Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Fairness Through Uncertainty

Theo Stephens Kehoe, BAI

April 15, 2024

A Dissertation submitted in partial fulfilment

of the requirements for the degree of

Master of Science in Computer Science

Fairness Through Uncertainty

Theo Stephens Kehoe, Master of Science in Computer Science

University of Dublin, Trinity College, 2024

Supervisor: Andrea Patane

Abstract As machine learning models become more prevalent in our lives and take on critical decision-making roles across various sectors of society, ensuring their decisions are not only accurate but also fair has become ever more crucial.

This paper focuses on individual fairness in Bayesian Neural Networks (BNNs). Individual fairness ensures that similar individuals receive similar outcomes by a model. Specifically, we undertake the definition of $\epsilon - \delta$ individual fairness. BNNs offer advantages over deterministic neural networks due to their ability to quantify uncertainty and effectively handle smaller datasets; a capability that has led to their adoption in critical fields such as medicine.

In this paper, we construct a fairness regularisation method for BNNs by transferring techniques from adversarial robustness training and employing the Fair-FGSM algorithm [17] for generating similar inputs. This approach draws on existing research that highlights the similarity between adversarial robustness and individual fairness definitions. The regulariser is designed to be simple, facilitating its integration into existing training procedures without extensive modifications. We also introduce a simple metric for measuring and comparing $\epsilon - \delta$ individual fairness models in an intuitive manner, the *Threshold-Fairness* metric.

Through our experimentation on various model architecture sizes and similarity metric parameters, encompassing a total of three-thousand models, we can attest that our devised regulariser is effective at improving the individual fairness of a BNN. However, due the to fairness-accuracy trade-off, there is a small degradation of model accuracy imposed by the regulariser. Additionally, we empirically evaluate that there is a relationship between the set of non-protected ϵ parameter values and the effectiveness of the regulariser at improving fairness. We hope that our devised regulariser acts a starting point for more sophisticated and adaptable individual fairness mechanisms, and that these findings will serve as a foundational piece for future research into individual fairness in BNNs.

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Theo Stephens Kehoe

April 15, 2024

Acknowledgments

This work would have never been achievable without the guidance and support of my supervisor, Dr Andrea Patane. I am eternally grateful that I was given the opportunity to contribute to such a relevant and important field in Computer Science. Your relaxed and understanding approach made a significant difference, and I deeply appreciate the freedom and encouragement you offered during my time researching with you. Thank you.

To all my friends who I dearly love, thank you for not only supporting me throughout my college years, but also giving me unforgettable memories full of laughter and happiness.

To my family, congratulations for getting me to the finishing line. I would never be here without you. Your love and support have molded me into the person I am today.

Theo Stephens Kehoe

University of Dublin, Trinity College

April 2024

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction and Motivation

Artificial Intelligence (AI) and Machine Learning (ML), a subset of the former, has transformed the landscape of technology and society. ML models have enabled systems to learn complex relationships from data and make accurate decisions with minimal human intervention. With the vast amounts of data generated by the internet and modern technology, these models have the capability to aggregate, analyse and identify patterns in large datasets within reasonable time frames - a task far beyond the capacity of any human team. Its effectiveness and versatility in tackling complex problems has been utilised in almost every sector of society. In healthcare, ML has contributed to improving diagnostics, personalising treatment plans, and enhancing patient care through predictive analytics and disease detection [41][35]. ML has also found use in finance [16], competitive gaming [21], music recommendation [58], computer networking [11], and construction [20], just to name a few. State of the art AI chat bots such as ChatGPT can even perform code analysis and bug detection using ML [48]. The application of these models is extensive and it is not an understatement to say it has become one of the cornerstones of modern technology.

With that said, the impact of these models on both the individual and at the societal level is substantial [55]. Numerous studies have shown that these models may risk perpetuating various forms of discriminatory bias found within data and their design [44][13][54]. Accounts of biased AIs and ML systems is significant. For instance, it was noted by a Harvard professor that AI-driven search engines routinely push predatory credit lending on those with African-American sounding names [1]. Recruitment ML algorithms have shown preferences for men over women [15], and facial recognition software has shown discriminatory behaviour towards black people [2]. It is therefore crucial that these models are not only accurate but *fair* in their decision making processes.

The study of fairness within ML encapsulates research into evaluating and enforcing fair decisions by such models towards certain groups or individuals with protected attributes. Protected attributes include but are not limited to gender, race, sexual orientation, nationality, and disability. It is a fast developing research area, with many methods deployed already to define and enforce fairness in a multitude of ML models [44]. Unfortunately, however, there is currently no 'silver bullet' for the issue of defining fairness in ML. Fairness, and what it means to make a 'fair' decision, is largely based on the context of the decision-making problem. As such, the concept of fairness is inherently subjective and can vary widely depending on cultural, social, and individual perspectives. Consequently, we are tasked with the complex task of navigating this subjectivity to develop models that are not only technically sound but also ethically aware. Despite the absence of this 'silver bullet', the ongoing pursuit of fairness in ML is a multidisciplinary task, encompassing ethics, law, sociology, and computer science, to come together to develop *systems* to enforce fairness. With this in mind, we are motivated to hopefully contribute **a piece** of the puzzle to furthering the progression of fair ML models.

As it stands, there is very little research into methods to define or improve fairness for uncertainty aware ML models, such as Bayesian neural networks (BNNs). Uncertainty aware, or more precisely, stochastic ML models have the advantage over deterministic models in that they can quantify and incorporate uncertainty directly into their decision making processes. This allows them not only to provide predictions but also measures of confidence in those predictions. This ability to assess and communicate uncertainty can lead to more informed and transparent decision making, potentially reducing the risk of overconfidence in inaccurate predictions. Considering these benefits and the existing gap in research on fairness within BNNs, we are motivated to investigate potential mechanisms to enhance fairness in these models.

1.2 Research Objectives

In current literature, ML fairness definitions can be categorised into group fairness and individual fairness [18]. Group fairness aims to treat similar individuals similarly on a group level by enforcing a form of *statistical parity* on each group's statistics. For example, having the same false positive rate between two groups. Individual fairness, however, aims to treat similar individuals by assigning them similar outcomes, as defined by [18], the authors who originally defined individual fairness. Although the former is generally easier to compute and more widely adopted, the latter makes more intuitive sense. Individual fairness also moves beyond the limitations of group characteristics, offering a fairness notion that does not rely solely on group-based attributes. This approach can be particularly beneficial in contexts where group definitions are fluid, overlapping, or not fully representative of individual identities, scenarios where group fairness falls short [10]. We describe these definitions in more detail later in Section 2.2 of our literature review. In essence, individual fairness is a more flexible and consistent definition of fairness compared to group fairness, hence we will be adopting a form of the definition for our project.

Recent work has drawn parallels between the definitions of individual fairness and adversarial robustness, and has exploited this parallel to transfer techniques to measure individual fairness in BNNs [17]. Adversarial robustness is an area of ML that attempts to make models more resistant to adversarial inputs; inputs that have minimal perturbations compared to correctly classified inputs, but get completely misclassified by the model.

We intend to go a step further than [17] and utilise their findings to further explore and harness this parallel by transferring techniques from adversarial robustness to devise a mechanism to *improve* individual fairness in BNNs. We then hope to investigate the devised mechanism's applicability and effectiveness. We can thus summarise our project's objectives as follows.

- Develop a mechanism to improve individual fairness within BNNs at training time by transferring and adapting adversarial robustness regularisation procedures to the context of individual fairness.
- Investigate and provide insight on the effectiveness of the devised mechanism on improving individual fairness by comparing to a standard BNN.
- Investigate and discuss any effects the devised mechanism may have on the predictive performance of a BNN.
- Empirically evaluate the overall base performance of the devised mechanism and discuss its applicability for improving individual fairness within BNNs.

1.3 Outline of Approach

In this paper, we develop a mechanism to improve $\epsilon - \delta$ individual fairness, as defined by [30], by making use of the *Fair-FGSM* algorithm developed by [17] to measure individual fairness for BNNs. In addition to this, we adapt the adversarial robustness training regulariser developed in [24] to our context of individual fairness in BNNs. By combining the Fair-FGSM algorithm with our adapted regulariser, we create an individual fairness regulariser for BNNs. This regulariser can thus improve the individual fairness for a BNN at training time. We also develop a simple metric to allow for easy comparison of the individual fairness levels of two models, the *Threshold-Fairness* metric. This metric is designed to be more intuitive and easier to comprehend than the existing maximum difference metric used for measuring individual fairness of BNNs.

Proceeding this, we rigorously test our regulariser through a series of experimental trials, testing different BNN architecture sizes and parameter values. We train all our models on the Adult dataset [8], a benchmark dataset for fairness tasks. We thus observe the effect of the regulariser on both the individual fairness levels and predictive performance of a BNN, as well as the influence of parameter values may have on the regulariser's effectiveness.

1.4 Contributions

In this paper, we provide a foundational piece for the investigation and development of individual fairness in BNNs. We develop a simple regulariser that can be used during training time to improve the individual fairness of a BNN; the first of its kind. Through our rigorous experimentation, we can deem that our regulariser has a base performance of being fast and effective with confidence. With that said, we also provide the insight that the regulariser may hinder predictive performance slightly due to the fairness-accuracy trade off, which we outline in detail in Section 2.2.4. Additionally, we describe the relationship between the values of the set of non-protected ϵ values, a parameter assigned to the regulariser, and its effectiveness in enhancing $\epsilon - \delta$ individual fairness of a BNN. It is with our hope that these contributions fill the current gap in literature, and act as an initial starting point that will inspire further developments in this crucial area of ML fairness research.

1.5 Report Outline

We structure our paper as follows.

- **Chapter 2 Literature Review.** We review existing literature on biases in AI and ML, as well as the definitions of fairness, particularly focusing on individual versus group fairness, and the existing methods for both measuring and improving fairness under these definitions in ML models.

- **Chapter 3 Formalisation.** We delve into the core technical concepts necessary for understanding our devised methodology. In this, we include a formalisation on neural networks (NNs), BNNs, and the definition of $\epsilon - \delta$ individual fairness.

- **Chapter 4 Methodology.** We describe the development of our fairness regulariser and the Threshold-Fairness metric. We detail our adaptation of adversarial robustness training procedures to our context of enhancing fairness in BNNs, as well as the use of Fair-FGSM.

- **Chapter 5 Experiments.** We outline our experimental setup. Here we detail the packages and tools used, the parameters and architectures tested, and the metrics collected for assessing fairness, accuracy, and uncertainty. In addition, we give a detailed account of the trials conducted to test our developed regulariser, as well as code snippets.

- **Chapter 6 Results.** Presents our findings from the experimental results. Through the observation of our produced visualisations of the data, we describe how the regulariser influenced the individual fairness and predictive performance of the BNNs. We also describe the observed relationship between the regulariser's parameters and its performance.

- **Chapter 7 Evaluation and Limitations.** We critically evaluate the strengths and weaknesses of our findings and discuss its limitations.

- **Chapter 8 Conclusion and Future Work.** We summarise the main insights and contributions of our paper, as well as the potential areas for future research to be conducted.

Chapter 2

Literature Review

In this chapter, we review the existing literature on fairness in machine learning. We examine the key sources of bias and the necessity to combat them. We then explore the existing definitions of fairness within ML literature, primarily focusing on individual fairness and the methodologies developed to both measure and evaluate it in deterministic NNs and BNNs.

2.1 Bias in AI and ML

As ML algorithms become more dominant in decision-making processes across society, the damage of biases present in the training data and algorithms' designs become more apparent and challenging to reverse [61]. These biases lead to individuals or groups being unfairly mistreated by such systems. To gain scope of the effect of these biases, we will first outline the main (non-exhaustive) sources of bias that have been in discovered in current ML literature.

2.1.1 Sources of Bias

Algorithmic bias is when bias is not present in the input space and is added purely by the algorithm [6]. The choice of optimisation functions, regularisations, the consideration of applying regression models to subgroups, and the use of statistically biased estimators in the algorithmic design can all lead to biased outcomes [44].

Historical bias is the already existing bias and socio-technical issues in the world which can be present in the data generation process, even given a perfect sampling and feature selection [60]. One of the most cited examples of this is that of Amazon's failed recruiting algorithm, which would discriminate against women due to the historical dominance of men hired in the tech industry [15]. Other examples of this include a 2018 image search that would produce fewer female CEO images due to only 5% of Fortune 500 CEOs being women, causing the system to be biased towards men [60].

Measurement bias is bias inherited from how we choose, utilise and measure specific features [60]. A prominent example of this type of bias was observed in the recidivism risk prediction tool COMPAS, in which it used prior arrests as a proxy variable for how likely an individual would recommit a crime. This was observed to be inappropriate, due to the fact that minority communities are policed more frequently, so they have higher arrest rates. This would lead to the conclusion that an individual a part of a minority group is considered more dangerous, which is incorrect as there is a difference in how these groups are treated and assessed [60].

Representation bias is bias that arises from the non-random sampling of subgroups [44]. For example, if smartphone data is collected in order to inform the design of city transportation system, individuals over the age of 65 years old will be under-represented, which could lead to services providing for the needs of the older population being inaccessible [14].

Population bias is bias that arises when a user population of a platform pertains statistics, demographics, representatives and characteristics different to the original target population [52]. For example, in a 2007 survey on college students it was found that Hispanic students were significantly less likely to use Facebook and more likely to use MySpace. On the other hand, White, Asian and Asian-American students were significantly less likely to use MySpace and more likely to use Facebook [25].

2.1.2 Necessity of Combating Bias

The necessity for implementing systems that mitigate bias and enhance fairness in ML systems is evident in both literature and society. Organisations around the world have begun advocating for the address of bias in artificial intelligence systems. CNN, for instance, has noted modern large-language models such as ChatGPT and Microsoft's chat-bots can be racist, sexist and creepy [64]. The Washington Post have described how modern AI image generators can amplify 'outdated Western stereotypes' [50]. The New York Times even questions the diversity issue underpinning the tech industry, which has contributed to biased systems, saying that it is a problem 'tech companies are reluctant to acknowledge' [46].

Evidence of the effects are widespread across society. In education, it was found algorithms predicting six-year college graduation generally have higher false positive rates for White students and higher false negative rates for Latino students [5]. In the medical field, a widely used algorithm in U.S. hospitals used to allocate health care to patients was found less likely to refer black people than white people, who were equally sick, to programs that improve care for patients with complex needs [51]. A 2022 study found that a state-of-the-art robot trained by AI was more likely to associate black men with criminals, or women being homemakers. It was concluded by researchers that use of the robot would only amplify "malignant stereotypes" that fuel racism and misogyny, and they issue a call to justice, imploring that "the Robotics, AI, and AI Ethics communities to collaborate in addressing racist, sexist, and other harmful culture or behavior relating to learning agents, robots, and other systems."[28].

The addressal of these issues is in its infancy and is fast developing, with the EU only introducing the first comprehensive AI regulation act in the world in 2023 [49]. These examples and calls to action illustrate the need for more research into developing systems and methodologies to reduce bias in AI and ML systems.

2.1.3 Developing Systems to Combat Bias

To combat these various forms of bias, it is imperative that systems are developed whereby there is a form of bias mitigation at every stage of the process. An argument can be made that as historical bias is one of the more prominent forms of bias affecting ML algorithms, attention should only be directed towards identifying and neutralizing these biases in the data before they are used to train models. And they would be correct, to a degree. However, a lack of in-built bias mitigation in terms of a regulariser or a fairness module can result in unforeseen biases to persist or even be amplified by the training process. In addition, biases hidden within the dataset can be made apparent during the training process. For example, an individual's address being used as a proxy variable to infer the religion or race of the individual, such as discussed in measurement biases in Section 2.1.1. We should not view bias mitigation and fair models as a one-time correction, but as an *ongoing process* that accompanies the model throughout its entire life-cycle. Integrating fairness techniques directly into the model architecture or training algorithm, such as fairness constraints and regularisers, ensures that the model actively works to counteract bias. As this is the case, we are keen to state that we intend to contribute to development of a *piece* of the system, rather than designing the entire system.

2.2 Defining Fairness

To measure and then improve fairness in a ML system based on a notion of fairness, it must first be clearly defined what it means to make a fair decision. This is where most complexity and issues arise in literature, as there is no strict objective definition of what it means to be “fair”. After all, what qualifies as fair can vary between contexts and remains somewhat subjective. The problem of defining fairness stems from sociology and philosophy; it is not a problem found only in computer science. Thus the solution hinges on the developments made in those fields. That being said, broad general definitions still exist within ML literature. As explained in our introduction, the definitions can be categorised into two main types: group fairness and individual fairness. We will explore these definitions by giving an overview of both and discussing their attributes. We will then briefly discuss the future and apparent difficulties of defining fairness, as well as explore the trade-off between fairness and accuracy.

2.2.1 Group Fairness

Group fairness, or better known as statistical parity, is a umbrella term encapsulating a few different mathematical definitions of fairness. These definitions primarily focus on ensuring that minority members should be treated at approximately the same rate as the majority members [59][6]. Definitions, such as equal opportunity and treatment equality, require that statistical properties remain the same across protected and non-protected groups [6][18]. Equal opportunity, for example, states that individuals part of the protected or unprotected groups should have equal true positive rates. Treatment equality, on the other hand, states that the ratio of false negatives and false positives is the same for both protected groups.

In current literature, group fairness is the most widely adopted notion of fairness. There are various methods for both verifying [4][59][7] and improving [27] [57] [56] group fairness for deterministic neural networks. There are some methods for controlling group fairness for BNNs [67] and Bayesian classifiers [27], but more methods still need to be developed for the former. With that said, in this paper we will be devising a method for improving individual fairness in BNNs as opposed to group fairness.

Group fairness appears desirable in many instances, but there are some drawbacks to the definitions. For instance, a system that is fair under statistical parity can produce outcomes that are blatantly unfair from the individual's point of view. Such a system would also produce reduced utility and still allow targeting of subsets of individuals while maintaining fairness [18]. A problem often raised with group fairness definitions is that they are also often suited only for coarse-grained protected groups. Group fairness does not account for biased outcomes against individuals who are at the intersection of multiple types of discrimination or are a part of groups which are not (yet) defined that may need protecting [10]. It, in essence, does not enforce that individuals that are similar by all means except the protected attributes receive similar outcomes by the system.

2.2.2 Individual Fairness

Individual fairness encapsulates fairness definitions that ensure individuals who are 'similar' with respect to the classification task receive similar outcomes [18][10]. The most demanding aspect of this definition is to define what it means for individuals to be 'similar'. Most utilise a similarity or distance metric that is calculated on pairs of individuals. In the case of a primitive distance metric, a pair of individuals are considered similar if their distance from each other is within a specified threshold. If this is the case, you would expect the outputs by a given fair model to also be similar; that is, the distances between their outputs are also within a specified threshold. If the model violates this expectation, then we consider the model *unfair*. In most contexts, it is challenging to define the similarity metric due to the lack of *availability of* or *access to* such a metric. Similarity metrics and their definitions are discussed later in Section 2.3.1.

Individual fairness appears very desirable, as enforcing it can create fair decisions for individuals and avoid the problems related to that of group fairness [18]. However, the major pitfall of individual fairness is that even if a perfect task-relevant similarity mapping over the individuals is obtained for a training set, there is a lack of a way of generalising to a new, unseen set of individuals. This problem was even noted by the authors of [18], who initially formulated individual fairness. The context-specific nature of individual fairness also requires a policymaker with domain-specific knowledge to define the similarity or distance metrics, unlike group fairness which can be applied easily to most models. It is also generally computationally infeasible to calculate the similarity metric over all pairings of individuals and their outcomes in a dataset, especially if the dataset is large. These problems have hindered the practical application of individual fairness measures, despite their theoretical appeal over group fairness. Researchers have noted their regret that individual fairness has been neglected in favour of group fairness, because of how individual fairness is "intuitive and captures aspects that group fairness does not handle" [39].

Fairness, and what it means to be fair, is a very complex and nuanced concept that has been subject to debate by philosophers and other humanities scholars for millennia. From a philosophical background, the flattening of a nuanced, complex moral concept to a small cluster of statistical definitions may seem worrying. It would be in this case arguable that having task-specific metrics that are derived from some informed domain-specific policy maker is preferable; that maybe the context-nature of individual fairness is not a hindrance, but a necessity. As such, as well as given the strengths previously outlined, we are motivated in this paper to contribute to the pursuit of making the definition more practical through our devised methods.

2.2.3 Progressing the Definition of Fairness

In current literature, there is a general notion that there is an inherent conflict between that of group and individual fairness definitions. New literature investigating this has suggested that this apparent conflict is actually a misconception that stems from the conflict of *specific variants* within each definition [10]. The literature argues that specific combinations of variants of each definition can reverse the apparent conflict, such as a *group-adjusted* variant of individual fairness. The general notions of fairness defined in ML literature is dissected in great detail, investigating whether there is a distinct difference in principle of the two definitions and analysing their attributes that are a part of ideas and theories found within various philosophies. We encourage the interested reader to read the paper [10] as it provides critical analyses of fairness definitions in ML.

2.2.4 Trade Offs Between Fairness and Accuracy

As models are optimised to make the most accurate predictions possible, they may inadvertently utilise and perpetuate biases present in the training data to increase prediction accuracy against test sets. Explicitly defining fairness constraints on models will restrict them from fully leveraging all available information in the data, biased or unbiased, reducing potential accuracy. The trade-off highlights a fundamental problem in machine learning; balancing the pursuit of high predictive performance with the moral and ethical responsibility to treat individuals and groups equitably.

The research into this fundamental trade-off is substantial [45]. Multi-objective frameworks have been constructed to optimise both (group) fairness and accuracy in logistic regression and decision tree classifiers [62][63], as well as “learning to defer” methods for deterministic neural networks [42]. We outline this trade-off as a “keep-in-mind” for the reader, since we will be mainly focusing on the effect of our fair training method on the fairness of a BNN. However, we will outline parameters in our methods that are able to adjust the model between fairness and accuracy.

2.3 Measuring Individual Fairness

In this section, we examine the methods for measuring individual fairness. We begin first by defining similarity metrics used within literature and the established methods for evaluating. We then explore the link between individual fairness and adversarial robustness and discuss how it is utilised for measuring fairness in BNNs.

2.3.1 Similarity Metric

As previously discussed, to measure individual fairness a similarity or distance metric is utilised. A similarity metric quantifies the degree of similarity between two individuals or inputs based on their attributes. Similarity metrics are task-specific and should be well-defined enough that they reflect meaningful comparisons between the inputs. For instance, in a hiring algorithm, the similarity metric might consider attributes such as skills, experience, and education level to determine how similar two job applicants are. The metric will also include a sensitivity to the protected attributes, such as gender or race, to ensure that it does not inadvertently perpetuate bias.

