compared to state-of-the-art baselines (with an ensemble size of 5), early exit ensembles can improve uncertainty metrics up to $2\times$ while providing test-time speed-up and memory reduction of approx. $5\times$. Additionally, our approach can improve accuracy by up to 3.8 percentage points compared to single model baselines reflecting the variance-reducing effect of averaging a set of diverse models with individually high variance and low bias. Finally, our results suggest that by providing well-calibrated predictive uncertainty for both in- and out-of-distribution inputs, early exit ensembles have the potential to improve the trustworthiness of models in high-risk medical decision-making.

Chapter 5—Towards adversarial robustness with early exit ensembles In this chapter, we propose an adversarial mitigation technique for biosignal classification tasks by leveraging our findings by interpreting early exit neural networks as an ensemble of weight-sharing sub-networks. This work is motivated by the fact that adversarial examples across different clinical domains can easily manipulate deep learning models. Their security and privacy vulnerabilities raise concerns about the practical deployment of these systems. The number and variety of adversarial attacks grow continuously, making it difficult for mitigation approaches to provide effective solutions. Current mitigation techniques often rely on expensive re-training procedures as new attacks emerge.

Our experiments on state-of-the-art deep learning models show that early exit ensembles can provide robustness generalizable to various white box and universal adversarial attacks. The approach increases the accuracy of vulnerable deep learning models up to 60 percentage points while providing adversarial mitigation comparable to adversarial training. This is achieved without previous exposure to the adversarial perturbation or the computational burden of re-training. Here, we notice that when the attacker is aware of the intermediate exits, early exit ensembles can still provide adversarial robustness, although the mitigation is weaker. Nevertheless, this attack is very expensive to produce, circa 10x more expensive than the others that do not aim the early exits. In this scenario, we show that we can exploit the orthogonality of early exit ensembles to adversarial training to compensate for and build even more robust models. In summary, we can benefit from early exit ensembles on mitigating attacks not seen before without the need for retraining. Since mitigation also means making it more difficult and expensive for the adversary to consider the attack worthy, the presence of intermediate exits can achieve this goal.

Chapter 6—Conclusion and future work This chapter summarizes the contributions presented in the thesis, reflects on the new insights resented in the previous chapters, and outlines

limitations along with potential future research directions.

1.4 List of publications

The research efforts presented in this thesis have led to several publications at peer-reviewed conferences and workshops, which gave us the opportunity to present and discuss these ideas with relevant mobile systems, machine learning, and health-focused communities. Most of them correspond to the main contributions in Chapters 3, 4, and 5, while the remaining ones, while not directly related to this dissertation, are highly related in nature and aim to these contributions.

Works related to this dissertation

- **Qendro L.**, Mascolo C. ***Towards Adversarial Mitigation with Early Exit Ensembles.*** Proceedings of the 44th IEEE International Engineering in Medicine and Biology Conference, EMBC'22 (Qendro and Mascolo, 2022).
- Campbell A.*, **Qendro L.***, Lio' P., Mascolo C. ***Robust and Efficient Uncertainty Aware Biosignal Classification via Early Exit Ensembles.*** IEEE International Conference on Acoustics, Speech, & Signal Processing, ICASSP'22 (Campbell et al., 2022).
- **Qendro L.***, Campbell A.*, Lio' P., Mascolo C. ***Early Exit Ensembles for Uncertainty Quantification.*** Proceedings of Machine Learning Research (PMLR) Machine Learning for Health, ML4H'21 (Qendro et al.).
  [Best Thematic Paper Award] for best contribution to "Real-world robustness and generalization in machine learning for health".
- **Qendro L.***, Campbell A.*, Lio' P., Mascolo C. ***High Frequency EEG Artifact Detection with Uncertainty via Early Exit Paradigm.*** Workshop on Human In the Loop Learning, ICML'21 (Qendro et al., 2021a).
- **Qendro L.**, Chauhan J., C. P. Ramos A. G., Mascolo C. ***The Benefit of the Doubt: Uncertainty Aware Sensing for Edge Computing Platforms.*** Proceedings of the Sixth ACM/IEEE Symposium on Edge Computing, SEC'21 (Qendro et al., 2021b).
- **Qendro L.**, Ha S., de Jong R., Maji P. ***Stochastic-Shield: A Probabilistic Approach Towards Training-Free Adversarial Defense in Quantized CNNs.*** Proceedings of the 1st

Workshop on Security and Privacy for Mobile AI, MobiSys'21 (Qendro et al., 2021c).

*equal contribution

Other works

- Xia T., Han J., **Qendro L.**, Dang T., Mascolo C. ***Hybrid-EDL: Improving Evidential Deep Learning for Uncertainty Quantification on Imbalanced Data.*** Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022 (Xia et al.).
- Ferlini A.*, Ma D.*, **Qendro L.***, Mascolo C. ***Mobile Health with Head-Worn Devices: Challenges and Opportunities.*** IEEE Pervasive Computing Magazine 2022 (Ferlini et al., 2022).
- Tarkhani Z.*, **Qendro L.***, O'Connor Brown M., Hill O., Mascolo C., Madhavapeddy A. ***Enhancing the Security & Privacy of Wearable Brain-Computer Interfaces.*** arXiv preprint 2021 (Tarkhani et al., 2022).
- Xia T., Han J., **Qendro L.**, Dang T., Mascolo C. ***Uncertainty-Aware COVID-19 Detection from Imbalanced Sound Data.*** Proceedings of the Annual Conference of the International Speech Communication Association, Interspeech'21 (Xia et al., 2021).
- Montanari A., Sharma M., Jenkus D., Alloulah M., **Qendro L.**, Kawsar F. ***ePerceptive: energy reactive embedded intelligence for batteryless sensors.*** Proceedings of the 18th Conference on Embedded Networked Sensor Systems, Sensys'20 (Montanari et al., 2020).

*equal contribution

Chapter 2

Background

Recent advances in machine learning, specifically in the domains of computer vision, natural language processing, and signal processing, have greatly facilitated the development of automated medical diagnostic and screening tools. This has been made possible by breakthroughs in deep learning techniques and their ability to automatically learn high-level representations from raw data. Deep learning-powered health applications such as disease pre-screening from clinical images (Wang et al., 2021; Li et al., 2021), self-diagnosis through medical chatbots (Divya et al., 2018), and health monitoring using smartwatches (King and Sarrafzadeh, 2018) have achieved remarkable success. Notably, these models have become more widely available to consumers through mobile devices, enabling remote testing and expediting medical diagnosis beyond hospital settings (Qudah and Luetsch, 2019; Wood et al., 2019). Given the remarkable capabilities of deep learning techniques in various fields, we have decided to focus specifically on deep learning-based approaches in this dissertation.

This chapter starts with an introduction to the mobile health ecosystem in Section 2.1. While Section 2.2 reviews the machine learning theory used as a foundation for this work. It continues by defining and identifying the sources of predictive uncertainty in mobile health in Section 2.2.4. Section 2.4 presents the state-of-the-art uncertainty aware approaches used as both baselines and foundation theories for the frameworks proposed in this thesis. The chapter concludes with an overview of the edge computing platforms targeted in this work and their characteristics (Section 2.5). The purpose of this chapter is to make the thesis self-contained, but the material presented here is by no means exhaustive with respect to the corresponding research sub-fields.

2.1 The mobile health ecosystem

Mobile health aims at providing cost-effective healthcare support, delivery, and intervention to a large population via mobile technologies. Specifically, it relies on modern mobile devices to monitor human-related biometrics. It uses the collected data for automatic disease diagnosis and tracking, thus reducing healthcare costs by obviating physical presence at hospitals or direct contact with doctors. Mobile health enables a variety of services and applications, such as: collecting community and clinical health data; sharing of healthcare information with practitioners, researchers, and patients; real-time monitoring of patient vital signs; direct provisioning of care; and training and collaboration of health workers (Germanakos et al., 2005).

Figure 2.1.1 represents the typical architecture of a mobile health system. Mobile health targets various user groups, including patients with chronic diseases, elderly people with health risks, children requiring special care, and adults wanting a better understanding of their physical fitness. Smart devices (for instance, smartwatches, smart earbuds, or glasses) are equipped with multiple sensors (such as photoplethysmography, inertial measurement units, electroencephalography, etc.) which can measure vital signs and keep a record of daily activities. The collected data is then offloaded and stored in healthcare platforms based either in the cloud or in edge devices, depending on the device's compute capacity and privacy requirements. Finally, health experts in the form of automatic systems, human clinicians, or a combination of both interpret and analyze the data for a better understanding of the health conditions of the patient. In the case of *intervention*, the information flow would go the reverse way, with the (AI or human) doctors sending treatments or rehabilitation instructions (such as therapy and drug delivery) to the patients via the mobile health platform.

Within the mobile systems community, a considerable number of health applications stem from coupling machine learning and sensor data. Detection of depression is possible using audio and text in conversation (Al Hanai et al., 2018). In addition, the camera and speaker can serve as eyes and voices for people with vision and speech disorders (Smaradottir et al., 2018). Now we are seeing more ML-based mobile health applications deployed in the real world. A recent example is the automatic skin lesion system deployed as a mobile application for end users (Liu et al., 2020; Bui, 2021). Acknowledging that the impact and applications of mobile health can be countless, the scope of this thesis is to study more in-depth the opportunities/challenges of smart devices with connected sensors and their associated deep learning-based mobile health applications. Additionally, the examples mentioned above show how the predictive distributions of the deep learning models are increasingly being used to make decisions in important scenarios. Such

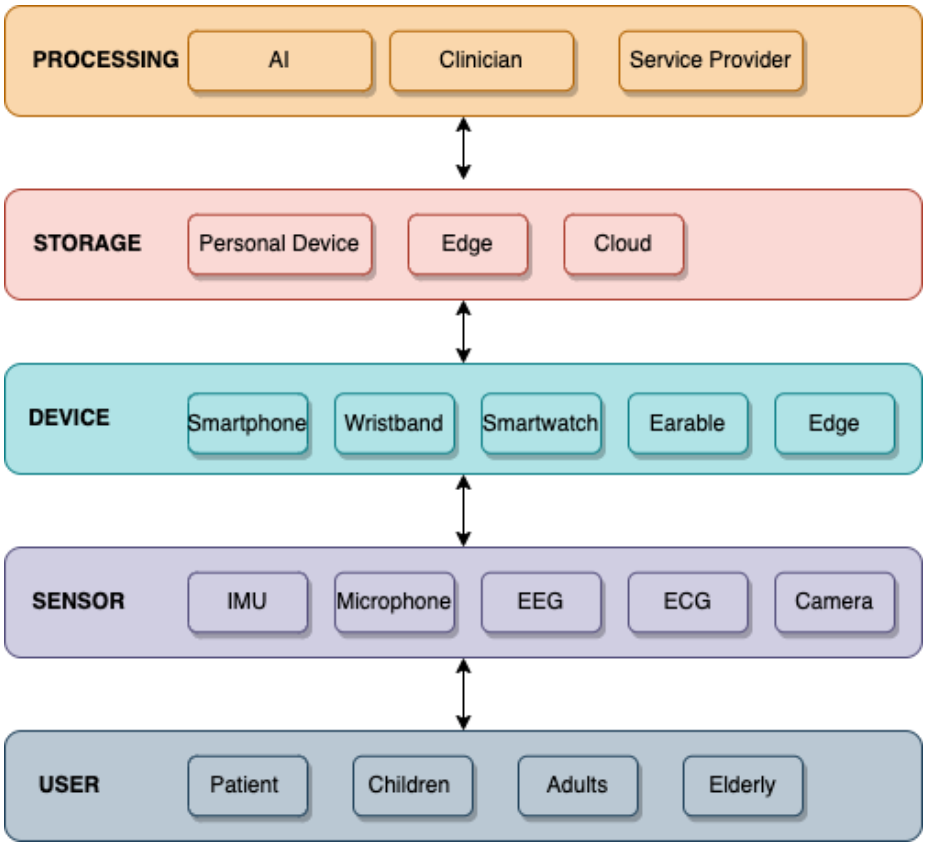


Figure 2.1.1: The architecture of a mobile health system. Sensing is provided by the multiplicity of sensors available in the mobile/wearable device(s). The collected data is stored and processed in the cloud, edge devices, or wearable for more privacy. Finally, a summary or clues are sent to the practitioner for a human-in-the-loop model, which provides the required human expertise for intervention.

high-stakes applications require not only accurate predictions but also accurate quantification of predictive uncertainty, i.e., meaningful confidence values in addition to class predictions. In such settings, calibrated predictive uncertainty is important because it enables accurate assessment of risk, allows practitioners to know how accuracy may degrade, and informs an automatic system to abstain from decisions due to low confidence. Uncertainty aware deep learning health applications not only provide increased accuracy and robustness (Leibig et al., 2017) but also help in establishing trust in these methods. In the following sections, we define predictive uncertainty and introduce the building blocks needed to build the uncertainty aware frameworks proposed in this thesis.

2.2 Deep neural networks

Deep learning aims to train a model on a set of data known as a training set and enable it to perform well on new and unseen data, referred to as a test set. The model's ability to achieve this objective is known as generalization, and its performance is typically quantified by a measure such as a cost function (or error). To evaluate the generalization capabilities of a given model, the difference between the test and train errors, referred to as the generalization gap, is calculated. Poor performance on the training set is termed as underfitting, whereas a substantial generalization gap is referred to as overfitting, where the test error is greater than the training error. To strike a balance between underfitting and overfitting, machine learning practitioners modify the model's capacity, which determines the model's ability to fit different functions.

A model with higher complexity, e.g. model parameters, usually also has a higher capacity. A machine learning algorithm performs best when its capacity is appropriate for the complexity of the task and the availability of data. To achieve optimal performance, a machine learning algorithm must strike a balance between the model's capacity, the task's complexity, and the amount of available data. The algorithm performs best when the capacity is appropriately matched to these factors. However, measuring performance only on these terms would bring several issues during deployment. For optimal performance in real-world scenarios, a deep learning model must provide indications of uncertainty when the prediction is uncertain and capable of handling data outside of its training data distribution. In this section, we present two fundamental deep learning models: multi-layer perceptrons and convolutional neural networks. Furthermore, we provide an introduction to early exit neural networks, which form the basis of the deep learning models used in Chapter 4 and 5.

2.2.1 Multi-layer Perceptrons

Multi-layer perceptrons (MLPs), are a class of machine learning models whose aim is to learn an optimal set of parameters θ to best approximate a function $f_\theta(\mathbf{x})$. In classification tasks, studied throughout this thesis, the neural network computes $y = f_\theta(\mathbf{x})$ which maps the input $\mathbf{x}$ to a category or real value y. A neural network consists of a composition of L differential operations or layers, such that:

$$\begin{aligned} f_\theta(\mathbf{x}) &= (f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)})(\mathbf{x}) \\ y &= (f^{(L)}(\ldots f^{(2)}(f^{(1)}(\mathbf{x}))) \end{aligned} \tag{2.1}$$

where $(f^{(i)} \circ f^{(j)})(\mathbf{x}) = f_{\theta_i}(f_{\theta_j}(\mathbf{x}))$ denotes function composition for $i \neq j$ and $\theta = \cup_{i=1}^{L} \theta_i$.

Let $\mathbf{h}^{(i)}$ denote the intermediary output of the i-th layer such that $\mathbf{h}^{(i)} = f_{\theta_i}(\mathbf{h}^{(i-1)})$, and $\mathbf{h}^{(0)} = \mathbf{x}$. A simple fully-connected layer i computes a linear transformation of its input $\mathbf{h}^{(i-1)}$:

$$\mathbf{h}^{(i)} = \mathbf{W}^{(i)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(i)} \tag{2.2}$$

The output of the fully-connected layer is then passed through an activation function, which accounts for non-linear patterns in the input features. A widely used activation function is the rectified linear unit (ReLU (Glorot et al., 2011)):

$$\text{ReLU}(\mathbf{h}^{(i)}) = max(0, \mathbf{h}^{(i)}) \tag{2.3}$$

Last, the output layer of the neural network activation can categorize the inputs via a special activation function, the softmax function such as:

$$\text{softmax}(\mathbf{h}^{(i)}) = \frac{\exp\left(\mathbf{h}^{(i)}\right)}{\sum_{c=1}^{C} \exp\left(\mathbf{h}^{(c)}\right)} \tag{2.4}$$

where C denotes the classification categories and the predicted class is defined by

$$y = \text{argmax}(\text{softmax}(\mathbf{h}^{(i)})). \tag{2.5}$$

2.2.2 Convolution Neural Networks

In MLPs, each neuron interacts with each element of the input introducing a high number of model parameters in each neuron and hurting the statistical generalization ability of the network. Convolutional neural networks (CNNs) were introduced to solve this issue by exploiting the idea of locality and spatial structure in inputs. A CNN contains a sequence of alternating convolutional and pooling layers. The convolutional layers output feature maps, which are then downsampled by the pooling operations building an increasingly compact representation of the input. A summarization layer is then applied, with the resulting feature vector being fed to an MLP that outputs predictions.

A convolutional layer takes as input a multidimensional tensor that contains intermediate features (or, in the case of the first network layer, the raw signal). The tensor is passed through a set of

kernels, each of them computing its own feature map of the input. The input to the layer is represented by $\mathbf{x} \in \mathbb{R}^{(H,W,C)}$ with height H, width W and C channels. Let $\mathbf{w} \in \mathbb{R}^{(h,w,c,f)}$ be the weight matrix with height h, width w, c channels and f filters, and $\mathbf{b} \in \mathbb{R}^{(f)}$ the bias vector. Consequently, the output will be represented as

$$\mathbf{y}_{[:,:,k]} := \mathbf{x} * \mathbf{w}_{[:,:,:,k]} + \mathbf{b}_{[k]} \tag{2.6}$$

for $\mathbf{y} \in \mathbb{R}^{(H-h+1,W-w+1,f)}$ and for $k = 1, \ldots, f$ where $*$ is the convolution operation.

The pooling operation, which follows the feature computation stage, compresses the feature representation by computing summary statistics for each input location. This step not only reduces the memory of the model but also enforces a translation-invariant representation, where small shifts in the inputs do not affect the resulting feature map.

2.2.3 Training neural networks

To train a neural network and learn the parameters θ, a differentiable cost function $J(\theta)$ is introduced as the objective function for an optimization algorithm. The parameters which minimize the cost function, $J(\theta)$, are considered to best fit the model. A very common guiding principle for training is the maximum likelihood, shown to be the best estimator asymptotically as the number of training examples goes to infinity, in terms of rate of convergence (Rao, 1992).

In a classification task, the learning procedure aims to minimize the negative log-likelihood cost function J that encodes the cross-entropy between the true, underlying data distribution, p_{data}, and the distribution modeled by the network, p_{model}. The cost function can be written as an average over the training set:

$$J(\theta) = \mathbb{E}_{\mathbf{x},y\sim\hat{p}_{data}} \mathcal{L}(f_\theta(\mathbf{x}, y)) \tag{2.7}$$

where $\mathcal{L}$ is the per-example loss function and $\hat{p}_{data}$ is the empirical data distribution.

We aim to reduce the expected generalization error by minimizing the empirical risk over the training dataset. Minimizing the cost function, $J(\theta)$, is achieved with an optimization algorithm that iteratively updates the parameters. For this task, the most commonly used property is the gradient $\nabla_\theta J(\theta)$. For the statistical estimation of the gradient, the expectation is computed by randomly sampling a small number of examples from the dataset, i.e., a batch b, then taking the average over only those examples. Most optimization algorithms converge faster using such

approximate estimates of the gradient. A canonical example is the stochastic gradient descent (SGD) which optimizes the network parameters using unbiased estimates of the exact gradient of the generalization error:

$$\theta = \theta - \mu \frac{1}{b} \nabla_\theta \sum_{i=1}^{b} \mathcal{L}(f_\theta(\mathbf{x}_i), y_i) \tag{2.8}$$

In summary, before training a neural network model on a dataset, the data is first split into batches, the batches are shuffled, and then a minibatch variant of SGD is run until all batches are exhausted. An entire pass through the training set is called an epoch, with training typically spanning multiple epochs. The number of epochs can vary, depending on the size of the dataset, the number of model parameters, the type of optimization algorithm used, and the learning rate μ employed. $\nabla_\theta J(\theta)$ is calculated using *backpropagation* by applying the chain rule for partial derivatives, starting from the output layer towards the input.

Training a deep learning model includes making decisions on the values of hyperparameters, such as the depth of a network or the architecture size. Usually, a grid search is performed over a hyperparameter range, in which the model is trained with a certain hyperparameter setting, and then the model performance on the validation set is recorded. Finally, the highest-performing hyperparameter tuning is chosen, and the respective model is tested on the test set.

2.2.4 Early exit neural networks

Most deep learning models offer a fixed computational graph that stays unchanged at inference time. This might not always be optimal as, in many cases, the system has to deal with various computation requests. In mobile and wearable devices, the battery imposes a limitation on the system's functionality; intuitively, a fixed-size network becomes a waste of resources or the cause for the system not being able to deliver the prediction. Early prediction models are a class of conditional computation models that exit once a criterion (e.g., sufficient accuracy) is satisfied at early layers. Leveraging that not all inputs are equally easy/difficult to process, they dynamically change their computational graph at inference time based on the input's complexity and the prediction confidence. Early exit networks comprise a backbone architecture and additional intermediate exit blocks (or classifiers) along its depth. At inference time, when a sample propagates through the network, it flows through the backbone and each of the exits sequentially, and the result that satisfies a predetermined criterion (exit policy) is returned as the prediction output,

without computing further the rest of the model. The exit policy can be informed not only by the output accuracy but also by the target device capabilities and load and dynamically adapt the network to meet specific runtime requirements (Montanari et al., 2019, 2020; Laskaridis et al., 2020a).

Early prediction methods embed a gating mechanism that informs the dynamic change of the neural network computation graph at inference time. For example, in Huang et al. (2017), and Teerapittayanon et al. (2016), the decision to exit the network is taken based on the output accuracy of each branch, meaning that the branches need to be executed until the end to determine if the input should exit or not. For inputs that need to exit at deeper branches, this constitutes overhead and wasted computation/energy. Similarly, in Bolukbasi et al. (2017), the network contains decision functions at the branching points to decide if an example should go out or continue further down the network. These functions are implemented as dense layers and trained iteratively one after the other. To address this computation overhead, other approaches (Wang et al., 2019; Li et al., 2019; Nan and Saligrama, 2017) propose a gating (exit) policy in the backbone. While more efficient, these approaches make an exit decision based on the features of the backbone model, assuming that the exit blocks do not contain any learning capacity materialized by more parameters or extra layers.

The design choice for the model architecture is crucial as it later affects the network's capacity and learning process, as each architecture scales and converges differently. The vast majority of early exit approaches start with a state-of-the-art architecture later modified by intermediate exits. The number and position of the early exits along the depth of the network affect both the granularity of results and the parameter/computation overhead compared to the vanilla approach (backbone only). Depending on the application and desired accuracy, intermediate exits can be placed equidistantly or at variable positions across the depth of the network; therefore, there is no predefined general rule on where to exit. Recent work proposes a way to define exit points based on the floating point operations (FLOPs) or parameters in the network (Kaya et al., 2019) allowing for a customizable distribution of the exits. Nevertheless, where to exit is a hyperparameter that should be tuned based on the application at hand. An architecture with too many exits can yield extreme overhead; instead, if the exits are too sparse, it would undermine the ultimate efficiency goal as the output takes longer to process. Moreover, too many early classifiers can negatively impact convergence when training end-to-end.

Training with early exits mainly falls into two categories: *end-to-end* and *exit-only* training. The *end-to-end* training represents a joint training of the network as a whole, including the backbone

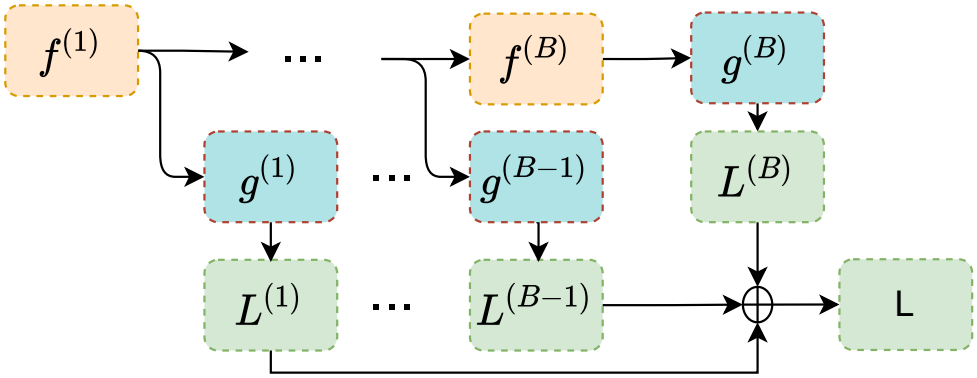


Figure 2.2.1: Training with early exits. Each exit branch $g^{(i)}$ propagates the prediction error in relation to the ground truth label $L^{(i)}$, training the blocks that precede each exit point $f^{(i-1)}$ for $i = 1, \ldots, B$.

and exit blocks. In this procedure, the loss function is a composition of the individual predictive losses of each exit (see Figure 2.2.1). As such, each prediction propagates the error in relation to the ground truth label to the blocks preceding that exit. This can be represented as a weighted sum

$$L = \sum_{i=1}^{B} \alpha_i L^{(i)}(\mathbf{y}, \hat{\mathbf{y}}^{(i)}) \tag{2.9}$$

where $L^{(i)}(\cdot, \cdot)$ is the loss function for the i-th block's exit and $\alpha_i \in \{0, 1\}$ is a weight parameter corresponding to the relative importance of the exit. Increasing the weight at a specific exit point forces the network to learn better features at the preceding layers (Scardapane et al., 2020). This training paradigm can increase accuracy for both the intermediate exits and backbone. However, this is not always guaranteed due to cross-talk between exits (Laskaridis et al., 2020b).

The second training approach consists of attaching early exits to existing pretrained models such that the backbone and the exits are trained separately (Equation 2.10). This procedure yields higher flexibility as the exits can be arbitrarily placed at a later stage (deployment phase), as well as higher efficiency as the exit blocks have fewer parameters allowing for on-device training, for instance. However, this training approach is more restrictive and thus can present lower accuracy than an optimized jointly trained variant.

$$L^{(i)}_{exit\_only} = L^{(i)}(\mathbf{y}, \hat{\mathbf{y}}^{(i)}) \tag{2.10}$$

All the aforementioned previous work is based on the traditional use of the early exit paradigm

as a framework for efficient computation. However, in Chapter 4 and Chapter 5, we give early exit neural networks a completely different interpretation in two important robustness aspects for deep learning models, such as uncertainty quantification and adversarial mitigation.

2.3 Predictive uncertainty

The predictions made by traditional deep neural networks, as described in Section 2.2, are typically point estimates. This is the case for regression networks and classification networks with a softmax layer. While the softmax layer can provide class probabilities, it is primarily a smooth approximation to the arg max function, which identifies the index with the maximum value (see Equation 2.5 and Equation 2.4). The softmax function predicts which classes are more likely relative to each other but does not provide information on how confident the network is in its prediction. The softmax probability only captures the relative probability of the input belonging to a particular class compared to other classes, but it does not reflect the overall model confidence. As a result, softmax outputs are not well calibrated and can be problematic (Gal and Ghahramani, 2015a; Guo et al., 2017). In contrast, predictive uncertainty provides a more informative interpretation of the inference as it can indicate whether the deep learning model is certain about its prediction or is just guessing randomly. While interpretation encompasses an understanding of all the elements that contributed to the prediction, uncertainty estimation is the first step to detecting anomalies and serves as a trigger for further investigation.

2.3.1 Shortcomings of softmax in predicting class probabilities

An exemplary benchmark for deep neural networks is to employ the softmax function to transform the continuous activations of the output layer into probabilities assigned to each class. The resultant model can be represented as a multinomial distribution, where the neural network's outputs specify the parameters, and hence the discrete class probabilities.

The neural network dynamically updates the probabilities of the different classes, but these probabilities are compressed into a simplex using the softmax function. As a result, the final output is a multinomial probability distribution that is optimized based on the neural network's parameters. To accomplish this optimization task, the cross-entropy loss is minimized, which has a probabilistic interpretation as a frequentist maximum likelihood estimation. However, this approach is

limited in that it cannot provide information on the variance of the predictive distribution.

The softmax function is known for overestimating the probability of the predicted class due to the exponent applied to the neural network outputs. This can lead to unreliable estimations of uncertainty because the distance of a newly observed label from the predicted label is not informative in and of itself, apart from its relative value compared to other classes. Numerous studies (Gal and Ghahramani, 2015a; Louizos and Welling, 2017; Sensoy et al., 2018), including our own in this dissertation, have demonstrated that even small and straightforward modifications to the input, such as adding noise or rotation, can lead to incorrect predictions by the model. Even for the misclassified samples, the probability of classification calculated using the softmax function is often quite high showing that softmax is inadequate in incorporating uncertainty in the prediction.

2.3.2 Uncertainty in mobile health applications

Uncertainty is manifested in various forms in mobile health applications. In traditional mobile systems, it can be at the physical layer due to node mobility, network topology, routing, and resource availability (Das and Rose, 2005). In medical EEG systems, both ambulatory and mobile can be caused, for instance, by biological calibration, signal acquisition, sensitivity settings, and human error and bias in the annotation. These factors, together with the sensor or electrode measurements, calibration, software, and device heterogeneity, add variability to the system feeding uncertainty. When using deep learning, the uncertainty induced by the model architecture and parameters is an additional factor that jeopardizes the trust in the prediction. Observed data can be consistent with many models, and therefore which model is appropriate, given the data, is uncertain. Similarly, predictions about future data and the future consequences of actions are uncertain.

2.3.3 Uncertainty estimation

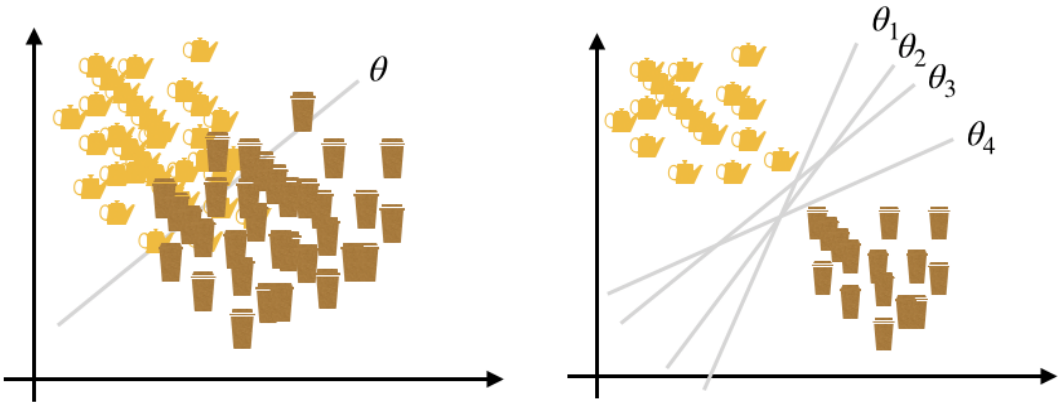
In the field of machine learning, it's often assumed that we possess knowledge of the loss function and can train algorithms to output decisions that minimize expected loss. However, in real-life situations, the loss function may be unknown or dependent on other factors that are determined later on. In such cases, it is critical to express uncertainties to prevent valuable information

from being lost. One way to capture uncertainty is to treat the unknown quantity of interest as a random variable and make predictions in the form of probability distributions, also known as predictive distributions. This discussion will focus on this particular method of representing uncertainty, as it's the interpretation used in this dissertation inspired by Quinonero-Candela et al. (2005). However, the questions of how to generate reasonable predictive uncertainties and what constitutes a reasonable predictive uncertainty still need to be addressed.

The Bayesian framework represents model parameters using posterior distributions that capture both noise-induced uncertainty and lack of knowledge about the true model. By integrating over this posterior, we arrive at the posterior distribution of the variables of interest, which leads to the predictive distribution. However, the choice of the prior distribution is critical to the validity of the resulting predictive distribution. In a frequentist framework, the problem is reduced to selecting the correct prior, whereas a Bayesian framework acknowledges that priors reflect subjective degrees of belief, rather than absolute truths. Sometimes, for convenience, overly restrictive priors are used, which can eliminate plausible hypotheses about the origin of the data and quantify inaccurate predictive uncertainties. Thus, it is crucial to use a reasonable prior that reflects a thorough understanding of the problem at hand. If one is confident in the prior's validity, then the resulting predictive distribution should be equally valid. However, it is essential to remember that the predictive distribution represents an updated belief and its accuracy will depend on how well the prior reflects the reality of the situation.

In traditional machine learning, the complete posterior distribution is simply disregarded and the focus is brought into *maximum a posteriori (MAP)* approaches which are tantamount to maximum penalized likelihood. However, the predictions made using the MAP approach lack any measure of uncertainty. Therefore, alternative methods for obtaining predictive uncertainties, like bagging (Breiman, 1996), bootstrap (Efron, 1982), ensembles (Lakshminarayanan et al., 2016), or dropout (Gal and Ghahramani, 2015a), are needed. Another option is to adopt simpler approaches that always generate the same predictive uncertainties regardless of the input, based on an estimate of the overall generalization error. This generalization error can be estimated empirically via cross-validation or theoretically through statistical learning bounds on the generalization error. Undoubtedly, any method that independently estimates predictive uncertainties based on the input has the potential to be superior to this simplistic approach.

Overall, it seems that generating accurate estimates of predictive uncertainty in machine learning is not a straightforward task. Additionally, there is a noticeable absence of agreement on the most effective methods for assessing predictive uncertainties. As a result, the field is currently priori-



(a) Aleatoric Uncertainty (b) Epistemic Uncertainty

Figure 2.3.1: Predictive uncertainty decomposed into epistemic (model) and aleatoric (data) uncertainty. The lines represent the deep learning models with parameter θ which can fit the data.

tizing the validation of different approaches through practical application rather than theoretical examination.

2.3.4 Types of uncertainties

In our work, we define *predictive uncertainty* as the probability of the target variable occurring given all available information. Unlike traditional approaches, we view the predictions of deep learning models as random variables (Quinonero-Candela et al., 2005). As a result, we represent predictions as probability distributions rather than point estimates. These distributions capture all the sources of uncertainty in a model, including structural, parametric, and noise-related uncertainties.