Various similarity metrics have been proposed within literature. A simple metric is that of a Lipschitz condition, which bounds the decision-making function with a Lipschitz constant with respect to a task-specific distance metric [18]. It is a metric that follows that of the individual fairness definition; the distance metric tending to be geometric distances, such as a ℓ_p metric. The disadvantage to this is that the metric may not capture complex relationships between protected and non-protected attributes. In contrast, the Mahalanobis Distance is a metric that considers the covariance among variables to measure the distance between two points in a multidimensional space, which is effective for when protected and non-protected attributes are correlated [66]. It is, however, more complicated in nature.

More novel approaches include that of learning a similarity metric through a causality-based approach to fairness [37], where a decision is considered fair if it is the same in both the actual world and a counterfactual world where the individual belonged to a different demographic group. One approach approximates a similarity metric using human judgements [29]; a model that assumes that it has access to a human arbiter, who can answer a limited set of queries concerning similarity of individuals for a particular task. This, however, assumes that the arbiter is "free of explicit biases", which may not be sufficient as implicit biases can still have a significant effect on decisions [19].

2.3.2 Established Methods

In current literature, there are various approaches to measuring and verifying individual fairness for deterministic NNs. Certifair [9], proposes a method to certify individual fairness of feed-forward NNs using a range of metrics, including the Mahalanobis distance. More novel approaches include RobustFair [40], which introduces a fairness evaluation technique in deep neural networks using a fairness confusion matrix guided by adversarial perturbation. For full insight into the established methods for measuring fairness we encourage the interested reader to explore [44], as it gives a comprehensive view of fairness definitions and methods to improve them in machine learning models. These methods, however, are all for deterministic networks and are inapplicable to BNNs, which we will be investigating in this paper. We build upon the research of [17], who outlines an adversarial approach to measuring individual fairness in BNNs.

2.3.3 Measuring Using Adversarial Robustness

When examining the definition of individual fairness, it is not hard to draw comparisons to that of adversarial robustness. Adversarial robustness refers to the verification that small perpetuations to a correctly-classified input does not cause a model to misclassify the input [34]. Individual fairness, in a similar manner, examines that the changing of a sensitive attribute for an input does not cause misclassification. The definitions differ in that of their scopes; adversarial robustness methods tend to analyse *local* robustness of a singular input, while individual fairness examines an equivalent to *global* robustness (i.e. robustness of all inputs). Adversarial robustness also considers average perpetuations in the input, while individual fairness considers the worst case change.

This similarity allows for the adaptation of methods designed to measure and enhance adversarial robustness for use in promoting individual fairness. Recent works have begun exploring and exploiting this similarity; such as developing statistical inferences on individual fairness using adversarial attacks [43] or adapting smoothing techniques from robustness to improve individual fairness for a deterministic model [65].

The authors of [17] utilise this parallel by introducing the *Fair-FGSM* algorithm, an adaptation of the FGSM algorithm [24]. Fair-FGSM, like FGSM, produces an adversarial example for a given individual. However, the pair can be conceptualised as similar inputs in respect to some similarity metric, in which the adversary attempts to maximise the difference in prediction outcome compared to the original input. If the difference in prediction outcome exceeds that of a defined threshold, the prediction is deemed unfair. Through this algorithm and the proposed methods we can evalulate the individual fairness of a deterministic and non-deterministic model. We introduce this formally and describe measuring individual fairness for BNNs in-depth later in Section 3.3.2.

2.4 Improving Individual Fairness

In this section we will discuss proposed methods for improving individual fairness in NNs, as well as the type of procedures they can be categorised into. We will then briefly outline the literature motivating us to take an adversarial robustness approach to improving fairness in BNNs.

2.4.1 Established Methods

As mentioned in Section 2.3.2, there are numerous methods for measuring individual fairness in deterministic NNs. Most papers establishing measuring techniques, such as Certi-Fair, also outline ways of improving fairness. The methods employed can take vastly different approaches, and can be categorised into three processing procedures. Pre-processing procedures manipulate the data before being given to the model, in-processing manipulates the model itself, and post-processing enforces fairness after prediction time [44]. As stated in Section 2.1.3, a full fairness system would incorporate some form of all three types of processes to ensure fairness.

Pre-processing procedures in literature tend to focus on mitigating biases found within datasets due to decisions made by the data curator. Some general techniques have been proposed, such as having labels for data, like having nutrition labels for food, that outline what task the data should be categorised into [26]. Others include preferential sampling, a type of sampling scheme, for making the data discrimination free [32]. Although effective, these methods can be cumbersome and may be impractical in some cases, especially if the dataset is a live dataset.

In-processing and post-processing procedures tend to leverage the individual fairness constraints to ensure fairness. For instance, [9] introduces an in-processing method to compute individual fairness guarantees by over-approximating bounds enforced by metric constraints through encoding the problem as a MLP problem, yielding fairer neural networks. Other methods include fairness regularisers [33] and joint-optimisers to optimise for both fairness and accuracy [22]. However, there still exists methods to ensure fairness without explicitly utilising these bounds. FETA [47], for instance, introduces a counter-example guided post-processing technique to enforce fairness at prediction time, as well as an in-processing technique that uses fairness as an inductive bias during the learning process.

Unfortunately, most devised methods for improving fairness are for deterministic networks and are inapplicable to BNNs, which we intend to inspect. As mentioned, this project will build upon individual fairness methods that adapt adversarial robustness techniques.

2.4.2 Adversarial Methods

As adversarial robustness procedures were originally devised to lower the sensitivity of networks against malicious inputs, as well as the previously discussed link to individual fairness, their procedures offer a good blueprint for mechanisms for improving fairness. Smoothing techniques akin to adversarial robustness have already been adapted for improving fairness in deterministic models [65]. We are motivated by this connection to adapt the adversarial training method of deep neural networks outlined in [24] alongside with [17]'s Fair-FGSM to create a fairness training algorithm of our own. We discuss and provide experimental results of the fair training method developed from this adaptation in Chapters 4 and 5.

2.5 Summary

In this chapter, we provided an overview of the existing literature pertaining to our project. We gave a brief synopsis of the various forms of bias found in data and algorithms and highlighted the significant appeals within both literature and society for action to reduce these biases. We then introduced the two main definitions of fairness within ML literature, group fairness and individual fairness. A dissection of each definition was given, where we outlined the advantages and pitfalls of each. Importantly, however, we highlighted the intuitive nature and advantages individual fairness has over group fairness. More specifically, how group fairness fails to ensure fair decisions in certain contexts whereby individual fairness does not. We then argued that the context-specific nature of individual fairness is not a hindrance but a necessity, and underscored literature that analyses ML fairness in-depth from a philosophical standpoint and provides insights on how to improve our definitions.

We then explored established methods for measuring individual fairness and the similarity metrics utilised. The intrinsic connection between adversarial robustness and individual fairness was also introduced, as well as the Fair-FGSM algorithm, which is a fundamental component to our devised methods.

Finally, we explored established methods for improving individual fairness in neural networks, as well as the type of procedures they can be categorised into. We also underscored how they are generally inapplicable to BNNs. We then finished by outlining how the connection between individual fairness and adversarial robustness lends us viable blueprints for mechanisms to improve fairness. This, coupled with the lack of research into individual fairness for BNNs, serves as the primary motivations for this paper.

Chapter 3

Formalisation of Preliminaries

In this chapter, we will introduce and formalise the core concepts essential to our constructed fairness mechanism, which we introduce in Chapter 4. We start by defining the fundamentals of a basic deterministic neural network, followed by an exploration of how BNNs differ from them. In this discussion, we will give an overview of Bayesian inference and briefly outline the inference method employed in our paper. Lastly, we will delve into how we define individual fairness within the context of BNNs and the methodology we adopt to assess it.

3.1 Neural Networks

Advancements in the field of ML have led to the rise of neural networks (NNs), the cornerstone of modern ML algorithms. NNs have the ability to learn complex relationships in data and produce highly accurate models, compared to more standard algorithms. Their structure draws inspiration from the biological neural networks seen within human and animal brains, and as such, they attempt to mimic the ability to recognise patterns and relationships in the world that is reminiscent of human cognition.

NNs are comprised of interconnected nodes, typically referred to as neurons, in a series of layers. A neuron can be viewed as a function with an associated weight and bias, by

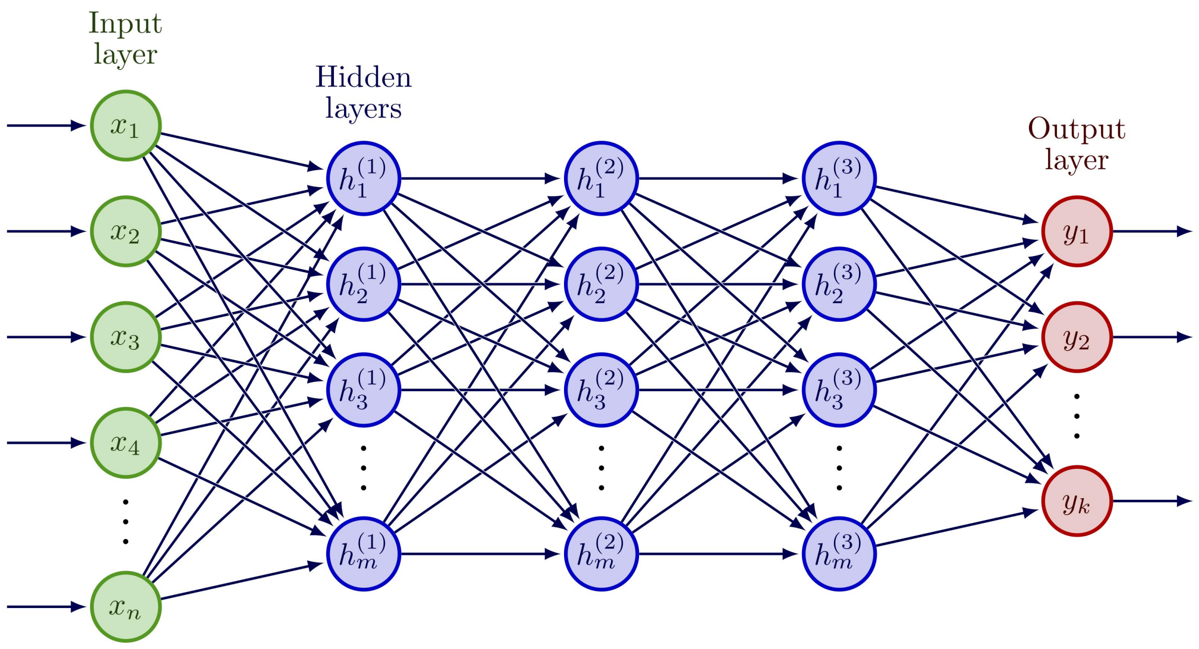


Figure 3.1: Structure of a neural network. Note the mathematical symbols we use do not align with the ones shown. Credit to Tex.com.

which its output is then used as the input in the following layer. The level of influence or strength of the neuron has on the following layer is determined by its weight. The output is controlled by an activation function, which applies a threshold. If the output exceeds this threshold, it is “activated” and its output will propagate to the next layer.

The value of the weight of a neuron is adjusted through a process known as learning, which is typically achieved through backpropagation or gradient descent [23]. During this learning phase the model’s performance is calculated using a loss function, referring to the difference or ‘loss’ between the model’s predictions and true values of the test set. The values of the weights are then adjusted accordingly in order to minimise the loss function until a local, or more ideally global, minimum is reached.

An example of the structure of a neural network is given in Figure 3.1. The number of layers in a NN can be referred to as its depth, while the number of neurons per layer as its width. The first layer is generally referred to as the input layer and the last as the output layer. The layers in between are subsequently referred to as the “hidden” layers.

Generally, most NNs are *feed-forward* NNs. This refers to the fact that the neuron signals are outputted in one direction towards the output layer and do not influence layers preceding the neuron or the neuron itself. Neural networks with this type of feed back loop are known as recurrent NNs [23]. For the purposes of this paper, we only investigate feed-foward NNs.

3.1.1 Formalising

To formalise NNs as a whole, we first define the transformation from input to output for a layer as

$$y_h = g(w_h x_h + b_h) \tag{3.1}$$

whereby w_h is the weight vector, b_h is the bias, g the hidden layer activation function and x_h is the input vector for a specified layer h s.t. $h \in L = [1, 2, 3, ..., N-1]$ for a neural network with width N. The activation function g in most modern neural networks is the *rectified linear unit* or ReLU for short [23].

The output layer, i.e. $h = N$, can therefore be defined in a similar manner, substituting out g for g_o.

$$y_N = g_o(w_N x_N + b_N) \tag{3.2}$$

The key difference here being that the activation function for the output layer can be altered depending on the context. In most cases, such as for classification tasks, the sigmoid function is used when a probability output $(0, 1)$ is needed. In this paper, we will be utilising the **Softmax** activation function [23]. Formally, this is defined as

$$\alpha(z_k) = \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.3}$$

whereby $\alpha : \mathbf{R}^C \longrightarrow (0,1)^C$, vector $\boldsymbol{z} = w_n x_N + b_N$ and C being the number of classes the input space can be classified into. Thus, the softmax function allows us to classify the output into multiple classes depending on which attains the highest probability outputted by the network. Although not entirely necessary for this paper, as we use only a single binary classification, it allows us to extend our methods to multi-classification networks.

With this in mind, we can now fully define a trained feed-foward NN simply as a function mapping f with an associated weights matrix $W \in \mathbf{R}^z$ such that $f^W : X \longrightarrow Y$. Where $X \subseteq \mathbf{R}^n$ and $Y \subseteq \mathbf{R}^C$, n being the number of inputs and z being the number of weights. Thus, for some input $x \in X$, $f^W(x)$ represents the probabilities of it being attributed to a certain class.

3.2 Bayesian Neural Networks

So far, we have been discussing the structure of *deterministic* NNs. The values assigned to weights and outputs are concrete scalar values and do not atone for uncertainty in their predictions. Uncertainty can arise from uncertainty already included in the data or uncertainty in the predictions due to the lack of expressiveness and/or transparency of a NN's inference model. Deterministic NNs do not deliver any means of quantifying their certainty and suffer from under or over confidence in their predictions. For this reason, researchers have developed methods for quantifying and embedding uncertainty into deep learning algorithms; one such method being the application of Bayesian inference.

3.2.1 Bayesian Inference

In section 3.1, we chose scalar values for our model's weights which were adjusted through the use of a learning algorithm such as backpropagation. In contrast, a Bayesian NN infers posterior distributions over the weights w and applies Baye's Theorem:

$$P(w|X) = \frac{P(X|w)P(w)}{P(X)} = \frac{P(X,w)}{\int_w P(X,w')dw'} \propto P(X|w)P(w) \tag{3.4}$$

where $P(w|X)$ is the probability of attaining weight w given evidence of X (posterior distribution), $P(w|X)$ is the probability of X given weight w (likelihood), $P(w)$ is the

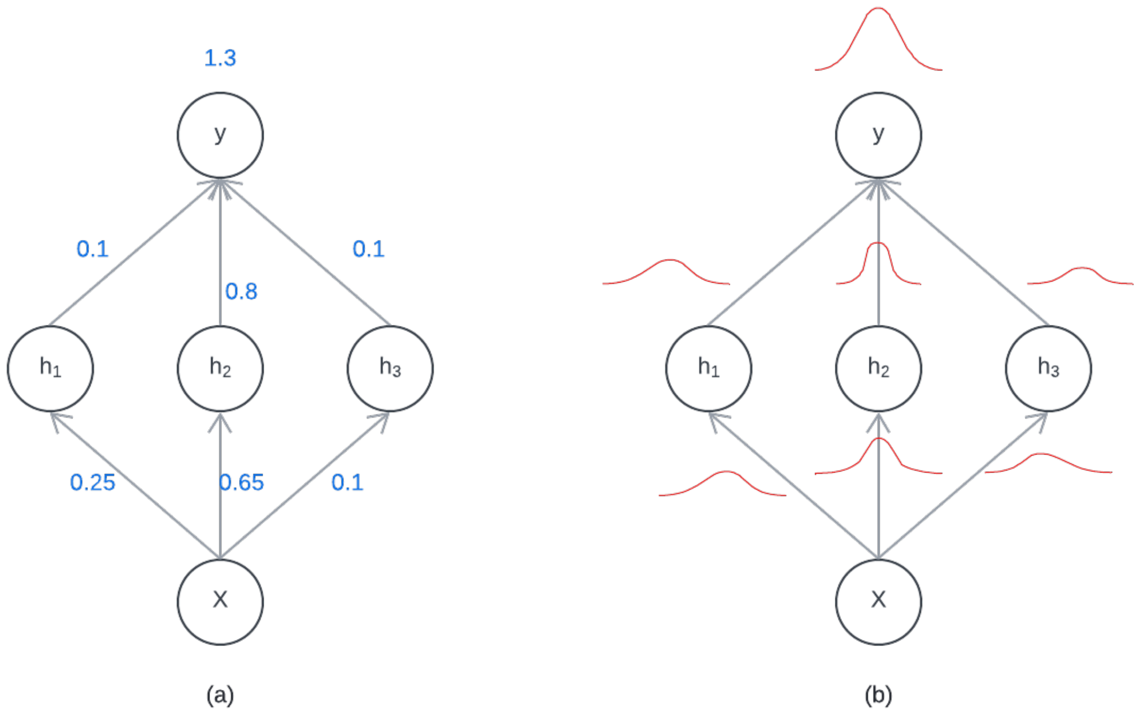


Figure 3.2: DNN vs BNN. (a) is a simple DNN and (b) is a BNN with stochastic weights.

prior likelihood of the weight (prior distribution), and $P(X)$ is the prior likelihood of evidence X (the marginal distribution). In our case, the evidence X can be our training dataset $D = \{(x_i, y_i)\}_{i=1}^{M}$ for a given size M. We should note that Bayesian NNs can include assigning prior distributions to *any* of the model's parameters, not being limited to just the weights. We can therefore learn from the dataset by iteratively applying Bayes theorem to the posterior distributions for a given parameter θ, such as the weights, as more evidence is discovered through the training data. The difference between a DNN and a BNN is illustrated in Figure 3.2.

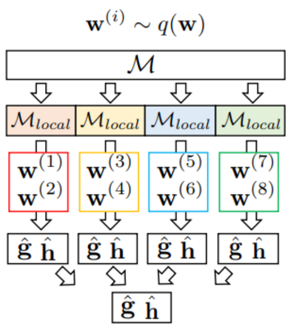
Once these posterior distributions are estimated, we can predict an output y^* for a unseen input x^* using Full Bayesian Analysis:

$$P(y^*|x^*, D) = \int P(y^*|x^*, \theta)P(\theta|D)d\theta \tag{3.5}$$

Applying Bayesian inference over the network's parameters allows us to obtain a good

Algorithm 1: Variational Online Gauss Newton (VOGN)

```
1: Initialise $\boldsymbol{\mu}_0$, $\mathbf{s}_0$, $\mathbf{m}_0$.
2: $N \leftarrow \rho N, \tilde{\delta} \leftarrow \tau\delta/N$.
3: repeat
4:    Sample a minibatch $\mathcal{M}$ of size $M$.
5:    Split $\mathcal{M}$ into each GPU (local minibatch $\mathcal{M}_{local}$).
6:    for each GPU in parallel do
7:       for $k = 1, 2, \ldots, K$ do
8:          Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
9:          $\mathbf{w}^{(k)} \leftarrow \boldsymbol{\mu} + \epsilon\boldsymbol{\sigma}$ with $\boldsymbol{\sigma} \leftarrow (1/(N(\mathbf{s} + \tilde{\delta} + \gamma)))^{1/2}$.
10:         Compute $\mathbf{g}_i^{(k)} \leftarrow \nabla_w \ell(\mathbf{y}_i, \mathbf{f}_{w^{(k)}}(\mathbf{x}_i)), \forall i \in \mathcal{M}_{local}$
            using the method described in Appendix B.
11:         $\hat{\mathbf{g}}_k \leftarrow \frac{1}{M} \sum_{i \in \mathcal{M}_{local}} \mathbf{g}_i^{(k)}$.
12:         $\hat{\mathbf{h}}_k \leftarrow \frac{1}{M} \sum_{i \in \mathcal{M}_{local}} (\mathbf{g}_i^{(k)})^2$ .
13:      end for
14:      $\hat{\mathbf{g}} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \hat{\mathbf{g}}_k$ and $\hat{\mathbf{h}} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \hat{\mathbf{h}}_k$.
15:   end for
16:   AllReduce $\hat{\mathbf{g}}, \hat{\mathbf{h}}$.
17:   $\mathbf{m} \leftarrow \beta_1 \mathbf{m} + (\hat{\mathbf{g}} + \tilde{\delta}\boldsymbol{\mu})$.
18:   $\mathbf{s} \leftarrow (1 - \tau\beta_2)\mathbf{s} + \beta_2 \hat{\mathbf{h}}$.
19:   $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \alpha\mathbf{m}/(\mathbf{s} + \tilde{\delta} + \gamma)$.
20: until stopping criterion is met
```


$\mathbf{w}^{(i)} \sim q(\mathbf{w})$
$\mathcal{M}$
$\mathcal{M}_{local}$
$\mathcal{M}_{local}$
$\mathcal{M}_{local}$
$\mathcal{M}_{local}$
$\mathbf{w}^{(1)}$ $\mathbf{w}^{(2)}$
$\mathbf{w}^{(3)}$ $\mathbf{w}^{(4)}$
$\mathbf{w}^{(5)}$ $\mathbf{w}^{(6)}$
$\mathbf{w}^{(7)}$ $\mathbf{w}^{(8)}$
$\hat{\mathbf{g}}$ $\hat{\mathbf{h}}$
$\hat{\mathbf{g}}$ $\hat{\mathbf{h}}$
$\hat{\mathbf{g}}$ $\hat{\mathbf{h}}$
$\hat{\mathbf{g}}$ $\hat{\mathbf{h}}$
$\hat{\mathbf{g}}$ $\hat{\mathbf{h}}$

Learning rate	α
Momentum rate	β_1
Exp. moving average rate	β_2
Prior precision	δ
External damping factor	γ
Tempering parameter	τ
# MC samples for training	K
Data augmentation factor	ρ

Figure 3.3: VOGN Algorithm. Note this has a few changes from the original outlined in [36], such as parallelisation, which is shown on the right. Bottom right table displays VOGN hyper parameters. Credit to [53], who designed this distributed version of VOGN.

idea of the uncertainty associated with the underlying processes in predicting outcomes. We can achieve this by comparing the predictions of multiple sampled model parameterisations of θ. If the differences between models is low, uncertainty is low, and if the difference is high, uncertainty is high [31]. Through this, we can make more accurate predictions even with an average performing BNN, as it is known that aggregating the predictions of a collection of average, yet independent, models can lead to better predictions than the predictions from a single high performance model [12].

3.2.2 Inference Algorithms

As previously stated, learning in BNNs is essentially establishing a posterior distribution from which we can sample from and then make a prediction. This however is easier said than done; although most aspects of Equation 3.4 are easy to compute, computing the integral for the evidence is generally computationally infeasible [31]. It is for this reason that most utilise specialised algorithms. The most common found are that of Markov chain Monte Carlo (MCMC) methods or approximate variational inference (VI). MCMC methods sample from the exact posterior, which can provide more accurate results, but suffer from high variance and long computation time. Variation inference, on the other hand, is faster in computation and is more suitable for large scale networks [68], although providing less accurate samples. For these reasons, we utilise the VI method Variational Online Gauss-Newton (VOGN), which approximates the posterior distribution using a Gaussian distribution [36].

3.2.3 VOGN and Regularisation

As we will be implementing a regularisation method to improve individual fairness, we must note that the posterior distributions are often considered to be soft constraints on the model, making them analogous to regulariation in traditional deterministic NNs. As this is the case, to impose a fairness constraint at training time we just need to manipulate VOGN's computation of estimating the posterior. More specifically, we will be manipulating the prediction made from the model just before the calculation of the loss. Calculation of the loss is shown in line 10 of Figure 3.3. We should note that the algorithm displayed has a few changes compared to the original VOGN algorithm described in [36], but it gives a good idea of the functioning of the algorithm. Details of how the fairness regularisation is conceived and implemented is outlined in Chapter 4.

3.2.4 Predictive Posterior Distribution

As one might expect, the outputs of a BNN are also distributions, as noted in Equation 3.5. We can observe this in Figure 3.2. We can re-write the Equation 3.5 to include the Softmax activation function α in order to formalise the predictive posterior distribution of our BNN.

$$\pi(x) = \int \alpha(f^W(x), W)P(W|D)dW \tag{3.6}$$

where f^W is our function mapping, W is our weights vector, D is our training dataset and $x \in X$. From here, we will refer to the predictive posterior distribution of our BNN as π.

3.2.5 Key Points on BNNs

Before moving on, we must note that *almost any* deterministic NN can be used as the functional model f^W for a BNN [31]. BNNs, or more generally stochastic deep learning models, are not stand alone separate models compared to DNNs. They are considered more of a 'wrapper' around the main functional model used in DNNs in order to give an ability to quantify uncertainity. It allows them to say "I don't know" about a prediction with the ability to give confidences in their answers. This not only helps us from a fairness standpoint, as this uncertainty has led to BNNs to be fairer compared to DNNs [17], but allows us to create more flexible and reliable models that can work well with even small datasets. The ability to work well on small, noisy datasets where uncertainty needs to be quantified is particularly effective for many sectors of society, especially in the medical field. Bayesian methods including BNNs have found use in medical imaging tasks, clinical signal processing, and electronic health records [3].

3.3 Individual Fairness

Previously in Section 2.2.2 we discussed the definition of individual fairness (IF) and its attributes. To recap, IF can be defined as a term that "encapsulates fairness definitions that ensure individuals who are 'similar' with respect to the classification task receive similar outcomes". We can interpret this as meaning that "the distributions assigned to similar people are similar" [18]. By [18], we can formalise this as a general Lipschitz mapping for our BNN function mapping $f_B^W : X \longrightarrow Y$, such that

$$D(f_B^W(x), f_B^W(x')) \leq d(x, x') \tag{3.7}$$

for an output metric D, input metric d and any pair of inputs $x, x' \in X$.

As this is quite general, we can rewrite this equation for two distance metrics with associated thresholds [30]. We will denote ϵ for the threshold to define similar inputs and δ for the threshold that their outputs must not exceed. Therefore, our model can be considered individually fair for a given value of ϵ and δ s.t. $\epsilon \geq 0, \delta \geq 0$ if and only if

$$d(x, x') \leq \epsilon \implies D(f_B^W(x), f_B^W(x')) \leq \delta \quad \forall x, x' \in X \tag{3.8}$$

3.3.1 Similarity Metric

What we classify as similar inputs is at our discretion through how we define d. As we are interested in defining similar individuals based on all aspects *except* their protected attributes (race, gender, etc.), we must define a way to account for both protected and non-protected attributes. As noted in [30], we can intuitively split up our input attributes into disjoint sets $S_1, ..., S_t$ with corresponding ϵ values $\epsilon_1, ..., \epsilon_t \geq 0$. We can therefore assign protected attributes a threshold of ∞, which implies a pair of individuals can differ in any way in their protected attributes and will be considered similar. Non-protected attributes (such as home address, education, etc.) can then be assigned a context specific threshold for what we define to be similar, 0 implying they will be classified as dissimilar if they are different in any way.

With this in hand, we can define our similarity metric d in a similar manner. As we build this paper off [18], we define our similarity metric ζ as a weighted ℓ_p metric such that

$$\zeta(x, x') = \sqrt[p]{\sum_{i=1}^{t} \theta_i |x_i - x'|^p} \tag{3.9}$$

where t is the number of input attributes, θ is the weight of the distance, and p denoting the ℓ_p in use. Through this we assign protected attributes a weighting $\theta = 0$ and non-protected attributes weights depending on their correlation with the sensitive attribute, if any. This captures the fact that the distance between two individuals should not be influenced by the distance between their sensitive attributes.

The choice of the ℓ_p metric will significantly affect the distance calculated. A common and well-known choice would be the ℓ_2 metric, which is the Euclidean distance between two points (i.e. straight line distance). ℓ_∞, on the other hand, measures the maximum absolute difference between the coordinates of a pair of points. It captures the single largest difference among all dimensions.

3.3.2 Measuring Individual Fairness

As mentioned in Section 2.4.2 we will be utilising the methods outlined in [17] to measure IF in our BNNs. We will also be adapting the *Fair-FGSM* method to create a fair training mechanism. Fair-FGSM is defined as

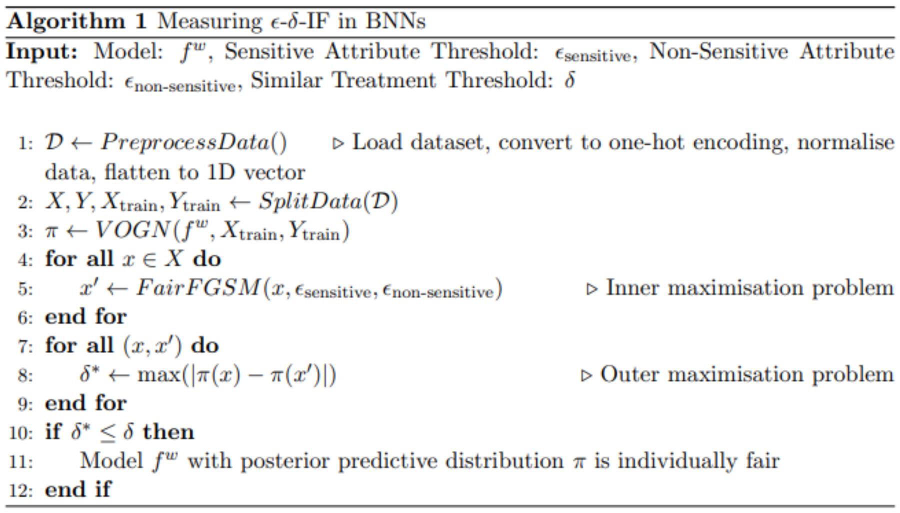
$$x' = x + \epsilon_{fair} \cdot \text{sign}(\mathbf{E}_{P(W|D)}[\nabla_x L(x, W)] \tag{3.10}$$

where ϵ_{fair} denotes our set of ϵ's and L is the loss function. Including our set of ϵ's allows us to generate an input x' for our model that is similar to an input x with regards to its non-protected attributes, but dissimilar in any manner in regards to its protected attributes. We choose the values of ϵ_{fair} by correlating it with the distance weightings defined in the similarity metric in Equation 3.9.

Based on our definition of IF in Equation 3.8 we are only concerned with finding *one* pair of inputs that produce dissimilar outputs. If one such pair exists then the model is considered unfair. The authors of [17] devise a method using Fair-FGSM to find the worst-case change in the model's output when all similar pairs are considered by solving equation 3.11.

$$\delta^* = \max_{x \in X} |\pi(x) - \pi(x')| \tag{3.11}$$

Figure 3.4 shows us the measuring of fairness in algorithmic form. As δ^* represents the maximum difference between two similar individuals for a model, we can consider



```
Algorithm 1 Measuring $\epsilon$-$\delta$-IF in BNNs
Input: Model: $f^w$, Sensitive Attribute Threshold: $\epsilon_{\text{sensitive}}$, Non-Sensitive Attribute
Threshold: $\epsilon_{\text{non-sensitive}}$, Similar Treatment Threshold: $\delta$

 1: $\mathcal{D} \leftarrow PreprocessData()$    ▷ Load dataset, convert to one-hot encoding, normalise
    data, flatten to 1D vector
 2: $X, Y, X_{\text{train}}, Y_{\text{train}} \leftarrow SplitData(\mathcal{D})$
 3: $\pi \leftarrow VOGN(f^w, X_{\text{train}}, Y_{\text{train}})$
 4: for all $x \in X$ do
 5:     $x' \leftarrow FairFGSM(x, \epsilon_{\text{sensitive}}, \epsilon_{\text{non-sensitive}})$    ▷ Inner maximisation problem
 6: end for
 7: for all $(x, x')$ do
 8:     $\delta^* \leftarrow \max(|\pi(x) - \pi(x')|)$    ▷ Outer maximisation problem
 9: end for
10: if $\delta^* \leq \delta$ then
11:     Model $f^w$ with posterior predictive distribution $\pi$ is individually fair
12: end if
```

Figure 3.4: Measuring fairness algorithm. Lines 4-12 encapsulate the main procedure for measuring fairness for a BNN. Credit to [17].

this as a metric for measuring the fairness of our model. However, later in Section 4.3 we construct a simple metric, the *Threshold-Fairness* metric, that takes into account the similarity threshold for the output δ.

Thus we can reformulate our definition of fairness for our BNN model as

$$\max_{x \in X} |\pi(x) - \pi(x')| \leq \delta \tag{3.12}$$

or more simply $\delta^* \leq \delta$ for some context-specific output similarity threshold δ. This, in essence, states that if the maximum of the differences between the outputs of inputs and their respective generated similar inputs is greater than our defined threshold δ, then the model is considered unfair.

Chapter 4

Methodology

In this chapter, we will describe the methodology behind the development of our fair training regulariser. We will also describe our devised fairness metric, the Threshold-Fairness score, designed to provide a clearer insight into fairness levels in relation to a specified similarity threshold δ.

4.1 Definitions

Firstly, we will revisit and fully encapsulate our definition of fairness within the context of a BNN with our similarity metric.

Definition 4.1.1 ($\epsilon - \delta$ Individual Fairness.)**.** *A BNN f^W with predictive posterior distribution π is said to be individually fair with respect to a given value of ϵ and δ s.t. $\epsilon \geq 0, \delta \geq 0$ iff:*

$$\zeta(x, x') \leq \epsilon \implies |\pi(x) - \pi(x')| \leq \delta \quad \forall x, x' \in X \tag{4.1}$$

the ζ function being our similarity metric defined in Equation 3.9.

As noted before, from observing the definition we can see we are only concerned with finding a *single* pair of ϵ-similar inputs that **do not** generate δ-similar outputs for us to classify the model as unfair. Fair-FGSM allows us to generate a similar input for every input, which we will refer to as a *fair example.* By getting the maximum of the differences of the predictive distributions of the inputs (x) and their respective fair examples (x'), we can assert if a model is fair or not. If the maximum difference is over the threshold δ, then the model is unfair.

Definition 4.1.2 ($\epsilon - \delta$ Individual Fairness with respect to maximum difference)**.**

$$\textit{Individually fair} \iff \delta^* \leq \delta \quad s.t. \tag{4.2}$$

$$\delta^* = \max_{x \in X} |\pi(x) - \pi(x')|$$

We can also use this maximum difference, δ^*, as a measurement for how fair our model is. The question now is how to improve the fairness of a given model, or more simply, lower the maximum difference between predictive distributions of pairs of similar inputs.

A possible solution to lowering the difference would be to simply pre-process the training dataset to include the fair examples. This, however, would be cumbersome, as a fair example would be needed to be generated for the entire dataset and then stored, doubling memory usage. It would also only guide the adjustment of parameters to account for fairness once, rather than continuously guiding the parameters throughout the training process, as a regulariser does. Additionally, it would not be able to handle new, unseen data as it would have no built-in functionality to ensure fairness. It is for these reasons that we devise and investigate a fairness regulariser by adapting adversarial learning methods.

4.2 Fairness Regularisation

To devise a method of fair training, we will be adapting the work of adversarial regularisation on deep NNs outlined in [24], the authors of FGSM. As stated before, adversarial examples are defined as examples that are only slightly different from correctly classified inputs but cause the model to completely misclassify them [24]. It was found that an adversarial objective function based on FGSM to produce adversarial examples was an effective regulariser for making a DNN more resistant to adversarial examples, given by the equation

$$L^* = \beta \cdot L(\theta, x, y) + (1 - \beta) \cdot L(\theta, x + \epsilon \cdot \mathrm{sign}(\nabla_x L(\theta, x, y)) \tag{4.3}$$

where L^* is the loss, L is the loss function, x, y, the input and output respectively, and θ is a model parameter.

It is clear that β controls the strength of the regulariser; higher β values would indicate that the original prediction's loss would be favoured over the adversarial example's and vice-versa for lower values.

Given our focus is on BNNs, we must manipulate the inference method of our stochastic model, VOGN, to include a preference for the distribution of our fair example. VOGN approximates the posterior distribution as a Gaussian distribution due to the infeasibility of computing the exact posterior. The algorithm attempts to minimise the loss between the posterior distribution and the true distribution by computing the *natural* gradient and taking the step in the steepest direction, similar to conventional optimisation procedures. The natural gradient, however, takes into account the geometry of the parameter space opposed to conventional Euclidean gradients, which can lead to better posterior approximations. We refer the reader to [36] for details on VOGN and the computation of natural gradients, as well as to the algorithm in Figure 3.3 on the exact computational steps it takes. The computation of the component gradient for a given mini-batch $i \in M$ of a training dataset is given by

$$g_i^{(k)} \longleftarrow \nabla_w \ell(y_i, f_{(k)}^w(x_i)) \tag{4.4}$$

where k is the current iteration of VOGN, w is the weights, ℓ is the loss function, $f_{(k)}^w(x_i)$ is our current model's predictive distribution for training data x_i, and y_i is the target (i.e. true) value. This can be seen in line 10 of Figure 3.3.

We can now manipulate this to include the combined predictive distributions of both x_i and x_i's fair example with a strength parameter β_{fair}.

$$g_i^{(k)} \longleftarrow \nabla_w \ell(y_i, z) \quad s.t. \tag{4.5}$$

$$z \longleftarrow \beta_{fair} \cdot f^w_{(k)}(x_i) + (1 - \beta_{fair}) \cdot f^w_{(k)}(x_i + \epsilon_{fair} \cdot \text{sign}(\mathbf{E}_{P(w|D)}))$$

where e_{fair} is our set of epsilons dictating what inputs we consider to be similar. It is not hard to see the similarity between Equation 4.5 and the adversarial regularisation Equation 4.3. The inclusion of the strength parameter β_{fair} allows us to control how strict we want our model to prefer the distributions of the fair examples over the predictions. Although it would be natural to want to set this to 0, i.e. *always* prefer the fair example distribution, it would be ill-advised due to the fairness-accuracy trade-off outlined in Section 2.2.4. A preference for such distributions may lead to a fair model but would heavily impede its accuracy, as for a given input it will generate a predictive distribution that is most appropriate for its fair example, rather than the input itself. It is key to note that the generated predictive distributions for a fair example are not guaranteed to be the same as the distributions of the original input. A good balance between the two is advisable, as we are looking for a fair model that is also highly accurate.

A key aspect that should be noted of our devised regulariser is that it is attempting to lower the *average* difference between outputs of inputs and their respective fair examples across the entire model. The lowering of the average is a key difference compared to measuring fairness; when measuring fairness, we are observing the maximum difference produced, as outlined in Section 3.3.2. Hence, our aim is to reduce the maximum difference by effectively lowering the average difference between inputs and their fair examples.

4.3 Threshold-Fairness Metric

Before discussing the implementation of our regulariser and experiment methods, we will first outline a simple metric devised to measure fairness that is related to a given output similarity threshold. Although using maximum difference between similar pairs in a training set δ^* is a good way of measuring fairness, it has no relation to our defined output similarity threshold δ. There is little to say about how fair a model is compared to another if only observing the maximum output differences for similar pairs; one value of δ^* may be within the similarity threshold for one model but not for another. Hence we devise the *Threshold-Fairness* metric.

Definition 4.3.1 (Threshold-Fairness)**.** *Given a similarity threshold δ and a maximum difference δ^*, the threshold-fairness score is:*

$$\text{Threshold-Fairness} = \frac{\delta - \delta^*}{\delta} \tag{4.6}$$

This captures measuring fairness in an intuitive manner; higher values indicate that the maximum difference between outputs of similar input pairs is much lower **relative** to the defined threshold to consider outputs similar. The higher the threshold-fairness, the less the model was influenced by a protected attribute, thus the more fair the model is. Negative threshold-fairness scores would also indicate that the model is unfair. We can thus compare the fairness of models that have different output similarity thresholds. For example, suppose we have a model m_1 with threshold $\delta = 0.8$ and maximum difference $\delta^* = 0.6$ and another model, m_2, with threshold $\delta = 0.7$ and maximum difference $\delta^* = 0.3$. We can compute their respective threshold-fairness scores, m_1 with a score of 0.25 and m_2 with 0.57. We can thus conclude that the model m_2 is fairer relative to its own threshold compared to m_1.

We acknowledge that whether this is an appropriate comparison is up for debate, as different thresholds would indicate that definitions of similarity may be different for each model. We advise that this metric is only comparable for models attempting to model the same classification task. Outside of this, we enter the blurred lines between machine learning and sociology, as the definitions of fairness and similarity utilised come into question. With that said, we think it is a useful metric in our context for observing the fairness of our models.

Chapter 5

Experiments

We will now describe how we tested our fair training regularisation mechanism. We will first outline how we constructed a BNN with and without our regularisation method, enabling us to compare their respective levels of fairness under varying architectures and non-protected epsilon values. We will then proceed to outline the metrics collected and then conclude with the difficulties faced during the experiments. All code for this project was conducted using Python 3.8.10.

5.1 Setup

To give us a full scope of the effectiveness of our regularisation method, we decided to run ten trials of an experiment comparing the levels of fairness of a BNN trained with the regularisation method against a standard BNN. These comparisons were made under various similarity metric parameters and model architectures. We will outline the details of the experiment as follows.

Database For the purposes of our experiment, we utilised the Adult database, a commonly used and certified database for ML fairness research [8]. The task of the database is to predict whether an individual will attain income that exceeds \$50K/yr based on

14 attributes, which includes protected attributes such as gender, race and age. For our experiments, we utilise an 80-20 training-test split. All categorical features, such as occupation, are one-hot encoded. As standard for neural networks, we normalise the inputs to the range of [0,1]. This has the benefit of speeding up the learning process.

BNN Architecture We test our regulariser under a range of different hidden layer widths (number of neurons per layer) and depths (number of layers). We vary the depth between 1 to 5, testing a width range of 2 to 64 at each depth. To put more accurately, for each depth [1,2,3,4,5] we test the fairness levels for each width [2,4,8,16,32,64]. The rest of the listed parameters are used across all architectures. We use a batch size of 128 and a maximum of 15 epochs. For the loss function we use categorical cross entropy, a standard loss function for BNN classification tasks. For the inference method, we use the aforementioned VOGN optimiser. The hidden layers utilise a typical RELU activation function and a Softmax function for the output layer, as outlined in Section 3.1.1. By testing a wide range of architecture sizes, we hope to capture the scalability of our fair training method.

Similarity metric - ϵ For our similarity metric, we will utilise only gender as our protected attribute. We will set the epsilon value for this protected attribute for maximum variability, as outlined in Section 3.3.1, $\epsilon_p = \infty$. In practice, this will be set to 1, as we are standardising the inputs. We then test the fair training regularisation method on a range of values for the set of epsilon values for our non-protected attributes. Specifically, we test the set of values [0.0, 0.05, 0.1, 0.15, 0.2]. This implies that by having a value of 0 non-protected attributes of individuals are only considered to be similar if there is no difference in their values, and if the value is 0.2, attributes of individuals are considered similar if their differences are within 0.2 of each other. Realistically, we would set this value based on prior research indicating the correlation between non-protected attributes and protected attributes, but here we are merely testing whether it has an affect on the fairness level outputted by a fair trained BNN. However, we note that we can extend this to include more protected attributes. Our objective of testing a wide range of non-protected ϵ values is to evaluate its role on the effectiveness of our fair training method.

Similarity metric - δ As we are interested in exploring how our devised fair training method effects the fairness levels of a BNN, rather than determining when a model is exactly unfair, we will set this to $\delta = \infty$. Again, as the inputs are standardised, this will be set in practice as $\delta = 1$.

Fairness Regularisation - β For our strength parameter, we will do a 50/50 split ($\beta = 0.5$) between accuracy and fairness for a given input, in accordance with what we outlined in Section 4.2.

Computer architecture We ran our experiments on two Microsoft Azure virtual machine with a Linux kernel, 2 virtual CPUs, x64 architecture and 4 GiB of RAM. This was implemented due to the considerable time required to train and evaluate the models under different architectures.

5.1.1 Trials

The following pseudocode describes a high level view of the sequence of operations for a single trial.

```python
x_train, x_test, y_train, y_test = preprocess_data()

def run_trial():
    # Initialise our parameters
    delta = 1 # Output threshold delta
    eps = [0.00, 0.05, 0.10, 0.15, 0.20] # Non-protected epsilon values
    layers = [1, 2, 3, 4, 5]  #Number of hidden layers in the model
```

```python
 neurons = [64, 32, 16, 8, 4, 2] #Number of neurons per hidden layer in the model

 for epsilon in eps:
     for neuron_num in neurons:
         for layer_num in layers:
             # Setting our epsilons
             epsilons = [epsilon for i in range(0, number_of_attributes)]
             epsilons[protected_attribute] = 1.0

             # Get our standard BNN and its training time
             BNN, time_BNN = trainBNNModel(neuron_num, layer_num, x_train, y_train, x_test, y_test, fairness_regularisation=False)
             # Get its metrics, including fairness levels using Fair-FGSM
             BNN_metrics = get_results(BNN, x_test, y_test, epsilon, delta)

             #Get our fair trained BNN and its training time
             BNN_fair_trained, time_fair_trained_BNN = trainBNNModel(neuron_num, layer_num, x_train, y_train, x_test, y_test, epsilons, fairness_regularisation=True)
             # Get its metrics, including fairness levels using Fair-FGSM
             BNN_fair_trained_metrics = get_results(BNN_fair_trained, x_test, y_test, epsilon, delta)

             # Store the results
             store_results(BNN_metrics, BNN_fair_trained_metrics, epsilon, delta, layer_num, neuron_num)
```

Listing 5.1: Python pseudocode for a single trial.