There are two primary types of uncertainty: *aleatoric* and *epistemic* (Der Kiureghian and Ditlevsen, 2009; Gal et al., 2022). *Aleatoric* uncertainty (shown in Figure 2.3.1b) is associated with noise in observations caused by factors such as sensor, motion, or label noise. This uncertainty cannot be reduced even if more data is collected. Aleatoric uncertainty is not influenced by the choice of the machine learning method. Even with the most sophisticated methodology, it is impossible to precisely determine the true value of a quantity that is only observed through a limited number of noisy measurements. To account for this type of uncertainty, it is necessary to model its distributions which can be done via various techniques such as neural density estimators or deep generative models. On the other hand, *epistemic* uncertainty (shown in Fig-

ure 2.3.1a) arises from our ignorance about which model generated the collected data, and it represents uncertainty in the model parameters. Typically, when estimating epistemic uncertainties in neural networks, ad hoc priors are employed for both the network architecture and the model weights. However, these priors do not have a direct interpretation in terms of physically meaningful priors on the functional space of the neural network output. Nonetheless, they can be valuable for detecting cases where a model is poorly constrained by data. In such situations, active learning strategies may be employed to acquire more data. In general, once the constraints of an application are understood, it is crucial to select appropriate metrics for quantifying how well the uncertainty is modeled. While frequentist statistical tools such as expected calibration error (Guo et al., 2017) can be used to quantify aleatoric uncertainty, epistemic uncertainty is inherently subjective and cannot be measured using such tools. A model can be well calibrated for aleatoric uncertainty but still produce meaningless epistemic uncertainty, and vice versa.

It is important to note that while the two types of uncertainty mentioned are the most relevant in machine learning, the use of approximation methods inevitably introduces approximation error. This is due to the limited computation available, which introduces yet another type of uncertainty, known as computational uncertainty. These approximation methods use a reasonable amount of computation to obtain an approximation of the mathematical posterior, which is theoretically possible but cannot be accessed given limited computational resources. The approximate posterior is then used as a direct replacement for the mathematical posterior in downstream tasks. It's crucial to recognize that limited computation induces approximation error captured by computational uncertainty, and accounting for it would generally enhance uncertainty quantification (Wenger et al., 2022). We leave the incorporation of computational uncertainty into our framework as future work.

Our approach to *predictive uncertainty* considers both *aleatoric* and *epistemic* uncertainties in a unified framework. By modeling both types of uncertainties, we gain a more comprehensive understanding of the underlying system, and we are better equipped to make accurate predictions.

2.3.5 Applications of uncertainty

The use of the estimated uncertainty is multi-fold. Firstly, using uncertainty aware approaches can provide better-calibrated models, which can give an extra indication if they are guessing at random to trigger further investigation. This further investigation could be performed by a clinician in a human-in-the-loop system where uncertain samples are sent to an expert, ensuring that the whole system, automatic and human-aided, is accurate. Uncertainty can be the metric of the

decision on how to choose samples for labeling in an active learning scenario to avoid sending too many samples and overwhelming the user. Furthermore, uncertainty can be used for balancing the computation in a distributed multi-tier (cloud) architecture in the presence of uncertainty. High uncertainty predictions might need further computation so that the sample might be sent for processing to the cloud or on bigger-capacity devices. Finally and most importantly, it can detect distributional shifts from deploying on different devices or environments, which can then be included in a retraining procedure.

Human-in-the-loop

Incorporating a human-in-the-loop system when deploying automated decision support is critical to providing reliable performance. Given the vital importance of health applications, uncertainty quantification is crucial not only to inform the confidence of the prediction but also to provide the opportunity to keep clinicians in the loop to correct the potentially wrong (or uncertain) automatic predictions. Leibig et al. (2017) found that when using deep learning to diagnose diabetic retinopathy from fundus images of the eye, incorrect prediction usually yielded higher uncertainty than true predictions therefore by excluding the most uncertain predictions, the automatic diagnosis' accuracy could be improved by 13 percentage points (from 87% to 96%) (Singh et al., 2021). Diabetic retinopathy is one of the leading causes of blindness in the working-age population of the developed world, so uncertainty estimations unlock two opportunities: (a) it provides better overall predictions, and (b) it allows for real-world deployment following recommendations from regulatory entities. For example, recommendations of the British Diabetic Association (now Diabetes UK) should comply with an 80% sensitivity and 95% specificity (Antal and Hajdu, 2014; Kapetanakis et al., 2015). Without a measure of uncertainty to isolate unreliable predictions, this algorithm would not be deployed, given the regulation thresholds, delaying symptom detection.

Active Learning

Supervised machine learning is revolutionizing healthcare, but acquiring labeled data in medical settings is a costly, long, and laborious process. Active learning (Cohn et al., 1996) (a special case of a human-in-the-loop system) provides a framework that allows the model to learn from a small amount of data and automatically select the most informative data points increasing its overall accuracy. In this paradigm, the model is initially trained on a small proportion of the

training data and leverages an acquisition function that decides what data points to ask labels on the fly (to an external automatic or human oracle). The acquisition function selects several points from a set of data points that lie outside the training set.

There is three main approaches of active learning representing different querying algorithms (Settles, 2009):

- **Membership query synthesis (Angluin, 1988).** In this scenario, during training, the model requests labels for any unlabeled sample in the input space, including new samples that the learner generates, rather than those sampled from an underlying natural distribution. The active learner can generate synthetic data and its own examples for labeling. This strategy might lead to asking for labels for arbitrary samples (such as unrecognizable symbols or artificial hybrid characters for handwritten character classification tasks) which are impossible to annotate by a human oracle (Baum and Lang, 1992). The following strategies have been proposed to solve this issue.
- **Stream-based selective sampling (Atlas et al., 1989; Cohn et al., 1994).** This approach is sequential in nature. The model is presented with a single data point and immediately must make the decision whether to create a query or discard it. The benefit of selective sampling relies on the fact that even if the input distribution is unknown and non-uniform, the queries still come from a real underlying distribution and are therefore significant for human annotators too.
- **Pool-based sampling (Lewis, 1995)** This sampling method is the most well-known scenario for active learning. It also fits very well for many real-world problems including those in the mobile health space where the labeled data is scarce but large collections of unlabeled data can be gathered at once. The pool-based active learning cycle starts with the selection of a small number of samples in the labeled training set $\mathcal{A}$. The learner then asks the oracle to label another set of carefully selected instances from the unlabeled set $\mathcal{B}$, which are further added to the labeled set $\mathcal{A}$. This process activates another learning iteration with the recently labeled data in a standard supervised manner allowing the learner to leverage the new knowledge for choosing which samples to query next. This process is repeated, with the training set increasing in size over time. Typically, samples are queried in a greedy fashion, according to an acquisition function used to evaluate all (or a subset of all) samples in the pool. This approach is sequential in nature to a certain extent however in contrast to selective sampling it does allow for managing a large set of unlabelled data all at once. It comes with the disadvantage of requiring more memory to store the data in

the pool, however different querying strategies can help with solving this issue.

All active learning scenarios involve leveraging an acquisition function whose key purpose is to determine which unlabeled data points need to be labeled, so it requires a way of assessing how informative each of them is (Settles and Craven, 2008). Uncertainty sampling is one of the most frequently used approaches to achieve that (Lewis and Catlett, 1994), as it only queries the clinician to label a data point if its classification uncertainty is significant. The intuition here is that the most informative examples are the ones that the classifier is more uncertain about. Specifically, the examples that lie near the class boundaries. If these difficult examples are chosen for the next training iteration the model will gain the most knowledge about the class boundaries compared to choosing more confident samples. In contrast to other learning approaches such as passive learning, which relies on random sampling (Figueroa et al., 2012), an uncertainty-based strategy is the best option for interactively labeling data points as it makes informed decisions reducing the overall labeling cost (Hong et al., 2020).

Distributed computation

While flagship devices can support the performance requirements of deep learning workloads, the current landscape is still very diverse, including previous generation and low-end models (Wu et al., 2019). In this context, the less powerful low-tier devices struggle to consistently meet the application-level performance needs. To address these limitations, two recent lines of work have been proposed to enhance the collaboration between device and cloud for model inference. One line of work (Hu et al., 2019; Kang et al., 2017; Li et al., 2019) treats the deep learning model as a computation graph and partitions it between device and cloud. At run time, the personal/edge device executes the first part of the model and transmits the intermediate results to a remote server. The server continues the model execution and returns the final result back to the device. The second type of approach is based on progressive inference; a mechanism that allows the system to exit early at different parts of the model during inference, based on the input complexity or device computational capacity (Montanari et al., 2020; Teerapittayanon et al., 2016; Bolukbasi et al., 2017). Different from the first class of approaches, these techniques always provide an output that might be intermediate (on the tiny device) or refined (on the cloud or bigger capacity device). The gating mechanism adopted is often based on the accuracy (based on a single softmax output) or/and the current battery of the device. However, the gating mechanism of these progressive approaches would benefit from incorporating the predictive uncertainty, which would provide a more accurate device-cloud system as well as a more principled way of deciding if the current local prediction can be trusted or needs further exploration on a bigger model

in the cloud. An uncertainty aware approach in this context can help in building a hierarchical inference, too, by providing a more abstract – and therefore easier to materialize – prediction (e.g., "steps") before specializing their prediction in deeper (more complex) models (e.g. "steps up" or "steps down") (Bilal et al., 2017; Zamir et al., 2017). This paradigm allows for fast inference when a very specific classification is not needed (depending on the task). It also helps in user retention as they are always presented with a prediction, but the specific prediction is only presented when the model is confident.

Out-of-distribution (OOD) detection

Despite the impressive performance achieved by deep learning, the underlying premise is that training and test data are from the same distribution. Yet, this assumption does not often hold in practice because dataset shift caused by user variability, device discrepancy, artifacts, and other factors is inevitable during in-the-wild signal (data) acquisition (Pooch et al., 2020). Most existing deep learning models cannot capture this distributional shift and tend to be overconfident during inference which may, in the long run, undermine trust in applying deep learning for healthcare (Guo et al., 2017). As it is not feasible to build a dataset that guarantees the coverage of all possible distributions for open-world scenarios, approaches to detect OOD inputs are important to ensure the safety of the overall system. For this purpose, an uncertainty aware deep learning approach would be able to isolate OOD samples and monitor covariate shift because a well-calibrated model will manifest much lower confidence when presented with unseen data from a distribution not included in the training procedure (Ovadia et al., 2019).

2.3.6 Uncertainty evaluation metrics

Throughout this dissertation, we use many evaluation metrics to better showcase the performance of our approaches in terms of accuracy and uncertainty quantification. The classification performance is evaluated based on class-weighted F1, precision, and recall. For uncertainty quantification, we use negative log-likelihood (NLL), Brier score (BS) (Brier et al., 1950), and expected calibration error (ECE) (Naeini et al., 2015). NLL and BS are proper scoring rules therefore their optimum values correspond to perfect predictions. Although not a proper scoring rule, ECE is intuitive and commonly used. The out-of-distribution behavior is evaluated using predictive confidence (PC) and predictive entropy (PE) since there is no ground truth label for fully out-of-distribution input.

F1 score is the harmonic mean of precision and recall, used to measure a model's accuracy.

$$F1 = \frac{2}{precision^{-1} + recall^{-1}} = 2\frac{precision * recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}, \tag{2.11}$$

where TP, FP, and FN represent true positive, false positive, and false negative predictions respectively.

Precision is the fraction of true positive samples among the samples classified as positive.

$$precision = \frac{TP}{TP + FP} \tag{2.12}$$

Recall (or sensitivity) is the fraction of samples classified as positive among the total number of positives.

$$recall = \frac{TP}{TP + FN} \tag{2.13}$$

Negative log-likelihood (NLL) measures how likely it is to observe the data under the model. NLL is defined as:

$$\begin{aligned} \text{NLL} = -\sum_{c \in \{1,\ldots,C\}} & 1(y = c) \log p_\theta(y = c|\mathbf{x}) \\ & +(1 - 1(y = c)) \log(1 - p(y = c|\mathbf{x})) \end{aligned} \tag{2.14}$$

where $1(\cdot)$ is the indicator function.

Brier score (BS) measures the accuracy of predicted probabilities. BS is defined as:

$$\text{BS} = \sum_{c \in \{1,\ldots,C\}} (p_\theta(y = c|\mathbf{x}) - 1(y = c))^2. \tag{2.15}$$

Expected calibration error (ECE) measures the expected difference (in absolute value) between accuracies and the predicted confidences on samples belonging to different confidence intervals. ECE is defined as:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{2.16}$$

where accuracy and confidence for bin B_m are

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{n \in B_m} 1(\hat{y}_n = y_n)$$
$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{n \in B_m} \text{p}_{c_n}$$

such that $\hat{y}_n = \arg\max_{c \in \{1,\ldots,C\}} p_\theta(y_n = c|\mathbf{x}_n)$ is the predicted class, M is the number of bins of size $1/M$, and bin B_m covers the interval $(\frac{m-1}{M}, \frac{m}{M}]$.

Predictive confidence (PC) is the probability of the top-class prediction. PC is defined as:

$$\text{PC} = \max_{c \in \{1,\ldots,C\}} p_\theta(y = c|\mathbf{x}) \tag{2.17}$$

Predictive entropy (PE) measures the average amount of information in the predicted distribution. PE is defined as:

$$\text{PE} = -\sum_{c \in \{1,\ldots,C\}} p_\theta(y = c|\mathbf{x}) \log p_\theta(y = c|\mathbf{x}) \tag{2.18}$$

2.4 Bayesian Deep Learning

MacKay (1992) showed the several potential benefits of a Bayesian approach to neural network learning; in particular, this work contained a convincing demonstration of naturally accounting for model flexibility in the form of the Bayesian Occam's razor, aiding comparison between different models, *accurate calibration of predictive uncertainty*, and to perform learning robust to over-fitting. This initial exploration of Bayesian neural networks led to two main approaches being developed. The first variational inference approximation, the variational Bayes (VB) approach for posterior inference (Hinton and Van Camp, 1993) was motivated by the use of minimum description length (MDL) to penalize a high volume of information contained in the weights. Second, Neal (1993) developed efficient gradient-based Monte Carlo methods, now known as Hamiltonian Monte Carlo (HMC). HMC is a method based on a dynamical simulation that does not rely on any prior assumption about the kind of posterior distribution. These are the first steps toward practical inference in Bayesian neural networks but are difficult to adapt to

modern practical needs.

2.4.1 Modern approaches to Bayesian neural networks.

Modern research in Bayesian NNs relies on either sampling-based techniques or the different flavors of variational inference. The field was restored by using Monte Carlo variational inference (MCVI) (Graves, 2011) to make variational Bayes practical and scalable, confirming gains in predictive performance in real-world tasks. Blundell et al. (2015) introduced *Bayes by Backprop* basing their work on Graves (2011) and further altering the Bayesian model by introducing a mixture of Gaussians prior to each weight and optimizing the mixture components. Although they improved the model performance, this variational approach is still computationally costly and difficult to use on edge devices. As an alternative to VB, probabilistic backpropagation (PBP) (Hernández-Lobato and Adams, 2015) employs approximate inference in the form of assumed density filtering (ADF) to refine a Gaussian posterior approximation. Each update to the approximate posterior requires propagating means and variances of activations through the network. Another rich family of approximate inference methods for Bayesian neural networks are stochastic gradient Markov chain Monte Carlo (SG-MCMC) methods (Welling and Teh, 2011; Ahn et al., 2012; Gong et al., 2018). These techniques allow for approximate posterior parameter inference using unbiased log-likelihood estimates.

The above methods are principled but often require sophisticated implementations - recently, a few methods aim to provide efficient approximations to the Bayes posterior. Bootstrap posteriors (Lakshminarayanan et al., 2017; Fushiki et al., 2005) have been proposed as a comprehensive, robust, and accurate method for posterior inference. However, obtaining a bootstrap posterior ensemble of size M is computationally intense M times the computation of training a single model. A few important and interesting methods used dropout as a Bayesian approximation. Dropout (Hinton et al., 2012; Srivastava et al., 2014) randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples form an exponential number of different ”thinned” networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network with smaller weights. This significantly reduces over-fitting and gives major improvements over other regularization methods. Gal and Ghahramani (2015b) show that a neural network with arbitrary depth and nonlinearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process (marginalized over its covariance function parameters).

They present Monte Carlo dropout sampling, which places a Bernoulli distribution over the network's weights. In practice, this means we can train a model with dropout. Then, at test time, we can stochastically sample from the network with different random dropout masks rather than performing model averaging. The statistics of this distribution of outputs will reflect the model's uncertainty. Concrete dropout (Gal et al., 2017) conducts a grid-search over the dropout probabilities to choose the optimal one adaptively by treating it as part of the model parameter. Still, this approach requires modifying the training phase. Another similar work used when batch normalization (BN) is adopted as a regularisation method (Teye et al., 2018) shows that training a deep network using batch normalization is equivalent to approximate inference in Bayesian models. An exciting aspect of BN is that the mini-batch statistics used for training each iteration depend on randomly selected batch members. Exploiting this stochasticity, they show that training with BN, like dropout, can be cast as an approximate Bayesian inference.

2.4.2 Dropout as Bayesian Approximation

Dropout training (standard dropout)

Srivastava et al. (2014) proposed dropout as a regularization method to prevent over-fitting. The idea is to drop units from layers to avoid feature co-adaptation (see Section 2.4). For a *fully connected (FC) neural network* the linear operation can be:

$$\begin{aligned} \mathbf{y}^{(l)} &= \mathbf{x}^{(l)}\mathbf{W}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{x}^{(l+1)} &= \rho^{(l)}\big(\mathbf{y}^{(l)}\big) \end{aligned} \tag{2.19}$$

where for each layer l, $\mathbf{x}^{(l)}$ and $\mathbf{y}^{(l)}$ are the input and output of that layer, and $\rho^{(l)}(\cdot)$ is a nonlinear activation function. $\mathbf{W}^{(l)}$ is the weight matrix of l with dimensions $K^{(l)}$ x $K^{(l-1)}$ and $\mathbf{b}^{(l)}$ is the bias vector of dimensions $K^{(l)}$. Using dropout at the l^{th} layer is mathematically equivalent to setting the rows of the weight matrix $\mathbf{W}^{(l)}$ for that layer to zero. The FC layer can, therefore, be represented with dropout:

$$\begin{aligned} \mathbf{z}_{[i]}^{(l)} &\sim \text{Bernoulli}\big(\,\cdot\,|\mathbf{p}_{[i]}^{(l)}\big) \\ \tilde{\mathbf{W}}^{(l)} &= \text{diag}\big(\mathbf{z}^{(l)}\big)\mathbf{W}^{(l)} \\ \mathbf{y}^{(l)} &= \mathbf{x}^{(l)}\tilde{\mathbf{W}}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{x}^{(l+1)} &= \rho^{(l)}\big(\mathbf{y}^{(l)}\big) \end{aligned} \tag{2.20}$$

Here $\mathbf{z}_{[i]}^{(l)}$ are Bernoulli distributed random variables with some probabilities $\mathbf{p}_{[i]}^{(l)}$. The $\mathrm{diag}(\cdot)$ maps vectors to diagonal matrices.

The described dropout operations convert a deterministic neural network with parameters $\mathbf{W}^{(l)}$ into a random Bayesian neural network with random variables $\tilde{\mathbf{W}}^{(l)}$, which equates to a neural network with a statistical model without using the Bayesian approach explicitly.

Dropout as Bayesian approximation

Gal and Ghahramani (2015a) proves the equivalence between dropout training in a neural network and approximate inference in a deep Gaussian process (GP). Differently from a non-probabilistic neural network, a deep GP is a powerful statistical tool that allows modeling distributions over functions. This means that to formulate the neural network layer as a Gaussian process, we would define its covariance function.

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \int p\big(\mathbf{W}^{(l)}\big)\rho^{(l)}\big(\mathbf{x}\mathbf{W}^{(l)} + \mathbf{b}^{(l)}\big)\rho^{(l)}\big(\mathbf{x}'\mathbf{W}^{(l)} + \mathbf{b}^{(l)}\big)d\mathbf{W}^{(l)}, \tag{2.21}$$

with an element-wise nonlinearity $\rho^{(l)}(\cdot)$ and distribution $p(\mathbf{W}^{(l)})$. If we consider now a deep GP with L layers and covariance function $\mathbf{K}(\mathbf{x}, \mathbf{x}')$, it can be approximated by setting a variational distribution over each element of a spectral decomposition of the covariance function of the GP. This spectral decomposition maps each layer of the deep GP to a layer of hidden units in the NN. For an L-layer neural network, this suggests that we can feed the output of one GP to the covariance of the next GP matching a deep GP (Damianou and Lawrence, 2013). Hence, the final predictive distribution can be formulated as

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \mathcal{W})p(\mathcal{W}|\mathbf{X}, \mathbf{Y})d\mathcal{W}, \tag{2.22}$$

where $p(\mathbf{y}|\mathbf{x}, \mathcal{W})$ is the whole Bayesian neural network posterior with random variables $\mathcal{W} = \{p\big(\mathbf{W}^{(l)}\big)\}$. To infer the predictive distribution estimation $p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y})$ we need to calculate $p(\mathbf{y}|\mathbf{x}, \mathcal{W})$ which is intractable. To address this, Gal and Ghahramani (2015a) proposed to use $q\big(\tilde{\mathbf{W}}^{(l)}\big)$, a distribution over the weight matrices as follows:

$$\begin{aligned} \mathbf{z}_{[i]}^{(l)} &\sim \mathrm{Bernoulli}\big(\,\cdot\,|\mathbf{p}_{[i]}^{(l)}\big) \\ q\big(\tilde{\mathbf{W}}^{(l)}\big) &= \mathrm{diag}\big(\mathbf{z}^{(l)}\big)\mathbf{W}^{(l)} \end{aligned} \tag{2.23}$$

The true posterior distribution is, therefore, approximated by the variational distribution $q\big(\tilde{\mathbf{W}}^{(l)}\big)$

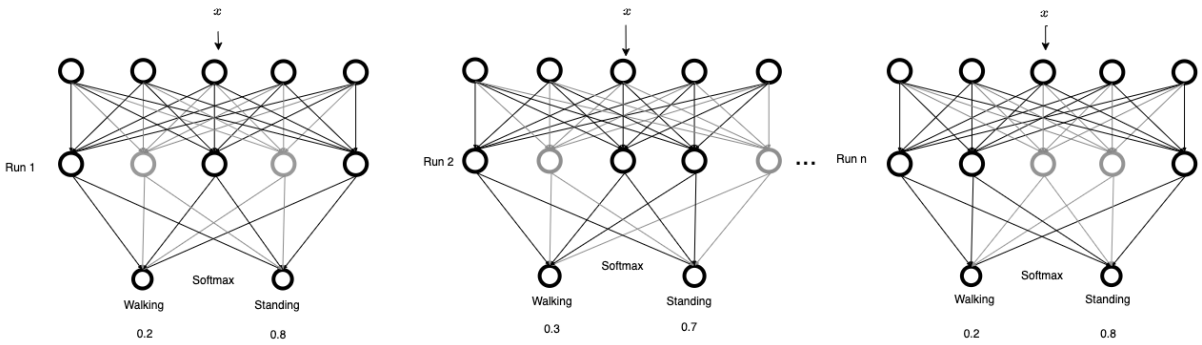


Figure 2.4.1: Monte Carlo dropout (MCDrop): activating dropout during inference creates an implicit ensemble of models. As shown, it requires running the same network with different dropout masks in order to provide uncertainty estimations. Grey nodes and connections denote the dropped ones.

where $\tilde{\mathbf{W}}^{(l)}$ represents the random variables used in dropout operations as described in 3.1. Finally, to approximate the predictive distribution $p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y})$ they perform Monte Carlo (MC) sampling of the random variables $\mathcal{W}$,

$$q(\mathbf{y}|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} q(\mathbf{y}|\mathbf{x}, \mathcal{W}_t), \tag{2.24}$$

where T is the number of MC samples. This method is called Monte Carlo Dropout (MCDrop) and is equivalent to performing T stochastic passes.

In the same way as using dropout on the FC layer, MCDrop can be applied to the individual convolutions in *convolution neural networks* (Gal and Ghahramani, 2015a). The convolution process is an iterative process that takes a sequence of steps in order to compute all elements in the input layer. Similarly to the FC layers, we can sample Bernoulli random variables $\mathbf{z}_{i,j,k}$ and apply them as masks to the weight matrix $\mathbf{W}_i \cdot diag([\mathbf{z}_i, j, k])$ which is equivalent to setting weights to 0 for different elements of the input.

Figure 2.4.1 illustrates how MC dropout is implemented. This technique relies on MC sampling and requires the whole network to run multiple times. Performing the multiple forward passes creates an implicit ensemble of models that differ from one other due to changes in the dropout masks. This implies that different runs will have different nodes and connections and provide the desired stochasticity in the deep learning model. The predictive distribution is, therefore, not produced by explicitly placing distribution over the layers but by running multiple stochastic

forward passes with dropout also activated during inference. These runs are then averaged and give an indication of the predictive uncertainty provided by the variance in the output, distribution variance in the regression context, and softmax variance in the classification.

While MCDrop can be seen as a variational approximation to Bayesian uncertainty from a Gaussian process several works describe it as a poor Bayesian approximation (Osband, 2016; Hron et al., 2017; Folgoc et al., 2021). Osband (2016) counterexamples revealing that even for simple networks you can analytically show that the posterior from MCDrop doesn't concentrate asymptotically. Additionally, using a fixed dropout rate, rather than optimizing this variational parameter can lead to an arbitrarily bad approximation. Hron et al. (2017) criticize the Bayesian interpretation of Variational Gaussian Dropout showing that the chosen improper log-uniform prior generally does not induce a proper posterior, and thus cannot be explained by the standard Bayesian and the related minimum description length arguments. Although, rather than specifically on MCDrop, this work focuses in general on approaches that reinterpret Gaussian multiplicative noise (commonly used as a stochastic regularisation technique in training of deterministic neural networks (Srivastava et al., 2014)) as a specific algorithm for approximate inference in Bayesian neural networks (Kingma et al., 2015; Molchanov et al., 2017; Neklyudov et al., 2017). Recently, Folgoc et al. (2021) question the properties of MCDrop for approximate inference, arguing that since MCDrop changes the Bayesian model, its predictive posterior assigns 0 probability to the true model on closed-form benchmarks; therefore the multimodality of its predictive posterior is not a property of the true predictive posterior. However, MCDrop remains an inexpensive way to quantify uncertainty, which is useful in many applications. Many works based on MCDrop have demonstrated reasonable empirical performance, and for some applications, it may be better to have some form of uncertainty than none at all, such as improving the performance of a reinforcement learning model quickly. Although MCDrop is a step forward towards optimizing and accelerating Bayesian deep learning techniques, it is not enough for running on edge platforms.

In Chapter 3, we use the theoretical foundations presented in this section to build an efficient framework that allows for uncertainty estimation in a single forward pass.

2.5 Edge Computing platforms

In recent years, the landscape of edge platforms has grown exponentially. Such platforms are required to provide the necessary components for sensing, data storage, and processing. Ad-

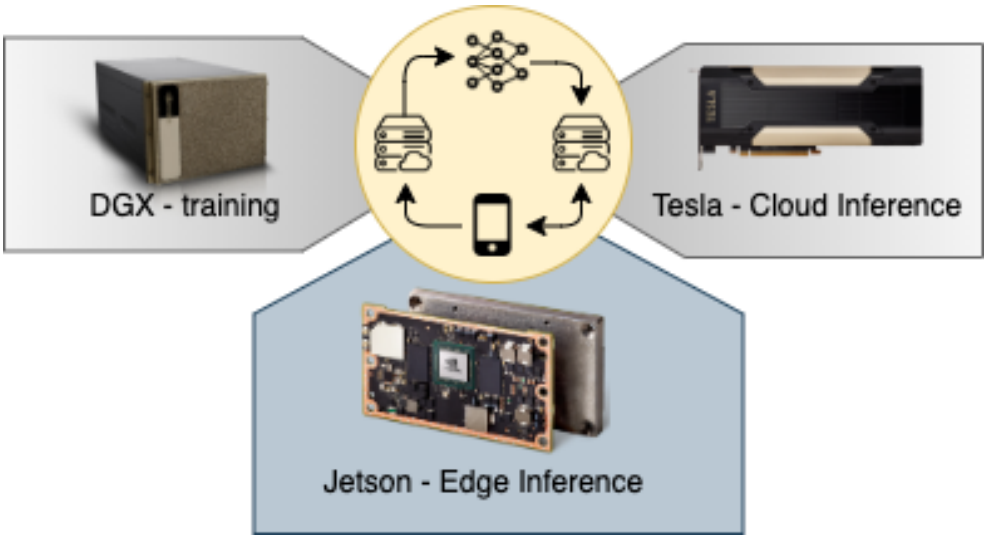


Figure 2.5.1: Deep learning workflow described in different training and inference phases on cloud or edge devices. Depending on the phase, a different capacity device is assigned.

ditionally, given the multiple functionalities these devices need to perform, there is a need for efficient algorithms running on such platforms. For example, Figure 2.5.1 represents a workflow for a deep learning framework where training is performed on very powerful tensor core GPUs such as the Nvidia DTX[1] optimized to accelerate the process of matrix multiplication. Inference on edge devices, such as Nvidia Jetson ones in this example, requires careful consideration in building efficient solutions that can maximize prediction throughput while keeping the resource footprint to a minimum. The range of edge devices is vast, and in this work, we consider two platforms that represent devices with different capacities, NVIDIA Jetson TX2[2] and Nano[3]. The TX2 is an embedded system-on-module, and it is representative of today's embedded platforms with capabilities (256-core Pascal GPU, 2-core Denver + 4-core ARM A57 CPU, 8GB RAM, input $\sim$19V) similar to high-end smartphones such as Samsung Galaxy 20 and OnePlus 7 Pro (Octa-core CPUs, Adreno 640/650 GPU, and 12GB RAM). The Nano, instead, has lower capabilities (128-core Maxwell GPU, 4-core ARM A57 CPU, 4GB RAM, input $\sim$5V) and represents more constrained embedded platforms. Both devices have CPU and GPU support for deep learning libraries (Tensorflow, PyTorch, CUDA, and cuDNN). The platforms have different frequency modes[4]. By choosing a frequency mode, it is possible to select a trade-off between speed and power consumption. A higher frequency results in a shorter computation time but higher power

[1] https://www.nvidia.com/en-us/data-center/dgx-a100/
[2] https://developer.nvidia.com/embedded/jetson-tx2
[3] https://developer.nvidia.com/embedded/jetson-nano-developer-kit
[4] https://jetsonhacks.com/2017/03/25/nvpmodel-nvidia-jetson-tx2-development-kit/

consumption. In this thesis, the frequency mode "Max-N" (2.0 GHz and 1.2 GHz for Arm CPU and GPU, respectively) was chosen, with the highest GPU frequency assigning the needed resources based on the task.

Both devices are equipped with power monitoring chips (Texas Instruments INA226[5]) which measure voltage supply and current draw. INA226 monitors both a shunt voltage drop and bus supply voltage. Programmable calibration value, conversion times, and averaging, combined with an internal multiplier, enable direct readouts (via software) of current in amperes and power in watts. Throughout this thesis, we use the edge devices presented here as hardware platforms for developing the proposed frameworks.

2.6 Summary

This chapter presented the background in deep learning and mobile health which we use as building blocks for this work. Additionally, it has provided a broad overview of the state-of-the-art in uncertainty quantification. Specifically, we outlined the limitations of current approaches highlighting the need for efficient uncertainty quantification for mobile health applications running on resource-scarce devices. This dissertation fills the gaps, stemming from the limitations of the prior art, by first introducing a framework that enables already trained deep learning models to quantify uncertainty during inference (Chapter 3). We further explore early exit neural networks and present a novel interpretation of this paradigm which can be exploited for uncertainty estimation and out-of-distribution detection (Chapter 4). Finally, leveraging on our early exit ensemble framework, we propose a new way to mitigate adversarial attacks (Chapter 5).

[5]https://www.ti.com/product/INA226

Chapter 3

Uncertainty aware mobile sensing for edge computing platforms

3.1 Introduction

In Chapter 1, we discussed the importance of deep learning in mobile and embedded application, including health and well-being monitoring (De Pessemier and Martens, 2018; Ryder et al., 2009; Subasi et al., 2018; Ballinger et al., 2018; Lu et al., 2012; Ronao and Cho, 2016). Additionally, we talked about how traditional deep learning models are scrutinized due to their lack of interpretability. Reliable uncertainty quantification, as an additional insight to point predictions, will improve the ease of understanding in deep learning, resulting in a more informed decision-making process.

Probabilistic approaches exist to provide frameworks for modeling uncertainty toward capturing the erroneous overconfident decisions, and we describe them thoroughly in Chapter 2. However, enabling such approaches on deep neural networks brings significant challenges to embedded devices. There is minimal work on alternatives to Bayesian deep learning or other approaches providing predictive uncertainty efficiently designed for embedded systems (Yao et al., 2018a,b) applicable only to multi-layer perceptrons (MLPs). However, as well known in the applications above, using convolution neural networks (CNNs) leads to more accurate predictions than MLPs (Hammerla et al., 2016; Yao et al., 2017). Indeed, most modern embedded deep learning models do not rely solely on MLPs but are often a combination of different neural layers, CNNs, and MLPs (Yao et al., 2017; Hammerla et al., 2016; Mathur et al., 2019). Consequently, these

approaches, although suitable for embedded devices, are not relevant for the types of deep learning models being deployed in practice. Moreover, they focus mainly on regression tasks, leaving many questions on how they can be used in classification contexts.

In light of the highlighted challenges, in this chapter, we propose a framework that addresses these limitations by enabling *predictive uncertainty* estimations for mobile and embedded applications and evaluating its efficiency on resource-constrained devices. Overall, we make the following contributions:

- We introduce an efficient framework that directly enables already trained deep learning models to generate uncertainty estimates without re-training or fine-tuning. Its core is based on theoretical developments casting dropout training as approximate inference in Bayesian Convolutional Neural Networks (Gal and Ghahramani, 2015a); we consider models already trained with dropout as a regularization method. This assumption is easily satisfiable since many modern deep learning networks use dropout during training (Yao et al., 2017; Hammerla et al., 2016). To achieve our goal of providing the uncertainty estimates, we propose an efficient layerwise distribution approximation, which transforms the single deterministic convolutional layer into a stochastic convolutional layer. Unlike previous methods that generate the prediction distribution via multiple runs (Lakshminarayanan et al., 2017; Gal and Ghahramani, 2015a, 2016b), our layerwise distribution is propagated through the network in a cascaded manner, massively reducing the computational complexity by allowing the model to produce uncertainty estimations in one single run. Therefore, this approach makes it possible to enable predictive uncertainty on a much more comprehensive range of small devices where running uncertainty aware deep learning models would be impossible with traditional techniques.

- Our approach focuses on classification tasks which make obtaining uncertainty estimates more challenging. Unlike regression, we cannot interpret the output distribution as the model prediction output in a classification scenario. To solve this problem, we introduce an efficient way to sample over the final distribution to capture the predictive uncertainty and present the class accuracy. Moreover, our approach can offer the desired flexibility by enabling predictive uncertainty in CNNs, which have better predictive power than MLPs. Combining CNNs with layerwise distribution approximations becomes a powerful tool to estimate uncertainty while offering higher accuracy compared to the existing works which utilize MLP-based models (Yao et al., 2018a,b).

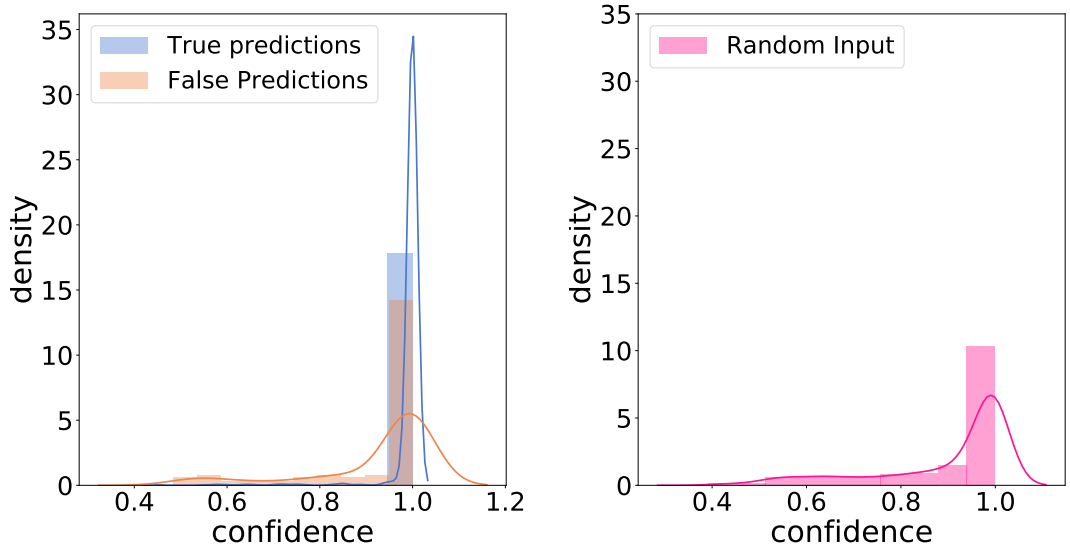
- We evaluate our framework on the Nvidia Jetson TX2 and Nano embedded platforms on

human activity recognition (HAR) and audio sensing applications. We compare our approach with the state-of-the-art Monte Carlo dropout (Gal and Ghahramani, 2016b), a fully connected network based approach (Yao et al., 2018a) as well as deep ensembles technique (Lakshminarayanan et al., 2017). For all approaches, we measure the resource consumption (latency and energy) and model performance, such as the accuracy and the quality of uncertainty estimations. Our approach can reduce inference and energy consumption by 8–fold to 28–fold while obtaining robust and accurate uncertainty estimation. We also significantly improve the accuracy of the deep learning models compared to previous work based on fully connected layered MLP models (Yao et al., 2018b) by a margin of at least 6% to 16% while being more cost-effective computationally. We make sure not to heavily contribute to the memory footprint by adding only a negligible runtime memory overhead (max 5%) compared to the vanilla deep learning model and improving (by 30%) on the MLP baseline. We show that our technique can run smoothly on CPU only, allowing devices without GPU to have fast and robust uncertainty aware predictions still.

The rest of the chapter is organized as follows: In Section 3.2, we use a case study to understand the motivation behind this work better. The review of the related works on uncertainty estimations and deep neural networks in mobile sensing and other applications is presented in Section 3.3. We introduce the technical details of the proposed framework in Section 3.4. The evaluation of our framework is presented in Section 3.5 and Section 4.5. After discussing some lessons learned and future work, we conclude the contributions of this chapter in Section 3.7.

3.2 Motivation

Limited previous work in mobile and embedded systems (Yao et al., 2018b,a) has empirically studied ways to provide uncertainty estimations in deep learning models. These techniques mainly focus on regression, leaving the classification scenario relatively unexplored. Classification tasks make the highest percentage of mobile sensing applications (Lane et al., 2015; Yao et al., 2017; Hammerla et al., 2016; Mathur et al., 2019; De Pessemier and Martens, 2018) but providing uncertainty estimations in the context of mobile sensor data and resource-constrained devices is still an open research area. Deterministic DNNs are trained to obtain maximum likelihood estimates and therefore do not consider the model parameters' uncertainty leading to predictive uncertainty. As a result, they provide overconfident decisions as the softmax probability only captures the relative probability that the input is from a particular class compared to the other classes but not the overall model confidence.



(a) True (correct) and false (incorrect) predictions from the test set.

(b) Random data samples.

Figure 3.2.1: Density histogram of vanilla DNN confidence measures. [The density histogram is a histogram with an area normalized to one. Plots are overlaid with kernel density curves for better readability.]

To visualize the issue mentioned above, we analyze experiments performed on the *Heterogeneous human activity recognition dataset* (Stisen et al., 2015) which consists of readings from two motions sensors (accelerometer and gyroscope) on nine users performing six activities (biking, sitting, standing, walking, climb stairs-up, and stairs-down). We chose a 5-layer network (four convolutional layers and one fully connected) and evaluated the behavior of softmax on the test set as well as on random input data [1]. In Figure 3.2.1a, we can see the confidence measures for true (correct) and false (incorrect) predictions. A distribution skewed towards the right (near 1.0 on the x-axis) shows that the model has higher confidence in predictions than the distributions skewed towards the left. This model gives a high probability for correct and incorrect predictions, while we would like to see low confidence for false predictions.

In Figure 3.2.1b, we plot the confidence given by the same model on entirely random data, which shows that vanilla DNNs are overconfident even when presented with random data the model has not seen during training. This limitation of conventional deep learning approaches motivates our work. We aim to provide accurate predictions while understanding if the model is

[1] More details on these experiments in Section 3.5.

guessing at random or if it is certain about the prediction. In addition, we want the deep learning models to run on resource-limited devices, therefore, to be latency and energy-savvy. We aim to overcome the computation overhead of sampling-based Bayesian techniques and other non-Bayesian approaches like deep ensembles. Running a single stochastic NN multiple times for each prediction, or needing to retrain or fine-tune existing model(s), is not feasible for many edge platforms. To close this gap, we build a new framework that can enable uncertainty estimates for currently deployed models under the constraints that it must require only one forward pass for each prediction, no retraining or fine-tuning, and incur only a residual increase in latency and memory. This is possible due to an approximation of the internal statistics of neural networks that allows an approximate propagation of the signal and confidence through the network layers. Its core is a layerwise distribution approximation that creates a stochastic convolution layer that enables uncertainty estimates to be approximately propagated from inputs to outputs in a single run, which incurs only negligible increased linear algebra cost.

3.3 Related Work

As previously highlighted in Chapter 2, the benefits that may result from providing uncertainty estimates for predictive models have long been recognized. Naturally, the vast majority of this rich literature (see Section 2.4) aims at small-scale problems and precludes deployability considerations. More recently, there has been a significant scale-up of the problems to which such techniques can be applied. Still, the focus is on investigating more accurate uncertainty estimates without considering the system implications of mobile and embedded computing. As a consequence, the proposed methods often *i*) require training new models from scratch or retraining/fine-tuning existing models with a development cost that might inhibit their use and/or *ii*) are computationally prohibitive, i.e., require a linear increase in latency or memory due to multiple forward passes through a single model or one forward pass through several models. We, on the other hand, take a different approach and focus primarily on providing a simple and effective solution that enriches existing deep learning models with predictive uncertainty estimations in a manner that does not require retraining/fine-tuning and ensures latency, memory, and energy consumption in the same ballpark as the original model.

3.3.1 Mobile Applications, Resource Constraints, and Uncertainty

Numerous works have investigated the use of deep neural networks for human activity recognition (Hammerla et al., 2016; Radu et al., 2018; Ravi et al., 2016; Khan et al., 2018; Peng et al.,

2018) and audio sensing (Lane et al., 2015; Georgiev et al., 2017b,a; Bhattacharya and Lane, 2016; Chauhan et al., 2018). These applications need intelligence at the edge and, therefore, deal with constrained resources. Recently, traditional DNNs have been modified to fit in memory, increase execution speed, and decrease energy demand to inference models on the edge (Yang et al., 2018; Cai et al., 2019; Lane et al., 2016). However, there is limited previous work that aims to enable uncertainty estimations on these models. ApDeepSense (Yao et al., 2018b) proposes a seminal approach to approximating the output distribution using standard dropout (Srivastava et al., 2014), which aims to reduce computation time and energy consumption. However, this work only applies to fully-connected NNs, leaving the challenges for more complex models, like Convolutional Neural Networks (CNNs), to be addressed. Modern architectures are very rarely solely MLP-based (Hammerla et al., 2016; Yao et al., 2017). This suggests that extending CNNs to obtain uncertainty estimations makes it possible to obtain not only higher accuracy but also more robust models in their predictive abilities. Moreover, being mainly engineered for regression, it does not use the distribution variance for the uncertainty in the classification task. Using the variance is fundamental to properly capture the uncertainty; without it, the variability in the prediction is not taken into consideration. Classification scenarios make up the majority of mobile sensing applications; therefore, we intentionally focus on these tasks in our work. We use the variance by adding sampling to the logit distribution to simulate the sampling in MCDrop but at a much lower computational cost.

Also, unlike previous works providing uncertainty estimation only in the last layer or after several runs, our approach provides flexibility through its layerwise approximation, which captures the uncertain unobserved quantities at the layer level. This is useful in scenarios where models cannot execute all layers, possibly due to energy and latency constraints (Lee and Nirjon, 2019; Montanari et al., 2020) or would like to exit early in a densely layered model to save energy while providing robust uncertainty estimates.

3.4 Methods

We present a framework that enables pre-trained deterministic deep learning models to generate uncertainty estimates on resource-limited devices *i*) without increasing their latency, memory, and energy profiles except for a minimal amount due to additional linear algebra and *ii*) without any retraining/fine-tuning which could prevent deployability on the same devices. This is in stark contrast to existing techniques for providing uncertainty estimates which have a linear (often unaffordable) increase in operational costs due essentially to their reliance on multiple forward

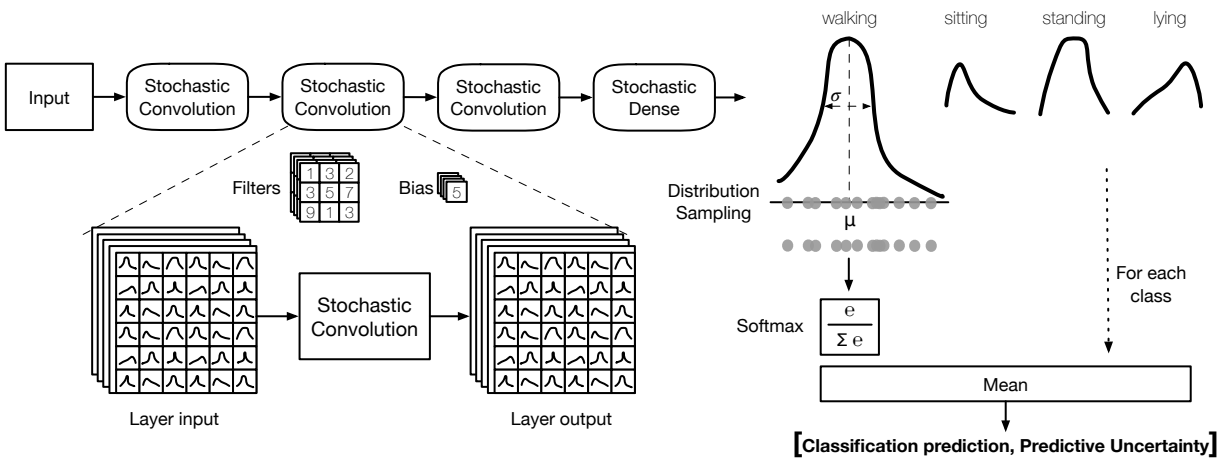


Figure 3.4.1: Overview of the framework: describing the composition of the stochastic convolution, its propagation throughout the network, and finally, the distribution sampling at the softmax layer.

passes, and which may require training new models.

3.4.1 High-level representation of our framework

Our framework is grounded on the fact that when using a neural network trained with dropout, the network does not have a deterministic structure anymore since it is described in part by random variables. Therefore, we enable the convolution operations in the stochastic network generated by dropout to apply to inputs described by probabilistic distributions and generate distributions as outputs (Figure 3.4.1). This means that for every layer, the network is described by inputs (and consequently outputs) which are Gaussian distributions with a mean and variance. The layerwise distribution is going to capture the predictive uncertainty since we are not considering point estimates anymore.

As described in the visual overview in Figure 3.4.1, these distributions are then propagated through the whole network carrying and transmitting the information on the prediction and the uncertainty of these predictions. Finally, we integrate over the culminating Gaussian distributions to ultimately sample point values (unaries) and pass them through the softmax function. At this point, we only sample from the final distribution, which is a tiny fraction of the network's compute (see Section 3.6.2), and therefore, it does not significantly increase the model's inference time. At the end of this process, the classification model produces the class prediction and

its predictive uncertainty based on the samples. This operation is extremely fast as we only run the model once, passing inputs to the models to get the output distribution.

While our approach uses the theoretical foundations of MCDrop (Gal and Ghahramani, 2015a) casting dropout training as an implicit Bayesian approximation, it radically distinguishes from it as in our technique, the distributions are embedded in the network and do not rely on multiple runs to produce them. In addition, we offer a novel mode to adapt the distribution output to predict the outcome in classification tasks while providing the desired predictive uncertainty estimations.

3.4.2 Our Approach to Efficient Uncertainty Estimation

We have developed an alternative approach that eliminates the need for a time-consuming and computationally intensive sampling process. Our method is designed to be resource-friendly by enabling convolution neural networks to generate both predictive uncertainty and predictions. To achieve this, we introduce a layerwise distribution approximation that allows us to incorporate the distribution at the layer level and propagate it throughout the network. We utilize a multivariate Gaussian distribution as our chosen distribution due to the equivalence of the layerwise approximations of deep Gaussian processes and neural networks trained with dropout.

A layer l of a neural network trained with dropout can be represented as:

$$\begin{aligned} \mathbf{z}_{[i]}^{(l)} &\sim \text{Bernoulli}\big(\,\cdot\,|\mathbf{p}_{[i]}^{(l)}\big) \\ \tilde{\mathbf{W}}^{(l)} &= \text{diag}\big(\mathbf{z}^{(l)}\big)\mathbf{W}^{(l)} \\ \mathbf{y}^{(l)} &= \mathbf{x}^{(l)}\tilde{\mathbf{W}}^{(l)} + \mathbf{b}^{(l)} \\ \mathbf{x}^{(l+1)} &= \rho^{(l)}\big(\mathbf{y}^{(l)}\big) \end{aligned} \tag{3.1}$$

where $\mathbf{x}^{(l)}$ and $\mathbf{y}^{(l)}$ are the input and output of the layer l, $\rho^{(l)}(\cdot)$ is a nonlinear activation function, while $\mathbf{W}^{(l)}$ is the weight matrix with dimensions $K^{(l)}$ x $K^{(l-1)}$ and $\mathbf{b}^{(l)}$ is the bias vector of dimensions $K^{(l)}$. Here $\mathbf{z}_{[i]}^{(l)}$ are Bernoulli distributed random variables with probabilities $\mathbf{p}_{[i]}^{(l)}$. The outlined dropout operations transform a neural network with deterministic parameters $\mathbf{W}^{(l)}$ into a random Bayesian neural network with random variables $\tilde{\mathbf{W}}^{(l)}$. This results in a statistical model for the neural network without the need to explicitly utilize Bayesian techniques.

In a Bayesian framework, we are interested in learning the posterior distribution which can be

applied to calculate the final predictive distribution given the input $\mathbf{x}$:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}|\mathbf{x}, \mathcal{W}) p(\mathcal{W}|\mathbf{X}, \mathbf{Y}) d\mathcal{W}, \tag{3.2}$$

where $p(\mathbf{y}|\mathbf{x}, \mathcal{W})$ is the whole Bayesian neural network posterior with random variables $\mathcal{W} = \{p\big(\mathbf{W}^{(l)}\big)\}$. However, computing the exact posterior distribution $p(\mathcal{W}|\mathbf{X}, \mathbf{Y})$ is not tractable. To address this, we can use variational inference as an approximate to the posterior distribution (Wainwright et al., 2008). As such, inspired by Gal and Ghahramani (2015a) we employ the variational distribution $q\big(\tilde{\mathbf{W}}^{(l)}\big)$, a distribution over the weight matrices as follows:

$$\begin{aligned} \mathbf{z}_{[i]}^{(l)} &\sim \text{Bernoulli}\big(\,\cdot\,|\mathbf{p}_{[i]}^{(l)}\big) \\ q\big(\tilde{\mathbf{W}}^{(l)}\big) &= \text{diag}\big(\mathbf{z}^{(l)}\big)\mathbf{W}^{(l)} \end{aligned} \tag{3.3}$$

where $\tilde{\mathbf{W}}^{(l)}$ represents the random variables used in dropout operations. Choosing the approximate posterior distribution as specified in Equation 3.3 establishes an equivalence between the layerwise approximation of a deep Gaussian process and the neural network, which is trained using either the mean square error or a cross-entropy loss function while incorporating dropout operations.

A Gaussian process is a stochastic process such that every finite collection of those random variables has a multivariate Gaussian distribution. Therefore, we initially considered our layerwise distribution approximation represented by multivariate normal distributions. We found, however, that this approximation was not enough for avoiding multiple forward passes and therefore explored an additional approximation of Gaussian distributions with diagonal covariance matrices. Indeed, by modeling inputs and outputs of each layer as approximately following normal distributions with diagonal matrices, it is possible to compute the output's mean and standard deviation in closed-form from those of the inputs as well as the layer operations for deterministic models trained with dropout, without requiring any retraining. Our approach is based on an approximation to the internal statistics of neural networks that permits an approximate propagation of the signal and confidence through the network layers in a manner that is applicable to convolution and dense layers.

To start, we enable the basic operations in convolution neural networks to output the expected value and a probability distribution of the output random variable. Computing the exact output distribution is intractable; consequently, we approximate it with the multivariate Gaussian distribution. This approximation is based on minimizing the Kullback-Leibler (KL) divergence

between the exact and approximate distributions as follows:

$$\begin{aligned}\min_q \mathrm{KL}(p(x)\|q(x)) &= \min_q \int p(x)\log\left(\frac{p(x)}{q(x)}\right)dx \\ &= \min_{\mu,\sigma^2} -\int p(x)\log\mathcal{N}\left(x|\mu,\sigma^2\right)dx \\ &= \min_{\mu,\sigma^2} \frac{\log(\sigma^2)}{2} + \frac{\int_x p(x)(x-\mu)^2dx}{2\sigma^2},\end{aligned} \tag{3.4}$$

where $p(x)$ is the exact output distribution and $q(s) \sim \mathcal{N}\left(x|\mu,\sigma^2\right)$ is the approximate output distribution. To obtain the optimal approximate output distribution, we take the derivative over the μ and σ^2; therefore, the approximation can be represented as matching the mean and variance between the two distributions:

$$\begin{aligned}\mu &= \int p(x)xdx \\ \sigma^2 &= \int p(x)(x-\mu)^2dx.\end{aligned} \tag{3.5}$$

We reformulate the convolution operation with dropout as follows. The input to the layer is represented by $\mathbf{x} \in \mathbb{R}^{(H,W,C)}$ with height H, width W and C channels. Let $\mathbf{w} \in \mathbb{R}^{(h,w,c,f)}$ be the weight matrix with height h, width w, c channels and f filters, and $\mathbf{b} \in \mathbb{R}^{(f)}$ the bias vector. Consequently, the output will be represented as $\mathbf{y}_{[:,:,k]} := \mathbf{x} * \mathbf{w}_{[:,:,:,k]} + \mathbf{b}_{[k]}$ for $\mathbf{y} \in \mathbb{R}^{(H-h+1,W-w+1,f)}$ and for $k = 1,\ldots,f$ where $\odot$ represents the element-wise operation and $*$ the convolution operation. Formally,

$$\mathbf{y}_{[i,j,k]} := \mathrm{vec}\big(\mathbf{x}_{[i+[0:h],j+[0:w],:]} \odot \mathbf{z}_{[i+[0:h],j+[0:w],:]}\big)^\top \mathrm{vec}\big(\mathbf{w}_{[:,:,:,k]}\big) + \mathbf{b}_{[k]}, \tag{3.6}$$

where

$$\begin{aligned}\mathbf{z}_{[i+[0:h],j+[0:w],l]} &\sim \mathrm{Bernoulli}\big(\,\cdot\,|\mathbf{p}_{[i+[0:h],j+[0:w],l]}\big) \\ \mathbf{x}_{[i+[0:h],j+[0:w],l]} &\sim \mathcal{N}\big(\,\cdot\,|\mu_{[i+[0:h],j+[0:w],:]}, \sigma^2_{[i+[0:h],j+[0:w],l]}\big).\end{aligned} \tag{3.7}$$

According to Equation 3.4, we need to calculate the mean and the variance of the output distribution $p(\mathbf{y}_{[i,j,k]})$. Given that the Bernoulli variables $\mathbf{z}$ and the input Gaussian variables $\mathbf{x}$, as shown in Equation 3.7, are independent random variables, we can have the mean of the output

as follows:

$$\begin{aligned}\mathbb{E}\big[\mathbf{y}_{[i,j,k]}\big] &= \mathbb{E}\left[\sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]}\right] + \mathbf{b}_{[k]} \\ &= \sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbb{E}\big[\mathbf{x}_{[u,v,a]}\big]\mathbb{E}\big[\mathbf{z}_{[u,v,a]}\big]\mathbb{E}\big[\mathbf{w}_{[u-i,v-j,a]}\big] + \mathbf{b}_{[k]} \\ &= \sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mu_{[u,v,a]}\mathbf{p}_{[u,v,a]}\mathbf{w}_{[u-i,w-j,a]} + \mathbf{b}_{[k]}.\end{aligned} \tag{3.8}$$

Since $\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]}$ are independent variables, we can measure the variance as follows:

$$\begin{aligned}\mathrm{Var}\big[\mathbf{y}_{[i,j,k]}\big] =& \mathrm{Var}\left[\sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]} + \mathbf{b}_{[k]}\right] \\ =& \sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathrm{Var}\big[\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]} + \mathbf{b}_{[k]}\big] \\ =& \sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbb{E}\big[\big(\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]} + \mathbf{b}_{[k]}\big)^2\big] \\ & - \big(\mathbb{E}\big[\mathbf{x}_{[u,v,a]}\mathbf{z}_{[u,v,a]}\mathbf{w}_{[u-i,v-j,a]} + \mathbf{b}_{[k]}\big]\big)^2 \\ =& \sum_{u=1}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbb{E}[\mathbf{x}_{u,v,a}^2]\mathbb{E}[\mathbf{z}_{u,v,a}^2]\mathbb{E}[\mathbf{w}_{u-i,v-j,a}^2] - \mathbb{E}^2[\mathbf{x}_{u,v,a}]\mathbb{E}^2[\mathbf{z}_{u,v,a}]\mathbb{E}^2[\mathbf{w}_{u-i,v-j,a}] \\ =& \sum_{u=1}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}(\mu_{u,v,a}^2 + \sigma_{u,v,a}^2)\mathbf{p}_{u,v,a}\mathbf{w}_{u-i,w-j,a}^2 - \mu_{u,v,a}^2\mathbf{p}_{u,v,a}^2\mathbf{w}_{u-i,w-j,a}^2 \\ =& \sum_{u=1}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\mathbb{E}[\mathbf{x}_{u,v,a}^2]\mathbb{E}[\mathbf{z}_{u,v,a}^2]\mathbb{E}[\mathbf{w}_{u-i,v-j,a}^2] - \mathbb{E}^2[\mathbf{x}_{u,v,a}]\mathbb{E}^2[\mathbf{z}_{u,v,a}]\mathbb{E}^2[\mathbf{w}_{u-i,v-j,a}] \\ =& \sum_{u=i}^{i+h-1}\sum_{v=j}^{j+w-1}\sum_{a=1}^{c}\big(\big(\mu_{[u,v,a]}^2 + \sigma_{[u,v,a]}^2\big)\mathbf{p}_{[u,v,a]} - \mu_{[u,v,a]}^2\mathbf{p}_{[u,v,a]}^2\big)\mathbf{w}_{[u-i,w-j,a]}^2.\end{aligned} \tag{3.9}$$

We can further represent the operations and efficiently compute the output distribution, namely

$$
\begin{aligned}
\mathbb{E}[\mathbf{y}] &= (\mu \odot \mathbf{p}) * \mathbf{w} + \mathbf{b} \\
\mathrm{Var}[\mathbf{y}] &= (((\mu \odot \mu) + (\sigma \odot \sigma)) \odot \mathbf{p} - (\mu \odot \mu) \odot (\mathbf{p} \odot \mathbf{p})) * (\mathbf{w} \odot \mathbf{w}) \\
\mathbf{y}_{[i,j,k]} &\sim \mathcal{N}\big(\cdot \,|\mathbb{E}\big[\mathbf{y}_{[i,j,k]}\big], \mathrm{Var}\big[\mathbf{y}_{[i,j,k]}\big]\big).
\end{aligned}
\tag{3.10}
$$

We have provided a mathematically grounded proof of calculating the mean and the variance of the output at each convolution layer on networks trained with dropout. This means we can have CNNs taking probabilistic distributions as inputs and generating distributions as outputs. Hence, avoiding the need for computationally costly sampling from these distributions.

Given our approximation of modeling the inputs and the outputs as Gaussian distributions with a diagonal covariance matrix, what is required now is to compute the mean and variance of the activation function that follows the linear mapping, which can then be plugged into the Gaussian model. Toward this end, the mean and variance

$$
\begin{aligned}
\mu_{\mathbf{y}} &= \mathbb{E}[\mathbf{y}] \\
\sigma^2_{\mathbf{y}} &= \mathbb{E}[\mathbf{y} - \mu_{\mathbf{y}}]^2,
\end{aligned}
\tag{3.11}
$$

can be represented as a sum of expectations of the output activations with respect to the Gaussian input distribution over the compact intervals where the activation function is linear. This computation can be done in closed-form via $\mathrm{erf}(\cdot)$ for any piece-wise linear activation as demonstrated in Yao et al. (2018b), and particularly for the ReLU activation used in our work.

Secondly, we propose an efficient way to exploit the output distribution of our stochastic neural network to provide a classification prediction and a predictive uncertainty measure. To this aim, we sample these Gaussian distributions in the logit space. We sample unaries (single elements) from the output distribution and then pass the point values from this distribution to the softmax

function, such as

$$\begin{aligned} \mathbf{y} &\sim \mathcal{N}\big(\cdot\,|\mathbb{E}[\mathbf{y}], \mathrm{Var}[\mathbf{y}]\big) \\ p(y=c|\mathbf{x},\mathbf{X},\mathbf{Y}) &\approx \frac{1}{S}\sum_{s=1}^{S}\xi(\mathbf{y}_s) \\ H(\mathbf{y}|\mathbf{x},D) &= -\sum_{i=1}^{S} p_{c\mu} * log p_{c\mu}, \end{aligned} \tag{3.12}$$

where $\mathbf{y}$ is the output distribution of the model; therefore, the prediction can be considered as the mean of the categorical distribution obtained from sampling S single values from the Gaussian output distribution and then squashed with the softmax function $\xi(\cdot)$ to get the probability vector and the predictive entropy. The sampling operation from the output distribution is extremely fast as we only run the model once, passing inputs to the models to get the output logits. At this point, we only sample from the logits, which is a tiny fraction of the network's compute, and as we can see in Section 3.6.2, it does not significantly increase the model's inference time.

In conclusion, we add a layerwise approximation to the convolutional layers, propagated throughout the network to produce a probability distribution in output. With the approximation presented in Equation 3.10 and the output distribution sampling in Equation 3.12, we can now enable classification models to output predictive uncertainties alongside the class inference in one single run.

3.5 Experiments

We design experiments to verify the improvement of our framework in providing well- calibrated uncertainty quantification compared to state-of-the-art uncertainty aware baselines.

3.5.1 Datasets

Heterogeneous human activity recognition dataset (HHAR). (Stisen et al., 2015) HHAR contains readings from two motion sensors (accelerometer and gyroscope). The data is gathered from nine users performing six activities (biking, sitting, standing, walking, climbing stairs up, and climbing stairs down) with six types of mobile devices. Conforming to the description on Yao et al. (2018b), we segment raw measurements into five seconds samples and take a Fourier transform on these samples as the input data. Each sample is further divided into time intervals

Dataset	#Training	#Testing	Output	One-hot Encoding
HHAR	28,314	1,686	Sit, Stand, Walk, Bike, StairUp, StairDown	✓
Opportunity	26,908	6,287	Stand, Walk, Sit, Lie	✓
Speech Commands	1,668	562	Yes, No, Up, Down, Left, Right, On, Off, Stop, Go	✓

Table 3.5.1: Statistical information of the publicly available datasets used for evaluation. *Heterogeneous* human activity recognition (HHAR) (Stisen et al., 2015), *Opportunity* (Chavarriaga et al., 2013) and *Speech Commands* (Warden, 2017) dataset.

of the length of 0.25s.

Opportunity. (Chavarriaga et al., 2013) This dataset consists of data from multiple accelerometers and gyroscopes placed on participants' bodies at different locations such as arms, back, and feet. They used three devices on a hip, a left lower arm, and a right shoe by default and target to detect the mode of locomotion: stand, walk, sit and lie. In total, for all users and all recordings, the dataset consists of 3,653 modes of locomotion instances of variable duration (between 0.2 and 280 seconds). Following the preprocessing proposed by Hammerla et al. (2016), we use the run 2 from user 1, and runs 4 and 5 from users 2 and 3 in our test set. The remaining data is used for training. For frame-by-frame analysis, we created sliding windows of duration 1 second with 50% overlap.

Speech Commands. (Warden, 2017) For audio sensing, we use the *Speech Commands* dataset, and the suggested preprocessing by Mathur et al. (2019). We train our network with the portion of the dataset that consists of 2,250 one-second-long speech files belonging to 10 keyword classes (yes, no, up, down, left, right, on, off, stop, and go). In this task, the goal is to identify the presence of a certain keyword class in a given speech segment. This 10-class dataset was then randomly split into training (75%) and test (25%) class-balanced subsets to make sure we get the same amount of data for each class. The input to the model is a two-dimensional tensor extracted from the keyword recording, consisting of time frames and 24 MFCC features.

Augmentations. All three of our datasets were collected in a controlled environment; therefore, we augmented real-life noise to the datasets in a principled manner to include real-world variability. Data augmentation can encode prior knowledge of the data, resulting in more robust

models and providing more resources to the deep learning platform. For both HAR datasets, we used the data augmentation techniques proposed in Um et al. (2017) which consist of seven variations for IMU data on wearables: rotation, permutation, time-warping, scaling, magnitude-warping, jittering, and cropping. For example, permutation consists of randomly perturbing the temporal location of events within the same time window. Rotation, instead, enriches the data to match different sensor placements like an upside-down position. These alterations allow for real-life noise considerations which cannot be witnessed in the data collected in a lab-controlled setting, e.g., rotating a smartwatch. We randomly chose a period between 20 seconds and two minutes and applied a randomly selected augmentation method; the intervals between noise periods were randomly selected between two and five minutes. For audio, we sampled examples of ambient noise from a publicly available environment sound dataset (Piczak, 2015) and added them to the audio dataset. We assume that only one type of noise is present at a given time, and each noise lasts between 15 and 60 seconds.

3.5.2 Baseline algorithms

We tested our proposed method on the three datasets described above and compared the outcome with other four state-of-the-art approaches:

Backbone. This is a conventional deep neural network. To show the benefits of a stochastic approach, we need to compare it to the traditional deep learning network. This network is used as the non-Bayesian baseline. As mentioned before, our technique and MCDrop rely on an already trained network; therefore, this network is the one we consider the already trained network we refer to.

MCDrop. Monte Carlo dropout (Gal and Ghahramani, 2016a) is based on Monte Carlo sampling and runs the neural network multiple times to generate the uncertainty estimation. Hence, we use MCDrop-k to represent MCDrop with k (3, 5, 10, and 30) runs (forward passes). This approach, like ours, assumes that the model has already been trained with dropout. For this baseline, we keep dropout activated during inference too.

ApDeepSense. ApDeepSense (Yao et al., 2018b) is an algorithm that enables fully-connected NNs to provide uncertainty estimations during inference. This technique uses dropout to perform the basic operations in the FC layers, too. Compared to our method, ApDeepSense works only with MLPs and considers mainly regression tasks where the variance of the distribution

represents the uncertainty. Therefore, it does not translate very well to classification tasks. For this baseline, as suggested in the original proposal Yao et al. (2018b), we use a 5-layer neural network composed of fully connected layers with 512 hidden dimensions and ReLU activation function.

Deep. Deep ensembles (Lakshminarayanan et al., 2017) rely on providing uncertainty estimations by training and running an ensemble of models (multiple networks). Although this baseline requires retraining, we include it to illustrate the upper bound of the uncertainty estimation quality that can be accomplished with retraining. Ensembles are created by training the models with different random weight initialization. To achieve this, we use the Backbone architecture (not its trained model) with random initialization for each model. We use Ensemble-*k* to represent an ensemble of *k* (3, 5, and 10) individual NNs.

3.5.3 Implementation details

To evaluate the performance of our approach, we build a five-layer deep neural network composed of four 2D convolutional layers with a ReLU activation function and one fully-connected output layer. The choice of the architecture was made to make a fair comparison with the other baselines, which mainly relied on five-layer deep networks (Yao et al., 2018b,a; Lakshminarayanan et al., 2017).

During training, the model is optimized by ADAM (Kingma and Ba, 2014) with a learning rate of $1e-4$. We add dropout of $p_{drop} = 0.5$ (default) at each internal layer to stabilize training, avoid overfitting and fulfill our requirement of having a model trained with dropout regularization. We use cross entropy as the loss function and a batch size of 64. For all the datasets, we isolate 5% of the training set for validation and hyperparameter tuning. We employ the described architecture for all the datasets. The models were implemented using Tensorflow (Abadi et al., 2015) (v.1.15.2) and trained on a 4-GPU Linux server with 64GB memory.

During inference, we enable our layerwise approximation to all layers and propagate it all the way to the output layer (Figure 3.4.1). This architecture allows having the output of the model represented as the mean and variance of the output distribution. However, since we are dealing with classification tasks, we sample unaries from the output distribution and pass them through the softmax function, as explained in Section 3.4.2. We have two inputs to the first layer of the neural network: the sample we need to do inference on and the standard deviation of the data calculated on the training set. We use this as a prior to feed to the stochastic network.

3.5.4 Evaluation metrics

In Section 4.5, we present the results in terms of accuracy, F1 score, and negative log-likelihood (NLL). The prediction accuracy expresses the correlation between the prediction of the deep neural network and the actual value; instead, the F1 score is the weighted average of precision and recall. NLL is a proper scoring rule that captures the correlation between the ground truth label and the model prediction, where lower NLL means they are highly correlated.