The mean of the metrics of 10 trials for each epsilon value was then calculated. Note that some trivial details, such as how the metrics are stored and computed, are omitted.

5.1.2 Packages Used

The implementation of this project relied on the use of several packages. For data storage, vector and matrix calculations the widely popular data analysis packages NumPy and Pandas were used. Scikit-learn, a popular ML package, was used for the train-test splitting of the dataset. For constructing our BNNs, we use DeepBayes, an extension of the industry standard deep learning toolkit Keras (which is subsequently an extension of the ML platform package Tensorflow). Keras unfortunately does not have in-built functionality to support the construction of BNNs, hence we must rely on DeepBayes. DeepBayes implements various basic BNN analysis including probabilistic safety, certifiable adversarial robustness and adversarial attacks such as FGSM. It also implements various optimisers such as VOGN. We extend this optimiser to include our fair training mechanism alongside Fair-FGSM. Full code of our implementation can be viewed at `https://github.com/tstephen22/Fairness-Through-Uncertainty`. For visualising the results of our experiments, we use a combination of the popular Python visualisation package Matplotlib and the data visualisation software Tableau.

5.1.3 Code Implementation

BNN

The following python pseudo-code describes how we implement our standard BNN and a fair trained BNN. It's important to mention that this omits some minor data manipulations for simplification, but the core concept remains the same.

```python
def trainBNNModel(neurons, layers, x_train, y_train, x_test, y_test,
   epsilons=[], fairness_regularisation=False):
        input_shape = x_train.shape[1]
```

```
    num_classes = y_train.shape[1]

    # Define our sequential (feed forward) neural network
    model_BNN = keras.Sequential()
    model_BNN.add(keras.Input(shape=input_shape))

    # Add our layers of neurons with RELU activation function
    for x in range(layers):
        model_BNN.add(keras.layers.Dense(neurons, activation="relu"))

    # Add our output layer with Softmax function
    model_BNN.add(keras.layers.Dense(
        num_classes, activation="softmax"))

    # Initialise our loss function - categorical cross entropy
    loss = keras.losses.CategoricalCrossentropy()

    # Intialise our VOGN optimiser using DeepBayes
    optimizer = optimizers.VariationalOnlineGuassNewton()

    # Parameters
    batch_size = 128
    epochs = 15

    # Finally compile our BNN with DeepBayes
    bayes_model = optimizer.compile(
        BNN_model, loss_fn=loss, batch_size=batch_size, epochs=epochs,
        # train with fairness regularisation (which is 5 in the optimisers list) if set to true
        rob_lam=0.5, # our beta value
        robust_train= 5 if fair_regularisation else 0,
        fair_epsilons=epsilons)

    # Train our model
    bayes_model.train(x_train, y_train, x_test, y_test)

    return bayes_model
```

Listing 5.2: Python pseudocode for the construction of a BNN with and without fairness regularisation.

Fairness Regularisation and Fair-FGSM

To implement our fairness regularisation, we rely on the code implementation of Fair-FGSM supplied by [17] to create our fair examples using our set of epsilons. Our regularisation is implemented as follows in the VOGN optimiser of DeepBayes.

```
    ...
    # Fair-FGSM training
    elif(int(self.robust_train) == 5):
        output = tf.zeros(predictions.shape)
        # Generate our fair example
        fair_example = BNN_FAIR_FGSM(self, features[0], self.attack_loss
, eps=self.fair_epsilons)
        # Get the predictive distribution of the fair example
        fair_example_dist = self.model(fair_example)
        # This (below) is the equalivalent of our equation 4.5, robust.
lambda being our strength parameter
        output = (self.robust_lambda * predictions) + ((1-self.
robust_lambda) * fair_example_dist)
        loss = self.loss_func(labels, output)
    ...
```

Listing 5.3: Python code for fairness regularisation.

5.2 Metrics

In this section we detail and give a brief discussion of the metrics we gathered to fully understand the impact of our proposed fair training method. For each metric, we will be taking the mean over the 10 trials as outlined.

5.2.1 Max Difference

This is one of the primary metrics for measuring the level of fairness of a BNN. It describes the maximum difference found between the predictive distribution of similar pairs in the input space, as described in Equation 3.11. To do this, for a given model we generate a fair example for every input in the test set. We then feed every pair into the model and compute their differences; the maximum difference found being our measure of fairness. As our predictive distributions are Softmax outputs, and we are dealing with a binary classification, we take the absolute difference of the two probabilities of an input belonging to the first class. Note that this could also be done with the second class; the value would remain the same. A value closer to 0 means that the model's prediction was not influenced by the input's protected features (gender), while a value closer to 1 means the model's prediction was more heavily influenced by a protected attribute [17]. We note that we collect this metric and visualise it in heat maps for completeness, but we do not provide an analysis of it as we will primarily focus on the threshold-fairness score.

5.2.2 Threshold-Fairness

We also calculate the threshold-fairness score as described in Section 4.3. As our δ is 1, the threshold-fairness score is simply an inverse of the maximum difference; values closer to 1 mean the model **was not** heavily influenced by a protected attribute (i.e. more fair) while values closer to 0 means the model **was** more heavily influenced by a protected attribute (i.e. less fair). This will be the primary metric we will observe in our results. We are thus looking to increase this value with our fairness regulariser.

5.2.3 Average Difference

We calculate and collect the average difference in a model's prediction distributions for pairs of similar inputs. The reason for this is our regularisation method will account for the difference between individual pairs, rather than accounting for the largest difference in the entire input space. To evaluate that our method is working, we would hope that the average difference is lower after our regularisation. For a number of pairs N, the average difference is

$$\text{Average Difference}: \quad \frac{\sum_{i=1}^{N} |\pi(x) - \pi(x^{'})|}{N} \tag{5.1}$$

5.2.4 Time

As standard with evaluating new regularisation methods, we collect and measure the time it takes to train with and without our method. We do not expect it to increase drastically as the complexity of Fair-FGSM, irrespective of the model's complexity, is $O(n)$ where n is the number of elements in the input. Fair-FGSM and the original FGSM method is quite fast compared to other adversarial computation methods which generally use iterative steps.

5.2.5 Accuracy, Precision, Recall

Given we are manipulating the inference method of our BNN, we compute and collect the accuracy, precision and recall of the models to see if our method has any effect on them. For context, True positives (TP) are the number of inputs classified correctly as the first class and false positives (FP) are the number of inputs incorrectly classified as the first class. True negatives (TN) are the number of inputs correctly classified as the second class and false negatives (FN) are inputs incorrectly classified as the second class. These metrics are computed as follows.

$$\text{Accuracy : } \quad \frac{TP + TN}{TP + TN + FN + FP} \tag{5.2}$$

$$\text{Precision : } \quad \frac{TP}{TP + FP} \tag{5.3}$$

$$\text{Recall : } \quad \frac{TP}{TP + FN} \tag{5.4}$$

5.2.6 Predictive Entropy

For completeness, we will calculate and collect the predictive entropy of the mean of the predictive distributions outputs of our BNNs. Entropy of the predictive distributions outputted by a BNN quantifies the level of uncertainty of the model's predictions [38]. For our classification task, if a BNN gives the predictive distributions over C classes for a given input as $p_1, p_2, ..., p_C$ then the predictive entropy is calculated as

$$\text{Predictive Entropy : } \quad -\sum_{c=1}^{C} p_c log(p_c) \tag{5.5}$$

Higher levels of predictive entropy indicate high uncertainty in the model's predictions (i.e. a high level of surprise in the predictions) for the given input. Low values indicate higher confidence. We use DeepBaye's built-in analyser to compute this for our models.

5.3 Difficulties

A number of issues were raised throughout both the experimentation setup and execution phase. For one, the number of architectures and non-protected ϵ values to be evaluated was significant. The number of models to evaluate over all is the number of trials (10) $\times$ the number of ϵ values (5) $\times$ number of layer values (5) $\times$ the number of neuron values (6) $\times$ with and without fair training (2) = 3000 models. If we take a rough estimate of 400 seconds to evaluate one model, it takes roughly 1,200,000 seconds to complete all trials, or roughly 2 weeks. It was for this reason that the trials were split between two identical Azure VMs, each completing 5 trials overall. Originally, we had planned to only do 5 trials overall given the significant amount of time. Fortunately, however, we found ourselves with additional time, so we extended the number of trials to 10. This allowed us to obtain a more precise assessment of the overall performance.

Another difficulty faced was devising the strength parameter β. Initially, a slightly different implementation of the regulariser was taken, such that there was no strength parameter. In this implementation, the regulariser would restrict the model to *always* balance for fairness over accuracy, essentially having $\beta = 0$. Although the regulariser did improve fairness levels to a very high degree, it penalised accuracy heavily and to such an extent that it would achieve a lower bound accuracy no matter the architecture. After observing this, we decided to adapt our methodology and implementation to allow the ability to balance between fairness and accuracy, thus including the parameter β in our regulariser. Given our objective was to investigate the base performance of the regulariser, we set our regulariser to balance between accuracy and fairness evenly.

Chapter 6

Results

In this chapter, we will present and discuss the results of our experiments outlined in Chapter 5. As we have experimented across a high number of architectures and non-protected ϵ values, we will primarily focus on the results and graphs of $\epsilon = 0.1$, while referring to general trends seen in other values. All graphs produced and discussed for every metric and ϵ value evaluated for our experiments can be viewed in the Appendix A of this paper. As mentioned previously, we have experimented on a total of three thousand models. The results for every model, as well as additional graphs such as visualisations of the maximum difference, can be viewed at `https://github.com/tstephen22/Fairness-Through-Uncertainty`. We assess the performance and impact of our proposed method on three key aspects of BNNs; fairness, predictive performance, and training duration.

6.1 Fairness

6.1.1 General Performance

We will now evaluate the general effectiveness of our devised method in improving the fairness of a BNN. From observation, we can provide several insights on the method's performance. We visualise the Threshold-Fairness scores produced by our experiments

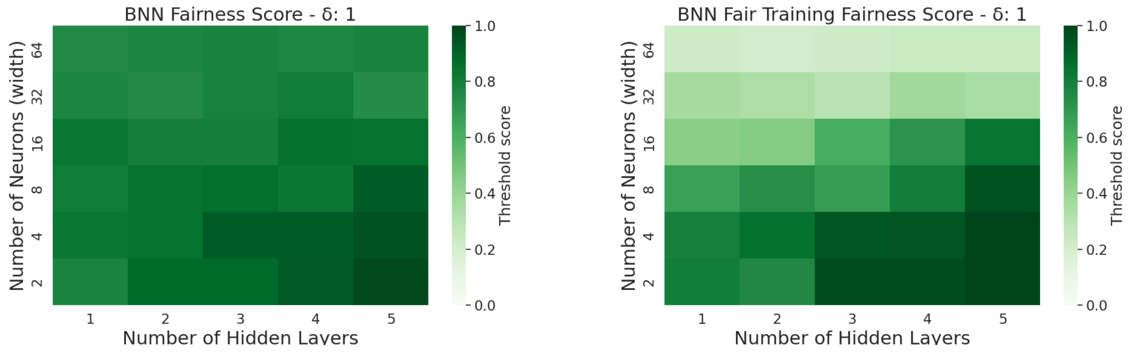


Figure 6.1: Heatmap of Threshold-Fairness score for BNN (left) vs BNN with fair training (right), $\epsilon = 0$

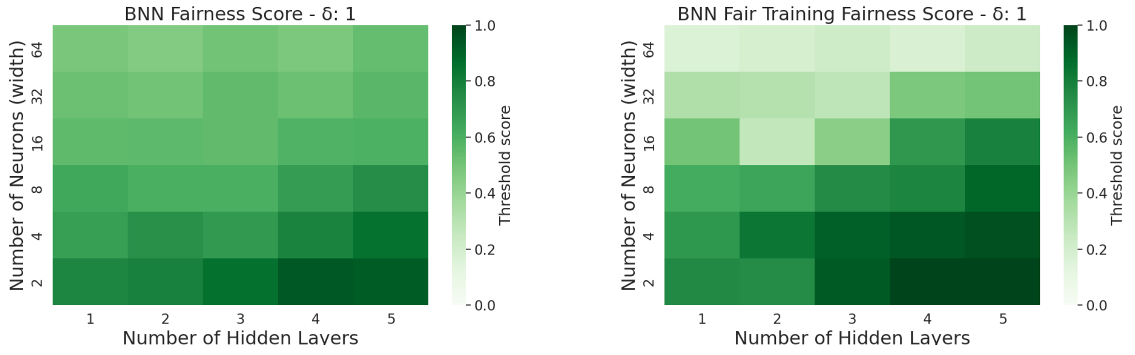


Figure 6.2: Heatmap of Threshold-Fairness score for BNN (left) vs BNN with fair training (right), $\epsilon = 0.05$

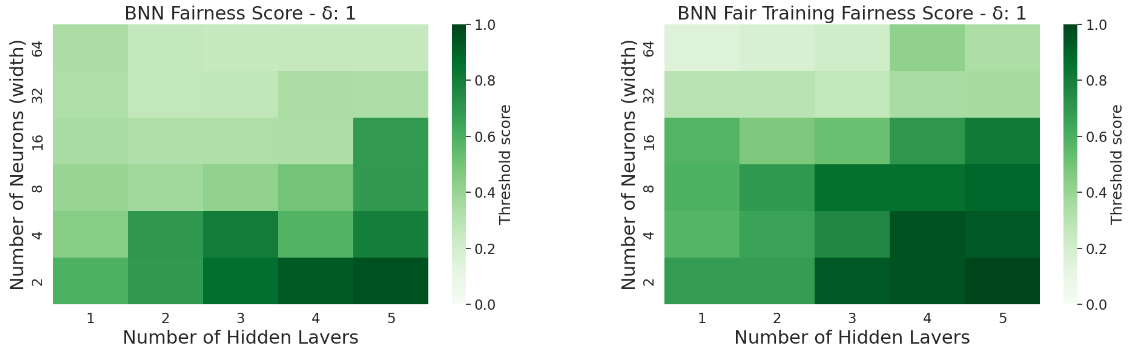


Figure 6.3: Heatmap of Threshold-Fairness score for BNN (left) vs BNN with fair training (right), $\epsilon = 0.10$

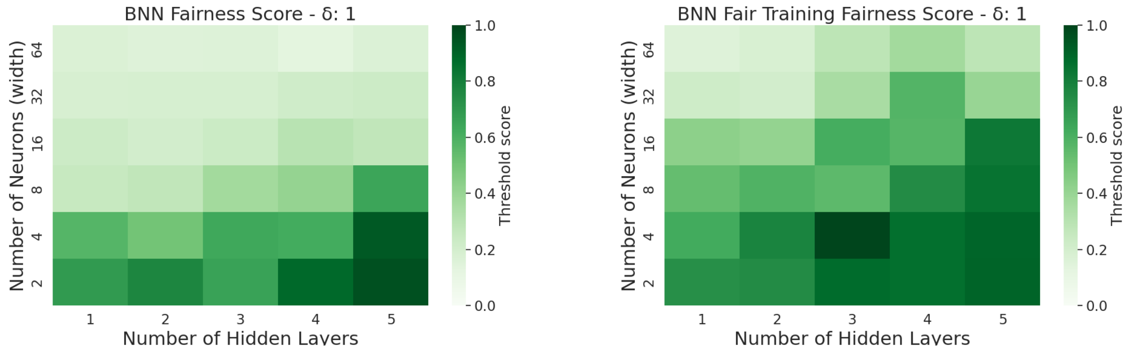


Figure 6.4: Heatmap of Threshold-Fairness score for BNN (left) vs BNN with fair training (right), $\epsilon = 0.15$

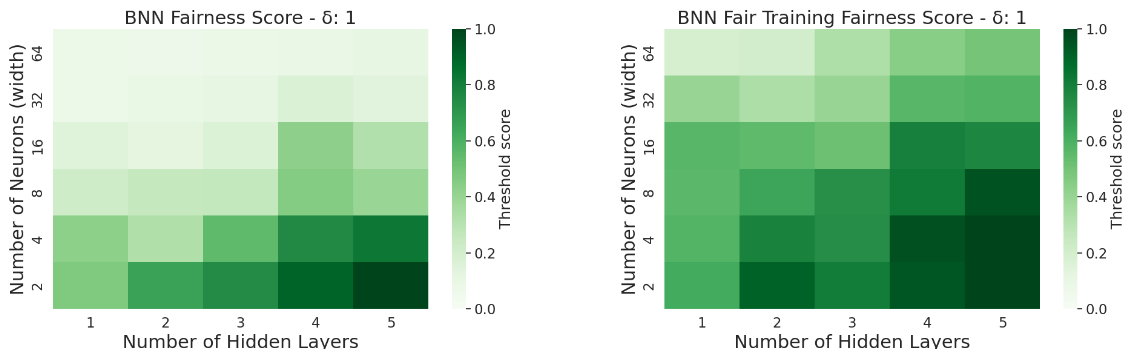


Figure 6.5: Heatmap of Threshold-Fairness score for BNN (left) vs BNN with fair training (right), $\epsilon = 0.2$

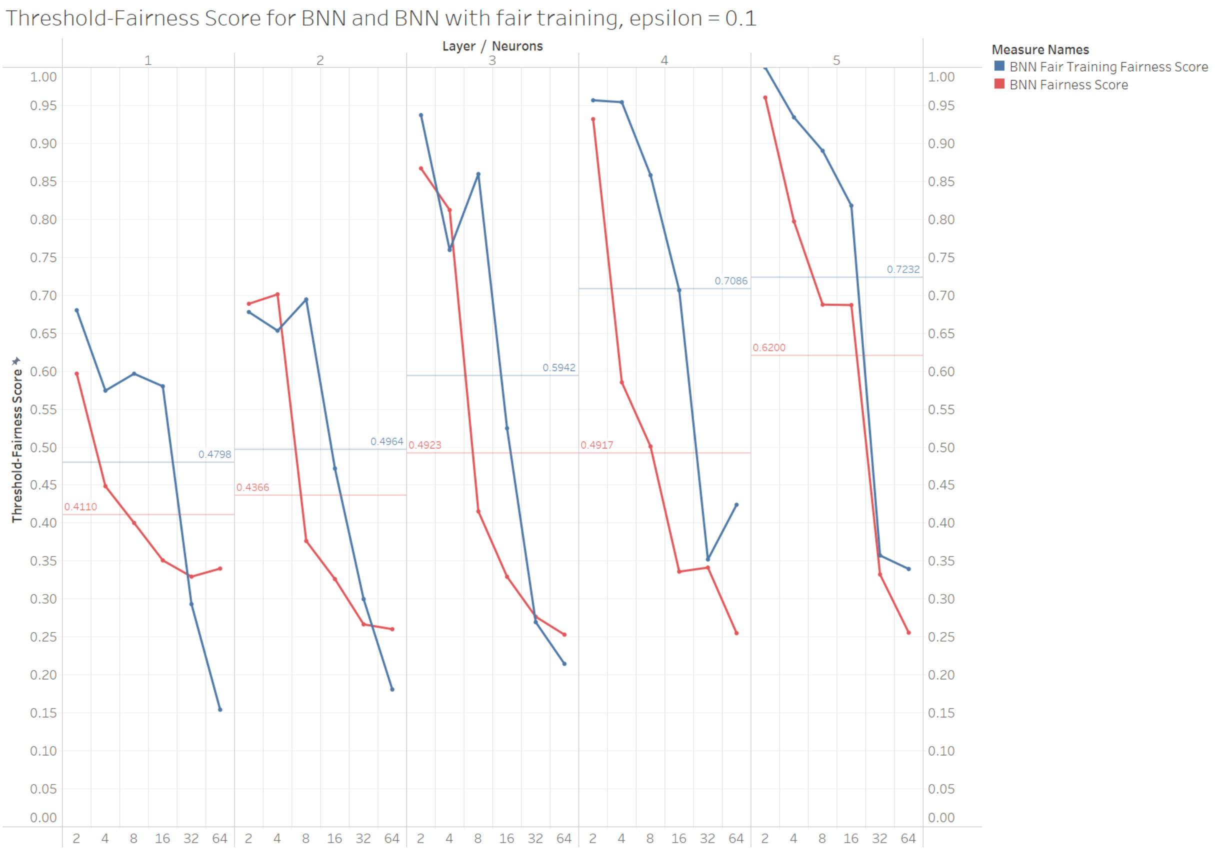


Figure 6.6: Threshold-Fairness Score against architecture for BNN (red) vs BNN with fair training (blue), $\epsilon = 0.1$. Constant lines indicate the average value over the layer size.

for each epsilon value in the heatmaps in Figures 6.1 - 6.5. We can observe a general pattern of our fair training method being quite effective for improving fairness levels of narrow architectures, but has a negligible or even a reducing effect on the fairness of wide architectures. On closer inspection of the fairness levels for $\epsilon = 0.1$ in Figure 6.6, it becomes evident that for every depth (layer) there is an average increase in fairness after fair training. This trend can be observed in the corresponding Threshold-Fairness graphs for other ϵ values. As well as that, the decline in fairness levels as the number of neurons increase is more noticeable. This can largely be explained by the fairness-accuracy trade off as mentioned. Accuracy increases as the model tends to become more precise and detailed with a greater number of neurons, reducing fairness as it employs additional ways to distinguish between individuals who are considered similar - effecting both a standard BNN and a fair trained BNN. Our fair training method does improve the fairness levels in some instances of high number of neurons (such as a 5 layer network with 64 neurons per layer, as seen in Figure 6.6), but generally it tends to decrease it for ϵ values < 0.1.

6.1.2 Effect of ϵ

An observable trend in the performance of the fair training method is that it improves as the ϵ values increase. When ϵ is below 0.1, such as $\epsilon = 0$ or $\epsilon = 0.05$, the regularisation tends to reduce fairness levels, sometimes even drastically, as observable in Fig 6.7. This trend can be further observed in graphs in ϵ plots for different architectures, such as in Figure 6.8 for a model with 4 layers and 64 neurons each. At values at and above 0.1 for ϵ the fair training method tends to improve fairness levels across all architectures.