In addition to the metrics above, we consider predictive confidence and entropy as measures of uncertainty. The confidence metric gives a better understanding of how the model behaves during inference. Ideally, we want to achieve high confidence for correct predictions and low confidence for incorrect ones. In classification tasks, it is considered as the confidence given from the softmax. In conventional DNN and APDeepsense, it is measured based on only one softmax operation. Instead, in the other baselines, including ours, it is the mean of the categorical predictive distribution. To evaluate the predictive uncertainty, we measure the predictive entropy $H(\mathbf{y}|\mathbf{x}, D)$ which captures the average amount of information contained in the predictive distribution. Formally,

$$H(\mathbf{y}|\mathbf{x}, D) = -\sum_{s=0}^{S-1} p_{c\mu} * logp_{c\mu}, \tag{3.13}$$

where $p_{i\mu}$ is the predictive mean probability of the c^{th} class from S Monte Carlo samples in the case of MCDrop, the S model predictions in the case of Deep, and finally, from the S unaries sampled from the output distribution and passed to the softmax function in our approach. To reliably capture the predictive uncertainty, we aim for a predictive entropy that is low for true and high for false inferences.

3.5.5 Embedded Edge Systems Setup

To evaluate our framework's performance, we run the inference on two edge platforms, NVIDIA Jetson TX2 and Nano (described in Section 2.5), and measure the metrics above while computing the latency and energy consumption per testing sample. The latency indicates the average time it takes for the model to predict the provided sample. For fairness, we measure only the time needed to pass a sample and do not consider the time needed to upload the model. In most of the considered baselines, including ours, the model is uploaded just once and kept in memory. However, for ensembles, this might be different depending on the capacity and scenario; therefore, we decide not to add that time and computation to the results. The energy consumed

is expressed as *power x time*. We measure voltage supply and current draw by accessing the TI INA226 power monitoring chip and computing power and energy from the logs. We perform ten runs (with the whole test set) and average the results. For an evaluation catering to more resource-limited devices, we evaluate our framework on the Jetson Nano with CPU only enabled (with all the datasets). This validates the applicability to these kinds of limited platforms.

3.6 Results

3.6.1 Model Estimation Performance

Table 3.5.2 shows the accuracy, F1 scores, and negative log-likelihood obtained for the three datasets. Negative log-likelihood (NLL) penalizes both over- and under-confident predictions. Overconfident predictions can be infinitely penalized, which discourages probabilities equal to zero or one. The minimum value is zero, which can be achieved if the prediction is correct with 100% confidence. Otherwise, the worse the prediction, the larger the loss. Since NLL reflects both the accuracy and the quality of predictive uncertainty, our approach is the best performing across the three baselines: we can provide accurate predictions with high-quality uncertainty estimates. We achieve higher accuracy compared to the other methods, especially ApDeepSense because we employ CNNs instead of MLPs. MCDrop does achieve similarly accurate uncertainty to our method; however, it takes 30 runs to reach an NLL close to ours, which makes it very power-hungry and not suitable for resource-constrained devices, as highlighted in Section 3.6.2. A similar argument holds for Deep–10. A deep ensemble not only requires running multiple models to get the predictive uncertainty, but it also demands keeping them in memory. In the case of many embedded devices, this would involve some kind of scheduling for memory allocation to perform them all. To apply this technique, it is required to train multiple models; therefore, it cannot be performed on already trained networks.

To have a closer look at what happens when we compare the best-performing approaches at a more fine-grained level, we present the results on confidence and predictive entropy in the following density histograms. We notice how the trust concern in the overconfident predictions of the conventional DNN Backbone is valid for all three datasets (see Figure 3.6.1). Even if the model has high accuracy, it loses its credibility when the confidence is so high for false predictions. We see a peak near higher confidence values for true predictions. However, most importantly, we want lower confidence values for false predictions. Our approach conforms to this as the best-performing approach, especially in the HHAR dataset. ApDeepSense performs

Model	Ours	Backbone	ApDeepSense	MCDrop–3/5/10/30	Deep–3/5/10
HHAR					
Accuracy	**0.95**	0.92	0.79	0.93/ 0.92/ 0.91/ 0.93	0.93/ 0.93/ 0.93
F1 Score	**0.79**	0.67	0.60	0.67/ 0.68/ 0.68/ 0.71	0.70/ 0.70/ 0.71
NLL	**2.56**	42.14	3.75	30.14/ 8.01/ 5.58/ 4.13	23.54/ 7.18/ 4.36
Opportunity					
Accuracy	**0.87**	0.84	0.80	0.83/ 0.83/ 0.84/ 0.85	0.82/ 0.83/ 0.84
F1 Score	**0.81**	0.73	0.70	0.74/ 0.74/ 0.74/ 0.77	0.74/ 0.75/ 0.76
NLL	2.39	12.35	4.00	12.31/ 9.38/ 8.38/ **2.28**	9.21/ 7.98/ 3.60
Speech Commands					
Accuracy	**0.82**	0.78	0.71	078/ 0.80/ 0.81/8 0.81	0.79/ 0.80/ 0.81
F1 Score	**0.73**	0.62	0.59	0.62/ 0.63/ 0.63/ 0.71	0.60/0.60/ 0.68
NLL	**1.01**	2.11	1.97	1.44/ 1.20/ 1.13/ 1.08	1.44/ 1.18/ 1.08

Table 3.5.2: Accuracy, F1 Score, and Negative Log-Likelihood (NLL) for HHAR, Opportunity, and Speech Commands dataset. MCDrop-k represents MCDrop with k (3, 5, 10, and 30) runs (forward passes), instead Deep-k represents an ensemble of k (3, 5, and 10) individual NNs. NLL measures the correspondence between the ground truth values and their predicted distributions. Lower NLL means higher correspondence. Bold entries indicate the best performance.

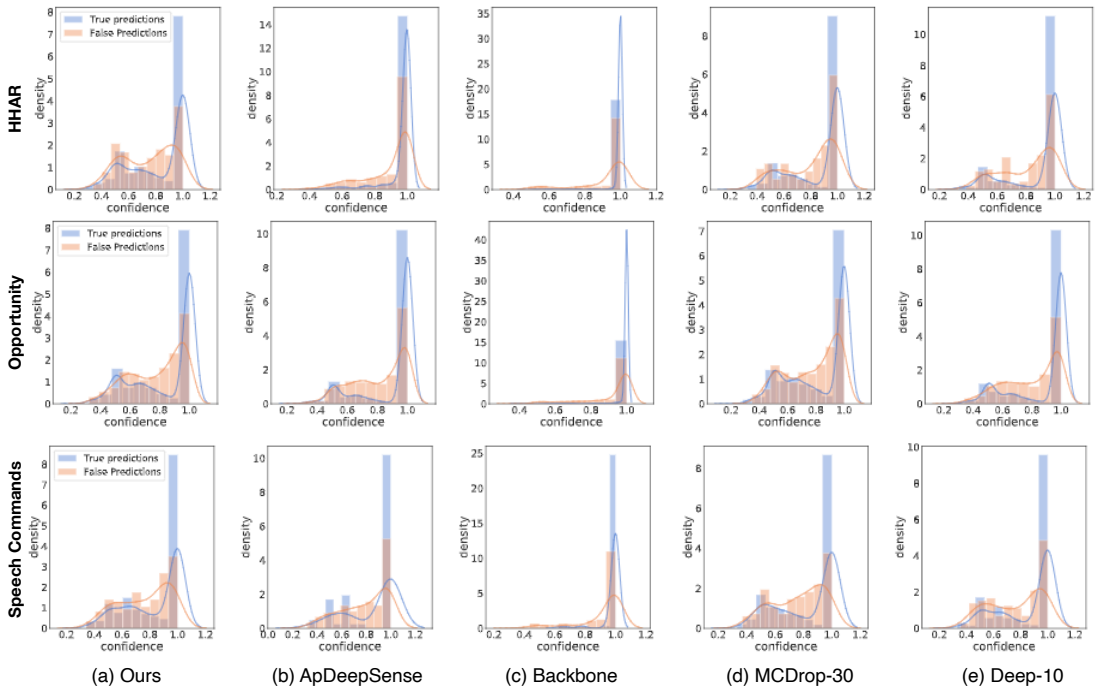
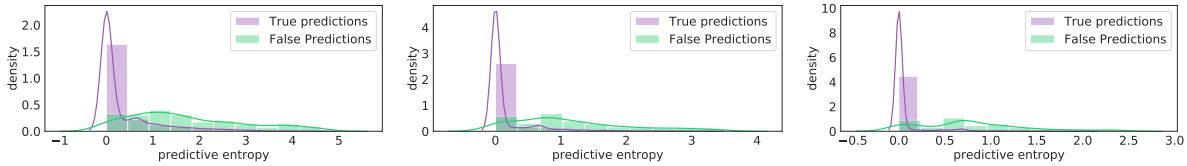


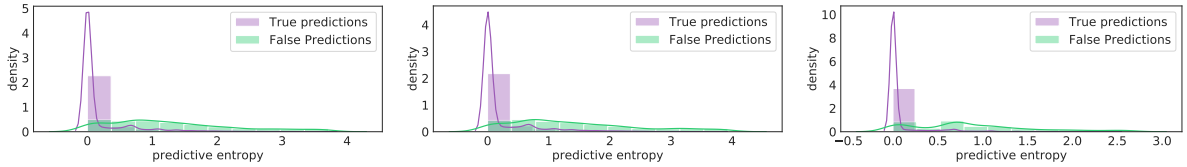
Figure 3.6.1: Density histogram of confidence measures for true (correct) and false (incorrect) predictions. A distribution skewed towards the right (near 1.0 on the x-axis) indicates the model has higher confidence in predictions than the distribution skewed towards the left. [The density histogram is a histogram with an area normalized to one. Plots are overlaid with kernel density curves for better readability.]

worse, as expected, given its limitations in relying only on MLPs, and it is not enabled to work well for classification.

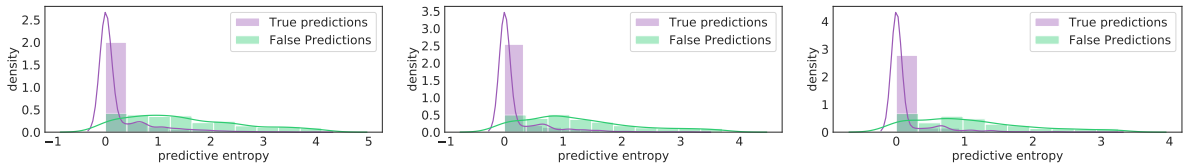
In Figure 3.6.2, the density histograms illustrate the predictive entropy as uncertainty estimate. Predictive entropy embodies the average amount of information in the predictive distribution. Given a data sample, the predictive entropy reaches its maximum value when all classes are predicted to have equal uniform probability (therefore, the model is highly uncertain), and its minimum value of zero when one class has probability 1 and all others probability 0 (i.e., the prediction is certain). We compare our approach versus MCDrop–30 and Deep–10, the best performing so far and the best state-of-the-art baselines. Although the three techniques can provide uncertainty, ours performs better than Deep–10 on all datasets and better than MCDrop–30 on the HHAR. Our results are similar to MCDrop–30 on the other two datasets; however, we obtain the illustrated uncertainty estimates in one single forward pass with great computational



(a) HHAR



(b) Opportunity



Ours MCDrop–30 Deep–10

(c) Speech Commands

Figure 3.6.2: Density histogram of predictive entropy for true (correct) and false (incorrect) predictions. Predictive entropy $H(\mathbf{y}|\mathbf{x}, D)$ captures the average amount of information contained in the predictive distribution. To reliably capture the predictive uncertainty, the predictive entropy should be low for true and high for false inferences. [The density histogram is a histogram with an area normalized to one. Plots are overlaid with kernel density curves for better readability.]

advantage and only a slight computation overhead compared to vanilla DNNs (see Section 3.6.2). Our approach performs better than the others in this respect since it assigns lower confidence to wrong predictions. Where our technique performs similarly to MCDrop or Deep, we still benefit from a significant computational gain.

The ideal confidence distribution would have a higher density towards 1.0 for true predictions and towards 0.0 for false predictions. This might be impossible in practice because, e.g., a correct prediction can be influenced by noise in input, resulting in low confidence. In our speech command task, although we include different background noises in training, there can still be other types of noise not seen in the training data which will cause uncertainty. The uncertainty in this prediction would trigger a reaction from the application, which might involve recording the command again and, ultimately, collecting these noisy data to include in the next training cycle in the cloud or smart device for personalization.

In the human activity recognition task, we see two prediction classes (stairs down and stairs up), which are occasionally confused by the model reflected in either a true prediction with high uncertainty or a wrong prediction with low uncertainty. While this is an undesired outcome, as the model should be well calibrated to handle these predictions, having the measure of uncertainty indicates that for these classes, the models should be engineered to extract more meaningful features. This glimpse of interpretability allows us to make decisions on the model architecture and how to process it in a hierarchical system, i.e., the user is momentarily given the parent class prediction (stairs), and then it is refined to a specific classification outcome (stairs up or down) after further computation and higher confidence.

While how the uncertainty is interpreted and the reaction to this trigger is application-specific, we provide an efficient framework that generalizes to different applications allowing us to dive deeper into the model reasoning, improving the overall system accuracy.

3.6.2 Latency and Energy Consumption

Figure 3.6.3 shows the latency (inference time) and energy consumption for all datasets on Nvidia Jetson TX2 and Nano. As we can see, our approach adds only a slight (max 20%) overhead over the conventional Backbone while being able to provide uncertainty estimates. Our latency is around 9-19ms per inference, depending on the dataset and the edge platform. The latency of MCDrop is significantly worse, and the time to perform inference increases with the number of runs (forward passes) being 20x times in the best scenario compared to our method and increasingly more in other cases (up to 28–fold). Similar trends can be observed for Deep (2x - 8x times) as they require running multiple neural networks.

The energy consumption measurements show similar patterns. Ours is also at least (20%) faster than one of the most recent approaches (ApDeepSense). In general, our method requires less energy than all the other approaches. It adds only a negligible or a tiny overhead (depending on the dataset) to the traditional DNN approach, which does not provide uncertainty estimates. Noticeably, our approach performs well on the Nano, highlighting that the applications can harness the utility of reliable predictions on many modern mobile and embedded devices, especially if latency could be slightly sacrificed (often the case for critical applications). Additionally, our CPU-only results on Nano demonstrate that our framework can run efficiently on resource-constrained devices that do not have a GPU.

As mentioned before, we want to ensure that our models can have a small footprint on these devices. For both embedded platforms, our investigations show that we add only a negligible

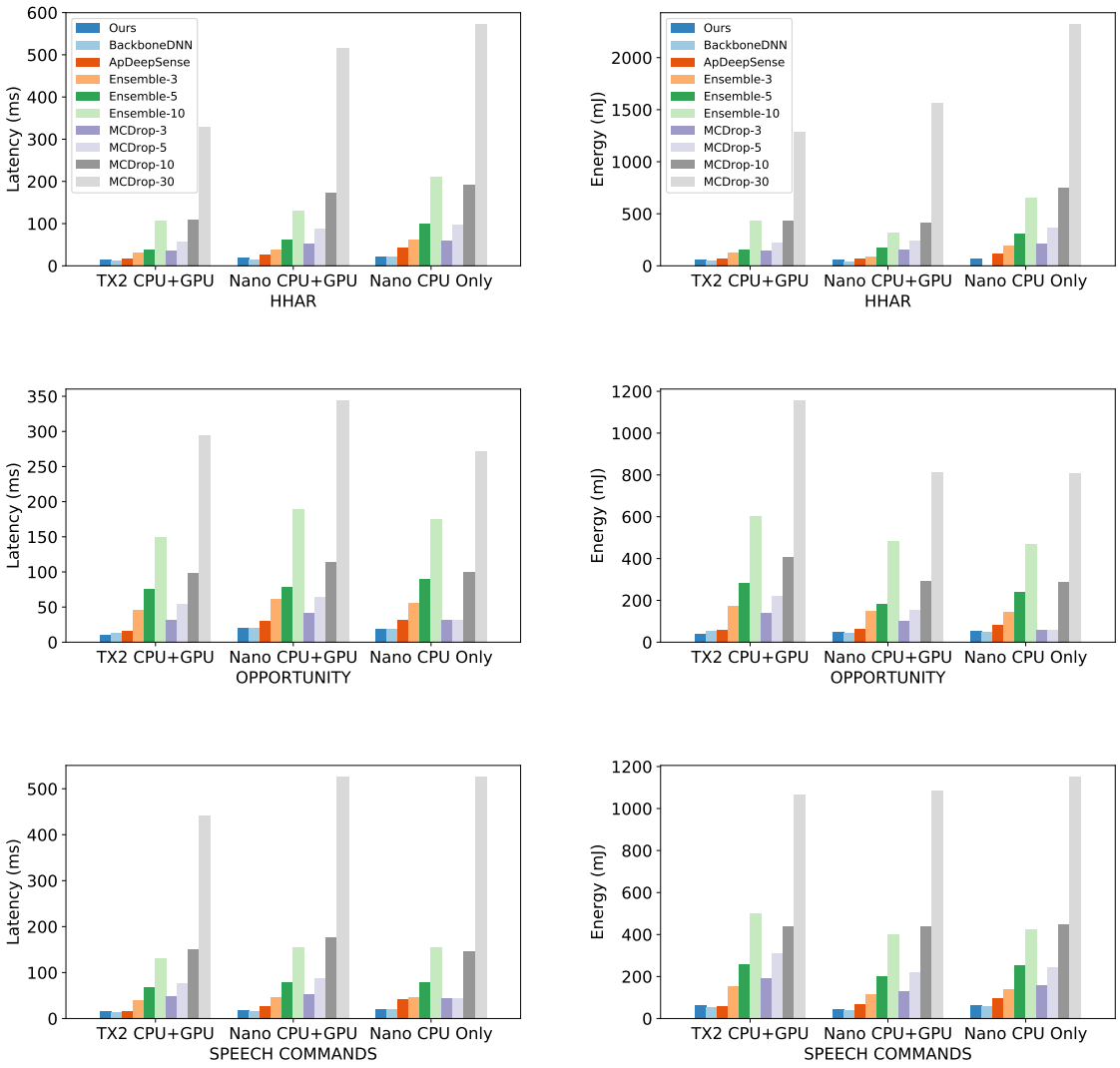


Figure 3.6.3: Inference time and energy consumption on two edge platforms with different CPU and GPU capacity. The latency indicates the average time it takes for the deep learning model to make a prediction on the provided sample. The energy consumed is expressed as *power x time*.

runtime memory overhead (max 5%) compared to the vanilla deep learning model while improving on the MLP baseline by 30%. However, deep ensembles start heavily using the memory swap on the Nano when passing five ensembles. Therefore, there is more need for memory sharing mechanisms. MCDrop does not contribute extra to the memory, but this technique relies on a lengthy computation time, as seen in all the results in this section.

3.7 Discussion and conclusions

In this chapter, we have introduced a framework able to provide uncertainty estimates through layerwise distribution approximations using only a single forward pass. It provides a simple and system-efficient solution that empowers convolutional neural networks to generate uncertainty estimations during inference for classification tasks. We evaluated our approach on multiple mobile sensing datasets and showed that it significantly outperforms state-of-the-art baselines in terms of computation, latency, and energy expenditure while providing reliable uncertainty estimates and accurate predictions.

In this chapter, we provide a powerful solution that does not require re-training or fine-tuning; however, it comes with some limitations too. The probabilistic baselines (MCDrop and Deep), due to their nature, can significantly improve their performance by continuously increasing the number of samples or models to apply but with a huge computational overhead. Instead, our approach considers hardware limitations by design by giving the uncertainty in one single forward pass; therefore, its performance improvements are limited to these constraints. Our approach includes the limitations of using Multivariate Gaussian distribution, too, which are common to Bayesian approximations using variational inference methods in general. For example, the problem of underestimation of the variance of the posterior distribution or the fact that the limited capacity of the posterior approximation can also result in biases in model parameters (and this is the case, e.g., in time-series models). Additionally, our framework focuses on providing predictive output distributions on models that have been trained with dropout regularization. A vast majority of modern architectures do not use dropout anymore but batch normalization. This slightly restricts the usage of this framework. To overcome the limitations of the solution presented in this chapter, in Chapter 4 we propose a new ensemble-based framework that can be applied to any feed-forward neural network independently of the type of architecture, layer, or regularization technique.

To conclude, uncertainty estimation brings the much-required element of interpretability and reasoning to the predictions made by neural network models. Such estimates are vital in the area of mobile, and embedded systems as these systems deal with different kinds of uncertainties. In this chapter, we have offered an avenue to provide them cheaply on these platforms while maintaining the needed level of performance.

Chapter 4

Early exit ensembles for uncertainty quantification

4.1 Introduction

In the previous chapter, we focused on already trained deep learning models and introduced a framework that enables efficient uncertainty estimation. While providing an excellent solution for already trained and deployed models, this approach comes with the limitations described in Sections 3.7 and further in detail in Section 4.5.9. The most important one is the condition of having models trained with dropout regularization; however, a vast majority of modern solutions use other methods for regularization, such as batch normalization, L1/L2 regularization, early stopping, data augmentation, or a combination of them. Therefore, there is a need for a more general solution that can support a wider range of deep learning architectures. In this chapter, motivated by the limitations of previous techniques (including ours in Chapter 3) described thoroughly in Section 2.4, the ease of implementation of deep ensembles for uncertainty estimation, as well as the wide adoption of early exit neural networks for efficient mobile applications (Montanari et al., 2019; Laskaridis et al., 2020a), we present early exit ensembles for uncertainty quantification - a novel framework for enabling uncertainty quantification in any feed-forward neural network in a single forward pass.

Ensembles of deep learning models can improve predictive performance by combining the output of multiple models (Peimankar and Puthusserypady, 2019; Zheng et al., 2020). Recently, these ensemble techniques have also been shown to provide reliable predictive uncertainty quan-

tification, where predictive uncertainty is defined as a probability distribution over multiple predictions on a single sample. These techniques can be broadly divided into implicit vs. explicit ensembles. Monte Carlo dropout, as extensively explained in Section 2.4.2, creates an implicit ensemble of networks by approximately sampling a weight distribution during inference. On the other hand, deep ensembles (Lakshminarayanan et al., 2017) and hyper-deep ensembles (Wenzel et al., 2020) are explicit ensembles composed of models of the same architecture independently trained with different weight initialization and hyperparameters, respectively. The techniques mentioned above, however, have limited use for real-time tasks due to their computational and memory overhead. In particular, Monte Carlo dropout requires multiple forward passes to create an ensemble which translates into higher latency and computational cost. On the other hand, for deep and hyper-deep ensembles, not only does the memory footprint scale linearly with the ensemble size, but also an increased computational burden is incurred from loading and running multiple models. In Chapter 3, we provided an extended analysis of this computation footprint on two edge platforms, NVIDIA Jetson TX2 and Nano (see Section 3.6.2).

To address the computational and memory bottleneck of the previous techniques for uncertainty quantification, we propose early exit ensembles via a novel interpretation of early exit neural networks (Teerapittayanon et al., 2016; Montanari et al., 2020). We summarize the main contributions of this chapter as follows:

- In our approach, early exit ensembles form an implicit ensemble of models from which predictive uncertainty can be quantified. Specifically, early exit ensembles are a collection of weight-sharing sub-networks that can be created by adding exit branches to any backbone deep learning architecture.
- We compare early exit ensembles to state-of-the-art deep learning uncertainty baselines in a series of experiments on the task of classification using real-world medical imaging datasets. Our evaluation includes data from different modalities (images and time series) and different model architectures (shallow and deep).
- Based on our experiments, early exit ensembles can provide significantly better uncertainty quantification compared to baselines such as Monte Carlo dropout and deep ensembles. On some datasets, negative log-likelihood, brier score, and expected calibration error are improved up to $2\times$. Furthermore, on out-of-distribution data, early exit ensembles can achieve a 77% higher predictive entropy compared to the state-of-the-art. We demonstrate that early exit ensembles are not only easy-to-implement but also perform optimally in terms of accuracy, uncertainty quantification, run-time, and memory on multiple model-

dataset combinations.

- Finally, our approach enables training all ensemble members jointly in a single model, as well as providing uncertainty estimations from a single forward-pass of input data. Early exit ensembles, therefore can be applied to a wide range of real-world applications in need of efficient uncertainty quantification.

4.2 Related Work

4.2.1 Uncertainty in medical imaging

For quantifying uncertainty in medical imaging tasks, the most commonly used techniques are approximate Bayesian methods such as Monte Carlo dropout and deep ensembles (Leibig et al., 2017; Rahman et al., 2021; Abdar et al., 2021). Few studies employ pure Bayesian methods such as Bayesian neural networks on medical data (Zhao et al., 2018; Lotter et al., 2021) due to the former techniques being simpler to implement. To date, the majority of applications of uncertainty quantification in deep learning on medical data have focused on images such as cancerous skin lesions (Abdar et al., 2021), diabetic retinopathy (Leibig et al., 2017), and brain magnetic resonance imaging (Zhao et al., 2018). Prior to the work in this chapter, there was no study of uncertainty quantification in deep learning that covers multiple medical imaging modalities (e.g. time series and images) across different architectures.

4.2.2 Early exit neural networks

As extensively described in Section 2.2.4, within the field of efficiency in deep learning, early exits are a class of conditional computation models that exit once a criterion (e.g., sufficient accuracy) is satisfied in order to save on computation (Wang et al., 2019; Li et al., 2019). Recent research leverages early exit predictions to dynamically change a neural network computation graph at test time (Nan and Saligrama, 2017; Montanari et al., 2019, 2020). The two most common use cases for early exits rely on either completing the full early exit inference before making a decision (Huang et al., 2017; Teerapittayanon et al., 2016) or applying a gating mechanism before the exit points in the backbone architecture (Bolukbasi et al., 2017). Under a different interpretation, the early exit paradigm can be used to mitigate the problem of model overthinking (Kaya et al., 2019) where the representations learned in earlier layers can be more accurate than those learned in the later layers which can sometimes not generalize well. As of yet, the

connection between early exit model architectures and uncertainty remains unexplored.

4.2.3 Multifidelity models

Multifidelity models are composed of multiple models which are able to estimate the same output quantity with varying approximation qualities and varying computational costs (Peherstorfer et al., 2018). These approaches are composed of low-fidelity and high-fidelity models, where a low-fidelity model is defined as a model that estimates the same output with a lower accuracy instead a high-fidelity model guarantees better accuracy at a higher computational cost. Similar to our early exit ensemble approach, a multifidely method combines predictions from computationally cheap low-fidelity models with outputs from the high-fidelity model leading to significant reductions in runtime and providing unbiased estimators of the statistics of the high-fidelity model outputs (Narayan et al., 2014; Ng and Willcox, 2014; Teckentrup et al., 2015; Peherstorfer et al., 2016). These multifidelity approaches aim at replacing the heavy-compute sampling-based approaches by shifting many of the model evaluations to low-fidelity models while evaluating the high-fidelity model a small number of times to establish unbiased estimators. In early exit ensembles for uncertainty quantification, however, the boundary between low- and high-fidelity models is not defined in a way to create only two categories but it includes models of incremental fidelity. Moreover, the fidelity of the models in the ensemble is not always fixed as their performance strictly depends on the specific input. Nevertheless, under the aforementioned considerations, early exit neural networks can be considered a special case of multifidelity models with the crucial distinction that although each of the exits proposes a different fidelity, they all share the same backbone.

4.3 Methods

We introduce a novel framework, *early exit ensembles*, which gives early exit neural networks a new probabilistic interpretation as an implicit ensemble. Herein, we present the methodology required to transform any feed-forward neural network into an efficient ensemble of weight-sharing sub-networks from which uncertainty can be quantified.

Problem formulation. Consider a classification problem where $\mathbf{x} \in \mathbb{R}^D$ denotes a D-dimensional input and $y \in \{1, \dots, C\}$ a corresponding discrete target taking one of C classes. We seek to learn a neural network that can model the probabilistic predictive distribution $p_\theta(y|x)$

over the ground truth labels, where θ are model parameters. This predictive distribution is obtained via the softmax transform $\xi(\cdot)$ of unnormalized log probabilities $f_\theta(\mathbf{x}) \in \mathbb{R}^C$.

Neural networks. By definition of a NN, $f_\theta(\cdot)$ consists of blocks of differential operations (e.g., convolution) (Goodfellow et al., 2016). We assume, therefore that $f_\theta(\cdot)$ can be decomposed into a composition of B blocks such that

$$f_\theta(\mathbf{x}) = (f^{(B)} \circ f^{(B-1)} \circ \cdots \circ f^{(1)})(\mathbf{x}) \tag{4.1}$$

where $(f^{(i)} \circ f^{(j)})(\mathbf{x}) = f_{\theta_i}(f_{\theta_j}(\mathbf{x}))$ denotes function composition for $i \neq j$ and $\theta = \cup_{i=1}^{B} \theta_i$ represents the union of each blocks parameters. Let $\mathbf{h}^{(i)} \in \mathbb{R}^{K_i \times D_i}$ denote the intermediary output of the i-th block having K_i features of dimension $D_i \leq D$ such that $\mathbf{h}^{(i)} = f_{\theta_i}(\mathbf{h}^{(i-1)})$ for $1 \leq i \leq B-1$, and $\mathbf{h}^{(0)} = \mathbf{x}$.

4.3.1 Early Exit Ensembles

With only minor architectural modifications, any multi-layered feed-forward neural network can be converted into an implicit ensemble of networks by adding early exits. We define an *early exit block* as a neural network $g_{\phi_i}(\cdot)$ with parameters ϕ_i which takes as input the intermediary output $\mathbf{h}^{(i)}$ from the i-th block of the backbone neural network $f_\theta(\cdot)$. We let each exit block learn a probabilistic predictive distribution $p_{\phi_i}(y|\mathbf{x})$ such that

$$\begin{aligned} p_{\phi_i}(y|\mathbf{x}) &= \xi(g_{\phi_i}(\mathbf{h}^{(i)})) \\ &= \xi((g^{(i)} \circ f^{(i-1)} \circ \cdots \circ f^{(1)})(\mathbf{x})) \end{aligned} \tag{4.2}$$

for $1 \leq i \leq B-1$. As such, any NN is able to output a set $\mathcal{M}$ containing up to $B-1$ probabilistic distributions from early exits blocks, in addition to the standard output from its final block

$$\mathcal{M} = \{p_{\phi_1}(y|\mathbf{x}), \ldots, p_{\phi_{B-1}}(y|\mathbf{x}), p_\theta(y|\mathbf{x})\} \tag{4.3}$$

where $|\mathcal{M}| = B$. In practice, the number of exit blocks and therefore the ensemble size $|\mathcal{M}|$ is a hyperparameter bounded above by B and bounded below by a combination of factors such as computational cost (Kaya et al., 2019) and quality of the uncertainty estimates (Ovadia et al., 2019). Figure 4.3.1 provides a visual representation of an early exit ensemble.

To train an early exit ensemble, we optimize a weighted sum of each exit's individual predictive loss. This procedure allows the training of the whole ensemble jointly. As such, each prediction

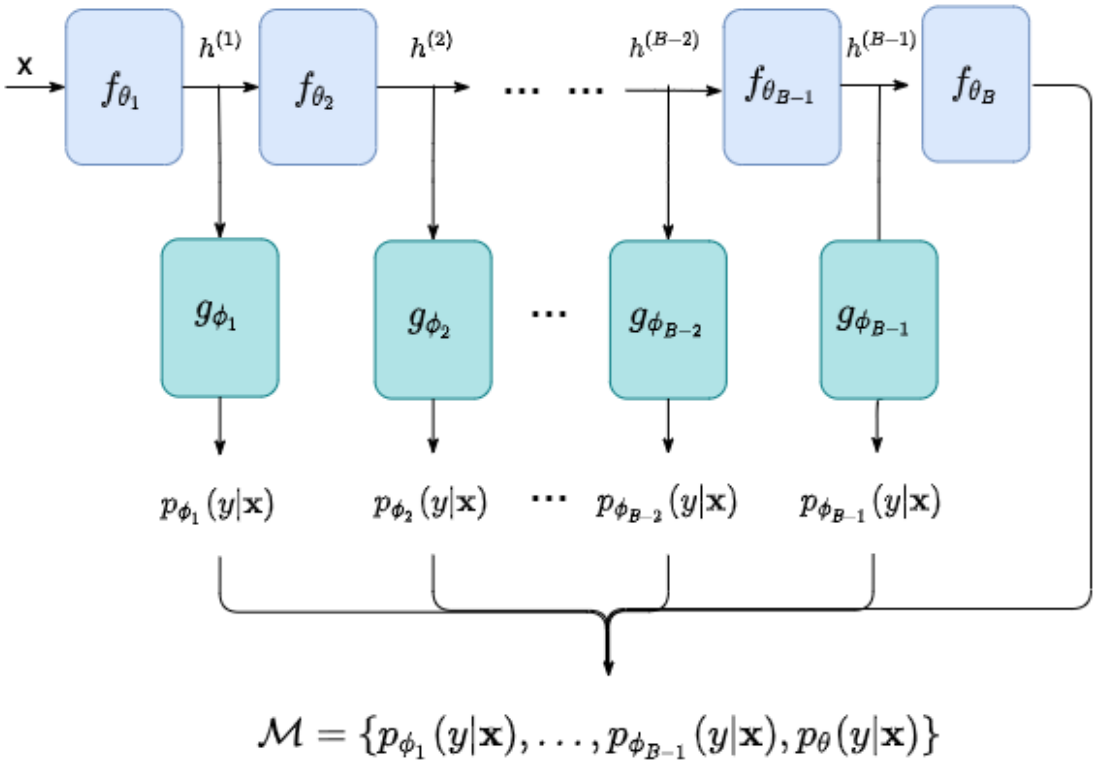


Figure 4.3.1: Representation of an early exit ensemble. Each f_{θ_i} represents a block from the backbone NN. Exit block g_{ϕ_i} take as inputs intermediary output $h^{(i)}$ shared by the backbone (and previous exits) and learns a predictive distribution $p_{\phi_i}(y|\mathbf{x})$.

propagates the error in relation to the ground truth label to the preceding exit blocks. More formally:

$$\mathcal{L} = L_{CE}(y, f_\theta(y|\mathbf{x})) + \sum_{i=1}^{B-1} \alpha_i L_{CE}(y, g_{\phi_i}(y|\mathbf{x})) \tag{4.4}$$

where $L_{CE}(\cdot, \cdot)$ is the cross-entropy loss function and $\alpha_i \in [0, 1]$ is a weight hyperparameter corresponding to the relative importance of each exit.

During inference, a single forward pass of a neural network with early exits produces an ensemble $\mathcal{M}$ of predictions. The overall prediction from $\mathcal{M}$ can be computed as the mean of a categorical distribution obtained from averaging the predictions from the individual exits:

$$p_{\theta_\mathcal{M}}(y|\mathbf{x}) \approx \frac{1}{|\mathcal{M}|}\big(p_\theta(y|\mathbf{x}) + \sum_{i=1}^{B-1} p_{\phi_i}(y|\mathbf{x})\big). \tag{4.5}$$

where $\theta_\mathcal{M} = \theta \cup \phi$ and $\phi = \cup_{i=1}^{B-1}\phi_i$. Compared to a single model prediction, an ensemble provides more information such as variance, entropy, and disagreement (as measured by Kullback-Leibler (KL) divergence), that can be exploited for better-calibrated predictive probabilities during both training and inference. Although other methods of aggregating the output of the ensemble exist (e.g. majority voting, geometric mean, or weighted average), previous works have shown that for uncertainty quantification, the arithmetic mean works well in practice (Gal and Ghahramani, 2015a; Lakshminarayanan et al., 2017).

4.3.2 Exit Block Architecture

The exit blocks are built to satisfy two important characteristics that have been shown to make a good ensemble: accuracy and diversity (Perrone and Cooper, 1992; Granitto et al., 2005). Since early exit blocks exit from different points in the backbone architecture, their predictions are naturally based on features learned from structurally distinct architectures, thereby promoting diversity. However, the exits from earlier blocks in the backbone inherit intermediary outputs with a weak representational capacity which can negatively impact the overall ensemble accuracy. While the earlier exits can learn from the deeper layers in the joint training procedure, they might not possess enough parameters to match the representational capacity of the last exits in the network. This might be acceptable in a traditional early exit neural network paradigm, but it is not valid for our ensemble technique since all exits have the same weight in the uncertainty quantification therefore the accuracy of each and every one of them cannot be compromised. To address this issue, we add an additional layer to the exit block which allows for increasing the

number of features and, consequently, the learning capacity of the exit. Of course, the deeper we go into the network fewer parameters are needed as the backbone compensates for them. For this purpose, we need the parameter increase in the exit block to be inversely proportional to the exit point in the backbone. Nevertheless, in some scenarios, there might not be a need for more parameters so we still need the option of having a lightweight exit block that only contains one layer needed for the intermediate classification.

For this purpose, we design a conditional architecture for the i-th exit block as follows:

$$g_{\phi_i}(\mathbf{h}^{(i)}) = \begin{cases} \mathbf{W}_2^{(i)}\rho(\mathbf{W}_1^{(i)}s(\mathbf{h}^{(i)}) + \mathbf{b}_1^{(i)}) + \mathbf{b}_2^{(i)} & \gamma > 0 \\ \mathbf{W}_3^{(i)}s(\mathbf{h}^{(i)}) + \mathbf{b}_3^{(i)} & \gamma = 0 \end{cases}$$

where $s(\cdot)$ denotes global average pooling used to reduce the spatial dimension D_i of each feature, $\rho(\cdot)$ is an activation function, $\mathbf{W}_1^{(i)} \in \mathbb{R}^{K_\gamma \times K_i}$, $\mathbf{W}_2^{(i)} \in \mathbb{R}^{C \times K_\gamma}$, $\mathbf{W}_3^{(i)} \in \mathbb{R}^{C \times K_i}$ and $\mathbf{b}_1^{(i)} \in \mathbb{R}^{K_\gamma}$, $\mathbf{b}_2^{(i)}, \mathbf{b}_3^{(i)} \in \mathbb{R}^C$ are weights and biases of linear layers respectively. The hyperparameter $\gamma \geq 0$ is a learning capacity factor used to increase the number of features from K_i to K_γ of the i-th intermediary output such that

$$K_i^\gamma = (\sqrt{1+\gamma})^{B-i} K_B, 1 \leq i \leq B-1 \tag{4.6}$$

where K_B is the number of features in the last block defined by the backbone. Intuitively, when $\gamma > 0$ the number of features in each exit block is inversely proportional to the exit point i.e. earlier exits use additional parameters to learn more complex relations between features. For instance, let us consider an architecture composed of $B = 10$ blocks with exits at $\{2, 5, 6, 8, 10\}^{th}$ block, $K_B = 512$ and $\gamma = 0.2$. Following Equation 4.6, for each exit block, we will have a decreasing number of parameters, specifically $K_2^{0.2} = 1062$, $K_5^{0.2} = 808$, $K_6^{0.2} = 737$, $K_8^{0.2} = 614$, and $K_{10}^{0.2} = 512$.

4.3.3 Exit Placement

In practice, the number of exit blocks is determined by the backbone architecture as well as computational cost (Kaya et al., 2019) and quality of the provided uncertainty estimates (Ovadia et al., 2019). Therefore the ensemble size $|\mathcal{M}|$ is a hyperparameter bounded above by B. As such, there is a combinatorial choice of exit points and therefore ensemble arrangements. To limit the search space, we introduce exit placement strategies in Table 4.5.4.

Strategy	Exit after
Block-wise	Every block.
Pareto	Blocks closest to 20% and 80% of total FLOPs.
Computation	Blocks closest to $\{15, 30, 45, 60, 75, 90\}$% of total FLOPs.
Residual	Residual blocks.
Last-k	Last k blocks.
Semantic	Blocks grouped by number and size of feature maps.

Table 4.3.1: Exit placement strategies for any backbone architecture. FLOPs: floating point operations.

For example, given a ResNet18 backbone (with blocks defined as convolution, batch normalization, and activation), the $Semantic$ strategy places exit blocks after each of the last blocks of kernel size and number features $(3, 64)$, $(3, 128)$, $(3, 256)$, and $(3, 512)$ resulting in size $|\mathcal{M}| = 5$. See Section 4.5.5 for more details.

4.3.4 Computational Cost

An early exit ensemble is an implicit ensemble since it constitutes a collection of sub-networks that share the weights θ from the backbone. Table 4.3.2 compares the computational cost of early exit ensembles vs. an ensemble of independent models and a single model (assuming the same backbone). Explicit ensembles like deep ensembles, linearly increase the number of parameters with the number of members, and therefore, they are not a cost-efficient technique. The only memory overhead introduced by early exit ensembles are the parameters from exit blocks $\phi = \cup_{i=1}^{|\mathcal{M}|-1}\phi_i$. Since in general $\phi << \theta$ where $\theta = \cup_{i=1}^{B}\theta_i$, early exit ensembles achieve computational and memory gains due to weight sharing.

Ensemble	Size	FLOPs	Compute
Single	θ	F	τ
Independent	$\theta * \|\mathcal{M}\|$	$F * \|\mathcal{M}\|$	$\tau * \|\mathcal{M}\|$
Early exit	$\theta + \phi$	$F + F_\phi$	$\tau + \tau_\phi$

Table 4.3.2: Computational and memory cost. F: FLOPs. τ: compute time. F_ϕ and τ_ϕ are FLOPs (floating-point operations) and compute time for all exit blocks respectively.

4.4 Experiments

We design experiments to verify the improvement of early exit ensembles in providing well-calibrated uncertainty quantification for in-distribution data compared to state-of-the-art deep learning ensemble baselines (Section 4.5.1). We then further verify the ability of early exit ensembles to detect out-of-distribution samples (ODD) (Section 4.5.2) as well as study its behavior on in-distribution shifts in Section 4.5.3. Finally, we analyze latency and power consumption on the NVIDIA Jetson TX2 embedded device while comparing it to the baselines in these efficiency terms (Section 4.5.4).

4.4.1 Datasets

We evaluate our proposed early exit ensembles on four medical imaging classification datasets: (1) ECG heart attack (ECG5000) (Dau et al., 2019), (2) EEG epileptic seizure (EEG-S) (Dua and Graff, 2017), (3) skin melanoma (ISIC2018) (Codella et al., 2019) and (4) EEG eye movement artifact (EEG-A). Table 4.4.1 contains a descriptive summary of each dataset. All datasets are split into 80% training, 10% validation, and 10% testing maintaining the original class proportions.

Dataset						
Name	**Type**	**Train**	**Valid**	**Test**	**Classes**	**Input dimension**
ECG5000	Time series	4050	450	500	2	1×140
EEG-S	Time series	9315	1035	1150	5	1×178
ISIC2018	Image	8268	919	1021	7	$3 \times 224 \times 224$
EEG-A	Time series	6400	800	800	2	1×2000

Table 4.4.1: Summary of the datasets used for experiments. All datasets are split into 80% training, 10% validation, and 10% testing maintaining the original class proportions.

ECG heart attack (ECG5000). (Dau et al., 2019) A dataset of univariate timeseries of ECG signals of length 140 was extracted from a single patient. Each signal falls into one of 5 classes which are combined to make two labels: Normal (N) and Abnormal (R-on-T, PVC, SP, UB). The original train and test datasets are combined creating a dataset of size 5000 which is re-split maintaining class proportions. No further data pre-processing is performed.

EEG epileptic seizure (EEG-S). (Dua and Graff, 2017) A dataset of univariate timeseries of single-channel EEG signal of length 178 extracted from 500 patients. Each signal falls into one of 5 classes: normal patient eyes open, normal patient eyes closed, tumor patient healthy area, tumor patient tumor area, epileptic patient seizure. The original train and test datasets are combined creating a dataset of size 11500 which is re-split maintaining patient and class proportions. During training, a combination of Gaussian noise, signal shift, and polarity inversion is applied with a probability of 0.5.

Skin melanoma (ISIC2018). (Codella et al., 2019) A dataset of multi-source dermatoscopic images of common pigmented skin lesions. Each image falls into one of 7 classes: melanocytic nevi, melanoma, benign keratosis-like lesions, basal cell carcinoma, actinic keratoses, vascular lesions, and dermatofibroma. The original train and validation datasets from Task 3 are combined creating a dataset of size 10208 which is re-split maintaining class proportions. All images are resized to 224×224 and then normalized using the training dataset mean and standard deviation. During training, a combination of Gaussian noise, horizontal and vertical flips, and jitter is applied to each image with a probability of 0.5.

EEG eye artifact (EEG-A). (Hamid et al., 2020) EEG-A consists of univariate timeseries of length 2000 extracted from 213 patients from the Temple University Artifact Corpus (v2.0). To create the dataset, a common set of 21 EEG channels were retained from all patients, and signals were resampled to 250 Hz. In addition, all EEG signals were bandpass filtered (0.3-40 Hz) using a second-degree Butterworth filter and notch filtered at the power lower frequency 60 Hz similar to previous work (Svantesson et al., 2021). Segments of clean and eye movement artifact signals were selected to form the EEG eye movement artifact dataset used here. Finally, all signals were min-max scaled to the range $[0, 1]$. During training, a combination of signal shift and polarity inversion was applied with a probability of 0.5.

For *out-of-distribution* analysis, we use three datasets containing the same type of signal but from completely different distributions: **ECG heart attack (ECG200)** (Dau et al., 2019), **EEG Steady-State Visual Evoked Potential Signals (SSEVP)** (Dua and Graff, 2017), and **CIFAR-10** (Krizhevsky and Hinton, 2010).

4.4.2 Backbone architectures

Each dataset is paired with a different architecture: FCNet for ECG5000, ResNet18 for EEG-S, MobileNetV2 for ISIC2018 and VGG16 for EEG-A (see Table 4.4.2).

Model		
Name	B	**Parameters**
FCNet	5	0.2 M
ResNet18	9	3.8 M
MobileNetV2	18	1.3 M
VGG16	14	23.8M

Table 4.4.2: Summary of the models used for experiments. B is the number of blocks in each architecture. Parameters represent the number of trainable model parameters (millions).

FCNet. (Wang et al., 2017) A fully convolutional network consisting of 4 blocks each containing a 1D convolution, batch normalization (BN), and Rectified Linear Unit (ReLU) activation. The output of the fourth block is averaged over the time dimension using global average pooling (GAP) and fed to a 1D convolutional layer with filter length 1. Convolutional layers all have 128 filters of lengths 8, 5, 5, and 3 all with a stride of 1 and zero padding to preserve the length of each time series input.

ResNet18. (He et al., 2016; Bizopoulos et al., 2019) A residual convolutional network composed of 8 blocks containing a 1D convolution, BN, and ReLU activation repeated twice. Convolutions in consecutive pairs of blocks have filters 64, 128, 256, and 512 with strides 1, 2, 2, and 2 all with length 3. The output of these blocks is fed to a GAP and a fully connected (FC) layer. Identity residual connections exist between consecutive blocks with the same number of filters. Downsample residual connections exist between blocks with different filter numbers.

MobileNetV2. (Sandler et al., 2018; Chaturvedi et al., 2020) A convolution neural network mobile architecture composed of 17 inverted residual blocks with bottleneck layers. Each inverted residual block contains a 2D convolution of size 1x1, a 2D depthwise convolution of size 3x3, a Rectified Linear Unit 6 (ReLU6), and finally a 2D convolution of size 1x1 followed by ReLU6.

VGG16. (Simonyan and Zisserman, 2014) A convolutional neural network composed of 13 blocks containing a 1D convolution, BN, and ReLU activation. Convolutions filters are distributed across the blocks following 2×64, 2×128, 2×256, 3×512, 3×512 all with length 3 and stride 1. Consecutive blocks with different numbers of filters are interleaved with a max pooling layer of length 2 and stride 2. The output of these blocks is fed to a GAP and 3 FC layers

with dropout and ReLU activation.

4.4.3 Baselines

We compare early exit ensembles against its backbone without exits (Backbone), Monte Carlo dropout (MCDrop) (Gal and Ghahramani, 2015a), deep ensembles (Deep) (Lakshminarayanan et al., 2017) and depth ensembles (Depth).

Backbone. The unchanged implementations of FCNet, ResNet18, MobileNetV2, and VGG16 represent a single model which outputs softmax probabilities over classes.

MCDrop. An approximate Bayesian probabilistic approach is implemented by adding dropout layers to each of the backbone architectures during training and activating them during inference. For fairness of comparison, dropout layers are added to the same places as the exit points. A total of 5 Monte Carlo samples are taken with a dropout rate of $p_{\text{MC}} = 0.2$.

Deep. A non-Bayesian probabilistic approach based on training an explicit ensemble of models with the same backbone architecture but trained with a different random weight initialization. A total of 5 models were used for the ensemble.

Depth. Our own baseline is composed of an ensemble of the same backbone where each member has a different depth determined by the exit points. Similar to Deep, all ensemble members are trained separately. In contrast, each of the 5 models ranges from shallow to deep and is trained with the same weight initialization.

Layerwise distribution approximation. This baseline represents the framework developed in Chapter 3 where we enable the convolution operations in the stochastic network generated by dropout to apply to inputs described by probabilistic distributions and generate distributions as outputs. For every layer, the network is described by inputs (and consequently outputs) which are Gaussian distributions with a mean and variance. A comparison to this baseline is provided in Section 4.5.9.

4.4.4 Placement of exits

Following findings from recent works on the optimal number of ensemble members to consider for well-calibrated uncertainty (Ovadia et al., 2019), we set the ensemble size $|\mathcal{M}| = 5$ for all

models. Since for FCNet this results in no choice of exit placement, we exit after each block. On the other hand, for ResNet18 we semantically group residual blocks based on the number of hidden features (based on findings from different exit strategies detailed in Section 4.5.5). For MobileNetV2, we follow a similar strategy to Kaya et al. (2019) by placing exit points based on the overall number of floating-point operations (FLOPs) in the model. As such, we place exit points closest to 45%, 60%, 75%, and 90% of the overall FLOPs for MobileNetV2. For VGG16, the best strategy was found to be following the inherent semantic grouping of the architecture.

4.4.5 Implementation details

We use the data preprocessing and hyperparameters described in the original implementation of each model-dataset combination as the initial starting point for hyperparameter tuning. All models are trained with the Adam (Kingma and Ba, 2014) optimizer using default parameters, except for the learning rate which is tuned over $\{1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}\}$. For MCDrop, the dropout rate is tuned over $\{0.2, 0.3, 0.5\}$. We empirically choose batch sizes and epochs from 50, 100, 200 and 100, 200, 250 respectively based on stability of training. To prevent overfitting, early stopping is implemented with patience of 5 based on the best validation accuracy. We train early exit ensembles with the best performing hyperparameters across all backbone models $\alpha_i = 1$. Finally, all models were implemented using PyTorch (Paszke et al., 2019) and trained on a 4-GPU Linux server with 64GB memory.

4.5 Results

We present the results of early exit ensembles on in-distribution (Section 4.5.1) and out-of-distribution (Section 4.5.2) experiments and compare them to the aforementioned baselines. Additionally, we analyze the effect of the number and position of exit blocks (Section 4.5.5), the learning capacity factor (Section 4.5.6), and diversity regularization (Section 4.5.7) as well as exit loss weighting (Section 4.5.8).

4.5.1 Uncertainty Estimation

Table 4.5.1 shows classification results on in-distribution test datasets. Our experiments show that the addition of early exits, and their joint training, improves accuracy compared to the backbone model across all datasets up to 3.8 percentage points as measured by the F1 score. These findings are in line with existing work on early exit architectures (Teerapittayanon et al., 2016).

Model	F1 (↑)	Precision (↑)	Recall (↑)
FCNet			
- Backbone	0.983 ± 0.010	0.983 ± 0.010	0.983 ± 0.010
- MCDrop	0.987 ± 0.002	0.987 ± 0.002	0.987 ± 0.002
- Depth	0.989 ± 0.007	0.989 ± 0.007	0.989 ± 0.007
- Deep	0.989 ± 0.003	0.989 ± 0.003	0.989 ± 0.003
- Early exit	$\mathbf{0.992 \pm 0.005}$	$\mathbf{0.992 \pm 0.005}$	$\mathbf{0.992 \pm 0.005}$
ResNet18			
- Backbone	0.847 ± 0.012	0.848 ± 0.012	0.847 ± 0.012
- MCDrop	0.844 ± 0.007	0.849 ± 0.004	0.844 ± 0.007
- Depth	0.861 ± 0.011	0.862 ± 0.011	0.861 ± 0.011
- Deep	$\mathbf{0.866 \pm 0.009}$	$\mathbf{0.866 \pm 0.008}$	0.866 ± 0.009
- Early exit	0.865 ± 0.002	0.865 ± 0.002	$\mathbf{0.866 \pm 0.002}$
MobileNetV2			
- Backbone	0.868 ± 0.005	0.869 ± 0.006	0.870 ± 0.003
- MCDrop	0.865 ± 0.002	0.872 ± 0.001	0.861 ± 0.002
- Depth	0.865 ± 0.011	0.873 ± 0.010	0.861 ± 0.011
- Deep	$\mathbf{0.887 \pm 0.011}$	$\mathbf{0.888 \pm 0.011}$	$\mathbf{0.888 \pm 0.009}$
- Early exit	0.885 ± 0.005	0.885 ± 0.004	0.886 ± 0.005
VGG16			
- Backbone	0.809 ± 0.011	0.810 ± 0.014	0.809 ± 0.011
- MCDrop	0.822 ± 0.011	0.830 ± 0.010	0.825 ± 0.013
- Depth	0.821 ± 0.023	0.830 ± 0.020	0.820 ± 0.022
- Deep	0.838 ± 0.010	0.839 ± 0.011	0.839 ± 0.010
- Early exit	$\mathbf{0.847 \pm 0.003}$	$\mathbf{0.848 \pm 0.004}$	$\mathbf{0.847 \pm 0.003}$

Table 4.5.1: Results for classification metrics. All results are for the optimally tuned learning rate, batch size, and number of epochs for each model: FCNet ($1e^{-2}$, 200, 250), ResNet18 ($1e^{-3}$, 200, 200), MobileNetV2 ($1e^{-5}$, 100, 200) and VGG16 ($1e^{-4}$, 200, 200). Results for MCDrop are for the optimal dropout rate $p_{\text{MC}} = 0.2$. Results for Early Exit are for the weighted classification loss $\alpha_i = 1$ and learning capacity factor: FCNet (0.0), ResNet18 (0.2), MobileNetV2 (0.7), VGG16 (0.5). We mark in bold the best results. All results are the mean plus/minus standard deviation across 3 independent splits of the test dataset.

Furthermore, Early Exit, Deep, Depth, and MCDrop all display significantly better F1 scores, precision, and recall compared to Backbone across each of the three datasets. We attribute these results to a reduction in variance caused by averaging a set of NNs that individually have a high variance and low bias (Perrone and Cooper, 1992). For ResNet18 and MobileNetV2, Deep is on average 0.2% better than Early Exit across classification metrics. However, this marginal

Model	NLL (↓)	ECE (↓)	Brier score (↓)
FCNet			
- Backbone	0.059 ± 0.031	0.009 ± 0.005	0.026 ± 0.015
- MCDrop	0.043 ± 0.019	0.011 ± 0.004	0.019 ± 0.004
- Depth	0.036 ± 0.008	0.017 ± 0.007	0.018 ± 0.006
- Deep	0.045 ± 0.020	0.014 ± 0.007	0.018 ± 0.005
- Early exit	$\mathbf{0.024 \pm 0.008}$	$\mathbf{0.007 \pm 0.001}$	$\mathbf{0.009 \pm 0.003}$
ResNet18			
- Backbone	0.432 ± 0.022	0.081 ± 0.007	0.233 ± 0.018
- MCDrop	0.362 ± 0.012	0.045 ± 0.005	0.216 ± 0.011
- Depth	0.318 ± 0.028	0.028 ± 0.003	0.194 ± 0.012
- Deep	0.316 ± 0.024	0.028 ± 0.004	0.189 ± 0.011
- Early exit	$\mathbf{0.306 \pm 0.013}$	$\mathbf{0.027 \pm 0.006}$	$\mathbf{0.189 \pm 0.005}$
MobileNetV2			
- Backbone	0.512 ± 0.022	0.080 ± 0.008	0.209 ± 0.010
- MCDrop	0.375 ± 0.016	0.041 ± 0.008	0.197 ± 0.008
- Depth	0.495 ± 0.005	0.133 ± 0.017	0.237 ± 0.004
- Deep	0.359 ± 0.019	0.037 ± 0.008	0.171 ± 0.007
- Early exit	$\mathbf{0.357 \pm 0.020}$	$\mathbf{0.033 \pm 0.010}$	$\mathbf{0.170 \pm 0.006}$
VGG16			
- Backbone	0.589 ± 0.040	0.109 ± 0.020	0.308 ± 0.015
- MCDrop	0.574 ± 0.051	0.093 ± 0.012	0.279 ± 0.014
- Depth	0.438 ± 0.018	0.057 ± 0.009	0.275 ± 0.010
- Deep	0.400 ± 0.004	0.039 ± 0.004	0.239 ± 0.007
- Early exit	$\mathbf{0.385 \pm 0.005}$	$\mathbf{0.033 \pm 0.010}$	$\mathbf{0.236 \pm 0.003}$

Table 4.5.2: Results for uncertainty metrics. We mark in bold the best results. All results are the mean plus/minus standard deviation across 3 independent splits of the test dataset.

sacrifice in accuracy is negligible compared to the improvement gain in uncertainty metrics (see Table 4.5.2). For instance, Early Exit improves model ECE by an estimated 4% and 11% for ResNet18 and MobileNetV2, respectively.

Early Exit clearly outperforms MCDrop both in terms of classification and uncertainty metrics. In particular (see Table 4.5.2), BS is improved by an estimated 53%, 13%, and 14% for FCNet, ResNet18 and MobileNetV2, respectively. In addition, compared to the current state-of-the-art Deep, Early Exit improves calibration as measured by ECE for ResNet18 (0.028 vs 0.027) and VGG16 (0.039 vs 0.033) representing a 3.8% and 15.4% decrease, respectively. Such improvement translates into the predicted probability distributions over the classes more accurately

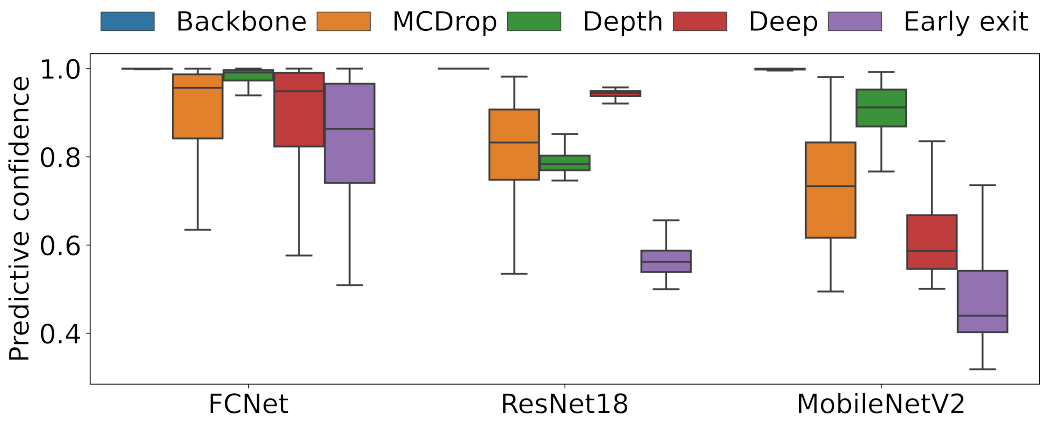


Figure 4.5.1: Predictive confidence (PC) on out-of-distribution samples.

reflecting the ground truth distribution in the data. Moreover, the generalizability of early exit ensembles is demonstrated by improvement in uncertainty quantification for both deep (ResNet18 and MobileNetV2) and shallow architectures (FCNet). In particular, for FCNet uncertainty metrics are roughly $2\times$ less compared to Deep and MCDrop. In general, this translates into Early Exit displaying greater/lesser uncertainty under out/in-distribution data shifts (see Section 4.5.3).

Finally, depth ensembles perform well across all metrics compared to MCDrop for both FCNet and ResNet18. In particular, Depth increases the F1 score by an estimated 2% and improves NLL by 14% compared to MCDrop for ResNet18. These results demonstrate that when forming an ensemble, diversity in terms of different NN architecture depths improves model calibration. Interestingly, Depth does not outperform Early Exit in either classification or uncertainty metrics. We attribute this difference to Early Exit having the benefit of exit blocks that are jointly trained with a backbone architecture. As a result, Early Exit not only benefits from shared features from the backbone that are informative for the task but is also able to diversify these features at each exit block allowing it to make accurate but independent predictions.

4.5.2 Out-of-Distribution Detection

Figures 4.5.1 and 4.5.2 show predictive confidence (PC) and predictive entropy (PE) on out-of-distribution datasets. A well-calibrated model should show low PC and high PE on out-of-distribution data. Our results clearly show across all datasets that Backbone is overly confident in making wrong predictions as reflected by an average PC of 0.99 and a PE entropy of 0.00. This demonstrates the risk of deploying deep learning models which are unable to provide proper uncertainty estimates potentially resulting in overconfident wrong diagnoses. Our proposed early

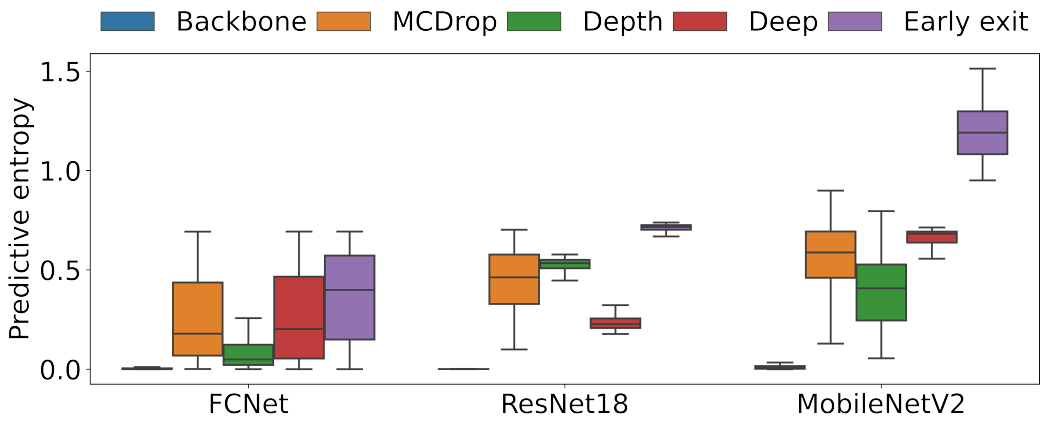


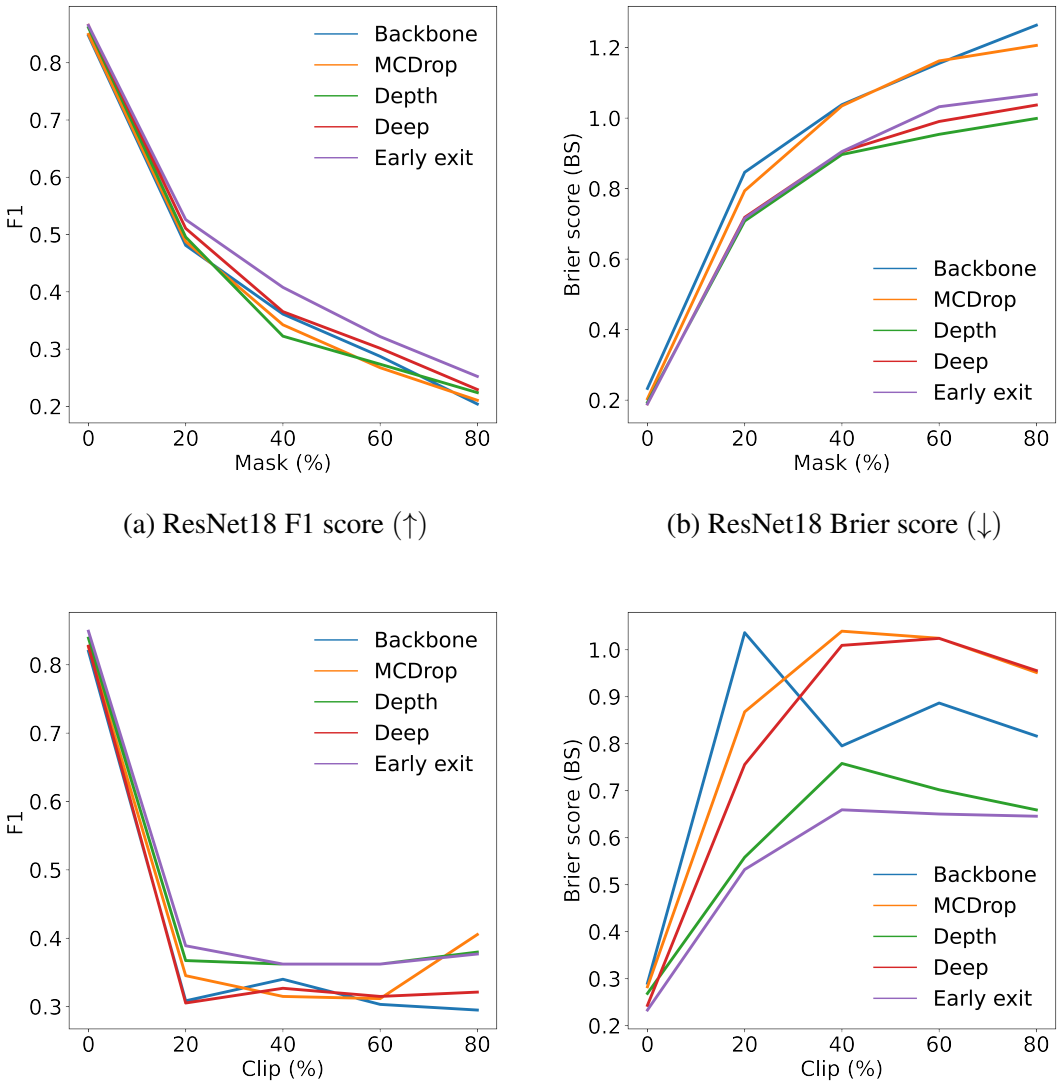
Figure 4.5.2: Predictive entropy (PE) on out-of-distribution samples.

exit ensemble framework provides the highest PE and lowest PC for out-of-distribution samples. In particular, Early Exit has an estimated 38%, 77%, and 34% higher entropy for FCNet, MobileNetV2, and ResNet18 compared to their respective best-performing baselines.

In contrast to previous works, Deep does not always out outperform MCDrop. For FCNet, Deep is comparable to MCDrop in terms of both average confidence and entropy. For ResNet18, MCDrop has a significantly lower PC (0.81 vs. 0.94) and PE (0.44 vs. 0.24). Depth ensembles do not show good calibration on OOD data having the worse performance for FCNet and MobileNet on both PC (0.97 and 0.90, respectively) and PE (0.10 and 0.40, respectively). While they showed acceptable uncertainty metrics in in-distribution, they cannot be trusted in OOD scenarios.

4.5.3 Calibration on in-distribution shifts

For biosignal classification, the input distribution is often shifted from the training distribution due to different hardware and/or data collection protocols (Mårtensson et al., 2020). In such scenarios, a model with well-calibrated uncertainty can indicate if a prediction should be trusted (Ovadia et al., 2019). Figure 4.5.3 displays the effect of in-distribution shifts on EEG-S and EEG-A test data for ResNet18 and VGG16, respectively. We apply two types of shifts: signal masking and amplitude clipping. Signal masking represents missing EEG signals caused by electrode movement or temporary malfunction. Signal clipping, instead, represents amplifier saturation caused by excess voltage. For all models, accuracy decreases as each shift intensifies. However, Early Exit displays greater robustness by maintaining a higher F1 score across all intensities for both shift types compared to baselines. In terms of the accuracy of predicted probabilities (as measured by BS), Early Exit performs better on clipping for VGG16 while following



(a) ResNet18 F1 score (↑)

(b) ResNet18 Brier score (↓)

(c) VGG16 F1 score (↑)

(d) VGG16 Brier score (↓)

Figure 4.5.3: In-distribution shifts applied to $[0\%, 20\%, 40\%, 60\%, 80\%]$ of test data for ResNet18 (masking) and VGG16 (clipping). A well-calibrated model shows a higher F1 and lower Brier score across all shift percentages.

a similar trend to Deep and Depth for ResNet18 up to a shift intensity of 40%.

			MCDrop	Depth	Deep	Early exit
FCNet	**Size**	(↓)	**0.20**	0.70	1.10	**0.20**
	FLOPs	(↓)	0.30	0.19	0.30	**0.06**
ResNet18	**Size**	(↓)	**3.80**	8.90	19.20	4.40
	FLOPs	(↓)	0.33	0.18	0.33	**0.07**
MobileNetV2	**Size**	(↓)	**1.36**	4.74	6.78	1.76
	FLOPs	(↓)	1.50	0.92	1.50	**0.39**
VGG16	**Size**	(↓)	**23.80**	30.10	119.00	25.80
	FLOPs	(↓)	11.60	6.30	11.60	**2.30**

Table 4.5.3: Memory and efficiency results. Size: number of parameters (millions). FLOPs: number of floating point operations (Giga). Best/second best results are indicated by bold/underline.

4.5.4 Efficiency analysis

Table 4.5.3 summarizes memory and efficiency analysis for all models. As expected, the memory footprint (as measured by Size) of Deep is the highest since it consists of 5 independently trained models. The most memory-efficient model is MCDrop, however, the FLOPs required for uncertainty estimation are approximately $5\times$ higher than Early Exit due to sampling at inference time. Dealing with 5 models (Deep and Depth) or performing 5 samples (MCDrop) translates into higher inference time compared to Early Exit. In Figure 4.5.4, we show how this reflects into latency and energy consumption on the NVIDIA Jetson TX2 board. Of course, depending on the model the latency and energy required increases as the number of FLOPs increase. Nevertheless, Early Exit can make uncertainty aware inferences much faster, therefore saving from 132 mJ to 835 mJ per inference (compared to Deep), establishing a great robust technique for battery-based devices and real-time applications.