There are several potential explanations for this effect. Firstly, low ϵ values can constrain the model heavily, as it essentially means there is little to no range for the non-protected attributes to be considered similar in their distances. Although this does imply the fair example is *only* different in the protected attribute, training the model on this

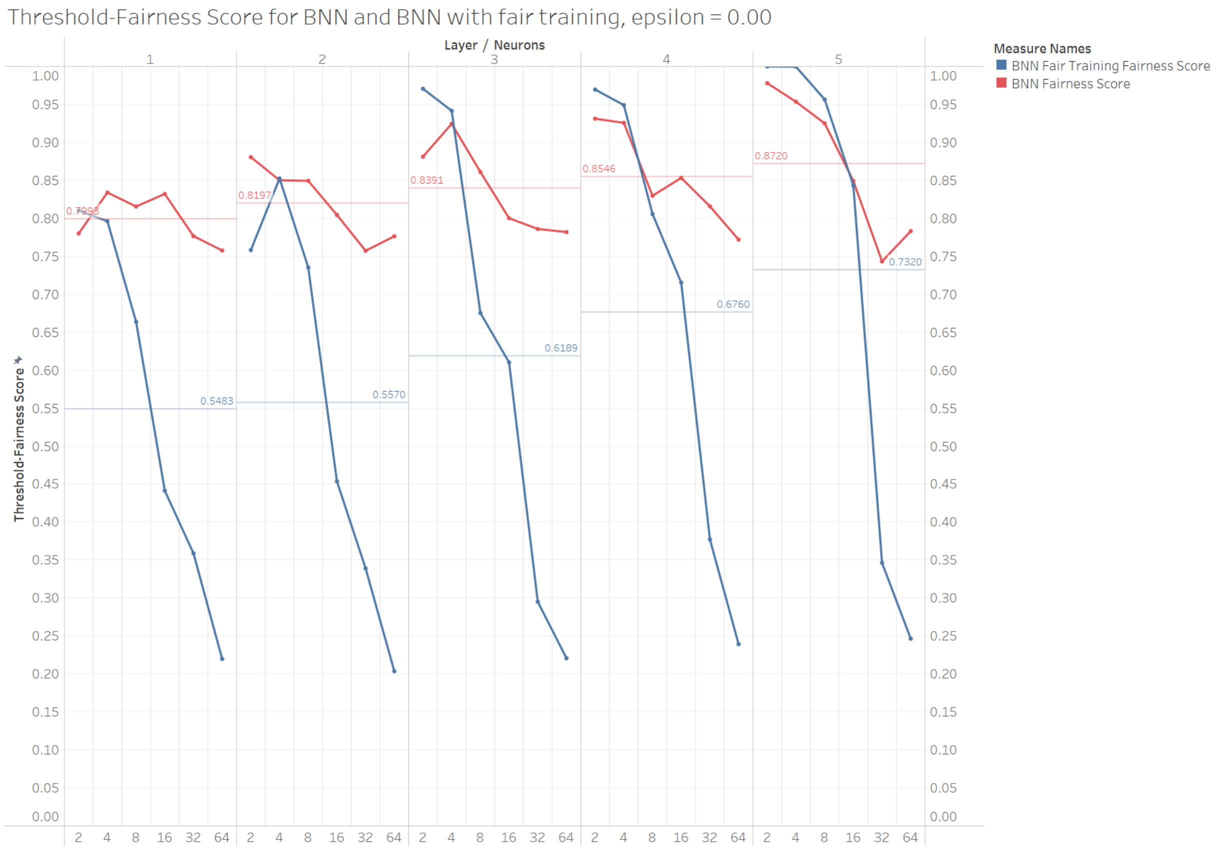
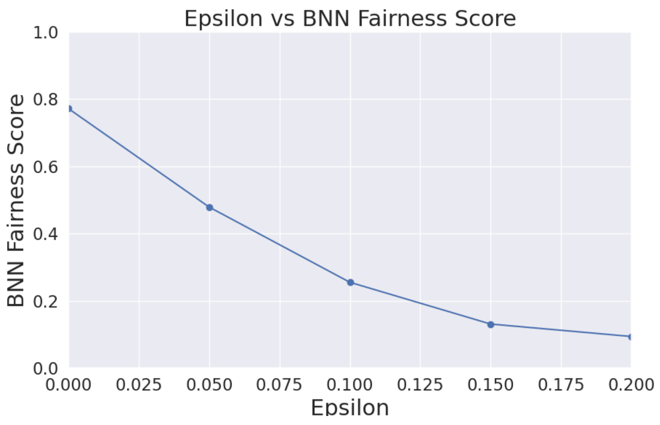


Figure 6.7: Threshold Fairness vs Architecture for BNN (red) and BNN with fair training (blue), $\epsilon = 0$. Constant lines indicate the average value over the layer size.



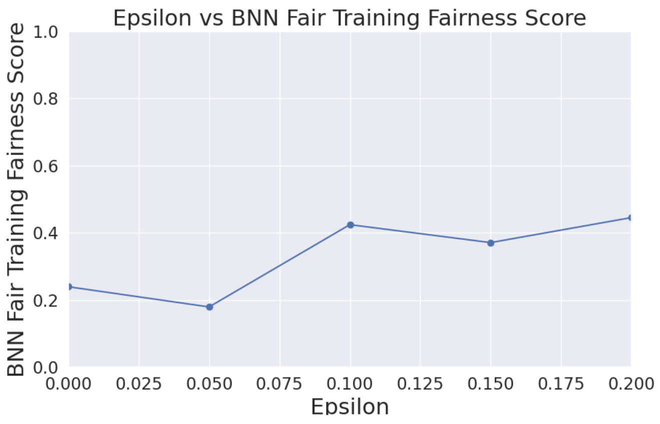


Figure 6.8: Example plot of Threshold-Fairness Score vs ϵ for a BNN (top) and a BNN with fair training (bottom). The model has 4 layers with 64 neurons each.

fair example could result in an excessive focus towards minimizing the differences strictly based on the non-protected attributes. It is key to note that the output generated by a BNN is a probability distribution, which is rarely the exact same for multiple outputs given the same input. As a result, the model might achieve high similarity in the outputs of inputs and their fair examples for non-protected attributes, but at the expense of not adequately addressing fairness concerning protected attributes.

A second potential explanation follows from this; BNNs thrive on the intricate variability in data, and by training the model on generated outputs that have very little difference from their real counterparts, we are in effect constraining the model. Setting $\epsilon = 0$ essentially doubles the data point given to the model and places an overemphasis on its outcome, potentially over-fitting the model to it. Although setting $\epsilon = 0$ for measuring fairness for a given model is acceptable, doing so for training would be naive given the complexities of both fairness and the inference methods of BNNs.

On the other hand, high ϵ values, particularly those at or above 0.2, are generally not recommended, unless the specific task of the model justifies such a substantial degree of variability in the differences for individuals to be considered similar. Such a high band would imply perturbations in the fair examples that are significant for the model, hindering its accuracy as it is not getting accurate data.

If anything, this illustrates the importance of a precise and carefully considered methodology for determining the ϵ values for non-protected attributes for the fair training regulariser. From our observations, we can provide the insight that our devised method works most appropriate for non-protected ϵ values at around 0.1 or 0.15, but this of course is context-specific to the model and fairness task.

6.1.3 Average Difference

As mentioned in Section 4.2, our devised regulariser's objective is lowering the maximum difference between inputs and their respective fair examples by lowering the average dif-

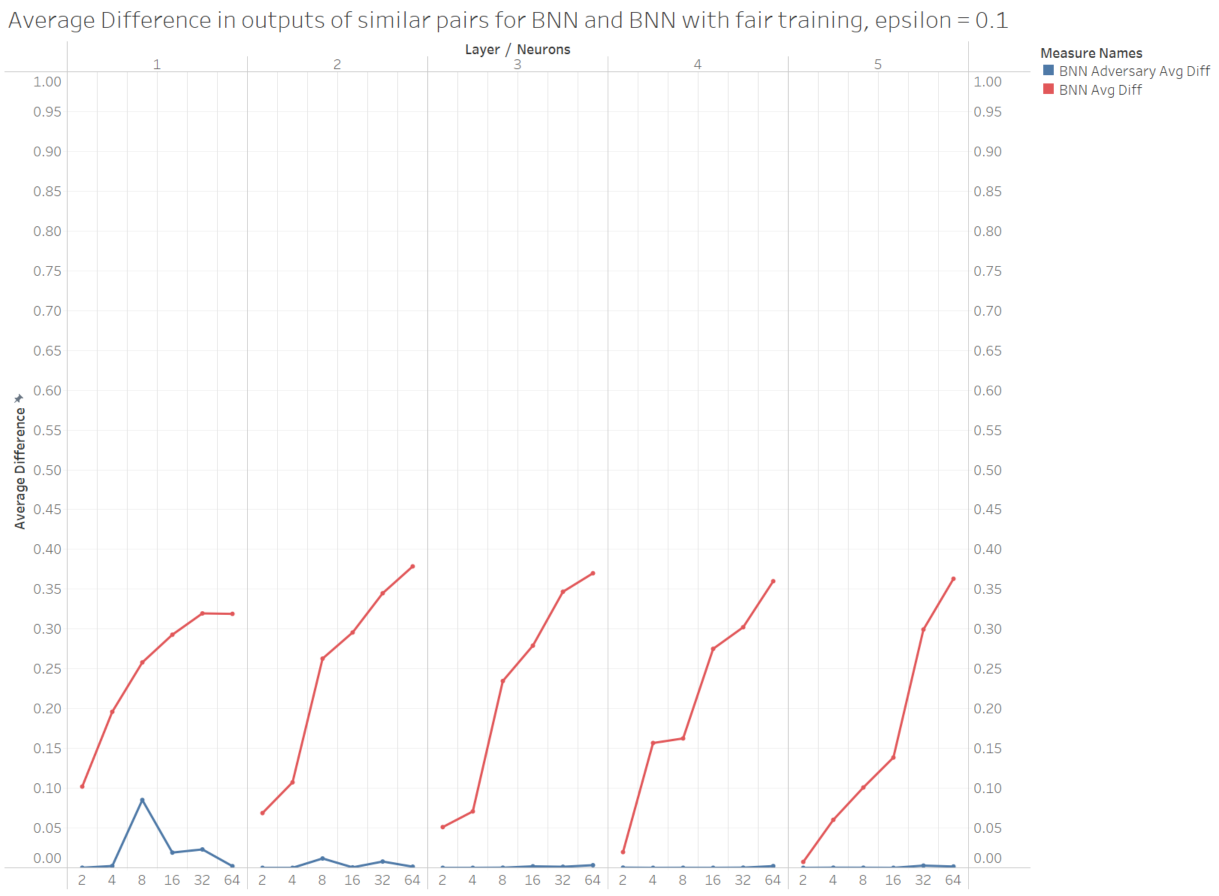


Figure 6.9: Average Difference against architecture for BNN (red) and BNN with fair training (blue), $\epsilon = 0.1$.

ference. Observing the average difference graphs, such as for $\epsilon = 0.1$ in Figure 6.9, we can safely conclude it is effectively completing this task. It is still, however, subjected to the effect of ϵ when viewing similar results for lower (< 0.05) non-protected ϵ values; increasing the average difference when $\epsilon = 0$.

6.2 Predictive Performance

When evaluating the performance of any regulariser, there must be consideration for its impact on the predictive performance of the model. We evaluate the predictive performance of the BNNs by observing the accuracy, precision and recall of the model. We then evaluate the potential impact on the stochastic nature of BNNs by observing the mean predictive entropy across architectures.

6.2.1 Accuracy

Observing the accuracy graphs such as in Figure 6.10, we can see that the increase in fairness has limited the maximum accuracy achievable of the model. Generally, the standard BNN reaches an average maximum accuracy of ≈ 0.84 ($\approx 84\%$), while the fair trained BNN reaches an average maximum accuracy of ≈ 0.82 ($\approx 82\%$) and at best an accuracy of ≈ 0.829 ($\approx 82.9\%$) across architectures and ϵ values. This is expected, given the fairness-accuracy trade-off. From observation of ϵ values > 0.1, we can see some indication of the accuracy dropping by at least 0.01 (1%) for every architecture. We can observe an example of this small fluctuation in Figure 6.11 for a model with 4 layers and 64 neurons each. We could attribute this to the threshold for similar outputs being too large, implying the model is balancing true inputs and fair examples with significant perturbations to their non-protected attributes. However, this isn't a significant enough drop for us to draw a relationship between ϵ and accuracy, as there could be other factors at play. There is also only a certain amount of accuracy achievable with a given benchmark dataset, such as the Adult dataset, so we can't draw any conclusive insights about the effect of non-protected

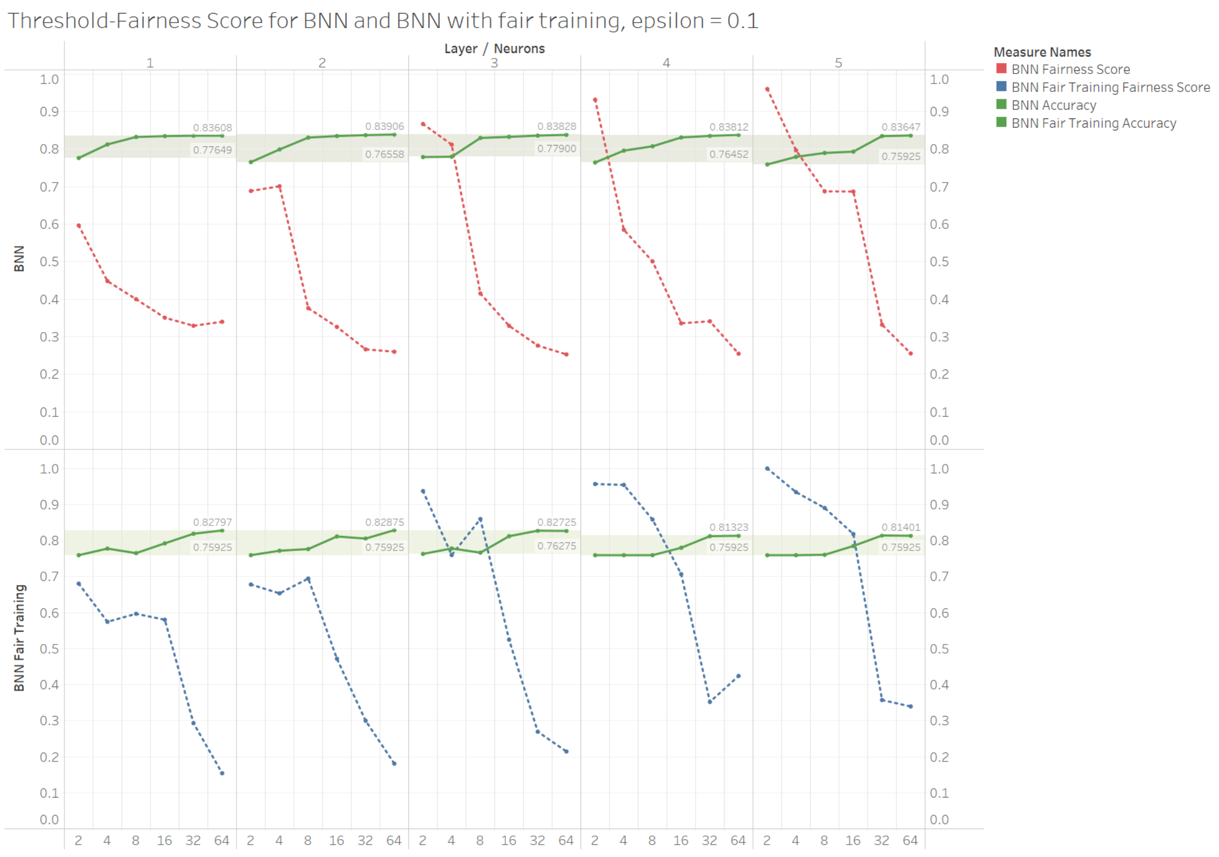


Figure 6.10: Accuracy and Threshold-Fairness Score against architecture for BNN and BNN with fair training, $\epsilon = 0.1$. Coloured bands indicate the range between the maximum and minimum accuracy over the layer size.

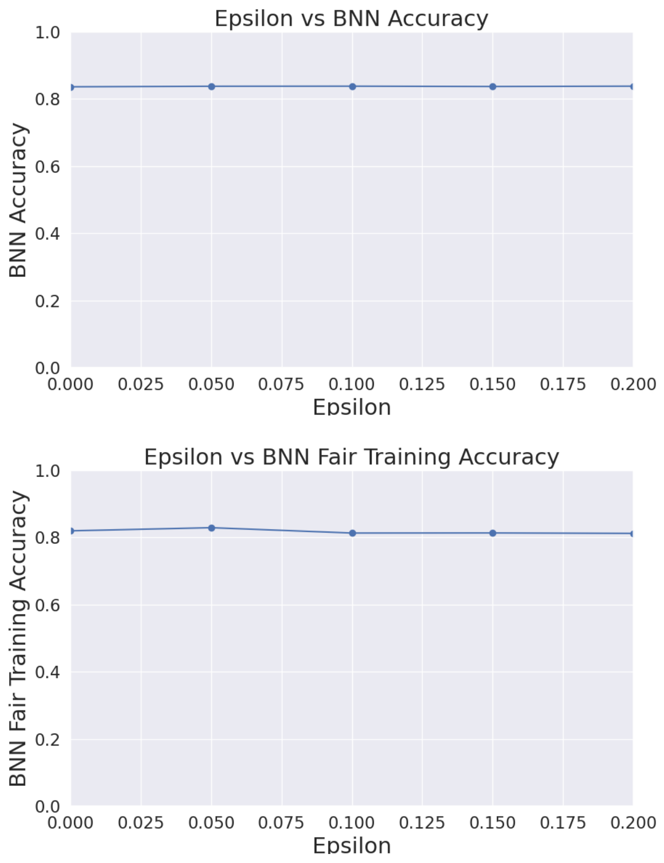


Figure 6.11: Example plot of Accuracy vs ϵ for a BNN (top) and a BNN with fair training (bottom). The model has 4 layers with 64 neurons each.

ϵ values on the accuracy. We can, however, confirm that the accuracy drops as fairness increases, as expected.

6.2.2 Precision and Recall

As precision and recall are generally analysed together, we will discuss the impact of the fair training method on both here. In a similar manner to accuracy, there is a potential trade-off between precision and recall with fairness. This is evident in our results across non-protected ϵ values. An example of precision and recall values for $\epsilon = 0.1$ can be viewed in Figure 6.12. For the fair trained model, precision tends to be much lower compared to a standard BNN as the number of neurons increase. Although it does increase as the number of neurons increases, it does not reach the same maximum precision as a standard BNN, which is ≈ 0.79.

The same can be observed for recall; it is generally much lower for the fair trained BNN compared to a standard BNN across different widths and ϵ values. There can be several factors contributing to this, such as the stochaistic nature of BNNs, which implies that each test essentially evaluates slightly different models. However, the general trend can be attributed to the increase in fairness. As the model ensures fairness across its predictions, it will inadvertently lower its accuracy in identifying true positives (leading to lower recall) and correctly classifying negatives (leading to lower precision) compared to the true values of the training data.

6.2.3 Predictive Entropy

From the mean predictive entropy across non-protected ϵ values, we can observe that a fair trained BNN's uncertainty in its predictions tends to fluctuate more across model architectures compared to a standard BNN. However, when the width of the architecture is at or above 32 neurons it is less uncertain in its predictions compared to a standard BNN; the predictive entropy is lower, implying a lower level of "surprise" in its predictions.

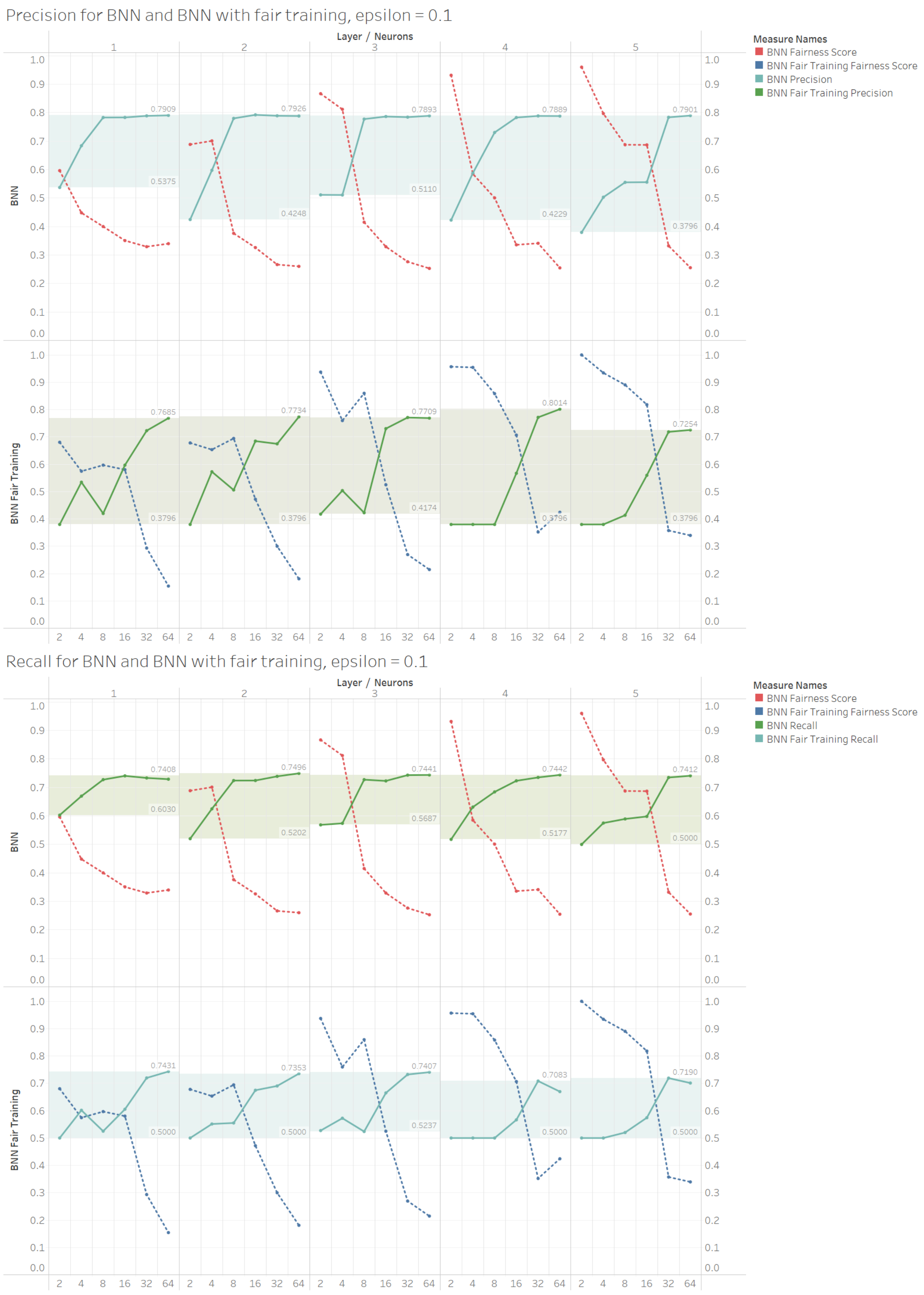


Figure 6.12: Precision (top) and Recall (bottom) with Threshold-Fairness against architecture, $\epsilon = 0.1$.