Overall, Early Exit presents the best trade-off between memory (max 16% increase), inference time (approx. $5\times$ less), and number of FLOPs (approx. $5\times$ less) since EEEs can produce all ensemble members' predictions in a single forward pass. Therefore, EEEs are better suited to a wider range of real-world applications in need of efficient uncertainty quantification.

4.5.5 Effect of Number of Exits

The number of early exits, and therefore ensemble size $|\mathcal{M}|$, is bounded above by the number of blocks B in each backbone model architecture (see Table 4.4.2). As such, each exit block

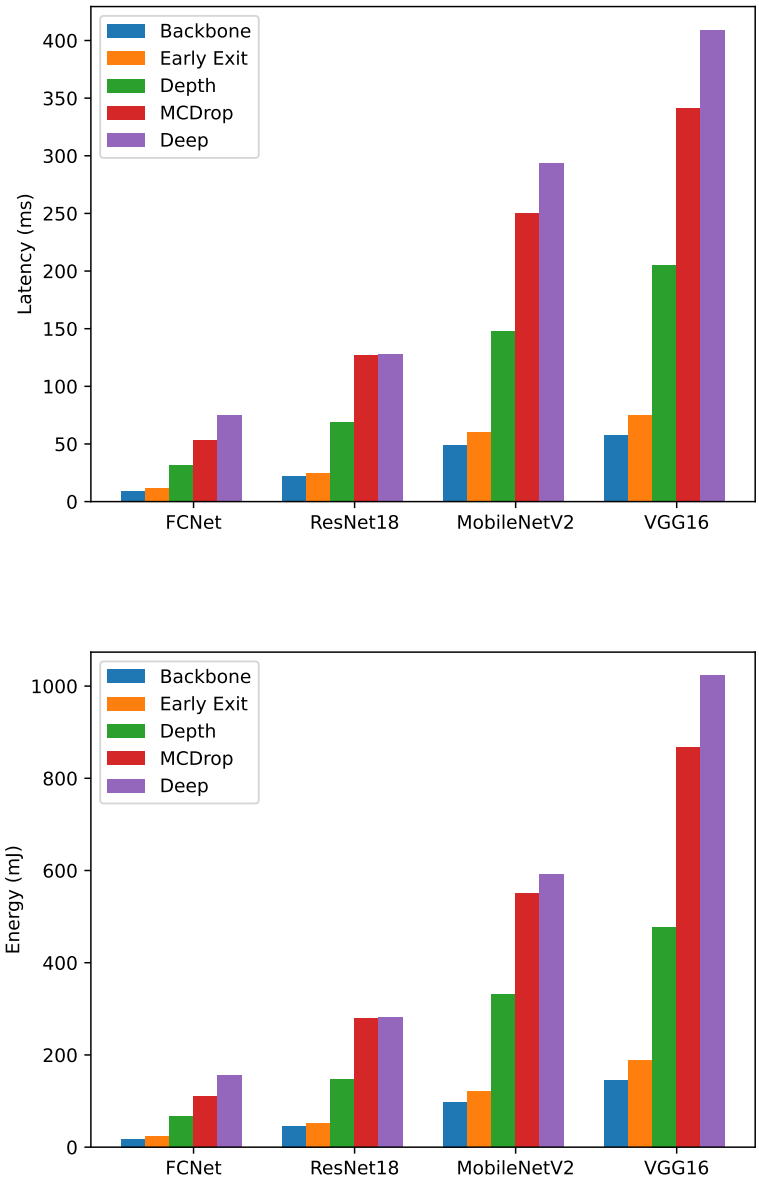


Figure 4.5.4: Latency and energy measured on the NVIDIA Jetson TX2 board. Latency measures inference time in ms. Energy is measured as power (W) x latency. Both values are reported for a single inference on a single sample. The value is an average of all samples in the training set.

is indexed $i \in \{1, \ldots, B-1\}$ which gives a combinatorial choice $\binom{B}{|\mathcal{M}|}$ of exit points and therefore ensemble arrangements. We investigate the effect of the number and placement of exits on predictive uncertainty using the ResNet18 backbone with $\gamma = 0.2$ (see Figure 4.5.5). Since

Strategy	$\lvert\mathcal{M}\rvert$	**F1 (↑)**	**ECE (↓)**	**Brier score (↓)**	**Exit index**	**Param. increase**
Pareto	$2+1$	0.853 ± 0.01	0.049 ± 0.01	0.201 ± 0.01	{4, 7}	9.8%
Last-k	4 +1	0.849 ± 0.00	0.063 ± 0.01	0.219 ± 0.01	{5 ,6, 7, 8}	23.3%
Semantic	$4+1$	$\mathbf{0.865 \pm 0.00}$	$\mathbf{0.027 \pm 0.01}$	$\mathbf{0.189 \pm 0.01}$	{2, 4, 6, 8}	15.2%
Computation	$6+1$	0.861 ± 0.01	0.057 ± 0.02	0.200 ± 0.01	{3, 5, 6, 7, 8}	25.6%
Residual	$8+1$	0.856 ± 0.01	0.032 ± 0.02	0.193 ± 0.02	{1, 2, 3, 4, 5, 6, 7, 8}	30.4%

Table 4.5.4: Effect of the number of exit blocks and exit strategy on the accuracy and uncertainty of the ensemble for ResNet18 with $\gamma = 0.2$. Reported results are the mean plus/minus standard deviation across three independent splits of the test dataset. Ensemble size $|\mathcal{M}|$ includes the backbone in addition to each exit block, represented as $+1$. For *Last-k* we set $k = 4$. Parameter increase is measured as a percentage increase over the number of parameters from the backbone. The exit index matches the indexes of the exit blocks in Figure 4.5.5.

for ResNet18, $B = 9$ and $|\mathcal{M}| = 5$ this yields 126 possible combinations of ensembles. To limit this search space, we introduce and test the following exit strategies:

- *Pareto* (Pareto, 1964): Exit at blocks closest to 20% and 80% of the overall number of FLOPs.
- *Computation*: Exit at blocks closest to 15%, 30%, 45%, 60%, 75% and 90% of the overall number of FLOPs (Kaya et al., 2019).
- *Residual*: Exit at residual blocks.
- *Last-k*: Exit at the last k blocks.
- *Semantic*: Exit at blocks semantically grouped by their number of features.

Table 4.5.4 summarizes the results for the listed exit strategies. A small ensemble of 3 members in the case of *Pareto* does not perform well in terms of accuracy but still shows well-calibrated uncertainty comparable to *Computation* with 5 members (ECE 0.049 vs. 0.057). Interestingly, a higher number of exits does not guarantee better uncertainty quantification (*Residual* BS 0.193 vs. *Semantic* BS 0.189), nor does choosing to exit deeper in the case of *Last-k* (BS 0.219). The strategy *Semantic* clearly gives the best results in terms of accuracy and uncertainty metrics reinforcing our findings in Section 4.5.1 which employs the same strategy. We attribute this result to the fact that when exiting at semantically group blocks, low-level and high-level features are integrated by summation at residual connections, therefore, representing the points of maximum diversity in the backbone.

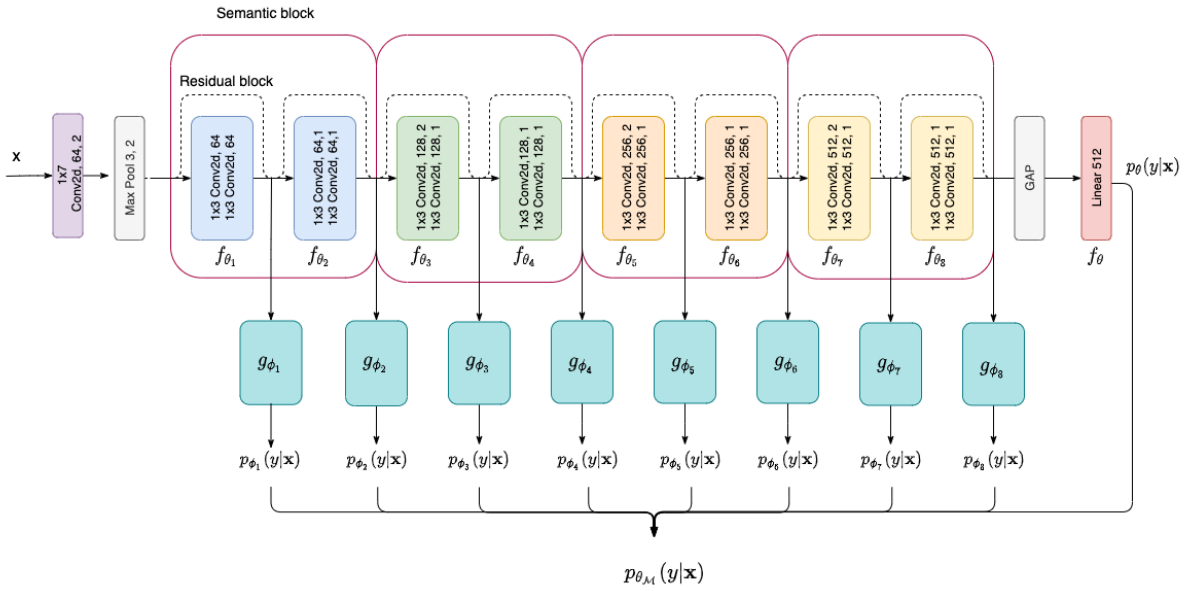


Figure 4.5.5: Representation of an early exit ensemble applied to a ResNet18 backbone decomposed into $B = 8$ blocks. The *Residual* strategy is used to create an ensemble of size $|\mathcal{M}| = 8 + 1 = 9$. For the *Semantic* strategy (best performing) only exits after each semantic block are included in the ensemble $|\mathcal{M}| = 4 + 1 = 5$. GAP represents Global Average Pooling. Block indexes match the exit indexes i in Table 4.5.4.

4.5.6 Effect of Learning Capacity Factor

In Figure 4.5.6, we show the effect of tuning the learning capacity factor γ for Early Exit in terms of NLL on each of the validation datasets. For FCNet, $\gamma = 0.0$ yields the best results for Early Exit in terms of lowest NLL and smallest percentage increase in parameters. On the other hand, for the deeper backbone architectures ResNet18 and MobileNetV2 on the larger datasets, higher learning capacity factors ($\gamma = 0.2$, $\gamma = 0.7$, respectively) yield the best results in terms of the lowest NLL. Intuitively, for these harder classification problems, the earlier exit blocks require more features in order to make accurate predictions that improve the overall ensemble performance. For ResNet18 and MobileNetV2 this translates to a small increase of $1.15\times$ and $1.38\times$ respectively in the number of overall parameters. In contrast, Deep with $|\mathcal{M}| = 5$ always increases parameters by $5\times$ regardless of the backbone and dataset. Additionally, unlike MCDrop, Early Exit can provide uncertainty quantification in one single forward pass.

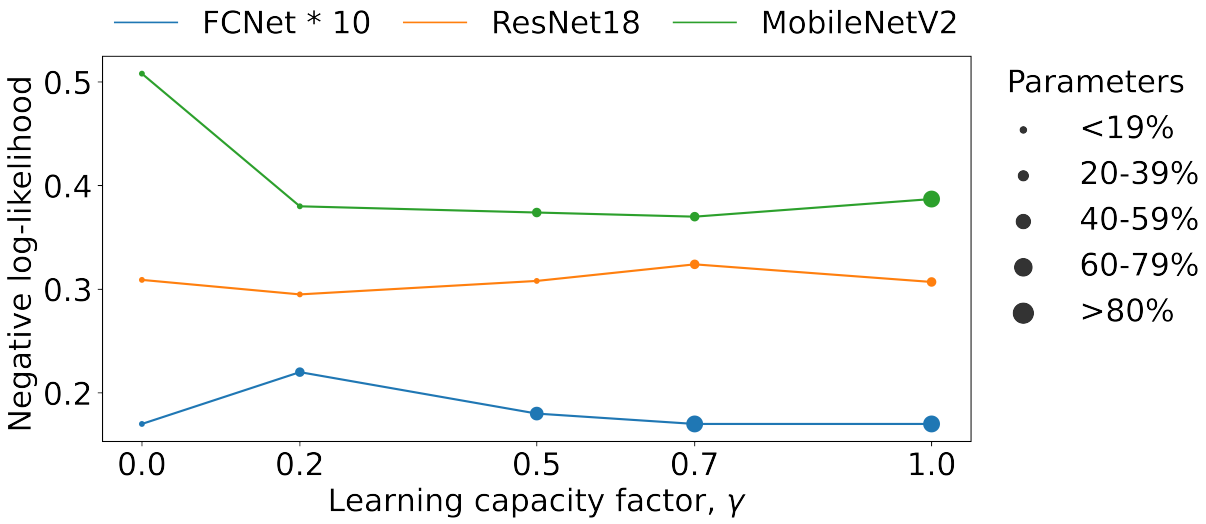


Figure 4.5.6: Effect of learning capacity factor γ on NLL as well the percentage increase in parameters over the backbone.

4.5.7 Diversity analysis

In order to further increase ensemble diversity, we experiment with a new loss function which is a composite classification and diversity loss function:

$$\mathcal{L} = \mathcal{L}_C + \beta \mathcal{L}_D \tag{4.7}$$

where $\mathcal{L}_C$ is the classification loss in Section 4.3.1, $\mathcal{L}_D$ is the diversity loss, and $\beta \geq 0$ represents the relative weight of the diversity loss.

Since the correlation between exit block predictions must be minimal, we propose a diversity loss such as:

$$\mathcal{L}_D = \frac{1}{M} \sum_{i=1}^{B-1} \sum_{j \neq i} L_{CE}(g_{\phi_i}(y|\mathbf{x}), g_{\phi_j}(y|\mathbf{x})) \tag{4.8}$$

where $M = |\mathcal{M}|(|\mathcal{M}| - 1)$. Since $\mathcal{L}_D$ minimizes the cross-entropy between all exit pairs it approximately maximizes pairwise mutual information (Boudiaf et al., 2020).

We further experiment with running all early exit models setting $\gamma = 0$ and tuning only the weight β of the diversity loss (Equation 4.7) over $\{0.3, 0.5, 1.0, 2.0\}$ using ResNet18. Intuitively, a higher value of β should allocate more importance to learning a diverse early exit ensemble. Results show that as β increases from 0.3 to 2.0, F1 score increases by 23.1% (from 0.85 to 0.86) and ECE decreases by 14.6% (from 0.047 to 0.041). Although performance improves,

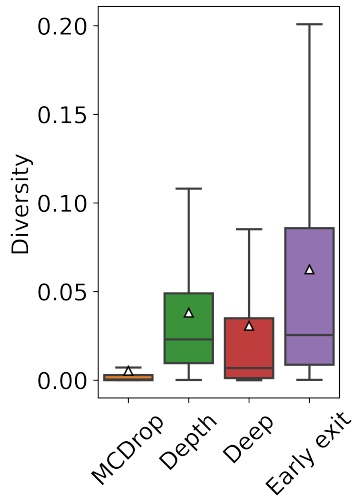


Figure 4.5.7: Visualizing diversity (↑) as measured by KL divergence for VGG16.

accuracy and calibration are still worse than the best result for early exit ensembles in Table 4.5.2. Therefore, given the diversity already embedded in early exit ensembles, introducing a diversity loss did not result in better-calibrated uncertainty quantification.

In Figure 4.5.7 we visualize diversity computed as the KL divergence between each ensemble member prediction and the average ensemble prediction (Fort et al., 2019). Given the comparable median diversity of Early Exit and Depth (approx. 0.024), it is clear that varying network structure leads to higher predictive diversity. Overall, we conclude that Early Exit is the best performing technique as it has a greater range of disagreement at a much lower computational cost since it can provide multiple predictions in a single forward pass. The heat maps of predictive confidence in Figure 4.5.8 (4 samples from EEG-A) further highlight the importance of ensemble diversity. Averaging the predicted probabilities of each member of Early Exit results in a correct prediction (highlighted in green) even though some individual members are wrong. This finding is in line with previous work suggesting that a good performing ensemble should be both accurate and diverse (Perrone and Cooper, 1992).

4.5.8 Effect of weighting exits.

We experiment with running early exit ensembles using a ResNet18 backbone setting $\gamma = 0$ and tuning only the importance weights α from the classification loss (Equation 4.7). Without the extra learning capacity, the early exits with $\alpha_i = 1$ tend to have a high misclassification rate. However, penalizing the earlier exits by setting $\alpha_i = 1/(B - 1 - i)$ for example, does not

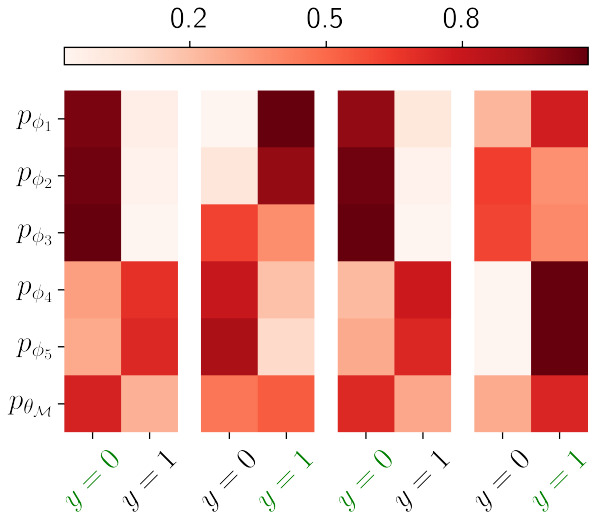


Figure 4.5.8: Visualizing predictive confidence (↑) (Early Exit, 4 samples) for VGG16. Ground truth in green.

improve classification performance and worsens uncertainty metrics overall. We conclude the performance of early exit ensembles requires each individual member to be accurate in line with the findings of Perrone and Cooper (1992).

4.5.9 Comparison to layerwise distribution approximation

In Chapter 3, we propose a layerwise distribution approximation that can enable uncertainty at inference time on deep learning models trained with dropout. Therefore, we want to compare our previous work with early exit ensembles in terms of in-distribution calibration, out-of-distribution detection, and energy efficiency. We run our experiments on the ECG dataset and FCNet architecture (replacing batch normalization layers with dropout layers) since this architecture is a very simple one. One of the limitations of our previous technique is that the behavior of other layers, such as batch normalization or bottleneck layers and (inverted) residual blocks need further investigation. However, this could lead to more complex implementations, which would hinder the adoption of our framework. For this reason, we believe that its most efficient use is in simple convolutional neural networks. Nevertheless, we compare the two techniques to give a better understanding of their behavior. Table 4.5.5 provides an overview of this comparison showing that both techniques can show good and comparable (early exits are slightly better) accuracy and calibration for in-distribution data; the same is valid in terms of efficiency. However, the layerwise distribution approximation proposed in Chapter 3 struggles at detect-

ing out-of-distribution samples since its diversity is much lower, as measured by a very low KL divergence. We believe that while the framework proposed in the previous chapter offers good tradeoffs in terms of providing well-calibrated uncertainty for already trained deep learning models, early exit ensembles are a more robust approach that can still guarantee great computation gains. In addition, early exit ensembles provide more flexibility in terms of types of layers and architectures that can be used, making it a more flexible option at the cost of training exits in addition to the backbone.

Approach	**F1**	**ECE**	**OOD PE**	**Diversity**	**Latency**	**Energy**
Layerwise	0.990 ± 0.005	0.009 ± 0.001	0.26	0.02 ± 0.01	11.99 ± 0.10	24.70 ± 0.23
Early Exit	0.992 ± 0.004	0.008 ± 0.001	0.43	0.05 ± 0.02	12.05 ± 0.09	24.98 ± 0.20

Table 4.5.5: Comparison between the two uncertainty quantification approaches proposed in this thesis. OOD PE measures the median predictive entropy under out-of-distribution samples. As measured by KL divergence, diversity is computed between each ensemble member prediction and the average ensemble prediction. Latency (ms) represents an average time of inference for one single sample, while energy (mJ) is measured as power x time on NVIDIA Jetson TX2 board. Results are mean and std of 3 different data splits, instead, latency and energy mean and std over all samples in the test set.

4.6 Discussion and conclusions

In this chapter, we introduce early exit ensembles, a novel framework for enabling uncertainty quantification in any feed-forward neural network. At the core of our methodology is a new interpretation of early exit neural networks as an implicit ensemble of weight-sharing sub-networks from which predictive uncertainty can be estimated. We evaluate early exit ensembles using four state-of-the-art deep learning architectures applied to different medical imaging datasets. Our results show that, compared to competitive deep learning ensemble baselines, early exit ensembles can provide better-calibrated uncertainty for both in-and out-of-distribution data. Moreover, our approach enables training all ensemble members jointly in a single model, as well as providing uncertainty estimations from a single forward pass of input data. Both the ease of implementation and the computational efficiency of training and inference mean that early exit can be applied to a wide range of real-world applications.

Early exit ensembles can be used in a variety of scenarios, however, they have a crucial impact on using uncertainty in patient-level decision-making. Here, we want to showcase a scenario where quantifying uncertainty is critical in health applications. As we explained previously, dur-

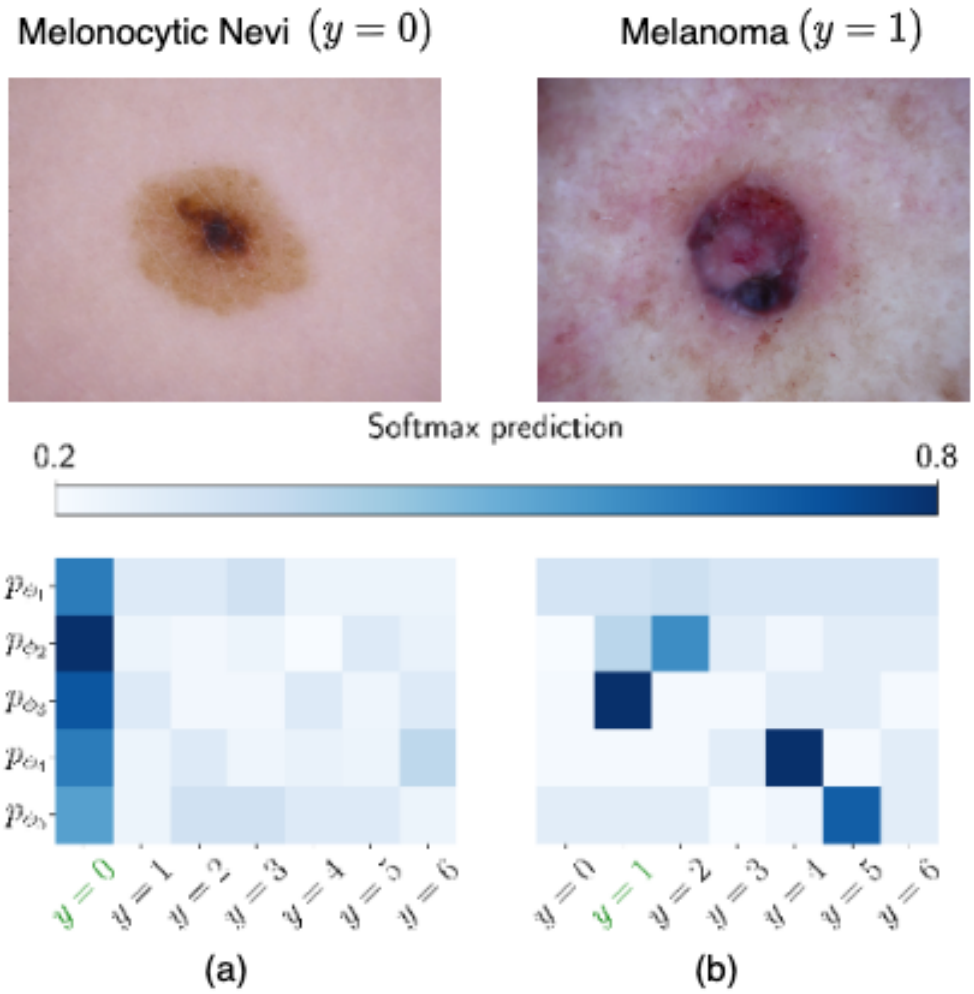


Figure 4.6.1: Early exit ensemble test predictions. In (a), all exits agree on the prediction. In (b) all exits highly disagree predicting different classes indicating the sample should be sent to a clinician for further checks. Correct class in green.

ing inference, each ensemble technique outputs $|\mathcal{M}| = 5$ softmax predictions over classes. The extent of agreement/disagreement between these predictions can be used to quantify uncertainty. Figure 4.6.1 visualizes two outputs of early exit ensembles using MobileNetV2 on the ISIC2018 (skin lesion) test dataset. It is clear from (a) that all exits are individually confident and correctly predict the same class $y = 0$ (melonocytic nevi). On the other hand, in (b) all the exits highly disagree predicting different classes to the true class $y = 1$ (melanoma) suggesting further investigation from an expert clinician. Both conditions, melonocytic nevi, and melanoma are very serious however melanoma indicates that the condition is at an advanced stage. Without uncertainty estimation, the automatic system would have missed this critical condition and had a

drastic impact on the patient's life.

In this chapter, we have shown the potential of early exit ensembles in aiding a human-in-the-loop system, detecting out-of-distribution samples, or informing clinicians on how the model's performance may degrade in different deployment settings. However, we know that in addition to naturally occurring noise or genuine mistakes, adversarial attacks are a constant threat to machine learning models compromising their security and privacy. In the next chapter, we use early exit ensembles as a building block to provide mitigation against adversarial attacks on biosignal tasks, further extending the robustness capability of our framework.

Chapter 5

Towards adversarial mitigation with early exit ensembles

5.1 Introduction

Detecting anomalous samples caused by statistical shifts in data distribution is a fundamental requirement for deploying a good classifier in the real world. In Chapter 4, we introduced early exit ensembles, a new paradigm for uncertainty quantification, and showed their performance in providing robustness through out-of-distribution detection. However, when deployed in the wild, deep learning models do not only have to deal with naturally occurring noise or distribution shifts: attackers can intentionally design synthetically transformed inputs to exploit the model and degrade the model's performance. Adversarial attacks introduce a new major failure mode of modern deep learning techniques, which needs to be addressed. In this chapter, we delve into the difficult task of adversarial mitigation. We use the early exit ensemble framework to build an attack-agnostic mitigation technique for classification tasks.

Recent work has shown that deep learning models are susceptible to a multiplicity of attacks (Goodfellow et al., 2016; Madry et al., 2017; Finlayson et al., 2019). One of the main reasons why DNNs are easily manipulated relies on the fact that current implementations mainly focus on improving accuracy only, overlooking robustness in general and especially towards adversarial attacks. Adversarial perturbations can very closely mimic physiologically plausible signals and become virtually indistinguishable from human perception. Nevertheless, they can drastically decrease the deep learning model accuracy and introduce security and privacy vulner-

abilities in safety-critical scenarios, such as health applications.

The diversity of attacks grows continuously, and traditional approaches that require retraining and re-deployment of the network are a significant burden and often not feasible. Currently, the state-of-the-art mitigation approaches rely on adversarial training (Madry et al., 2017) to minimize the risk of misclassifying the perturbed signal. However, its side-effects are not negligible since classifiers trained with adversarial examples learn fundamentally different representations compared to standard classifiers reducing accuracy (Tsipras et al., 2018); they also can cause a disparity in accuracy between classes for both clean and adversarial samples Xu et al. (2020). In addition, they are (i) resource-consuming, (ii) primarily mitigate only against the attacks they have been exposed to during training, and (iii) cannot be applied to already deployed or trained networks, suggesting limited generalization and applicability in the real world.

To solve the abovementioned issues, we propose a mitigation technique that relies on our findings in Chapter 4 interpreting early exit neural networks as an implicit ensemble of models. Early exit neural networks are a class of conditional computation models that exit once a criterion (e.g., sufficient accuracy) is satisfied to save on computation (Teerapittayanon et al., 2016). Under a different interpretation, the early exit paradigm can be used to mitigate the problem of model overthinking (Kaya et al., 2019) or, as previously explored in Chapter 4, used as an ensemble for uncertainty quantification. Our intuition on exploiting these networks for adversarial mitigation arises from the success of ensemble defense in improving both accuracy and robustness (Tramèr et al., 2017), however, without the computational burden of training and maintaining multiple single models.

In this chapter, we make the following contributions:

- We introduce a novel adversarial mitigation technique that provides run-time robustness to adversarial attacks it has never been exposed to before, presenting a highly desired attack agnostic technique. Furthermore, our approach is generalizable to various adversarial attacks (universal and white box attacks) while maintaining a low computational burden.
- We evaluate our method on state-of-the-art deep learning architectures applied to biosignal classification tasks. Our results show that we can increase the accuracy up to 60 percentage points (pp) compared to undefended deep learning models and provide well-calibrated adversarial mitigation comparable to adversarial training.
- Our experiments on four major adversarial attacks show the potential of early exit ensembles in providing adversarial robustness and how we can exploit its orthogonality to

adversarial training and Monte Carlo dropout to build robust models we can trust.

5.2 Related Work

In this section, we provide a simplified taxonomy of common adversarial attacks applied to deep learning models. Additionally, we explore state-of-the-art approaches for adversarial detection and mitigation. The literature in this space of deep learning has been expanding significantly in the last years, therefore here, we try to mainly include prior art that is closely related to our work.

5.2.1 A taxonomy of adversarial attacks

Current deep learning models are trained to assume that all samples are representative of the task at hand and trustworthy. However, malicious entities can impact these decision-making algorithms by either targeting the training data or forcing the model to alter its output at inference. These types of attacks, known as poisoning and evasion attacks, respectively, allow adversaries to significantly decrease overall performance, causing targeted misclassification or malicious behavior, and insert backdoors and neural Trojans (Tabassi et al., 2019; Chakraborty et al., 2018).

A simplified taxonomy of adversarial attacks is provided in Figure 5.2.1 where we identify two main attack scenarios:

- **Evasion (inference time) attacks.** This class of attacks is the most common in the adversarial machine learning ecosystem. Here, the adversary evades the system by perturbing input samples during inference. This setting does not assume any perturbation over the training data as the model is already trained and its weights are frozen.
- **Poisoning (training time) attacks.** In this attack scenario, the adversary poisons the training data by injecting carefully designed samples to compromise the whole learning process.

The objective of the adversary is to compromise the integrity of the deep learning model by causing:

- **Confidence reduction.** The adversary aims at reducing the confidence of the model which might lead to misclassification, higher entropy, and, ultimately, an increase in the predictive uncertainty.

- **Misclassification.** Given an input sample, the adversary aims at assigning it to any other class but the original one the model would have inferred.
- **Targeted misclassification.** Here, the adversary targets a specific class for misclassification. Instead of a naive misclassification, there is a specific target class to channel all samples in.
- **Source/Target misclassification.** In this attack, the goal is to assign a specific model to a specific class, creating a relationship between the input at the output.

The amount of information about the model available to an adversary defines the type of attacks that can be performed. An adversary who has access to the model architecture constitutes a stronger attack compared to a weaker adversary having access only to the signal fed to the model during inference time. Training time attacks attempt to influence or corrupt the model directly by altering the dataset used for training. Nevertheless, the amount of data and parts of the model the adversary has access to are very important in establishing the strength of the adversarial attack. There exist three broad attack strategies for altering the model based on the adversarial access to the model and data and, consequently what the attacker can exploit them for:

- **Data Injection.** This scenario happens when the adversary does not have access to the training data or the deep learning model, nevertheless it can inject new data. This corruption can alter the model outcome at a cheap computation cost.
- **Data Modification.** Here, the adversary does not have access to the model, but it can fully access the training data. This allows for augmenting or perturbing the current training set to alter the model predictions.
- **Logic Corruption.** The attacker has access to the whole model architecture and parameters. In this scenario, the adversary can completely alter the logic of the model by modifying the model parameters, such as updating to new ones for a more targeted attack or presenting randomly initialized ones for confidence reduction.

Adversarial attacks operating at inference time do not interfere with the training of the targeted model, but they aim to fool the model into producing incorrect predictions. Their success is determined by the amount of information available to the adversary about the model. Depending on this information, inference time attacks can be broadly classified into white-box or black-box attacks.

White-box attacks are performed by adversaries that have total knowledge about the model,

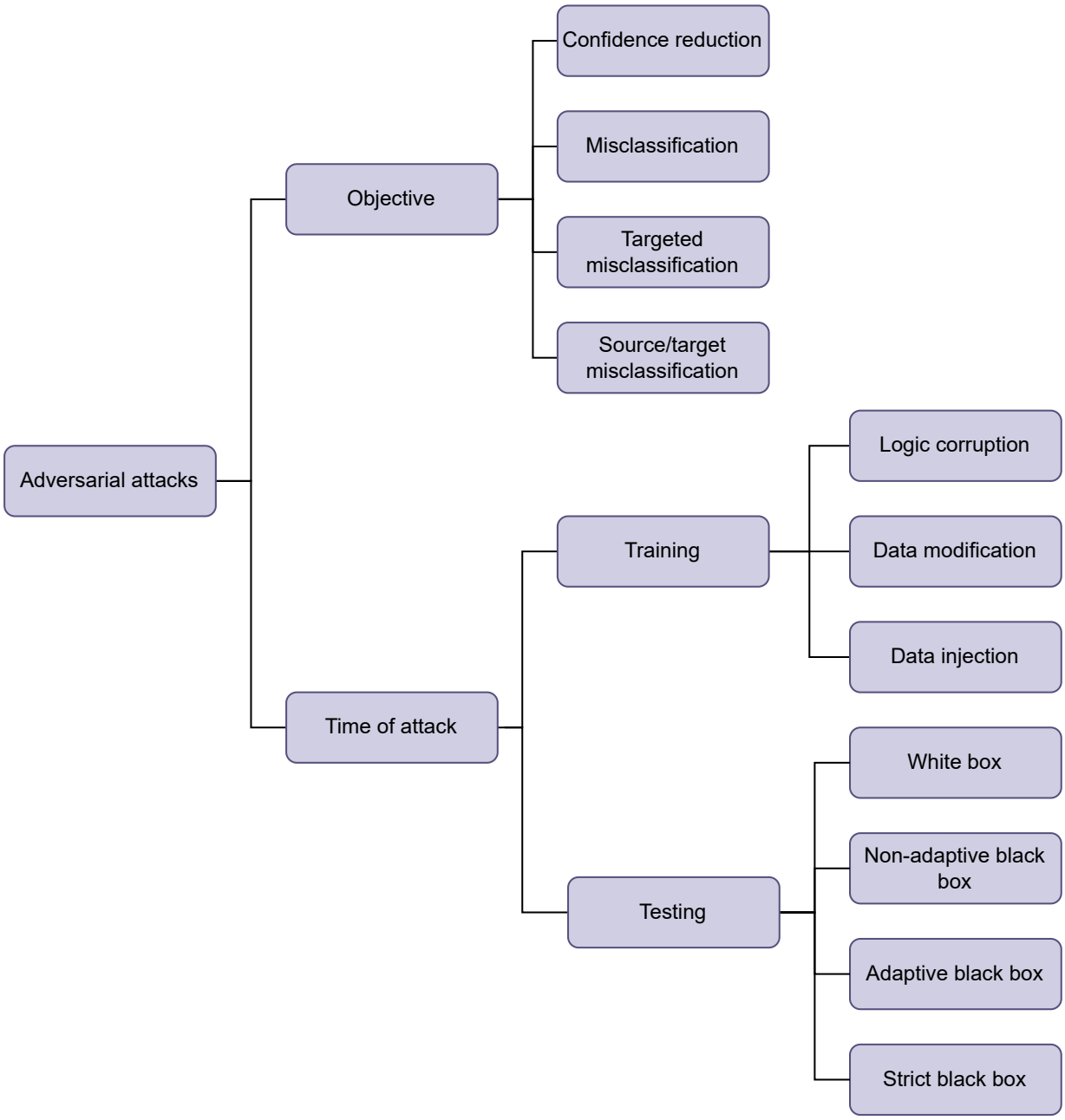


Figure 5.2.1: A simplified taxonomy of the adversarial attacks. Poisoning (training time) and evasion (inference time) attacks allow adversaries to significantly decrease overall performance, causing targeted misclassification or malicious behavior, and insert backdoors and neural Trojans.

including the model architecture, parameters, and training data distribution. This depth of information allows the attacker to extensively explore the model vulnerabilities and to produce

perturbed samples which cause targeted/non-targeted misclassification.

Black-box attacks represent the scenario where the attacker only has access to outputs of the target model without knowing its architecture and weights. Not having enough available information or past inputs to analyze the vulnerability of the model, black-box attacks exploit the model by providing a series of carefully crafted inputs and observing the outputs. Black-box attacks can be primarily classified into:

- **Non-adaptive black-box attacks.** For this attack, the adversary, has only access to the distribution of the training data of the targeted model. Given this information, it trains a proxy model with samples from the same training data distribution, aiming to approximate the target model. This is done by using white-box attack strategies. Ultimately the perturbed samples produced by this procedure are fed into the target model inference pipeline to force misclassification.
- **Adaptive black-box attacks.** These attacks can access the model by making adaptive queries and collecting the outputs stemming from the model. The attacker then uses the formed input-output dataset to train a proxy model. Once the model is trained, it produces adversarial samples by following the white-box attack technique forcing the target model to behave inaccurately.
- **Strict black-box attacks.** Differently from the adaptive black-box attacks, these attacks cannot alter the input to observe the output of the model. The only information is the current input to the model and its prediction. For this attack to work well, the adversary needs access to a larger set (compared to both adaptive and non-adaptive) of observed input-output pairs to build a representative training set for the proxy model.

In this chapter, we focus on inference time white-box attacks for two main reasons: (1) we are interested in applications that run on resource-constrained edge devices, where currently only inference is performed; (2) white-box attacks are the strongest attacks that can be performed during inference, therefore, mitigating them highlight the level of robustness our approach can provide.

5.2.2 Adversarial detection and defense

While the number of adversarial attacks grows exponentially, the solutions for their detection and mitigation are developed at a slower pace. This is mainly because most techniques are attack specific and do not transfer well between attacks. Additionally, a majority of proposed prior art

focuses on detection only, without proceeding to a defense solution. Adversarial detection is half of the work as the ultimate goal for achieving trustworthy and robust deep learning models is to provide adversarial mitigation. In this section, we explore state-of-the-art techniques in adversarial detection and defense, which we categorize into four main classes: adversarial training, input filtering, profiling techniques, and stochastic approaches.

Adversarial training (Goodfellow et al., 2014; Madry et al., 2017; Zhang et al., 2019) minimizes the risk that a perturbed sample can be misclassified by augmenting the training set with adversarially perturbed samples in addition to the clean original data. This approach assumes prior knowledge of which adversarial attacks are going to be encountered during deployment. Therefore, every time a new attack emerges, the model must be retrained with new adversarial samples in addition to the previous ones. Although adversarial training is a very resource-consuming approach, it is state-of-the-art in adversarial mitigation. However, its side effects are not negligible. Classifiers trained with adversarial examples learn fundamentally different representations compared to standard classifiers reducing accuracy (Tsipras et al., 2018). They can cause a disparity in accuracy for both clean and adversarial samples between different classes (Xu et al., 2020) by favoring certain classes even in perfectly balanced datasets. In addition, they are extremely resource-consuming as they need to be always retrained for new emerging attacks to guarantee robustness to them during inference, suggesting a very limited generalization and applicability in the real world.

Input filtering techniques are an interesting approach to adversarial detection because, differently from other approaches which operate at the model level, they work on the input features. Their main goal is to reduce the degrees of freedom available to an adversary by "squeezing" out unnecessary input features, e.g., by reducing the number of bits for color depth in images. This class of approaches relies on the fact that adversarial attacks usually operate on these unnecessary features in order to be less detectable. Feature squeezing (Xu et al., 2017) is the main work in this space and consists of running the deep learning model with both the original and squeezed/filtered input. This allows for detecting the adversarial if the difference in the prediction exceeds a certain threshold. Although its computational overhead is minimal, it provides detection only and can be easily bypassed (Sharma and Chen, 2018). Moreover, these techniques are mainly engineered for images; therefore, they are not directly applicable to other types of data.

Profiling techniques exist to monitor the path followed by clean samples during training or inference and consequently detect adversarial attacks by profiling their path from input to inference. Ptolemy (Gan et al., 2020), for instance, detects adversarial samples at runtime based

on the observation that malign samples tend to activate distinctive paths from those of benign inputs. These techniques are attack agnostic which is a great advantage however they rely on extensive profiling, which is a time-consuming task. Additionally, their main drawback is having a backdoor in the model since the attack can learn which specific paths get activated by profiling too. In contrast, our approach does not depend on any specific path; therefore, profiling its activation path does not give the adversary extra information. Another approach in this space is the Taboo Trap (Shumailov et al., 2018), which does not rely only on profiling but trains the model by applying restrictions on activations. During inference, any sample violating these restrictions is considered adversarial. This solution, however, is not generic as it would fail in the presence of covariate shifts. In addition, the attack can discover the limit of activations used in the restriction too.

Stochastic approaches assess adversarial attacks by looking at the likelihood of perturbed examples and the distance between perturbed and clean examples for each classifier (Alemi et al., 2016; Papernot et al., 2016; Li and Gal, 2017). Bradshaw et al. (2017) propose a Gaussian Process (GP) hybrid deep NN to help mitigate adversarial attacks; however, a closed-form solution for GPs has a $\mathcal{O}(n^3)$ computational complexity. In addition, GPs are very sensitive to quantization, and thus implementation on embedded hardware is challenging. A promising direction, Monte Carlo Dropout (MCDrop) (Gal and Ghahramani, 2015a), casts dropout training as approximate inference in Bayesian CNNs. Feinman et al. (2017) investigate model confidence on adversarial samples through MCDrop by looking at uncertainty estimates. They operate a two-feature approach, however, where in addition to MCDrop, they need to observe the density estimates during training to detect the points that lie far from the data manifold, which might lead to issues adapting to data shifts. While these techniques study the uncertainty for adversarial detection, we envision using the early exit ensemble framework to prevent the attack from changing the prediction by making it more difficult for the attacker to overcome all the ensemble members.

5.3 Methods

In Chapter 4, we introduced early exit ensembles, a collection of weight-sharing sub-networks created by adding exit branches to any backbone neural network architecture. During inference, they provide an ensemble of predictions in a single forward pass which allows for efficient and robust adversarial mitigation by aggregating the predictions from individual exits. Our intuition on using early exit ensembles as an adversarial defense relies on the fact that traditional adversarial attacks are built to fool the backbone of the model and generally discard or are unaware of the in-

termediate exits. Additionally, previous work has shown that for a single network, promoting the diversity among learned features of different classes can improve adversarial robustness (Pang et al., 2018). This is valid for ensembles of models too (Pang et al., 2019), and since early exit ensembles naturally incorporate ensemble diversity given by the structural diversity between its members, we propose our framework as an efficient way to provide adversarial mitigation.

Figure 5.3.1 shows a representation of an early exit ensemble. As depicted, any neural network $f_\theta(\cdot)$ can be decomposed into B blocks such that $f_\theta(\mathbf{x}) = (f^{(B)} \circ f^{(B-1)} \circ \cdots \circ f^{(1)})(\mathbf{x})$ where $(f^{(i)} \circ f^{(j)})(\mathbf{x}) = f_{\theta_i}(f_{\theta_j}(\mathbf{x}))$ denotes function composition for $i \neq j$ and $\theta = \cup_{i=1}^{B} \theta_i$, $\mathbf{x} \in \mathbb{R}^D$ denotes a D-dimensional input. Let $\mathbf{h}^{(i)} \in \mathbb{R}^{K_i \times D_i}$ denote the intermediary output of the i-th block having K_i features of dimension $D_i \leq D$ such that $\mathbf{h}^{(i)} = f_{\theta_i}(\mathbf{h}^{(i-1)})$ for $1 \leq i \leq B-1$, and $\mathbf{h}^{(0)} = \mathbf{x}$.

An early exit block is defined as a NN $g_{\phi_i}(\cdot)$ which takes as input the intermediary output $\mathbf{h}^{(i)}$ from the i-th block of $f_\theta(\cdot)$, henceforth referred to as the backbone. Each exit block learns a predictive distribution $p_{\phi_i}(y|\mathbf{x}) = \xi(g_{\phi_i}(\mathbf{h}^{(i)}))$ where $\xi(\cdot)$ is the softmax transform and $y \in \{1, \ldots, C\}$ a corresponding discrete target taking one of C classes. As such, any NN is able to output a set $\mathcal{M} = \{p_{\phi_1}(y|\mathbf{x}), \ldots, p_{\phi_{B-1}}(y|\mathbf{x}), p_\theta(y|\mathbf{x})\}$ which represents an early exit ensemble (see Figure 5.3.1). The ensemble $\mathcal{M}$ contains up to $B-1$ distributions from early exit blocks, in addition to the standard output from its final block. As such, ensemble size $|\mathcal{M}| = B$.

During training, a weighted sum of each exit's individual predictive loss is optimized. This procedure allows the training of the whole ensemble jointly. More formally:

$$\mathcal{L} = L_{CE}(y, f_\theta(y|\mathbf{x})) + \sum_{i=1}^{B-1} \alpha_i L_{CE}(y, g_{\phi_i}(y|\mathbf{x})) \tag{5.1}$$

where $L_{CE}(\cdot, \cdot)$ is the cross-entropy loss function and $\alpha_i \in [0, 1]$ is a weight hyperparameter corresponding to the relative importance of each exit.

During inference, a single forward pass of a NN with early exits produces an ensemble $\mathcal{M}$ of predictions. The overall prediction from $\mathcal{M}$ can be computed as the mean of a categorical distribution obtained from averaging the predictions from the individual exits:

$$p_{\theta_\mathcal{M}}(y|\mathbf{x}) \approx \frac{1}{|\mathcal{M}|}\big(p_\theta(y|\mathbf{x}) + \sum_{i=1}^{B-1} p_{\phi_i}(y|\mathbf{x})\big). \tag{5.2}$$

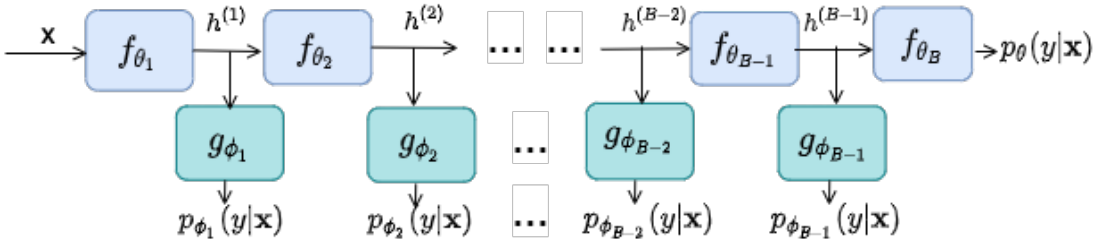


Figure 5.3.1: Representation of an early exit ensemble. A collection of weight-sharing sub-networks created by adding exit branches to any backbone neural network architecture. During inference, they provide an ensemble of predictions in a single forward pass by aggregating the predictions from individual exits.

Further information and analysis on early exit ensembles are described in Chapter 4.

5.4 Adversarial attacks

In this chapter, we consider 4 types of adversarial attacks: three white-box attacks (PGD, PGD-AVG, and PGD-MAX) aimed at early exit neural networks (Hu et al., 2020) and one universal adversarial perturbation attack (Moosavi-Dezfooli et et al., 2017). We group them into two attack vectors based on the assumption that the adversary is aware of the intermediate exits or unaware of them, respectively.

Formally, given a clean signal $\mathbf{x}$, an adversarial attack introduces small perturbations ∇ such that the prediction for $\mathbf{x}$ and $\mathbf{x}^{adv}$ differs.

5.4.1 Attack Vector 1

Assumption

*The adversary is **unaware** of the intermediate exits.*

This might be because (i) they do not have access to the full model; (ii) the attack cannot handle multiple exits (which is often the case); (iii) considering the intermediate exits is too expensive.

Projected Gradient Descent (PGD)

PGD (Kurakin et al., 2016) in an iterative universal first-order attack which aims to find the adversarial perturbations by moving in the opposite direction to the gradient of the loss function $L(\mathbf{x}, y)$ w.r.t. the signal (∇):

$$\mathbf{x}_0^{adv} = \hat{\mathbf{x}}, \mathbf{x}_{n+1}^{adv} = clip_{\mathbf{x}}^{\epsilon}\{\mathbf{x}_n^{adv} + \beta sign\big(\nabla_{\mathbf{x}} L\big(\mathbf{x}_n^{adv}, y\big)\big)\}, \tag{5.3}$$

where ϵ is the step size that restricts the l_∞ of the perturbation, and

$$\hat{\mathbf{x}} = \mathbf{x} + \epsilon_1 * sign\big(\mathcal{N}\big(\mathbf{0}^d, \mathbf{I}^d\big)\big) \tag{5.4}$$

(with parameters ϵ_1, ϵ such as $\epsilon_1 < \epsilon$) is an additional prepended random step that avoids going towards a false direction of ascent. This represents a single attack defined to fool $f_\theta(\cdot)$ only, without considering the intermediate exits g_{ϕ_i}, expressed as:

$$\mathbf{x}^{adv} = \underset{\mathbf{x} \in |\mathbf{X}' - \mathbf{x}|_\infty \leq \epsilon}{arg\,max}\ |L(f_\theta(\mathbf{x}', y))|. \tag{5.5}$$

Universal Adversarial Perturbation (UAP)

UAP (Moosavi-Dezfooli et et al., 2017) seeks a perturbation ∇ to fool $f_\theta(\cdot)$ on most data samples:

$$f_\theta(\mathbf{x} + \nabla) \neq f_\theta(\mathbf{x}) \tag{5.6}$$

where perturbations are constrained to $l_\infty \leq \epsilon$ to be visually imperceptible to humans. UAP aims at crafting a single perturbation for all data samples. The version used in this work is based on DeepFool (Moosavi-Dezfooli et al., 2016) which crafts perturbations iteratively by updating the gradient with respect to the model's decision boundaries. At each iteration, the attack chooses the perturbation direction of the minimum magnitude that is orthogonal to the decision hyperplane with the goal of finding pseudo-minimal perturbations to fool the model.

5.4.2 Attack Vector 2

Assumption

*The adversary is **aware** of the intermediate exits.*

This happens when (i) the adversary has access to the full model and is aware that the intermediate exits are used, and (ii) considering the intermediate exits is seen as worth the expensive additional computation.

PGD Average Attack (PGD-AVG)

PGD-AVG (Hu et al., 2020) considers all intermediate exits g_{ϕ_i} by maximizing the average of all losses such that the adversarial sample $\mathbf{x}^{adv}$ can attack any exit in the ensemble:

$$\mathbf{x}^{adv}_{avg} = \underset{\mathbf{x} \in |\mathbf{x}' - \mathbf{x}|_\infty \leq \epsilon}{arg\,max} \Big| \frac{1}{B} \big(L(f_\theta(\mathbf{x}', y)) + \sum_{i=1}^{B-1} L(g_{\phi_i}(\mathbf{x}', y)) \big) \Big| \tag{5.7}$$

PGD Max-Average Attack (PGD-MAX)

PGD-MAX (Hu et al., 2020) emphasizes on the individual exits, not just at maximizing an all-averaged loss like in eq. 5.7. It creates M single attacks $\mathbf{x}^{adv}$ such as eq. 5.5 and denoting the collection of single attacks as Ω:

$$\begin{aligned} \mathbf{x}^{adv}_{max} &\leftarrow \mathbf{x}^{adv}_{j^*}, where\, \mathbf{x}^{adv}_{j^*} \in \Omega \\ j^* = \underset{j}{arg\,max} \Big| \frac{1}{B} \big(L(f_\theta(\mathbf{x^{adv}}, y)) + \sum_{i=1}^{B-1} L(g_{\phi_i}(\mathbf{x^{adv}_j}, y)) \big) \Big| \end{aligned} \tag{5.8}$$

This attack is the strongest of the three since $\mathbf{x}^{adv}_{max}$ not only maximally fools the individual exit but is also transferable between exits.

5.5 Experiments

We design experiments to verify the performance of early exit ensembles in providing adversarial mitigation. We, firstly, test what level of defense it can offer when presented with attacks that are not aware of the intermediate exits and perform traditional attacks on the backbone (see Section 5.6.1). We then further explore the scenario where the adversary is aware of the intermediate exits and includes them when generating perturbed malign samples (see Section 5.6.2). Finally,

in Section 5.6.2, we also explore how its orthogonality to other techniques such as adversarial training and Monte Carlo dropout can be exploited for further mitigation.

5.5.1 Datasets & architectures

The plethora of sensors embedded in wearable and mobile devices has enabled affordable biosignal collection, delivering fast solutions for monitoring physical and mental health. Biosignals, such as brain electroencephalography (EEG) or heart electrocardiography (ECG), can be non-invasively measured, allowing for automatic detection of many health conditions such as epilepsy (Acharya et al., 2013), depression (de Aguiar Neto and Rosa, 2019) or heart failures (Konstantinos et al., 2021).

For the evaluation, two publicly-available biometric signal datasets are used: (1) electrocardiogram heart attack (ECG) (Dau et al., 2019), and (2) electroencephalogram eye-movement artifacts (EEG-A) (Hamid et al., 2020). Datasets are split into 80% training, 10% validation, and $/10\%$ testing maintaining class proportions. Each dataset is paired with a different architecture: FCNet (Wang et al., 2017) for ECG and VGG16 (Simonyan and Zisserman, 2014) for EEG-A.

ECG (Dau et al., 2019) is a dataset of univariate time series of ECG signals of length 140 extracted from a single patient. Each signal falls into one of 5 classes combined to make two labels: Normal (N) and Abnormal (R-on-T, PVC, SP, UB).

EEG-A (Hamid et al., 2020) consists of univariate time series of length 2000 extracted from 213 patients from the Temple University Artifact Corpus (v2.0). 21 EEG channels were retained from all patients, and signals were resampled to 250 Hz. All EEG signals were bandpass filtered (0.3-40 Hz) using a second-degree Butterworth filter and notch filtered at the power lower frequency 60 Hz. Segments of clean and eye movement artifact signals were used for this specific task.

FCNet (Wang et al., 2017) is a fully convolutional network (CNN) consisting of 4 blocks each containing a 1D convolution, batch normalization (BN), and Rectified Linear Unit (ReLU) activation. The output of the fourth block is averaged over the time dimension using global average pooling (GAP) and fed to a 1D convolutional layer with filter length 1. Convolutional layers all have 128 filters of lengths 8, 5, 5, and 3, all with a stride of 1 and zero padding to preserve the length of each time series input.

VGG16 (Simonyan and Zisserman, 2014) is a CNN composed of 13 blocks containing a 1D convolution, BN, and ReLU activation. Convolutions filters are distributed across the blocks following 2×64, 2×128, 2×256, 3×512, 3×512 all with length 3 and stride 1. Consecutive

blocks with different numbers of filters are interleaved with a max pooling layer of length 2 and stride 2. The output of these blocks is fed to a GAP and 3 FC layers with dropout and ReLU activation.

5.5.2 Baselines

We compare early exit ensembles against the undefended backbone model without any exits (No mitigation), the backbone model trained completely with PGD adversarial attacks (PGD AT), and combinations of early exit ensembles with orthogonal existing techniques: adversarial training (AT) and Monte Carlo dropout (MCDrop).

No mitigation. The unchanged backbones FCNet and VGG16 represent naive deep learning models without any mitigation or robustness techniques applied to it.

PDG adversarial training (PGD AT). The backbone model trained with adversarial samples generated with the PGD attack.

Early exit + PGD AT. Early exit ensemble trained with adversarial examples generated with the PGD attack, which had access to only the last exit (i.e., meaning the backbone), the averaging of the predictions from the individual exits is applied. A total of 5 exits were used for the ensemble.

Early exit + PGD-ADV AT. Early exit ensembles trained with samples generated by the PGD average attack. Here, the attack has access to all individual exits which form the ensemble.

Early exit + PGD-MAX AT. Early exit ensembles trained with samples generated by the PGD max-average attack.

Early exit + MCDrop. An approximate Bayesian probabilistic approach implemented by adding dropout layers to each of the exit blocks (including the last one of the backbone) during training and activating them during inference. Further increases diversity and adds stochasticity to the model. A total of 5 exits were used for the ensemble. Additionally, a total of 5 Monte Carlo samples are taken with a dropout rate of $p_{\text{MC}} = 0.2$.

Early exit + PGD AT + MCDrop. Early exit ensembles trained with samples generated by the PGD attack and dropout in the exit blocks. At inference time, the averaging of the predictions from the individual exits and MC samples are applied.

Early exit + PGD-AVG AT + MCDrop. Early exit ensembles trained with samples generated by the PGD average attack and dropout in the exit blocks.

Early exit + PGD-MAX AT + MCDrop. Early exit ensembles trained with samples generated by the PGD max-average attack and dropout in the exit blocks.

PGD-AVG and PGD-MAX, both as attacks and adversarial training, are not applied to the backbone (without exit) model since they would be the same as a naive PGD since the ensemble size is 1.

5.5.3 Hyperparameters

All models are trained using the Adam optimizer and an optimally tuned learning rate, batch size, and epochs: FCNet ($1e^{-2}$, 200, 250), and VGG16 ($1e^{-4}$, 200, 200). All results for our approach are based on a loss with α_i =1 as well as a learning capacity factor and exit strategy optimally tuned as follows: FCNet (γ=0.0, *Block-wise*), and VGG16 (γ=0.5, *Semantic*) with an ensemble size $M = 5$. To prevent overfitting, early stopping is based on the best validation accuracy with a patience of 5.

For the PGD (incl. AVG and MAX) adversarial attacks, 20 maximum iterations are used. The perturbation strength is set at $\epsilon = 100mV$ and $\epsilon = 10mV$ for EEG and ECG, respectively, while $\beta = \epsilon/4$ for both. PGD attacks (especially on EEG data) can bring strong and perceptible square-wave displacements, which would allow both experts and non-experts to distinguish them. With the aforementioned values, we provide the strongest perturbations possible while keeping the attack imperceptible to at least a non-expert eye. For UAP, we used the default values in the original work from Moosavi-Dezfooli et et al. (2017) where min_fool_rate=0.8 and a sample_size=100. For adversarial training, we use the same configurations as the attacks with 10 iterations for the PGDs.

5.6 Results

We present the results of early exit ensembles on defending against attacks from the attack vector 1 (Section 5.6.1) and 2 (Section 5.6.2) experiments and compare them to the aforementioned baselines. Additionally, in Section 5.6.2, we analyze the power of combining early exit ensembles with existing techniques for better mitigation. Performance is evaluated using class weighted F1 and expected calibration error (ECE). While F1 indicates model accuracy, ECE measures model calibration as the expected difference between accuracy and predicted confidence. See Section 2.3.5 for further details.

ECG - FCNet						
	No attack		**UAP**		**PGD**	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
No mitigation	0.98	**0.01**	0.70	0.16	0.33	0.27
PGD AT	0.92	0.07	**0.93**	**0.03**	0.73	0.13
Early exit	**0.99**	**0.01**	0.89	0.15	**0.77**	**0.11**

EEG-A - VGG16						
	No attack		**UAP**		**PGD**	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
No mitigation	0.81	0.11	0.76	0.05	0.18	0.66
PGD AT	0.80	**0.03**	**0.80**	**0.03**	**0.82**	0.05
Early exit	**0.85**	**0.03**	**0.80**	0.06	0.81	**0.04**

Table 5.6.1: Impact of adversarial perturbations on F1 score and ECE for undefended, PGD 5.5 adversarial training (AT) and EarlyExit. PGD: $\epsilon = 100mV$ and $\epsilon = 10mV$ for EEG and ECG, respectively, and $\beta = \epsilon/4$ (iterations=20, AT_iterations=10). UAP: min_fool_rate=0.8 and sample_size=100. The best results are in bold.

5.6.1 Attack Vector 1

Assumption

*The adversary is **unaware** of the intermediate exits.*

This scenario represents the attacks in Attack Vector 1, PGD, and UAP, described in Section 5.6.1 where the adversary is unaware of the intermediate exits either because it could not access them or for efficiency reasons. Table 5.6.1 summarizes the accuracy (as measured by the F1 score),

and the expected calibration error (ECE) results when the models are under adversarial attack.

As expected, Early exit provides better accuracy on clean data (no attack) compared to the undefended (no mitigation) model provided by averaging results from an ensemble of models. Additionally, PGD AT shows a lower accuracy on clean data demonstrating one of the disadvantages of adversarial training as mentioned in Section 5.2.2. As a universal adversarial attack, UAP degrades accuracy less than PGD; however, early exit ensembles can significantly mitigate it with a 19pp improvement on ECG-FCNet. As we know, the PGD attack applied in this scenario is a white box attack that can see only the output of the final exit, allowing for the intermediate exits in the ensemble to disagree with the attacked exit and provide the desired mitigation. Compared to the undefended model, our technique improves accuracy by 44pp and 63pp for ECG-FCNet and EEG-A-VGG16, respectively. With the decrease in accuracy, we see an increase in ECE when the model is attacked. This is expected behavior that adversarial mitigation techniques try to improve on. Early exit can decrease the ECE compared to the vanilla model and, in most cases, provides a better-calibrated model compared to PGD AT, except for the FCNet under UAP, where PGD AT provides better accuracy and lower ECE. Nevertheless, Early exit shows that our approach can provide comparable (even better in some instances) mitigation to the state-of-the-art adversarial training without any previous exposure to the adversarial attack presenting a general (non-attack-specific) approach.

5.6.2 Attack Vector 2

Assumption

*The adversary is **aware** of the intermediate exits.*

In a second scenario, Table 5.6.3, shows the impact of adversarial attacks which are aware of the intermediate exits of the model and maximize the overall loss accordingly. Here, early exit ensembles can still provide adversarial robustness, although the more the attacker aims the individual exits weaker the mitigation. The strength of our approach still relies on the fact that it does not depend on a specific attack or training procedure, making it a perfect technique to combine with adversarial training. In a real-world deployment, early exit ensembles have the potential to have a high level of robustness provided by adversarial training of known attacks as well as

ECG - FCNet						
	No attack		**PGD-AVG**		**PGD-MAX**	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
Early exit	**0.99**	**0.01**	0.71	0.10	0.55	0.11
Early exit + PGD AT	0.92	0.11	**0.95**	**0.06**	**0.94**	0.06
Early exit + PGD-AVG AT	0.93	0.18	0.87	0.10	0.86	0.08
Early exit + PGD-MAX AT	0.92	0.28	0.90	0.03	0.86	**0.01**

EEG-A - VGG16						
	No attack		**PGD-AVG**		**PGD-MAX**	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
Early exit	**0.85**	**0.03**	0.76	**0.04**	0.48	0.24
Early exit + PGD AT	0.83	0.07	0.80	**0.04**	**0.82**	0.07
Early exit + PGD-AVG AT	0.82	0.11	**0.82**	0.10	0.81	0.08
Early exit + PGD-MAX AT	0.82	0.04	0.78	0.27	**0.82**	**0.04**

Table 5.6.3: Impact of adversarial perturbations on F1 score and ECE for Early exit and or combined with traditional PGD AT and exit aware AT (PGD-AVG, PGD-MAX). The best results are in bold.

robustness towards new and unknown attacks. As discussed in Section 5.4, PGD-MAX is the strongest attack given its transferability characteristics. We can see this in our results too, where early exit ensembles benefit more from the PGDs adversarial training to compensate for the accuracy loss caused by PGD-MAX. However, contrary to the lightweight approach provided by early exit ensembles, PGD-MAX is very expensive to produce, circa 10x more expensive than PGD and PGD-AVG. Crafting a PGD-MAX attack for training purposes on a VGG16 model (batch size of 250 samples), costs 475 seconds on an Nvidia Tesla-V100 GPU, while PGD and PGD-AVG cost 38 and 48 seconds, respectively **?**. Early exit ensembles, instead, only add a slight memory overhead of 16% and computation overhead of 12% in the worst-case scenario (VGG16). Another interesting observation stemming from the results is the fact that, when combined with Early Exit, naive PGD AT can defend well against its exit aware versions while significantly reducing the cost of training too.

ECG - FCNet						
	No attack		PGD-AVG		PGD-MAX	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
Early exit + MCDrop	**0.99**	**0.01**	0.74	0.08	0.60	0.10
Early exit + PGD AT + MCDrop	0.92	0.10	**0.95**	**0.05**	**0.95**	0.06
Early exit + PGD-AVG AT + MCDrop	0.93	0.15	0.89	0.10	0.86	0.07
Early exit + PGD-MAX AT + MCDrop	0.92	0.26	0.90	0.03	0.86	**0.01**
EEG-A - VGG16						
	No attack		**PGD-AVG**		**PGD-MAX**	
Mitigation	**F1**	**ECE**	**F1**	**ECE**	**F1**	**ECE**
Early exit + MCDrop	**0.85**	**0.03**	0.77	**0.04**	0.55	0.22
Early exit + PGD AT + MCDrop	0.83	0.06	0.80	**0.04**	0.82	0.07
Early exit + PGD-AVG AT + MCDrop	0.82	0.09	**0.81**	0.10	0.81	0.09
Early exit + PGD-MAX AT + MCDrop	0.82	0.04	0.78	0.30	**0.83**	**0.04**

Table 5.6.5: Impact of adversarial perturbations on F1 score and ECE for Early exit combined with MCDrop, traditional PGD AT, and exit aware AT (PGD-AVG, PGD-MAX). For MCDrop, a dropout layer is added at each exit block (including the final). A total of 5 exits were used for the ensemble. Additionally, a total of 5 Monte Carlo samples are taken with a dropout rate of $p_{\text{MC}} = 0.2$. Best results are in bold.

Finally, we run experiments on increasing the diversity of the early exit ensemble and adding stochasticity to the model through the addition of Monte Carlo dropout. Our intuition is that even if the adversary has access to the intermediate exits, it might not be aware that the dropout is activated during inference, and even if it knows that, many queries to the model are needed to understand the number of samples or capture the different implicit architectures created when dropout is activated. In Table 5.6.5, we show how the introduction of MCDrop impacts the accuracy and the calibration error in the presence of the attacks. When no attack is applied, the addition of MCDrop does not improve the accuracy of the prediction; however, it shows a slight improvement in ECE compared to the results in Table 5.6.3. It is when the adversary starts to attack the individual exits that MCDrop comes into play by increasing the accuracy up to 7pp. When using adversarial training with Early Exit, the introduction of MCDrop does not impact the accuracy (max 1pp), suggesting that the early exit ensembles and adversarial training are sufficient for mitigation. Nevertheless, MCDrop decreases the ECE overall, presenting better-

calibrated models in the presence of adversarial attacks.

5.7 Discussion and conclusions

In this chapter, which concludes the empirical works of this thesis, we propose an adversarial mitigation technique for biosignal classification tasks. Deep learning techniques are increasingly used for decision-making in health applications; however, these can easily be manipulated by adversarial examples across different clinical domains. Their security and privacy vulnerabilities raise concerns about the practical deployment of these systems. The number and variety of adversarial attacks grow continuously, making it difficult for mitigation approaches to provide effective solutions. Current mitigation techniques often rely on expensive re-training procedures as new attacks emerge.

The approach proposed in this chapter is based on our previous findings in Chapter 4 interpreting early exit neural networks as an ensemble of weight-sharing sub-networks. Our experiments on state-of-the-art deep learning models show that early exit ensembles can provide robustness generalizable to various white-box and universal adversarial attacks. Our framework increases the accuracy of vulnerable deep learning models up to 60 percentage points while providing adversarial mitigation comparable to adversarial training. This is achieved without previous exposure to the adversarial perturbation or the computational burden of re-training.

Additionally, we show that the orthogonality of early exit ensembles to existing techniques such as adversarial training and Monte Carlo dropout can be exploited for further mitigation. The aggregation of different adversarial mitigation techniques is well known to be a good strategy under the condition that the single approach is a strong defense mechanism even if used separately. The strength of early exit ensembles is the ability to work well on unknown attacks, making it a universal strategy. We envision future deep learning models to be still trained with adversarial examples for known attacks and use attack-agnostic techniques such as early exit ensembles to handle, to some extent, future attacks.

Finally, considering the findings in this chapter and in Chapter 4, we can conclude that early exit ensembles are a very powerful framework that can be used for uncertainty estimation, in-distribution calibration, out-of-distribution detection, and adversarial mitigation. These benefits come with ease of implementation since the framework can be directly applied to the already known early exit networks, which did not exploit the full potential of this paradigm on robustness until this work. Additionally, the small computation footprint combined with the strong

transferability across architectures and datasets positions early exit ensembles as a *de facto* baseline for future work on robustness.

Chapter 6

Conclusion and future directions

At the beginning of this dissertation, we highlighted the importance of uncertainty estimation in health-critical scenarios. We discussed the limitations of traditional methods and current technology and focused our research efforts on investigating different techniques to efficiently and accurately provide uncertainty aware mobile health applications. Contribution-wise, we first proposed a solution to enable already trained deep learning models to output uncertainty estimates alongside the classification prediction in mobile sensing scenarios (Chapter 3) and followed with a new framework that proposes an efficient ensemble-based approach for uncertainty quantification with an analysis of in-distribution calibration and out-of-distribution detection (Chapter 4). We concluded by exploring the latter framework as an effective attack-agnostic method for adversarial mitigation in biosignal classification tasks (Chapter 5). Our findings can be used better to inform machine learning systems with respect to downstream tasks and also to yield real-world robustness against distributional shifts and adversarial attacks. In this chapter, we summarize our key contributions and propose several avenues for further research.

6.1 Summary of contributions

In this section, we reflect on the research questions introduced in Chapter 1 and summarize the major contributions of this dissertation.