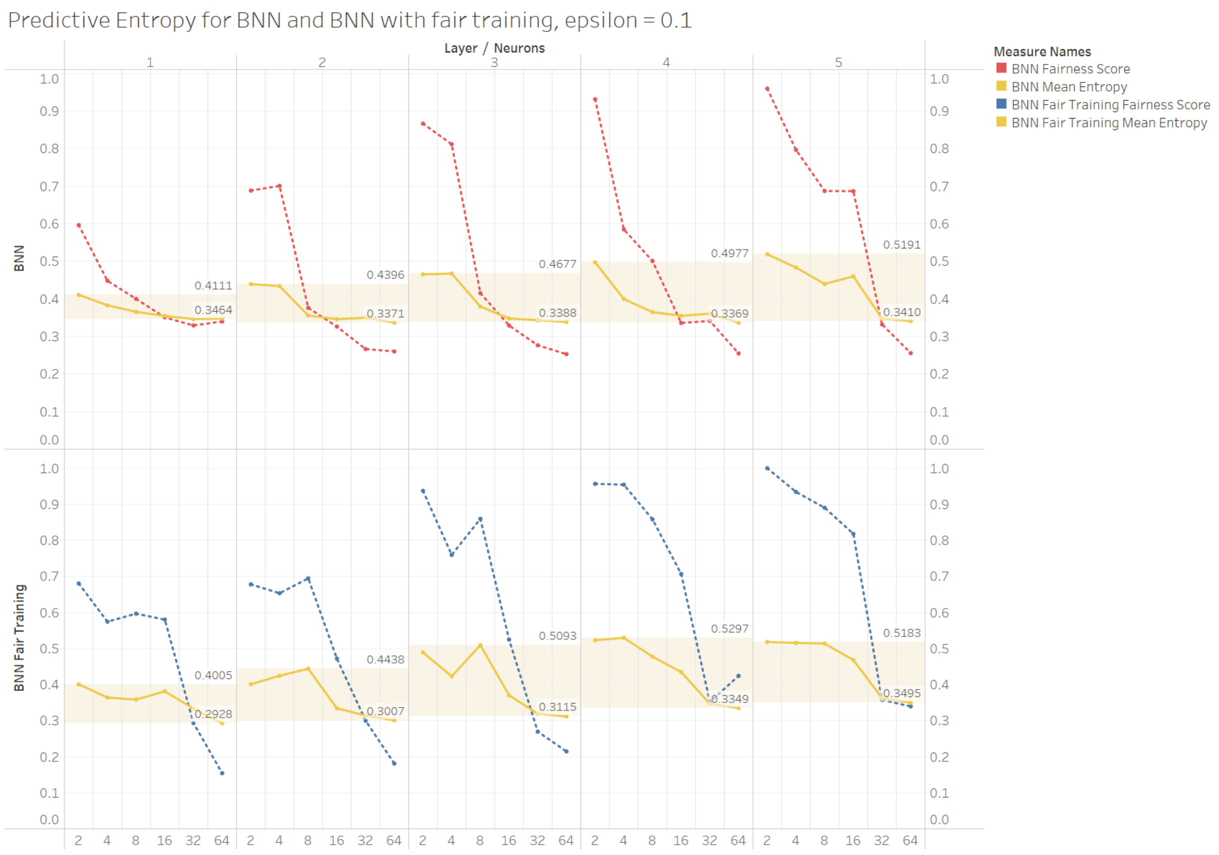


Figure 6.13: Mean Predictive Entropy and Threshold-Fairness score against architecture, $\epsilon = 0.1$.

This is observable in Figure 6.13. For a single layer with 64 neurons, the predictive entropy for a BNN with fair training is 0.2928 while for the standard BNN it is 0.3464. This can be shown likewise for 2, 3 and 4 layers. From observation, we can also assert that the increase in ϵ values has little effect on the predictive entropy. Across a majority of the trials for each ϵ value, the predictive entropy remains within the interval of [0.29, 0.52].

We could attribute this increase in confidence of the model's predictions to the fairness constraint. By design, we are trying to homogenise the predictions made across groups of individuals. This would naturally lead to a decrease in the level of surprise in outputs, as they become less variable and more predictable under our fairness guided training methodology. We, however, make these observations with caution, as predictive entropy is not an all encompassing measure of uncertainty.

6.3 Training Time

As described in Section 5.2.4, we expect an increase in training time as we are implementing an additional regulariser. Strictly speaking, we would expect an increase of O(N) to the model's training method as we are using Fair-FGSM, where N is the dimensionality of the input. We can clearly see this overall linear increase in Figure 6.14; the training time increases by $\approx$ 100 - 200 seconds. Outliers in the data can be associated with the methods Azure employs for handling long-running Virtual Machines over time, resulting in slightly inconsistent run times. Generally, our method is very fast and adds negligible time to the training time of a BNN.

6.4 Overall Performance

Based on our analysis of the results, we can conclude that our devised fairness regulariser measurably improves individual fairness levels for BNNs under suitable parameter values. The regulariser has a small impact on predictive performance, lowering the BNN

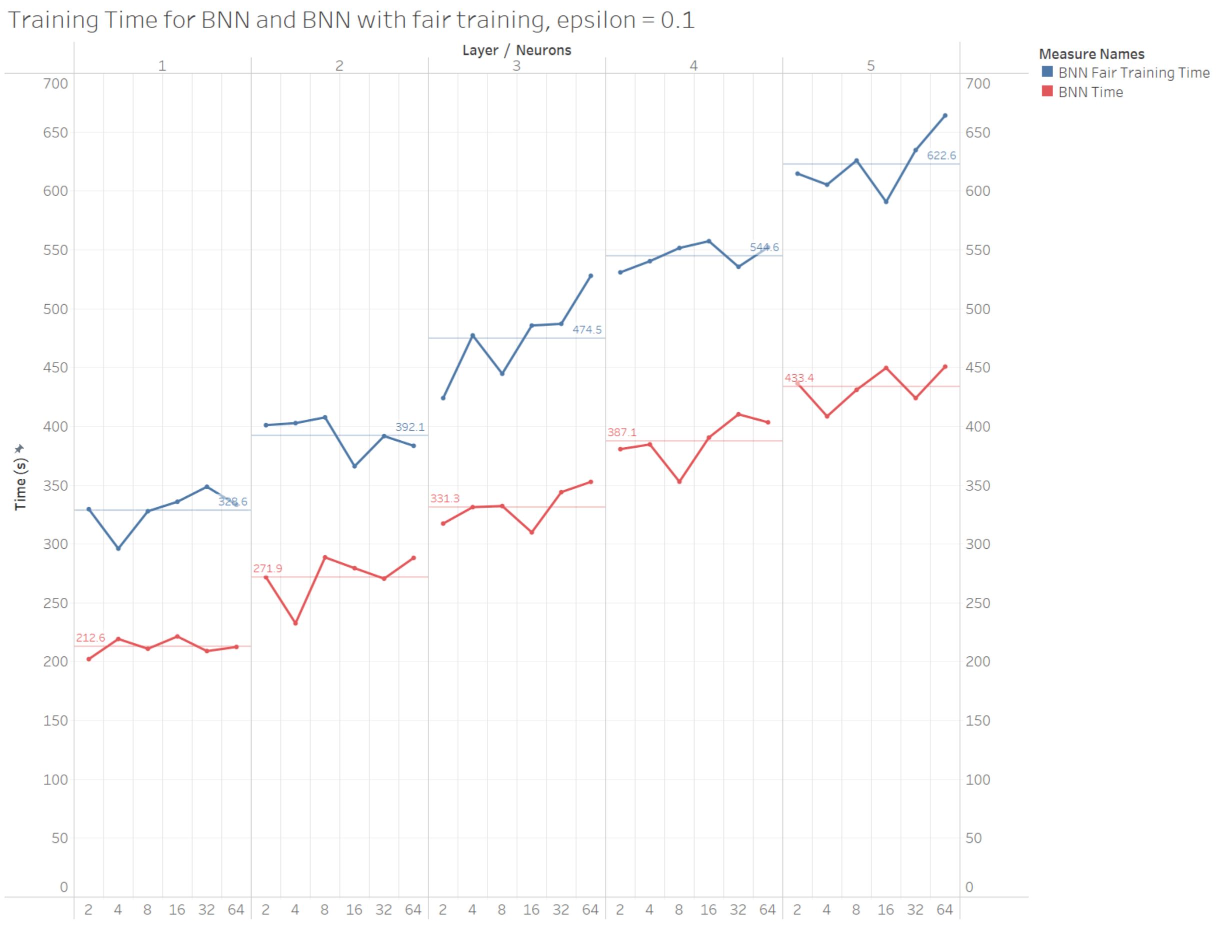


Figure 6.14: Training time against architecture for BNN and BNN with fair training, $\epsilon = 0.1$. Constant lines represent averages.

accuracy, precision and recall. This, however, is expected for any method with an objective to improve fairness. There is some indication that our method improves the model's confidence in its outputs, but we can not definitively say without further testing of the model's uncertainty. From our observations, it is clear that there is an inherent relationship between the values for the set of non-protected ϵ values and the effectiveness of the regulariser. For $\epsilon < 0.1$, the regulariser can be ineffective or detrimental to the fairness level of a BNN. As a result, we make the general recommendation to use non-protected ϵ values at or above 0.1 for the regulariser. However, this is not a rigid requirement; the values selected depend on the model's specific task and the context of the fairness scenario relevant to the user.

Chapter 7

Evaluation and Limitations

In this chapter, we will evaluate the findings of the paper as well as discuss its limitations.

7.1 Evaluation of Findings

As we concluded in Section 6.4, our fairness regulariser performs fast, effectively and reliably at improving individual fairness in BNNs when given non-protected ϵ values above 0.1. We can also empirically conclude that the non-protected ϵ values play a major role in determining the effectiveness of the regulariser. Unfortunately, we can not compare this to any method or benchmark for improving individual fairness in BNNs as this work is the first of its kind. However, as we have tested our regulariser on a high number of models with varying architecture sizes and taking the average over 10 trials, evaluating a total of 3000 models, we can conclude on the effectiveness and observed trends with confidence.

7.2 Limitations

In this section, we will discuss the primary limitations of our experiment.

7.2.1 Role of Strength Parameter β

In our experiments, we thoroughly tested multiple non-protected ϵ values, a key parameter for our regulariser. However, we did not investigate the role of the strength parameter β, which plays the role of balancing between accuracy and fairness. As outlined in Section 4.2, we set this to $\beta = 0.5$, implying an even balance between accuracy and fairness. Although valid, an argument can be made that the strict setting of this parameter is naive, as there may be scenarios where fairness *would be* preferred over accuracy (or vice-versa) for a specific input. For example, in hiring practices using ML, placing a higher emphasis on fairness may be more critical than achieving the highest accuracy, especially when considering candidates from underrepresented groups. In such cases, a higher weight on fairness could help mitigate biases and promote equal opportunities, despite potentially sacrificing some accuracy in the model's predictions. In retrospect, investigating the role of the strength parameter on the effectiveness of the regulariser by using a range of values would be beneficial. However, we think it would be more beneficial if this parameter was dynamically adapted based on the input received by the model.

7.2.2 Testing of Proxies

As noted in Section 3.3.1, we can account for correlations between non-protected and protected attributes by setting the similarity threshold (ϵ) for the non-protected attributes accordingly. For example, if there is a correlation between the ethnicity (protected) and home address (non-protected) of an individual for a hiring algorithm, we set the ϵ value for the home address accordingly such that the threshold to consider another home address as similar is much larger. Although we note this feature, we do not test the regulariser on various non-protected ϵ values for a single input, but rather use a single uniform value for all non-protected attributes. While ideally exploring its effectiveness in these types of scenarios would have been beneficial, our primary focus was to assess the foundational performance of the regulariser using uniform ϵ values across all non-protected attributes.

7.2.3 Testing of Multiple Protected Attributes

Similarly to the limitation in Section 7.2.2, we only evaluate the effectiveness of the regulariser when there is only a single protected attribute, which was gender. It would be beneficial to establish the regulariser's performance with multiple protected attributes, such as gender and race. As our work is the first of its kind for BNNs, our primary focus was to establish the foundational performance of the regulariser, hence we did not investigate this area. With that said, we posit that it would have a comparable performance if more protective attributes were specified. This lays grounds for future work to investigate as to whether multiple protected attributes has an effect on the regulariser.

7.2.4 Effectiveness on Deterministic NNs

Given the effectiveness observed for BNNs, we would be interested in seeing the performance of the regulariser when adapted for deterministic NNs. This would not be a difficult adaptation, given that Fair-FGSM can be used on deterministic NNs [17] and the adversarial training method outlined in [24] can be used as guidance, such as we have done. Although it is of our interest, we did not investigate this given our time constraints.

7.3 Review

Although with some limitations, our findings suggest there is a lot of value in our devised method. Additionally, it illustrates that the similarity between the definitions of individual fairness and adversarial robustness can be utilised and leveraged practically to enhance the fairness of BNNs, as well as other machine learning models. As our method is relatively simple and fast to compute, we are hopeful that it can act as a starting point to be developed and expanded upon. We are optimistic that with further development, it will evolve into a more rigorous and flexible regulariser, adaptable to a variety of scenarios where improving individual fairness is of high importance.

Chapter 8

Conclusions and Future Work

In this work, we have presented a simple fair training regulariser to improve the individual fairness of a Bayesian neural network (BNN) during training time; the first of its kind. Our motivation for the construction of this training method stems from the increasing need to effectively address and mitigate biases present in ML models. In addition to this, we were motivated to fill the gap in current research of methodologies for enhancing individual fairness for BNNs. BNNs are powerful machine learning models, with the potential to deliver substantial advancements to various fields of society, such as medicine, finance and security. As such, we saw it to be vital to develop a mechanism that serves as a starting point for counteracting biases and improving fairness within these models.

To achieve this objective, we explored the link between individual fairness and adversarial robustness. We transferred techniques from adversarial robustness training procedures into our context of individual fairness. We expanded on existing research that measures individual fairness in BNNs, embedding the *Fair-FGSM* algorithm directly into our regulariser's development. With this integration in hand, we developed a regulariser that can be applied during training to enhance the individual fairness of a BNN.

We then rigorously tested our developed regulariser across numerous experiment trials, testing various architecture sizes and parameter values, as well as collecting various metrics. In total, our experiments encompassed three thousand BNN models. Given this, we are confident in our findings. We can attest that our regulariser has a foundational performance of being fast and effective at improving individual fairness for a BNN, but does come at a slight degradation of model accuracy. We also found that the set of non-protected ϵ values chosen is influential to the overall performance of the regulariser. It is therefore our general recommendation to choose these parameters in a well-informed and correct manner, and ideally to use ϵ values at or greater than 0.1. However, we do outline that this is not a strict requirement, and that values should be set depending on the specific context the regulariser is being employed.

As stated before, we find that our work illustrates there is a lot of value to be gained from the similarity between the definitions of individual fairness and adversarial robustness in their definitions. Although there are some limitations to our work, we hope it will offer grounds for the construction of future work, which we will outline in the proceeding section. In conclusion, we hope that our developed regulariser and our empirical findings serve as a foundational piece in the investigation and development of individual fairness in BNNs.

8.1 Future Work

In this section, we will describe the possible avenues for future research in the area of individual fairness for BNNs.

8.1.1 Development of Fairness-Enhancing Mechanisms for Bayesian Neural Networks

As we have stated, to the best of our knowledge this is the first regulariser devised for BNNs to improve individual fairness. BNNs are increasingly being adopted in prominent areas in society, such as the medical field, to aid in making critical decisions where uncertainty needs to be correctly quantified. As such, we find it crucial that these models have tools in place to ensure that they are not only making accurate but fair decisions. We are aware that our devised regulariser is simplistic in nature, and may not capture complex qualities of fairness and what it means to be 'fair', but as it is the first of its kind we intend for it to act as a starting point for more complex and rigorous tools to be developed in the future. One potential development is to make the strength parameter β dynamic, allowing it to adjust based on the input and its respective fair example. This adaptation could enhance the regulariser's responsiveness and effectiveness in different scenarios. With this in mind, we emphasise the need for further research into the development of mechanisms that enhance individual fairness in BNNs going forward.

8.1.2 Testing of Regulariser in Complex Scenarios

As outlined in our limitations in Section 7.2, we do not test our regulariser with non-protected ϵ sets with various values, but rather a single uniform value for all non-protected attributes. We also do not test the effect of having multiple protected attributes. This was the case as we were interested in investigating and establishing the base performance of the regulariser. With that said, the performance of the regulariser on such scenarios where there are multiple protected attributes and various non-protected ϵ values would be highly insightful and beneficial; such scenarios are far more realistic anyway, as individuals can be part of multiple protected groups in the real world. We also noted that there is a relationship between non-protected ϵ values and the effectiveness of the regulariser at improving individual fairness, so further investigation into this relationship, such as when there is a set of non-protected ϵ values with various values above and below 0.1, would be quite insightful.

8.1.3 Adapting the Regulariser to Improve Individual Fairness on Deterministic Neural Networks

From our findings we can attest that the regulariser works well at improving individual fairness in BNNs. However, minimal adaptations are needed to integrate the regulariser into deterministic NNs. Given the extensive time required for our experimental setup, exploring this integration was beyond our current scope. Nevertheless, it presents an interesting area for future work.

8.1.4 The Relationship Between Fairness and Uncertainty

A limitation of our work was that we could not give a full comprehensive observation of the effect of the regulariser's on the uncertainty of the BNN, as it was out of our scope. We do make the observation that it may improve confidence in the model's predictions, as we are making the predictions of similar individuals more homogeneous, but we can't say it with complete certainty without more rigorous testing. Additionally, the work of [17] suggests that the uncertainty of BNNs results in them being fairer in their decisions compared to deterministic NNs. This, coupled with our small insight, opens up an area for further investigation into the relationship between uncertainty and individual fairness in deep learning models such as BNNs.

8.2 Reflection

As this paper develops the first individual fairness regulariser for BNNs, we are pleased to contribute a foundational piece to this area of research. The importance of ensuring deep learning models, such as BNNs, are fair in their decision making is becoming an ever increasing issue in society, especially as these models are deployed in critical areas of our lives. We are grateful that we had the opportunity to contribute to the advancement of this important endeavour.

On a personal note, the author would like to express that he is especially grateful to be given such an opportunity to not only learn about this important research area, but to also be able to contribute to it.

Declaration of AI Usage

We would like to state that the use of AI, such as ChatGPT, was used in parts of the writing of this paper to aid in sentence structure and phrasing.

Bibliography

[1] Pit-un inaugural convening: Keynote panel, technology, race, and our future, 2019.

[2] Racial discrimination in face recognition technology. 2020.

[3] Abdullah A. Abdullah, Masoud M. Hassan, and Yaseen T. Mustafa. A review on bayesian deep learning in healthcare: Applications and challenges. *IEEE Access*, 10:36538–36562, 2022.

[4] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. Fairsquare: probabilistic verification of program fairness. *Proc. ACM Program. Lang.*, 1(OOPSLA), oct 2017.

[5] Henry Anderson, Afshan Boodhwani, and R. Baker. Assessing the fairness of graduation predictions. In *Educational Data Mining*, 2019.

[6] Ricardo Baeza-Yates. Bias on the web. *Commun. ACM*, 61(6):54–61, may 2018.

[7] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration, 2019.

[8] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[9] Elias Benussi, Andrea Patane, Matthew Wicker, Luca Laurenti, and Marta Kwiatkowska. Individual fairness guarantees for neural networks, 2022.

[10] Reuben Binns. On the apparent conflict between individual and group fairness, 2019.

[11] R. Boutaba, M. A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada Solano, and O. Rendón. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9:1–99, 2018.

[12] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, aug 1996.

[13] Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM Computing Surveys*, 2020.

[14] Alexis Cook. Identifying bias in ai. *Kaggle*, Apr. 2023.

[15] Jeffrey Dastin. Insight - amazon scraps secret ai recruiting tool that showed bias against women. *Reuters*, Oct. 2018.

[16] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine learning in finance*, volume 1170. Springer, 2020.

[17] Alice Doherty, Matthew Wicker, L. Laurenti, and A. Patané. Individual fairness in bayesian neural networks. *ArXiv*, abs/2304.10828, 2023.

[18] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Rich Zemel. Fairness through awareness, 2011.

[19] Chloë Fitzgerald and S. Hurst. Implicit bias in healthcare professionals: a systematic review. *BMC Medical Ethics*, 18, 2017.

[20] M. Flah, Itzel Nunez, Wassim Ben Chaabene, and M. Nehdi. Machine learning algorithms in civil structural health monitoring: A systematic review. *Archives of Computational Methods in Engineering*, 28:2621 – 2643, 2020.

[21] Maite Frutos-Pascual and Begoñya García Zapirain. Review of the use of ai techniques in serious games: Decision making and machine learning. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(2):133–152, 2017.

[22] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. Fairneuron: improving deep neural network fairness with adversary games on selective neurons. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 921–933, New York, NY, USA, 2022. Association for Computing Machinery.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[25] Eszter Hargittai. Whose Space? Differences among Users and Non-Users of Social Network Sites. *Journal of Computer-Mediated Communication*, 13(1):276–297, 10 2007.

[26] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards, 2018.

[27] Max Hort, Zhenpeng Chen, Jie M. Zhang, Mark Harman, and Federica Sarro. Bias mitigation for machine learning classifiers: A comprehensive survey, 2023.

[28] Andrew Hundt, William Agnew, Vicky Zeng, Severin Kacianka, and Matthew Gombolay. Robots enact malignant stereotypes. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22. ACM, June 2022.

[29] Christina Ilvento. Metric learning for individual fairness, 2020.

[30] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. Verifying individual fairness in machine learning models, 2020.

[31] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, May 2022.

[32] Faisal Kamiran and Toon Calders. Classification with no discrimination by preferential sampling. 2010.

[33] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650, 2011.

[34] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Towards proving the adversarial robustness of deep neural networks. *Electronic Proceedings in Theoretical Computer Science*, 257:19–26, September 2017.

[35] Justin Ker, Lipo Wang, J. Rao, and Tchoyoson C. C. Lim. Deep learning applications in medical image analysis. *IEEE Access*, 6:9375–9389, 2018.

[36] Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and scalable bayesian deep learning by weight-perturbation in adam, 2018.

[37] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness, 2018.

[38] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics Data Analysis*, 142:106816, 2020.

[39] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. ifair: Learning individually fair data representations for algorithmic decision making. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1334–1345, 2019.

[40] Xuran Li, Peng Wu, Kaixiang Dong, and Zhen Zhang. Robustfair: Adversarial evaluation through fairness confusion directed gradient search. *ArXiv*, abs/2305.10906, 2023.

[41] Maxwell W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16:321–332, 2015.

[42] David Madras, Toni Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[43] Subha Maity, Songkai Xue, Mikhail Yurochkin, and Yuekai Sun. Statistical inference for individual fairness. *arXiv preprint arXiv:2103.16714*, 2021.

[44] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021.

[45] Aditya Krishna Menon and Robert C. Williamson. The cost of fairness in binary classification. In Sorelle A. Friedler and Christo Wilson, editors, *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 107–118. PMLR, 23–24 Feb 2018.

[46] Cade Metz. Who is making sure the a.i. machines aren't racist? Mar. 2021.

[47] Kiarash Mohammadi, Aishwarya Sivaraman, and Golnoosh Farnadi. Feta: Fairness enforced verifying, training, and predicting algorithms for neural networks, 2023.

[48] Adel Nehme. How to use chatgpt code interpreter, 2023.

[49] European Parliament News. Eu ai act: first regulation on artificial intelligence. Dec. 2023.

[50] Szu Yu Chen Nitasha Tiku, Kevin Schaul. These fake images reveal how ai amplifies our worst stereotypes. *Washington Post*, Nov. 2023.

[51] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.

[52] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Emre Kıcıman. Social data: Biases, methodological pitfalls, and ethical boundaries. *Frontiers in Big Data*, 2, 2019.

[53] Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical deep learning with bayesian principles, 2019.

[54] Ravi B. Parikh, Stephanie Teeple, and Amol S. Navathe. Addressing Bias in Artificial Intelligence in Health Care. *JAMA*, 322(24):2377–2378, 12 2019.

[55] Dino Pedreschi, Luca Pappalardo, Ricardo Baeza-Yates, Albert-Laszlo Barabasi, Frank Dignum, Virginia Dignum, Tina Eliassi-Rad, Fosca Giannotti, Janos Kertesz, Alistair Knott, Yannis Ioannidis, Paul Lukowicz, Andrea Passarella, Alex Sandy Pentland, John Shawe-Taylor, and Alessandro Vespignani. Social ai and the challenges of the human-ai ecosystem, 2023.

[56] Andrija Petrović, Mladen Nikolić, Sandro Radovanović, Boris Delibašić, and Miloš Jovanović. Fair: Fair adversarial instance re-weighting. *Neurocomputing*, 476:14–37, 2022.

[57] Yuji Roh, Kangwook Lee, Steven Whang, and Changho Suh. Sample selection for fair and robust training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 815–827. Curran Associates, Inc., 2021.

[58] Markus Schedl. Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics*, 5, 2019.

[59] Bing Sun, Jun Sun, Ting Dai, and Lijun Zhang. Probabilistic verification of neural networks against group fairness, 2021.

[60] Harini Suresh and John Guttag. A framework for understanding sources of harm throughout the machine learning life cycle. In *Equity and Access in Algorithms, Mechanisms, and Optimization*, EAAMO '21. ACM, October 2021.

[61] Shubhi Upadhyay. Algorithmic bias and its impact on society. *Medium*, Feb. 2023.

[62] Ana Valdivia, Javier Sánchez-Monedero, and Jorge Casillas. How fair can we go in machine learning? assessing the boundaries of accuracy and fairness. *International Journal of Intelligent Systems*, 36(4):1619–1643, 2021.

[63] Yuyan Wang, Xuezhi Wang, Alex Beutel, Flavien Prost, Jilin Chen, and Ed H. Chi. Understanding and improving fairness-accuracy trade-offs in multi-task learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1748–1757, New York, NY, USA, 2021. Association for Computing Machinery.

[64] Zachary B. Wolf. Ai can be racist, sexist and creepy. what should we do about it? *CNN*, March 2023.

[65] Samuel Yeom and Matt Fredrikson. Individual fairness revisited: Transferring techniques from adversarial robustness, 2020.

[66] Mikhail Yurochkin, Amanda Bower, and Yuekai Sun. Training individually fair ml models with sensitive subspace robustness, 2020.

[67] Xianli Zeng, Edgar Dobriban, and Guang Cheng. Bayes-optimal classifiers under group fairness, 2022.

[68] Hua Zhong, Lin Liu, and Shicheng Liao. A survey on bayesian neural networks. *Advances in Computer, Signals and Systems*, 2022.

Appendix A

Graphs of Results

The following pages contain the graphs for each measured metric for each ϵ tested. All metrics are averages over 10 trials, as outlined in Section 5.1. The graphs go in order of increasing ϵ value - 0 to 0.2. Note we do not include the individual ϵ plots for every architecture, such as in Figure 6.8, as there too many. All graphs, including the ϵ plots, can be viewed at `https://github.com/tstephen22/Fairness-Through-Uncertainty`.

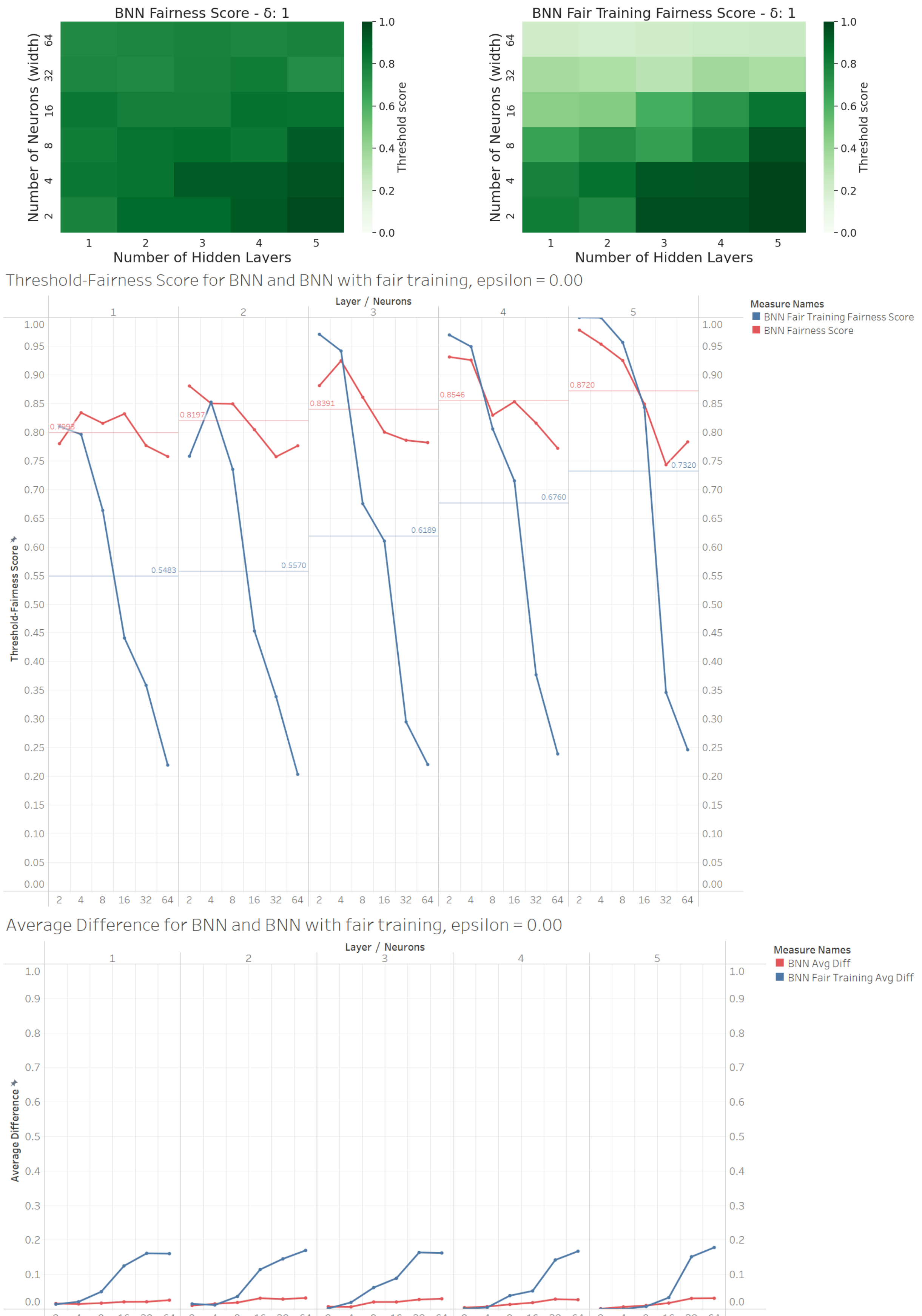
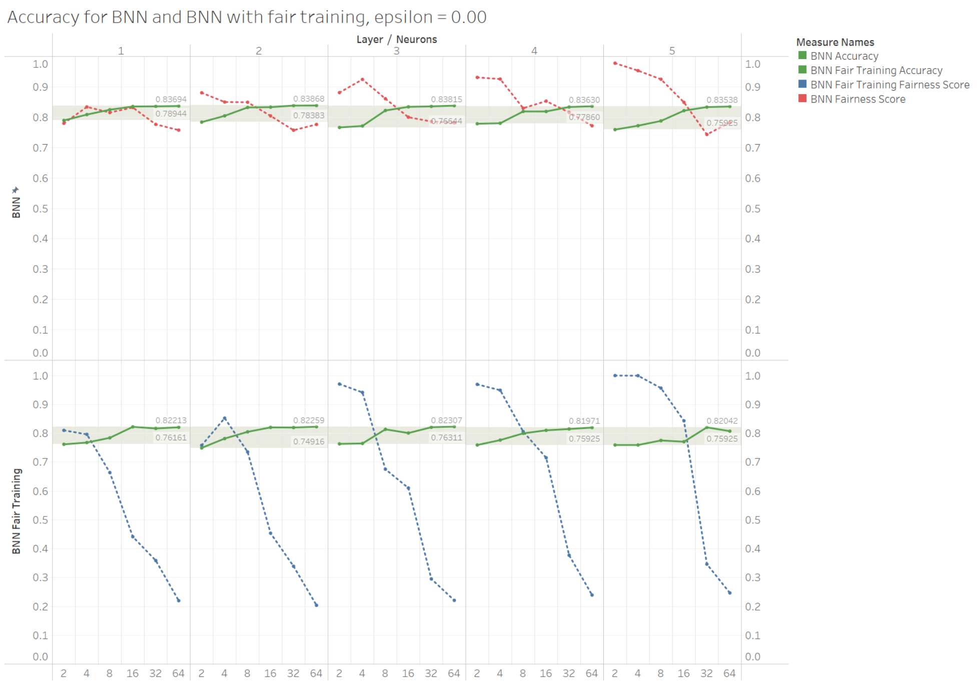


Figure A.1: Fairness metrics for $\epsilon_{fair} = 0$. Constant lines represent averages and bands represent min-max range.



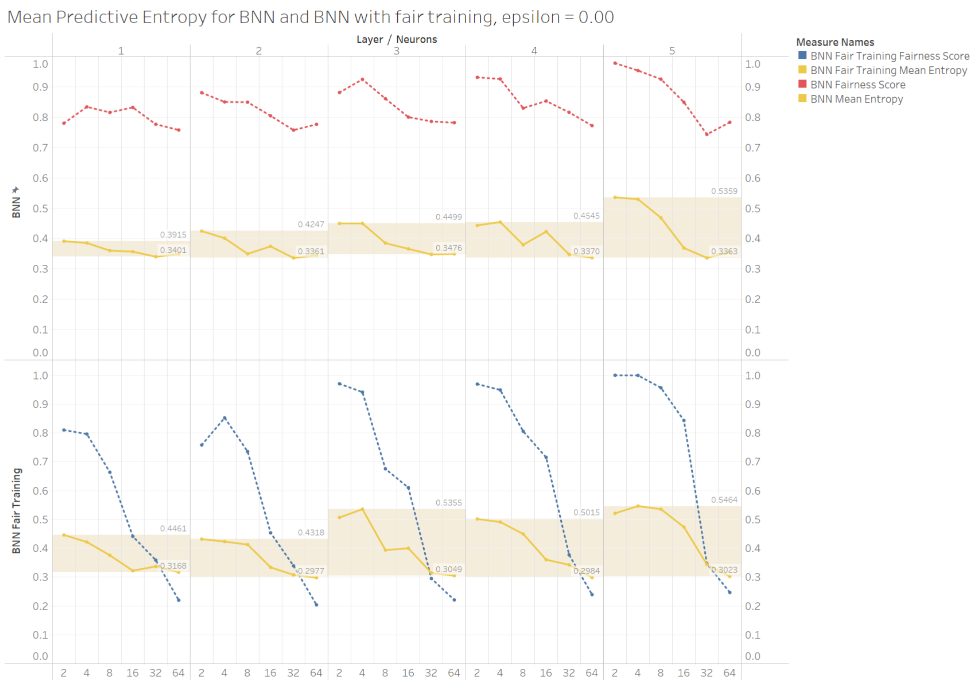


Figure A.2: Accuracy and Predictive Entropy metrics for $\epsilon_{fair} = 0$. Constant lines represent averages and bands represent min-max range.

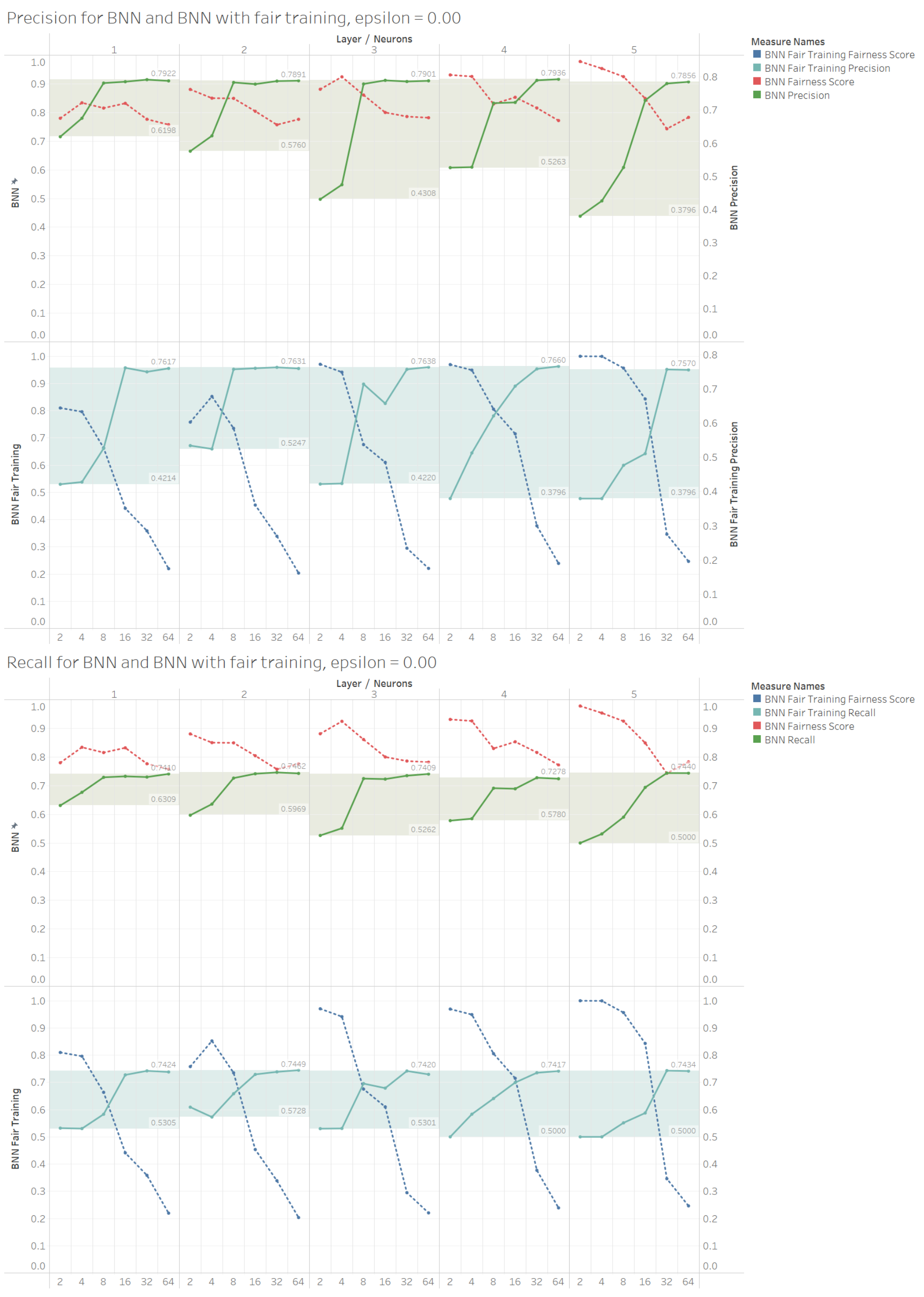


Figure A.3: Precision and Recall metrics for $\epsilon_{fair} = 0$. Constant lines represent averages and bands represent min-max range.

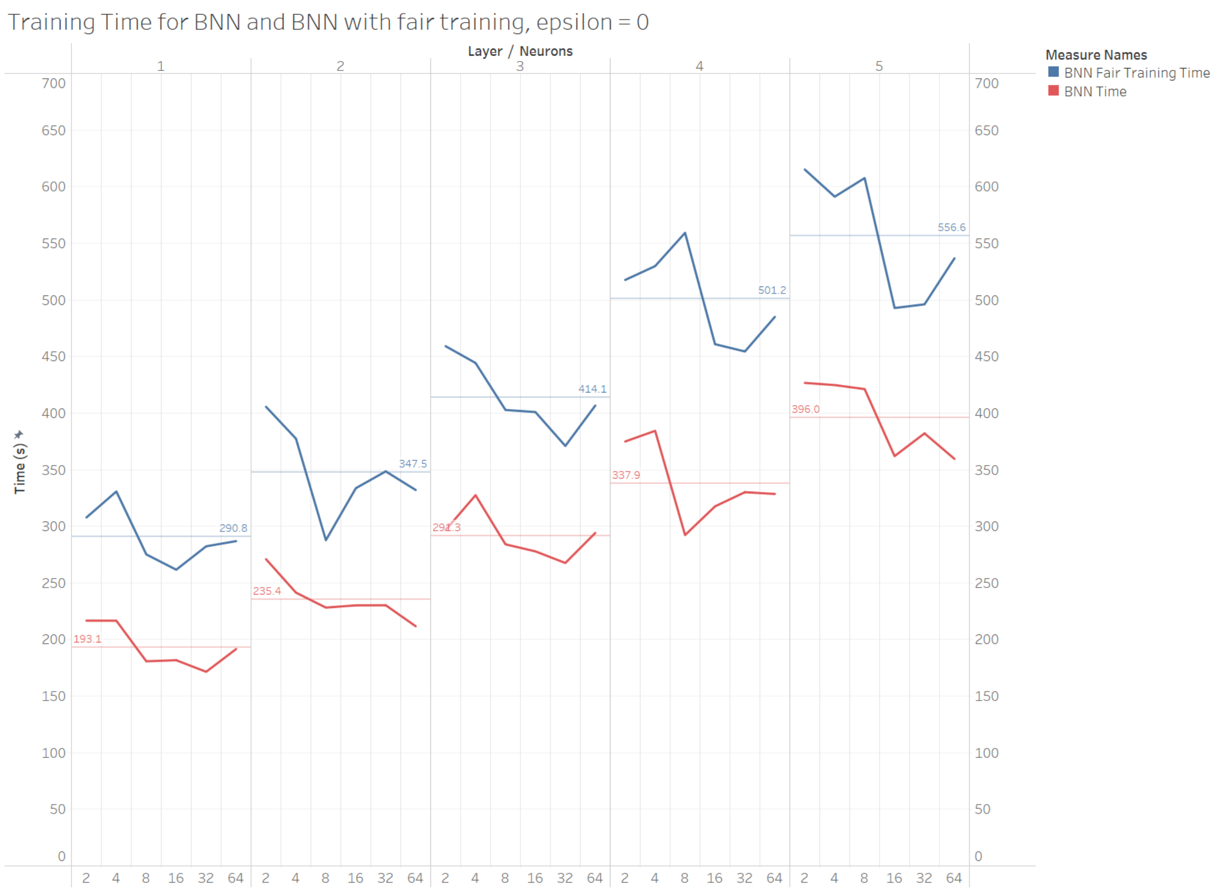
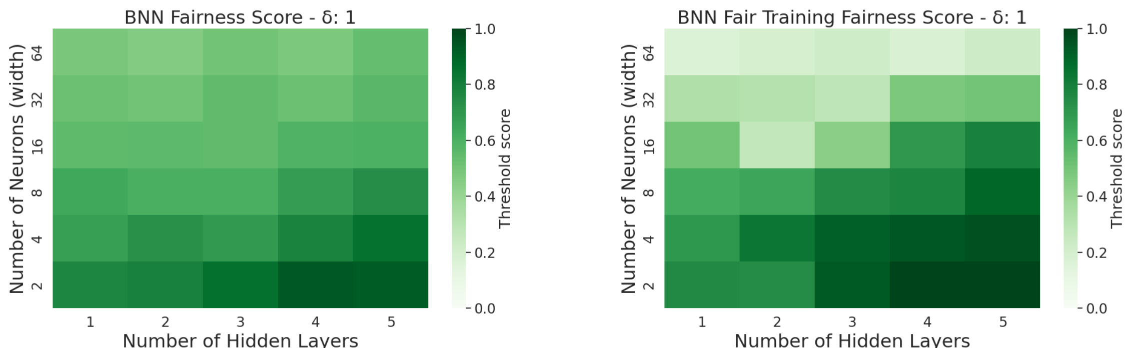
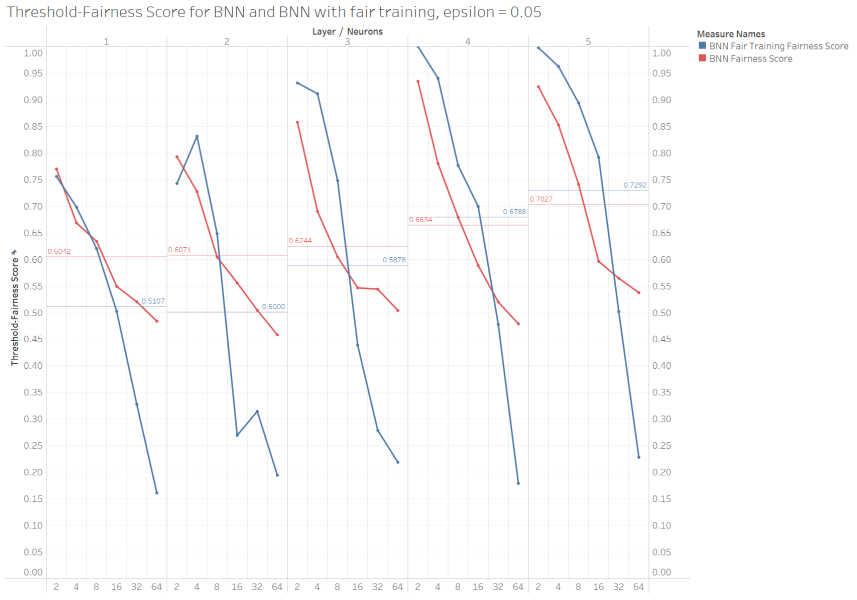


Figure A.4: Training time for $\epsilon_{fair} = 0$. Constant lines represent averages and bands represent min-max range.