6.1.1 Uncertainty aware mobile sensing for edge computing platforms

Research Question 1: *How can we provide uncertainty awareness on deep learning models which have already been trained and deployed?*

Contribution 1: In Chapter 3, we proposed an efficient framework that enables already trained deep learning models to provide uncertainty estimates without the need for re-training or fine-tuning strategies. Our framework is grounded on the fact that when using a neural network trained with dropout, the network does not have a deterministic structure anymore since it is partly described by random variables. Based on this, we enabled the convolution operations in the stochastic network generated by dropout to apply to inputs described by probabilistic distributions and generate distributions as outputs. These distributions are then propagated through the whole network carrying and transmitting the information on the prediction and the uncertainty of these predictions. Our framework is efficient by design and built from the ground up to generate predictive uncertainty, in addition to the classification task, from a single forward-pass of input data. This pipeline ensures minimal overhead, especially when compared with techniques that rely on multiple forward passes or several models, and an equal linear rise in energy and latency requirements, making them unfeasible for edge devices. Our performance and efficiency evaluation with mobile applications datasets on Nvidia Jetson TX2 and Nano edge platforms showed that our approach obtains not only robust and accurate uncertainty estimations but also outperforms state-of-the-art methods in terms of systems performance, reducing energy consumption (up to 28–folds), keeping the memory overhead at a minimum while still improving accuracy (up to 16%). Findings from this chapter were published at the *ACM/IEEE Symposium on Edge Computing, SEC, 2021*.

6.1.2 Early exit ensembles for uncertainty quantification

Research Question 2: *How can existing efficient paradigms be exploited for uncertainty quantification?*

Contribution 2: Chapter 4 introduced early exit ensembles, a novel framework for enabling uncertainty quantification in any feed-forward neural network. At its core, there is a new interpretation of early exit neural networks capable of capturing predictive uncertainty via an implicit ensemble of early exits. In particular, early exit ensembles are a collection of weight-sharing sub-networks created by adding exit branches to any backbone neural network architecture. Our

approach facilitates training all ensemble members jointly in a single model, as well as providing uncertainty estimations in a single forward-pass. The computational efficiency of training and inference facilitated by our framework means that early exit ensembles can be applied to a wide range of real-world applications. To showcase this, we evaluated our approach on several classification tasks using state-of-the-art deep learning architectures applied to medical datasets. Our experiments suggested that early exit ensembles have the potential to improve the trustworthiness of models in high-risk medical decision-making, given their ability to provide well-calibrated predictive uncertainty for both in- and out-of-distribution input samples. Finally, additional empirical evaluation of the proposed solution demonstrated strong performance in accuracy and uncertainty metrics as well as computation gain highlighting the benefit of combining multiple structurally diverse models that can be jointly trained. Work associated with Chapter 4 is included in several publications at the *Workshop on Human In the Loop Learning, HILL, at ICML, 2021*; *Machine Learning for Health, ML4H, 2021* conference; and the *IEEE International Conference on Acoustics, Speech, & Signal Processing, ICASSP, 2022*. At ML4H'21, this work was selected for a spotlight talk and granted *Best Thematic Paper Award* for best contribution to *Real-world robustness and generalization in machine learning for health*, validating the relevance of this study to the research community, especially, the one working on health-related problems.

6.1.3 Towards adversarial mitigation with early exit ensembles

Research Question 3: *To what extent can the introduced ensemble-based framework mitigate against inference time adversarial attacks? How can its orthogonality to existing mitigation techniques be exploited for better robustness?*

Contribution 3: In Chapter 5, we proposed an attack-agnostic adversarial mitigation approach for mobile health tasks. To achieve the desired adversarial defense, we leverage our findings in Chapter 4 interpreting early exit neural networks as an ensemble of weight-sharing sub-networks. The number and variety of adversarial attacks grow continuously; therefore, our solution becomes crucial to health-critical deep learning techniques as new attacks emerge. Our intuition on using this defense mechanism relies on the fact that the majority of adversarial attacks are built to fool traditional feed-forward neural networks and generally discard or are unaware of the intermediate exits. To test our hypothesis, we considered various white-box and universal adversarial attacks showing that our framework increases the accuracy of attacked deep learning models up to 60 percentage points while providing adversarial mitigation comparable to adversarial train-

ing. When delving deeper into other scenarios, we noticed that when the adversary is aware of the intermediate exits, our technique can still provide some level of robustness, but the mitigation is generally weaker. To solve this, we showed that we could exploit the orthogonality of early exit ensembles to existing techniques such as adversarial training and Monte Carlo dropout for further mitigation. Our work on adversarial mitigation was published at the *IEEE International Engineering in Medicine and Biology Conference, EMBC, 2022*; and the *Workshop on Security and Privacy for Mobile AI, MobiSys, 2021*.

6.2 Future work

In this dissertation, we have answered some research questions introduced in Chapter 1; nevertheless, many future directions were naturally uncovered.

6.2.1 Framework enhancements and optimizations

Model diversity is a very important characteristic when building a strong framework that can accurately capture predictive uncertainty. This is especially crucial in ensemble-based techniques, where having diverse members leads to diverse predictions better at recognizing out-of-distribution or adversarial samples. Therefore, new methodologies for enforcing model diversity in order to boost ensemble performance is an interesting research direction. A limitation of early exit ensembles, presented in Chapter 4, is the ensemble size tied to the underlying backbone architecture. To overcome this issue, future work could add dropout layers and/or add multiple heads to the exit blocks to further increase ensemble size and diversity between ensemble members. Additionally, in Chapter 4 we introduced several exit placement strategies and empirically evaluated their performance in terms of accuracy and uncertainty quality. Future work could include other optimization strategies, such as traditional greedy search algorithms or the use of Neural Architecture Search, to find the best performing members of the early exit ensemble.

Several performance enhancements could be developed for the framework introduced in Chapter 3. Our approach, based on Gaussian approximations of the internal statistics of the network, is a feasible solution to providing uncertainty estimates on edge devices. It is also a powerful solution considering that the approach does not require re-training or fine-tuning; however, it comes with some limitations too. Our approximation could be improved by considering the fact that even if the inputs were perfectly Gaussian, the outputs of non-linear activations naturally yield skewed distributions with values possibly in a limited subset of the domain, which are

not perfectly Gaussian distributed. Future work, therefore, could consider other approximations that minimize this discrepancy while simultaneously yielding uncertainty estimates for existing neural networks without increased operational costs, as is the case with our approach. In this work, we focus on enabling convolution neural networks to generate predictive uncertainty estimations. Our approach could be extended to recurrent neural networks. However, these alterations require additional effort in providing the right mathematical foundations and testing their feasibility on real-life datasets. Additionally, our framework focuses on providing predictive output distributions on models that have been trained with dropout regularization. Another widely used technique for regularization is batch normalization which could eventually be used to approximate the basic operations in neural networks. In this way, the networks trained with batch normalization can get the same benefits at inference time. Nevertheless, a key advantage of this framework is the fact that we model each layer to output predictive distributions. This can be a very flexible instrument for many applications, e.g., early prediction models, a class of conditional computation models that exit once a criterion (e.g., sufficient accuracy and low uncertainty) is satisfied at early layers. Such models can be very useful in intermittent learning systems which rely on harvested energy (Lee and Nirjon, 2019; Montanari et al., 2020).

6.2.2 Including uncertainty in the training procedure

The majority of efficient uncertainty aware approaches, including ours, only exploits uncertainty at inference time. However, considering this valuable information during training can lead to better performance. Current methods that incorporate this functionality tend only to be pure Bayesian approaches that struggle with underfitting at scale and parameter efficiency (Dusenberry et al., 2020). For instance, unlike Monte Carlo dropout or deep ensembles, early exit ensembles have access for free to uncertainty within the ensemble during training. We envision that our framework could be extended to exploit the uncertainty during training for adaptive decision-making on joint tasks such as signal interpolation, to be included in the classification loss for better performance, or to handle human annotation uncertainty in the downstream task (Kwon et al., 2019).

6.2.3 End-to-end systems for adversarial mitigation

Currently, the majority of research on adversarial attacks and defense focuses on image data, with only 5% of the work dedicated to other types of sensor data and signals (such as EEG, ECG, IMU, etc.) (Sadeghi et al., 2020). In Chapter 5, we have presented a solution working on

biosignal mobile tasks. We strongly believe that future work should invest more in the robustness of mobile sensing applications since a vast majority of health-related real-world applications are completely different from clinical settings in terms of the type of data used, noise, and efficiency considerations.

Additionally, as we saw in Chapter 5, the more information about the model and the training data the adversary has, the stronger the attack and the lower the mitigation effect. However, in order to retrieve the model and underlying data distribution, for instance, the attack needs to gain access to the hardware itself or the device-device/cloud-device communication. The same is valid for later stages of the attack where the adversary needs access to the peripherals (camera, BCI device, earbuds, or smartwatch) to inject the perturbed malign signal. Given this behavior, we believe that the way forward to guarantee robustness to adversarial attacks is to provide end-to-end systems built from the ground up to guarantee defense against adversarial attacks by using both traditional system security functionalities and machine learning mitigation approaches. We have already started working on this (Tarkhani et al., 2022) by providing an information flow control mechanism to detect information flow violations in BCI applications. However, we believe that many other hybrid approaches can be introduced. A good direction is the utilization of trusted execution environments (TEE). However, edge devices have limited computational resources; therefore, running the whole model inside TEEs is unfeasible. Several works have proposed running only specific layers inside the TEE; nevertheless, choosing which layers is a difficult task (Mo et al., 2020, 2021). We believe that early exit ensembles can facilitate this task by offering an intuitive way of choosing to run only the exit blocks inside TEEs. Therefore, future work can further explore this direction.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Moloud Abdar, Maryam Samami, Sajjad Dehghani Mahmoodabad, Thang Doan, Bogdan Mazoure, Reza Hashemifesharaki, Li Liu, Abbas Khosravi, U Rajendra Acharya, Vladimir Makarenkov, et al. Uncertainty quantification in skin cancer classification using three-way decision-based bayesian deep learning. *Computers in biology and medicine*, page 104418, 2021.

U Rajendra Acharya, S Vinitha Sree, G Swapna, Roshan Joy Martis, and Jasjit S Suri. Automated eeg analysis of epilepsy: a review. *Knowledge-Based Systems*, 45:147–165, 2013.

Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Tuka Al Hanai, Mohammad M Ghassemi, and James R Glass. Detecting depression with audio/text sequence modeling of interviews. In *Interspeech*, pages 1716–1720, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Fayez Alharbi and Katayoun Farrahi. A convolutional neural network for smoking activity recog-

nition. In *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–6. IEEE, 2018.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Dana Angluin. Queries and concept learning. *Machine learning*, 2:319–342, 1988.

Bálint Antal and András Hajdu. An ensemble-based system for automatic screening of diabetic retinopathy. *Knowledge-based systems*, 60:20–27, 2014.

Les Atlas, David Cohn, and Richard Ladner. Training connectionist networks with queries and selective sampling. *Advances in neural information processing systems*, 2, 1989.

Brandon Ballinger, Johnson Hsieh, Avesh Singh, Nimit Sohoni, Jack Wang, Geoffrey H Tison, Gregory M Marcus, Jose M Sanchez, Carol Maguire, Jeffrey E Olgin, et al. Deepheart: semi-supervised sequence learning for cardiovascular risk prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Eric B Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *International joint conference on neural networks*, volume 8, page 8. Beijing China, 1992.

Sourav Bhattacharya and Nicholas D Lane. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, pages 176–189. ACM, 2016.

Alsallakh Bilal, Amin Jourabloo, Mao Ye, Xiaoming Liu, and Liu Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2017.

Paschalis Bizopoulos, George I Lambrou, and Dimitrios Koutsouris. Signal2image modules in deep neural networks for eeg classification. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 702–705. IEEE, 2019.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. *arXiv preprint arXiv:1702.07811*, 2017.

Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-

entropy vs. pairwise losses. In *European Conference on Computer Vision*, pages 548–564. Springer, 2020.

John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

Glenn W Brier et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Peggy Bui. Using ai to help find answers to common skin conditions, 2021. URL https://blog.google/technology/health/ai-dermatology-preview-io-2021/.

Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Alexander Campbell, Lorena Qendro, Pietro Liò, and Cecilia Mascolo. Robust and efficient uncertainty aware biosignal classification via early exit ensembles. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3998–4002. IEEE, 2022.

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Justin Chan, Thomas Rea, Shyamnath Gollakota, and Jacob E Sunshine. Contactless cardiac arrest detection using smart devices. *arXiv preprint arXiv:1902.00062*, 2019.

Saket S Chaturvedi, Kajol Gupta, and Prakash S Prasad. Skin lesion analyser: An efficient seven-way multi-class skin cancer classification using mobilenet. In *International Conference on Advanced Machine Learning Technologies and Applications*, pages 165–176. Springer, 2020.

Jagmohan Chauhan, Jathushan Rajasegaran, Suranga Seneviratne, Archan Misra, Aruna Seneviratne, and Youngki Lee. Performance characterization of deep learning models for breathing-based authentication on resource-constrained devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(4):1–24, 2018.

Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. The opportunity challenge: A benchmark

database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15): 2033–2042, 2013.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

E Commission et al. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts. 2021.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

Sajal K Das and Christopher Rose. Coping with uncertainty in mobile wireless networks. In *Emerging Location Aware Broadband Wireless Ad Hoc Networks*, pages 189–204. Springer, 2005.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Fernando Soares de Aguiar Neto and João Luís Garcia Rosa. Depression biomarkers using non-invasive eeg: A review. *Neuroscience & Biobehavioral Reviews*, 105:83–93, 2019.

Toon De Pessemier and Luc Martens. Heart rate monitoring, activity recognition, and recommendation for e-coaching. *Multimedia Tools and Applications*, 77(18):23317–23334, 2018.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009.

S Divya, V Indumathi, S Ishwarya, M Priyasankari, and S Kalpana Devi. A self-diagnosis medical chatbot using artificial intelligence. *Journal of Web Development and Web Designing*, 3(1):1–7, 2018.

Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792. PMLR, 2020.

Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande. Deep neural networks as point estimates for deep gaussian processes. *Advances in Neural Information Processing Systems*, 34:9443–9455, 2021.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Andrea Ferlini, Dong Ma, Lorena Qendro, and Cecilia Mascolo. Mobile health with head-worn devices: Challenges and opportunities. *IEEE Pervasive Computing*, 2022.

Davide Figo, Pedro C Diniz, Diogo R Ferreira, and João M Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7): 645–662, 2010.

Rosa L Figueroa, Qing Zeng-Treitler, Long H Ngo, Sergey Goryachev, and Eduardo P Wiechmann. Active Learning for Clinical Text Classification: Is it Better than Random Sampling? *Journal of the American Medical Informatics Association*, 19(5):809–816, 06 2012. ISSN 1067-5027. URL https://doi.org/10.1136/amiajnl-2011-000648.

Samuel G Finlayson, John D Bowers, Joichi Ito, Jonathan L Zittrain, Andrew L Beam, and Isaac S Kohane. Adversarial attacks on medical machine learning. *Science*, 363(6433):1287–1289, 2019.

Loic Le Folgoc, Vasileios Baltatzis, Sujal Desai, Anand Devaraj, Sam Ellis, Octavio E Martinez Manzanera, Arjun Nair, Huaqi Qiu, Julia Schnabel, and Ben Glocker. Is mc dropout bayesian? *arXiv preprint arXiv:2110.04286*, 2021.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Tadayoshi Fushiki et al. Bootstrap prediction and bayesian prediction under misspecified models. *Bernoulli*, 11(4):747–758, 2005.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015a.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015b.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016a.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016b.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.

Yarin Gal, Petros Koumoutsakos, Francois Lanusse, Gilles Louppe, and Costas Papadimitriou. Bayesian uncertainty quantification for machine-learned models in physics. *Nature Reviews Physics*, 4(9):573–577, 2022.

Yiming Gan, Yuxian Qiu, Jingwen Leng, Minyi Guo, and Yuhao Zhu. Ptolemy: Architecture support for robust deep learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 241–255. IEEE, 2020.

Carlos García Rodríguez, Jordi Vitrià, and Oscar Mora. Uncertainty-based human-in-the-loop deep learning for land cover segmentation. *Remote Sensing*, 12(22):3836, 2020.

Petko Georgiev, Sourav Bhattacharya, Nicholas D Lane, and Cecilia Mascolo. Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–19, 2017a.

Petko Georgiev, Nicholas D Lane, Cecilia Mascolo, and David Chu. Accelerating mobile audio

sensing algorithms through on-chip gpu offloading. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 306–318, 2017b.

Panagiotis Germanakos, Constantinos Mourlas, and George Samaras. A mobile agent approach for ubiquitous and personalized ehealth information systems. In *Proceedings of the Workshop on 'Personalization for e-Health'of the 10th International Conference on User Modeling (UM'05)*, 2005.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient mcmc. *arXiv preprint arXiv:1806.04522*, 2018.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Pablo M Granitto, Pablo F Verdes, and H Alejandro Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139–162, 2005.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Jessica Guynn. Google photos labeled black people'gorillas'. *Usa Today*, 1, 2015.

Ahmed Hamid, Katherine Gagliano, Safwanur Rahman, Nikita Tulin, Vincent Tchiong, Iyad Obeid, and Joseph Picone. The temple university artifact corpus: An annotated corpus of eeg artifacts. In *2020 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–4. IEEE, 2020.

Nils Y Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880*, 2016.

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

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

Geoffrey Hinton and Drew Van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *in Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Citeseer, 1993.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

SeulGi Hong, Heonjin Ha, Junmo Kim, and Min-Kook Choi. Deep Active Learning with Augmentation-Based Consistency Estimation. *CoRR*, abs/2011.02666, 2020. URL https://arxiv.org/abs/2011.02666.

Jiri Hron, Alexander G de G Matthews, and Zoubin Ghahramani. Variational gaussian dropout is not bayesian. *arXiv preprint arXiv:1711.02989*, 2017.

Shih-Chung Hsu, Cheng-Hung Chuang, Chung-Lin Huang, Ren Teng, and Miao-Jian Lin. A video-based abnormal human behavior detection for psychiatric patient monitoring. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1–4. IEEE, 2018.

Chuang Hu, Wei Bao, Dan Wang, and Fengming Liu. Dynamic adaptive dnn surgery for inference acceleration on the edge. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1423–1431. IEEE, 2019.

Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang. Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference. *arXiv preprint arXiv:2002.10025*, 2020.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.

Venediktos V Kapetanakis, Alicja R Rudnicka, Gerald Liew, Christopher G Owen, Aaron Lee,

Vern Louw, Louis Bolter, John Anderson, Catherine Egan, Sebastian Salas-Vega, et al. A study of whether automated diabetic retinopathy image assessment could replace manual grading steps in the english national screening programme. *Journal of medical screening*, 22(3):112–118, 2015.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning*, pages 3301–3310. PMLR, 2019.

Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. Scaling human activity recognition via deep learning-based domain adaptation. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE, 2018.

Christine E King and Majid Sarrafzadeh. A survey of smartwatches in remote health monitoring. *Journal of healthcare informatics research*, 2(1):1–24, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

Siontis C. Konstantinos et al. Artificial intelligence-enhanced electrocardiography in cardiovascular disease management. *Nature Reviews Cardiology*, 18(7), 2021.

Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Hyeokhyen Kwon, Gregory D Abowd, and Thomas Plötz. Handling annotation uncertainty in human activity recognition. In *Proceedings of the 23rd International Symposium on Wearable Computers*, pages 109–117. ACM, 2019.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

Nicholas D Lane, Petko Georgiev, and Lorena Qendro. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294. ACM, 2015.

Nicholas D Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, page 23. IEEE Press, 2016.

Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2020a.

Stefanos Laskaridis, Stylianos I Venieris, Hyeji Kim, and Nicholas D Lane. Hapi: Hardware-aware progressive inference. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020b.

Seulki Lee and Shahriar Nirjon. Neuro. zero: a zero-energy neural network accelerator for embedded sensing and inference systems. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 138–152. ACM, 2019.

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports*, 7(1):17816, 2017.

David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *Acm Sigir Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995.

David D. Lewis and Jason Catlett. Heterogeneous Uncertainty Sampling for Supervised Learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 148–156. Morgan Kaufmann, San Francisco (CA), 1994. ISBN 978-1-55860-335-6. URL https://doi.org/10.1016/B978-1-55860-335-6.50026-X.

En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19 (1):447–457, 2019.

Xirong Li, Yang Zhou, Jie Wang, Hailan Lin, Jianchun Zhao, Dayong Ding, Weihong Yu, and

Youxin Chen. Multi-modal multi-instance learning for retinal disease recognition. *arXiv preprint arXiv:2109.12307*, 2021.

Yingzhen Li and Yarin Gal. Dropout inference in bayesian neural networks with alpha-divergences. In *International conference on machine learning*, pages 2052–2061. PMLR, 2017.

Yuan Liu, Ayush Jain, Clara Eng, David H Way, Kang Lee, Peggy Bui, Kimberly Kanada, Guilherme de Oliveira Marinho, Jessica Gallegos, Sara Gabriele, et al. A deep learning system for differential diagnosis of skin diseases. *Nature medicine*, 26(6):900–908, 2020.

William Lotter, Abdul Rahman Diab, Bryan Haslam, Jiye G Kim, Giorgia Grisot, Eric Wu, Kevin Wu, Jorge Onieva Onieva, Yun Boyer, Jerrold L Boxerman, et al. Robust breast cancer detection in mammography and digital breast tomosynthesis using an annotation-efficient deep learning approach. *Nature Medicine*, 27(2):244–249, 2021.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR, 2017.

Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast, Gokul T Chittaranjan, Andrew T Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. Stresssense: Detecting stress in unconstrained acoustic environments using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 351–360. ACM, 2012.

David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

Aleksander Madry et al. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Gustav Mårtensson, Daniel Ferreira, Tobias Granberg, Lena Cavallin, Ketil Oppedal, Alessandro Padovani, Irena Rektorova, Laura Bonanni, Matteo Pardini, Milica G Kramberger, et al. The reliability of a deep learning model in clinical out-of-distribution mri data: a multicohort study. *Medical Image Analysis*, 66:101714, 2020.

Akhil Mathur, Anton Isopoussu, Fahim Kawsar, Nadia Berthouze, and Nicholas D Lane. Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, pages 169–180. ACM, 2019.

Fan Mo, Ali Shahin Shamsabadi, Kleomenis Katevas, Soteris Demetriou, Ilias Leontiadis, Andrea Cavallaro, and Hamed Haddadi. Darknetz: Towards model privacy at the edge using trusted execution environments. *arXiv preprint arXiv:2004.05703*, 2020.

Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 94–108, 2021.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

Alessandro Montanari, Mohammed Alloulah, and Fahim Kawsar. Degradable inference for energy autonomous vision applications. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp/ISWC '19, 2019. ISBN 978-1-4503-6869-8. doi: 10.1145/3341162.3349337. URL http://doi.acm.org/10.1145/3341162.3349337.

Alessandro Montanari, Manuja Sharma, Dainius Jenkus, Mohammed Alloulah, Lorena Qendro, and Fahim Kawsar. eperceptive: energy reactive embedded intelligence for batteryless sensors. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 382–394, 2020.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Seyed-Mohsen Moosavi-Dezfooli et, Alhussein Fawzi, Omar Fawzi, and Pascal. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Feng Nan and Venkatesh Saligrama. Adaptive classification for prediction under a budget. In *Advances in neural information processing systems*, pages 4727–4737, 2017.

Rajalakshmi Nandakumar, Shyamnath Gollakota, and Nathaniel Watson. Contactless sleep apnea

detection on smartphones. In *Proceedings of the 13th annual international conference on mobile systems, applications, and services*, pages 45–57, 2015.

Akil Narayan, Claude Gittelson, and Dongbin Xiu. A stochastic collocation algorithm with multifidelity models. *SIAM Journal on Scientific Computing*, 36(2):A495–A521, 2014.

Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. *Advances in Neural Information Processing Systems*, 30, 2017.

Leo WT Ng and Karen E Willcox. Multifidelity approaches for optimization under uncertainty. *International Journal for numerical methods in Engineering*, 100(10):746–772, 2014.

NHTSA. Nhtsa. pe 16-007. jan 2017. tesla crash preliminary evaluation report. technical report. u.s. department of transportation, national highway traffic safety administration. Jan 2017.

Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS workshop on bayesian deep learning*, volume 192, 2016.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. *Advances in Neural Information Processing Systems*, 31, 2018.

Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979. PMLR, 2019.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

Vilfredo Pareto. *Cours d'économie politique*, volume 1. Librairie Droz, 1964.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Benjamin Peherstorfer, Tiangang Cui, Youssef Marzouk, and Karen Willcox. Multifidelity importance sampling. *Computer Methods in Applied Mechanics and Engineering*, 300:490–509, 2016.

Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018.

Abdolrahman Peimankar and Sadasivan Puthusserypady. An ensemble of deep recurrent neural networks for p-wave detection in electrocardiogram. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1284–1288. IEEE, 2019.

Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(2):74, 2018.

Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS, 1992.

Mathias Perslev, Michael Hejselbak Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. *arXiv preprint arXiv:1910.11162*, 2019.

Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

Eduardo HP Pooch, Pedro Ballester, and Rodrigo C Barros. Can we trust deep learning based diagnosis? the impact of domain shift in chest radiograph classification. In *International Workshop on Thoracic Image Analysis*, pages 74–83. Springer, 2020.

Lorena Qendro and Cecilia Mascolo. Towards adversarial robustness with early exit ensembles. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 313–316. IEEE, 2022.

Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. Early exit ensembles for uncertainty quantification. In *Machine Learning for Health*.

Lorena Qendro, Alexander Campbell, Pietro Liò, and Cecilia Mascolo. High frequency eeg

artifact detection with uncertainty via early exit paradigm. *arXiv preprint arXiv:2107.10746*, 2021a.

Lorena Qendro, Jagmohan Chauhan, Alberto Gil CP Ramos, and Cecilia Mascolo. The benefit of the doubt: Uncertainty aware sensing for edge computing platforms. *arXiv preprint arXiv:2102.05956*, 2021b.

Lorena Qendro, Sangwon Ha, René de Jong, and Partha Maji. Stochastic-shield: A probabilistic approach towards training-free adversarial defense in quantized cnns. *arXiv preprint arXiv:2105.06512*, 2021c.

Bonyan Qudah and Karen Luetsch. The influence of mobile health applications on patient-healthcare provider relationships: a systematic, narrative review. *Patient education and counseling*, 102(6):1080–1089, 2019.

Joaquin Quinonero-Candela, Carl Edward Rasmussen, Fabian Sinz, Olivier Bousquet, and Bernhard Schölkopf. Evaluating predictive uncertainty challenge. In *Machine Learning Challenges Workshop*, pages 1–27. Springer, 2005.

Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. Multimodal deep learning for activity and context recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):157, 2018.

Zillur Rahman, Md Sabir Hossain, Md Rabiul Islam, Md Mynul Hasan, and Rubaiyat Alim Hridhee. An approach for multiclass skin lesion classification based on ensemble learning. *Informatics in Medicine Unlocked*, 25:100659, 2021.

C Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. In *Breakthroughs in statistics*, pages 235–247. Springer, 1992.

Daniele Ravi, Charence Wong, Benny Lo, and Guang-Zhong Yang. Deep learning for human activity recognition: A resource efficient implementation on low-power devices. In *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 71–76. IEEE, 2016.

Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

Jason Ryder, Brent Longstaff, Sasank Reddy, and Deborah Estrin. Ambulation: A tool for

monitoring mobility patterns over time using mobile phones. In *2009 International Conference on Computational Science and Engineering*, volume 4, pages 927–931. IEEE, 2009.

Koosha Sadeghi, Ayan Banerjee, and Sandeep KS Gupta. A system-driven taxonomy of attacks and defenses in adversarial machine learning. *IEEE transactions on emerging topics in computational intelligence*, 4(4):450–467, 2020.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12(5):954–966, 2020.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31, 2018.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Burr Settles and Mark Craven. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 1070–1079, USA, 2008. Association for Computational Linguistics. URL https://dl.acm.org/doi/10.5555/1613715.1613855.

Yash Sharma and Pin-Yu Chen. Bypassing feature squeezing by increasing adversary strength. *arXiv preprint arXiv:1803.09868*, 2018.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 3–18. IEEE, 2017.

Ilia Shumailov, Yiren Zhao, Robert Mullins, and Ross Anderson. The taboo trap: Behavioural detection of adversarial samples. *arXiv preprint arXiv:1811.07375*, 2018.

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

Amitojdeep Singh, Sourya Sengupta, Mohamed Abdul Rasheed, Varadharajan Jayakumar, and Vasudevan Lakshminarayanan. Uncertainty aware and explainable diagnosis of retinal disease.

In *Medical Imaging 2021: Imaging Informatics for Healthcare, Research, and Applications*, volume 11601, page 116010J. International Society for Optics and Photonics, 2021.

Berglind F Smaradottir, Jarle A Håland, and Santiago G Martinez. User evaluation of the smartphone screen reader voiceover with visually disabled participants. *Mobile Information Systems*, 2018, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Cosmin Stamate, George D Magoulas, Stefan Küppers, Effrosyni Nomikou, Ioannis Daskalopoulos, Marco U Luchini, Theano Moussouri, and George Roussos. Deep learning parkinson's from smartphone data. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 31–40. IEEE, 2017.

Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 127–140. ACM, 2015.

Abdulhamit Subasi, Mariam Radhwan, Rabea Kurdi, and Kholoud Khateeb. Iot based mobile healthcare system for human activity recognition. In *2018 15th Learning and Technology Conference (L&T)*, pages 29–34. IEEE, 2018.

Mats Svantesson, Håkan Olausson, Anders Eklund, and Magnus Thordstein. Virtual eeg-electrodes: Convolutional neural networks as a method for upsampling or restoring channels. *Journal of Neuroscience Methods*, 355:109126, 2021.

Elham Tabassi, Kevin J Burns, Michael Hadjimichael, Andres D Molina-Markham, and Julian T Sexton. A taxonomy and terminology of adversarial machine learning. *NIST IR*, pages 1–29, 2019.

Zahra Tarkhani, Lorena Qendro, Malachy O'Connor Brown, Oscar Hill, Cecilia Mascolo, and Anil Madhavapeddy. Enhancing the security & privacy of wearable brain-computer interfaces. *arXiv preprint arXiv:2201.07711*, 2022.

Aretha L Teckentrup, Peter Jantsch, Clayton G Webster, and Max Gunzburger. A multi-

level stochastic collocation method for partial differential equations with random input data. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1046–1074, 2015.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. *arXiv preprint arXiv:1802.06455*, 2018.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Terry Taewoong Um, Franz Michael Josef Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. *arXiv preprint arXiv:1706.00527*, 2017.

Ivan Ustyuzhaninov, Ieva Kazlauskaite, Markus Kaiser, Erik Bodin, Neill Campbell, and Carl Henrik Ek. Compositional uncertainty in deep gaussian processes. In *Conference on Uncertainty in Artificial Intelligence*, pages 480–489. PMLR, 2020.

Michel Valstar, Björn Schuller, Kirsty Smith, Florian Eyben, Bihan Jiang, Sanjay Bilakhia, Sebastian Schnieder, Roddy Cowie, and Maja Pantic. Avec 2013: the continuous audio/visual emotion and depression recognition challenge. In *Proceedings of the 3rd ACM international workshop on Audio/visual emotion challenge*, pages 3–10, 2013.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Junwen Wang, Xin Du, Katayoun Farrahi, and Mahesan Niranjan. Deep cascade learning for optimal medical image feature representation. 2022.

Liuan Wang, Li Sun, Mingjie Zhang, Huigang Zhang, Wang Ping, Rong Zhou, and Jun Sun. Exploring pathologist knowledge for automatic assessment of breast cancer metastases in whole-

slide image. In *Proceedings of the 29th ACM International Conference on Multimedia (ACM MM)*, pages 255–263, 2021.

Yan Wang, Shuang Cang, and Hongnian Yu. A data fusion-based hybrid sensory system for older people's daily activity and daily routine recognition. *IEEE Sensors Journal*, 18(16): 6874–6888, 2018.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H Tran, and Albert Y Zomaya. See: Scheduling early exit for mobile dnn inference during service outage. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 279–288, 2019.

Pete Warden. Speech commands. https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html, 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Jonathan Wenger, Geoff Pleiss, Marvin Pförtner, Philipp Hennig, and John P Cunningham. Posterior and computational uncertainty in gaussian processes. *arXiv preprint arXiv:2205.15449*, 2022.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

Christopher S Wood, Michael R Thomas, Jobie Budd, Tivani P Mashamba-Thompson, Kobus Herbst, Deenan Pillay, Rosanna W Peeling, Anne M Johnson, Rachel A McKendry, and Molly M Stevens. Taking connected mobile-health diagnostics of infectious diseases to the field. *Nature*, 566(7745):467–474, 2019.

Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 331–344. IEEE, 2019.

Tong Xia, Jing Han, Lorena Qendro, Ting Dang, and Cecilia Mascolo. Hybrid-edl: Improving evidential deep learning for uncertainty quantification on imbalanced data. In *Workshop on Trustworthy and Socially Responsible Machine Learning, NeurIPS 2022*.

Tong Xia, Jing Han, Lorena Qendro, Ting Dang, and Cecilia Mascolo. Uncertainty-aware covid-19 detection from imbalanced sound data. *arXiv preprint arXiv:2104.02005*, 2021.

Han Xu, Xiaorui Liu, Yaxin Li, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. *arXiv preprint arXiv:2010.06121*, 2020.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *Proceedings of the 26th International Conference on World Wide Web*, pages 351–360. International World Wide Web Conferences Steering Committee, 2017.

Shuochao Yao, Yiran Zhao, Huajie Shao, Aston Zhang, Chao Zhang, Shen Li, and Tarek Abdelzaher. Rdeepsense: Reliable deep mobile computing models with uncertainty estimations. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4): 173, 2018a.

Shuochao Yao, Yiran Zhao, Huajie Shao, Chao Zhang, Aston Zhang, Dongxin Liu, Shengzhong Liu, Lu Su, and Tarek Abdelzaher. Apdeepsense: Deep learning uncertainty estimation without the pain for iot applications. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 334–343. IEEE, 2018b.

Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1308–1317, 2017.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

Gengyan Zhao, Fang Liu, Jonathan A Oler, Mary E Meyerand, Ned H Kalin, and Rasmus M Birn. Bayesian convolutional neural network based mri brain extraction on nonhuman primates. *Neuroimage*, 175:32–44, 2018.

Xiao Zheng, Wanzhong Chen, Yang You, Yun Jiang, Mingyang Li, and Tao Zhang. Ensemble deep learning for automated visual classification using eeg signals. *Pattern Recognition*, 102: 107147, 2020.

Miankuan Zhu, Jiangfan Chen, Haobo Li, Fujian Liang, Lei Han, and Zutao Zhang. Vehicle driver drowsiness detection method using wearable eeg based on convolution neural network. *Neural computing and applications*, pages 1–16, 2021.