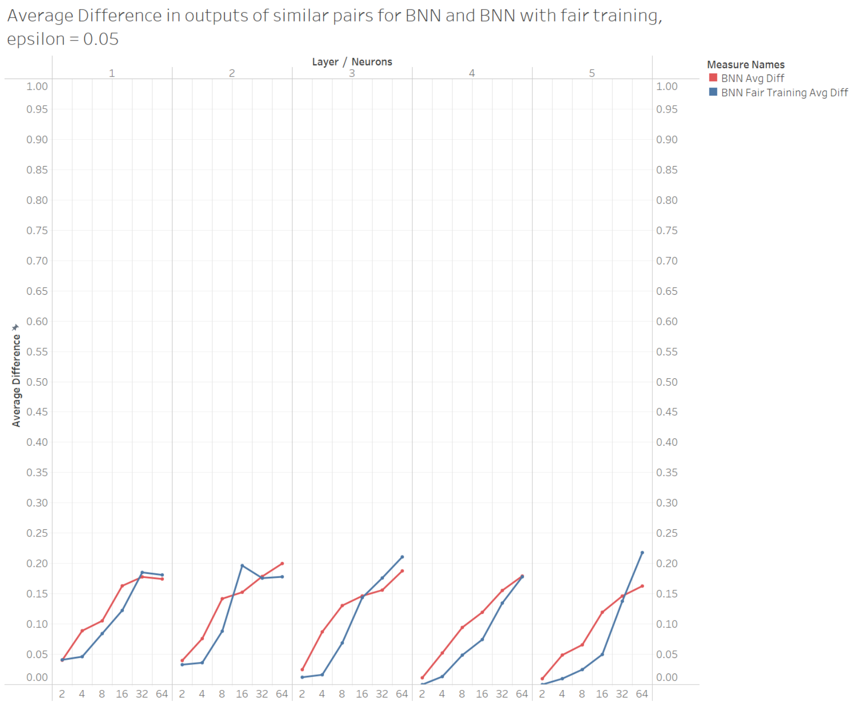
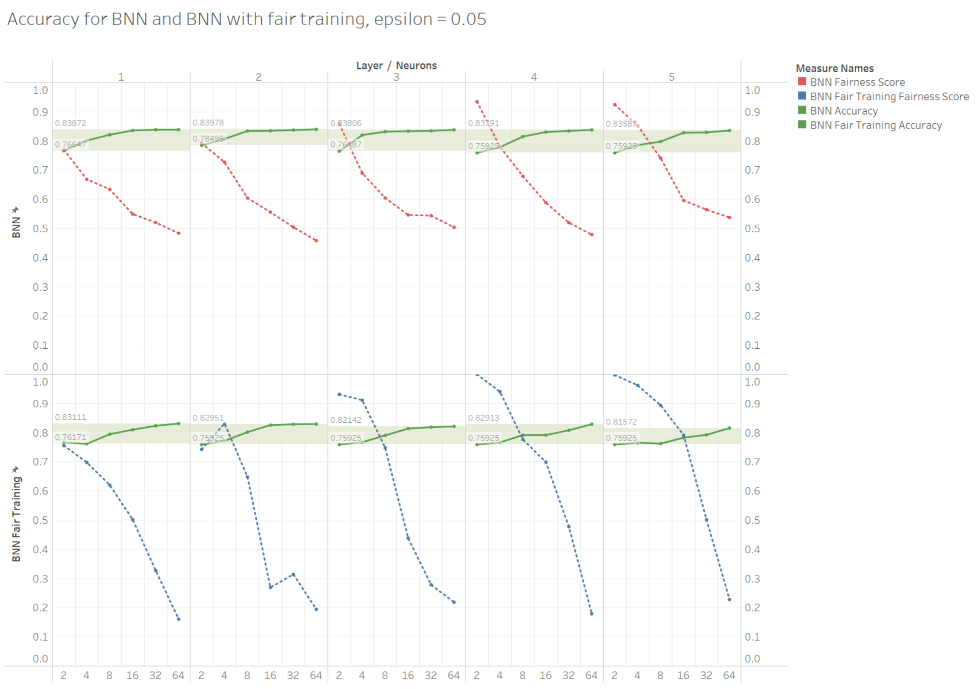


Figure A.5: Fairness metrics for $\epsilon_{fair} = 0.05$. Constant lines represent averages and bands represent min-max range.



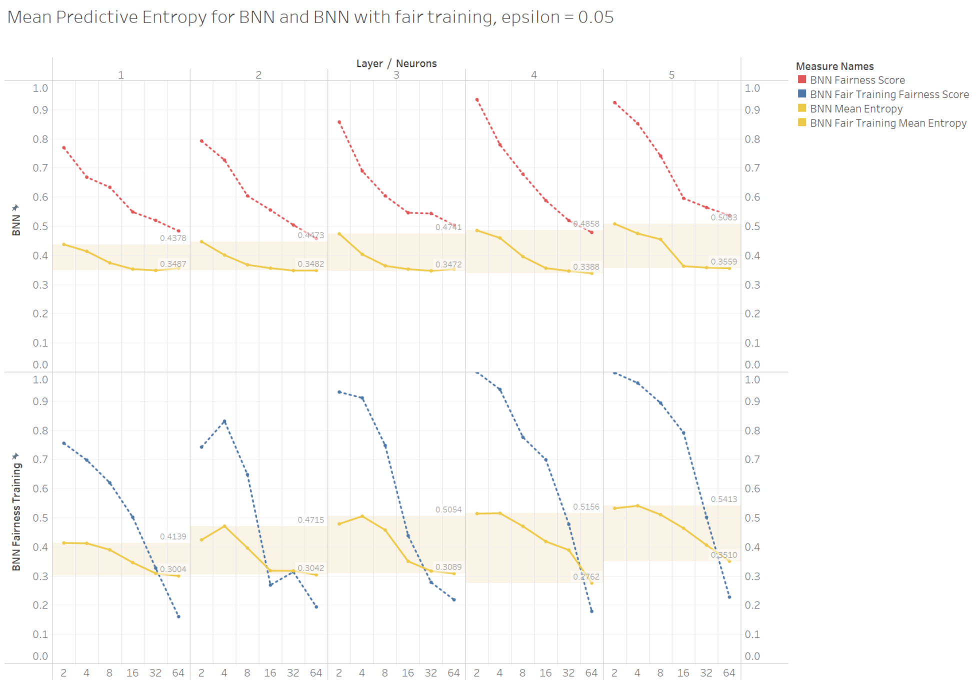
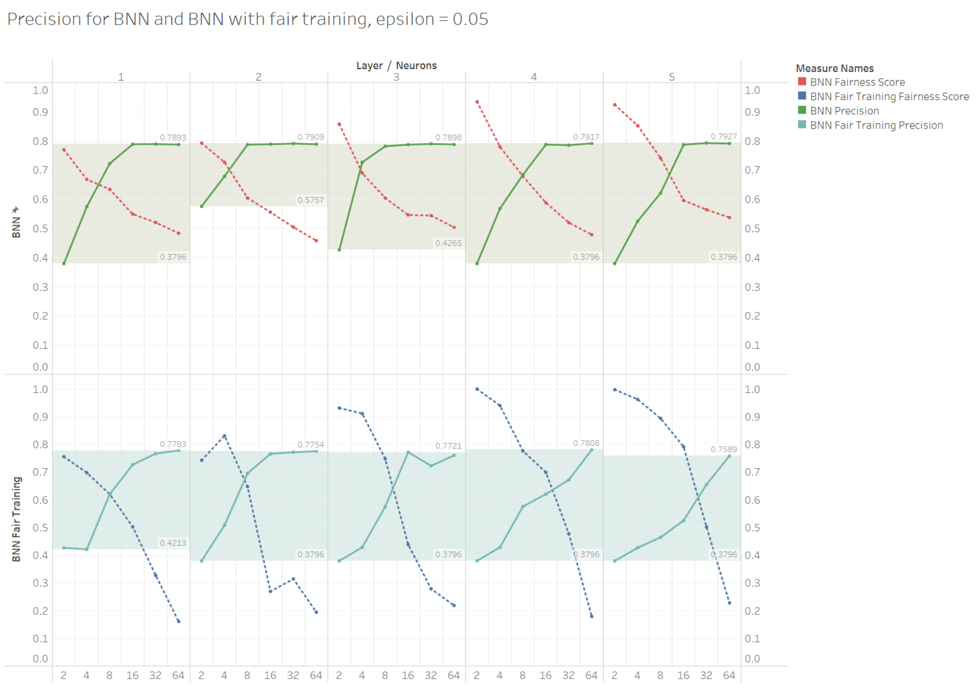


Figure A.6: Accuracy and Predictive Entropy metrics for $\epsilon_{fair} = 0.05$. Constant lines represent averages and bands represent min-max range.



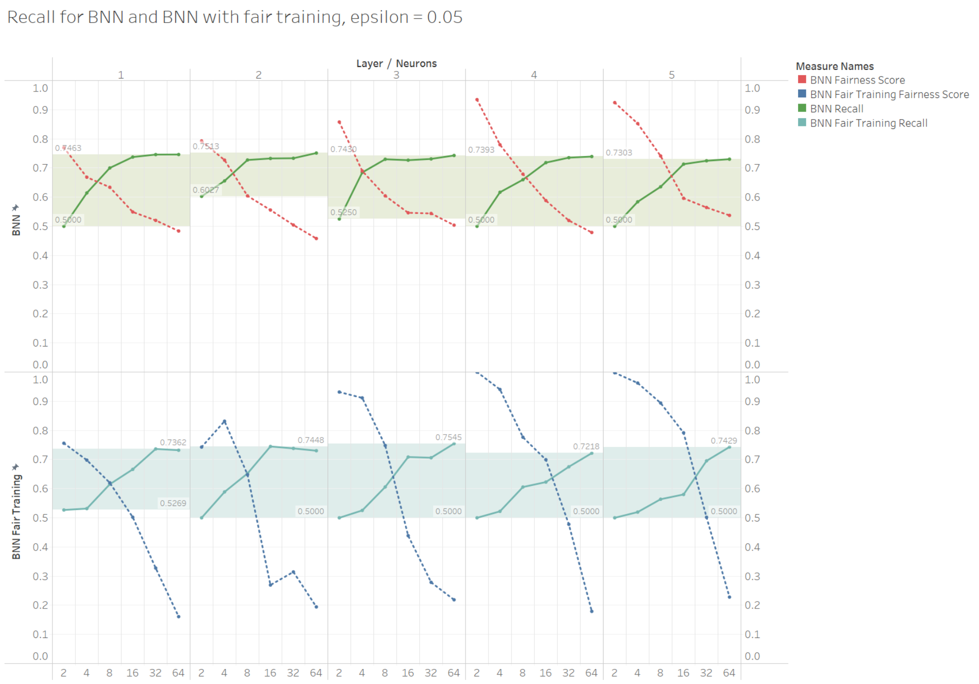


Figure A.7: Precision and Recall metrics for $\epsilon_{fair} = 0.05$. Constant lines represent averages and bands represent min-max range.

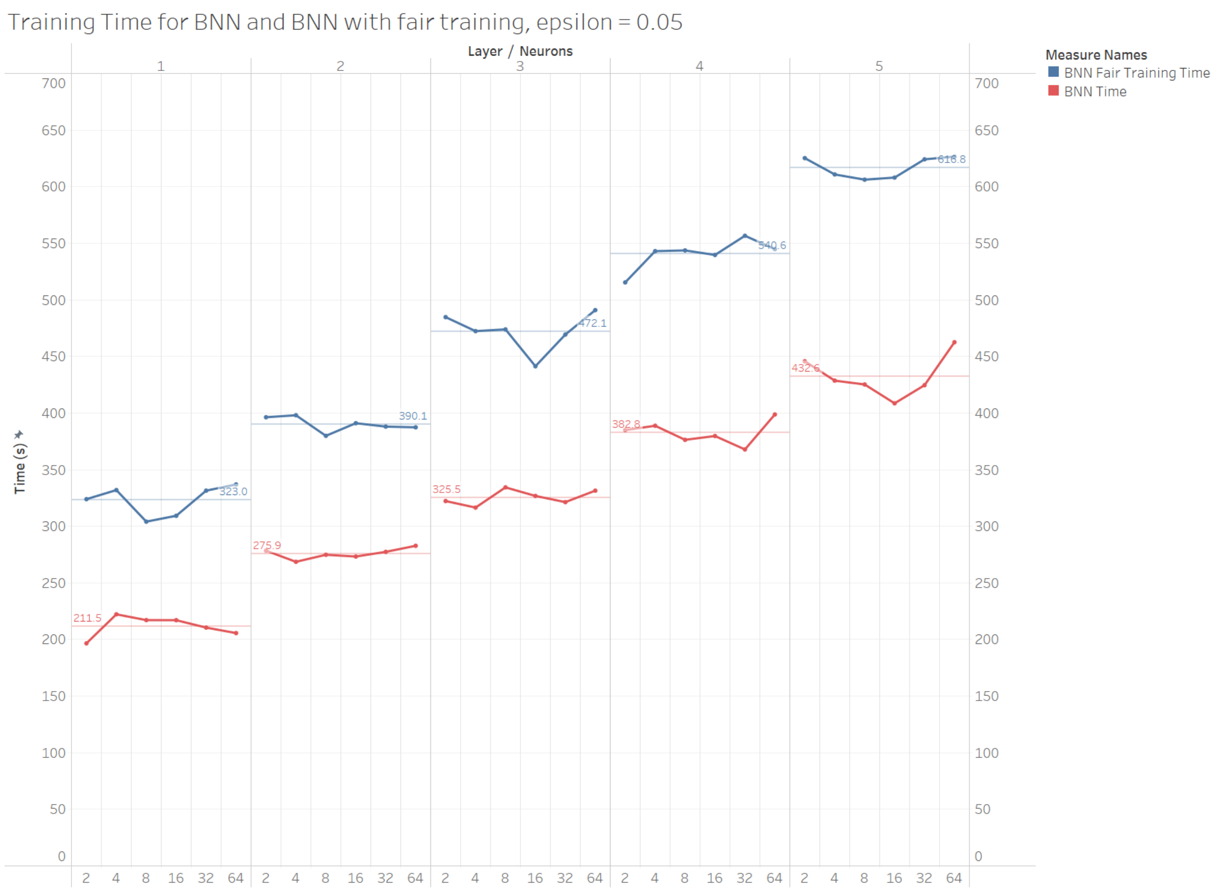


Figure A.8: Training time for $\epsilon_{fair} = 0.05$. Constant lines represent averages and bands represent min-max range.

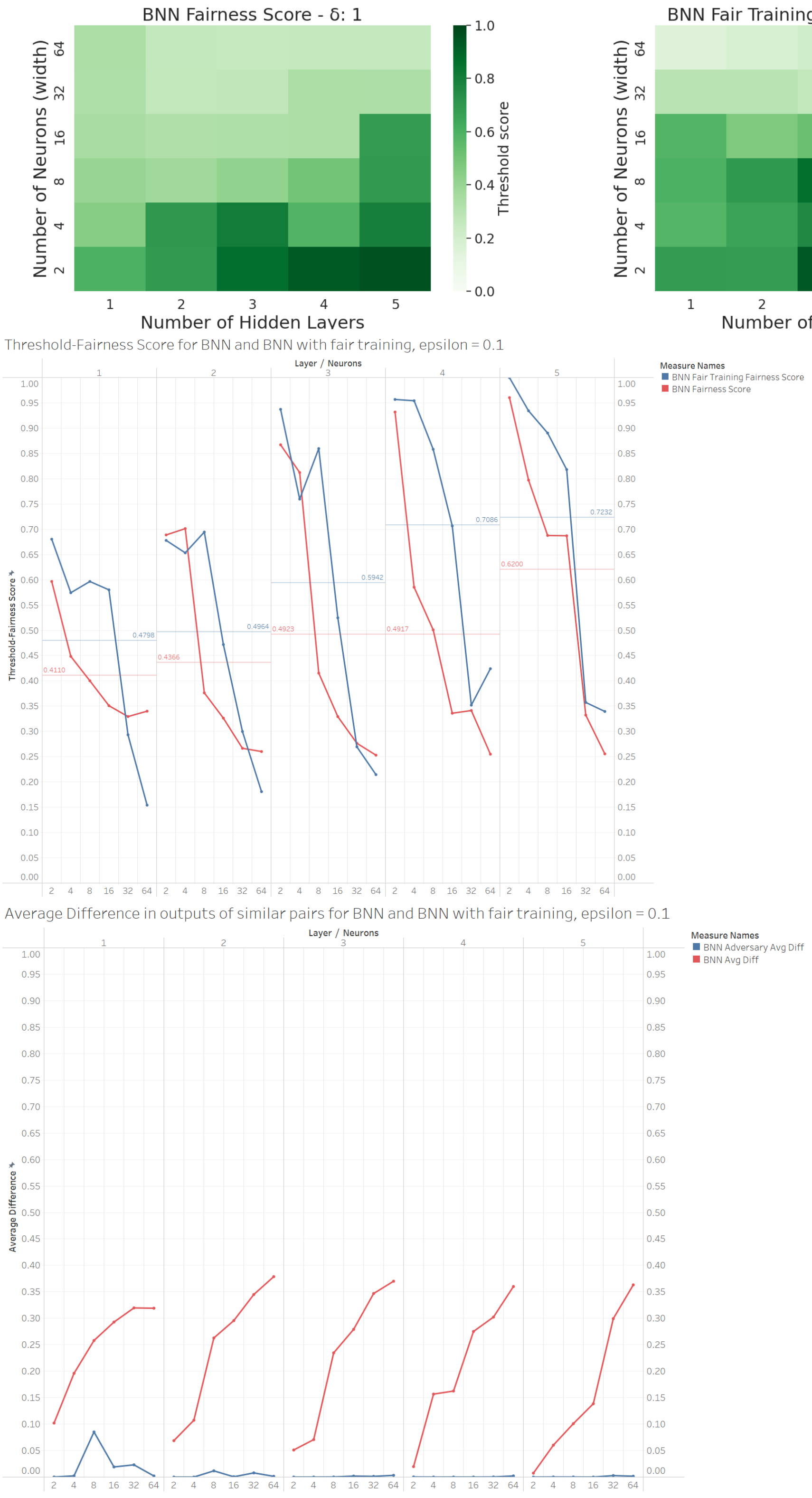
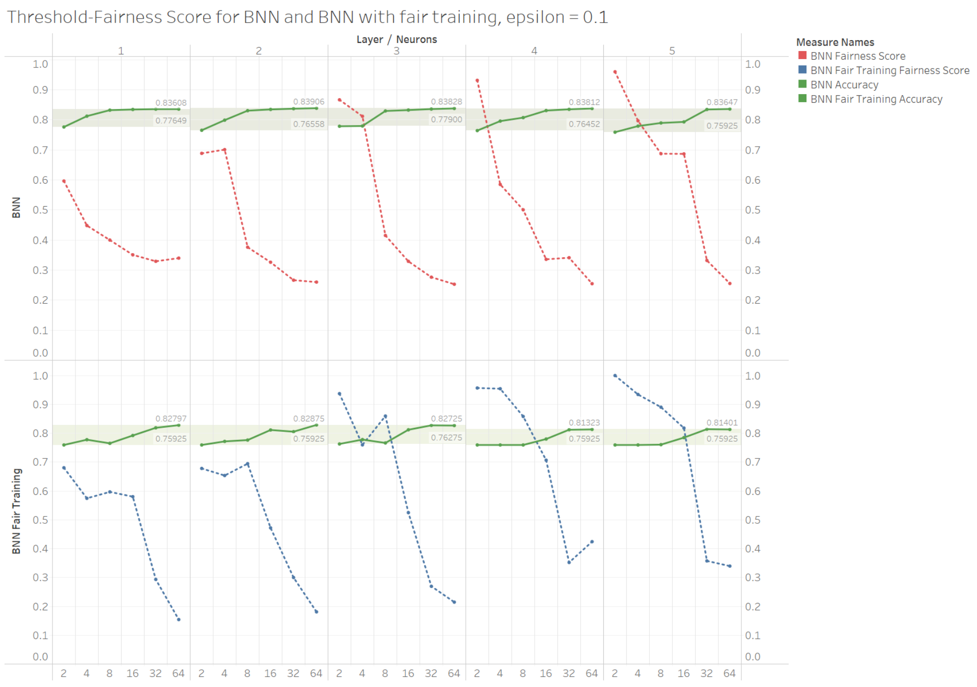


Figure A.9: Fairness metrics for $\epsilon_{fair} = 0.1$. Constant lines represent averages and bands represent min-max range.



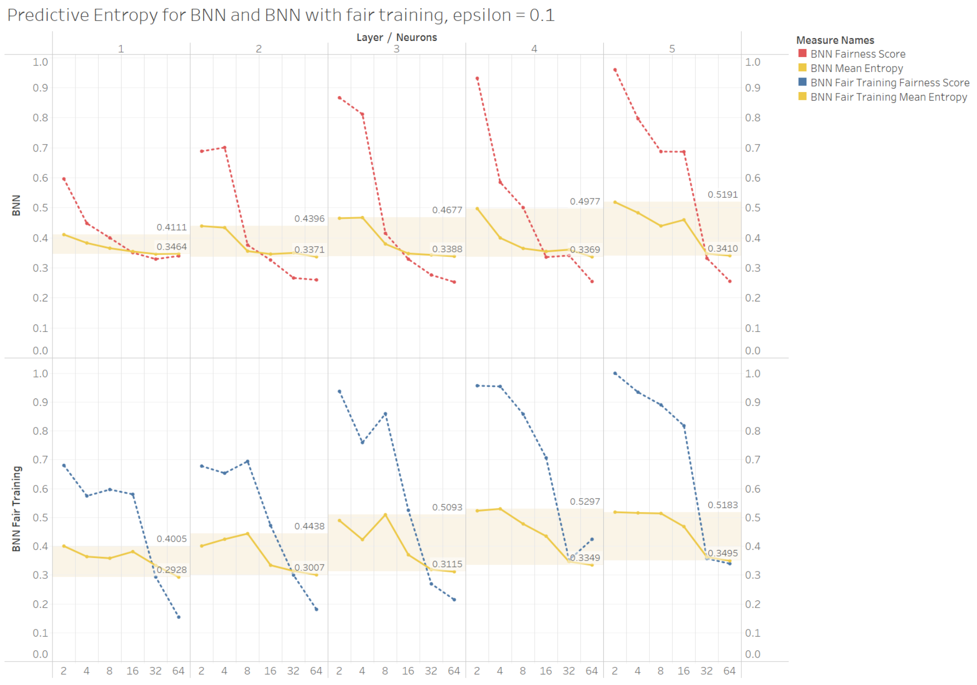
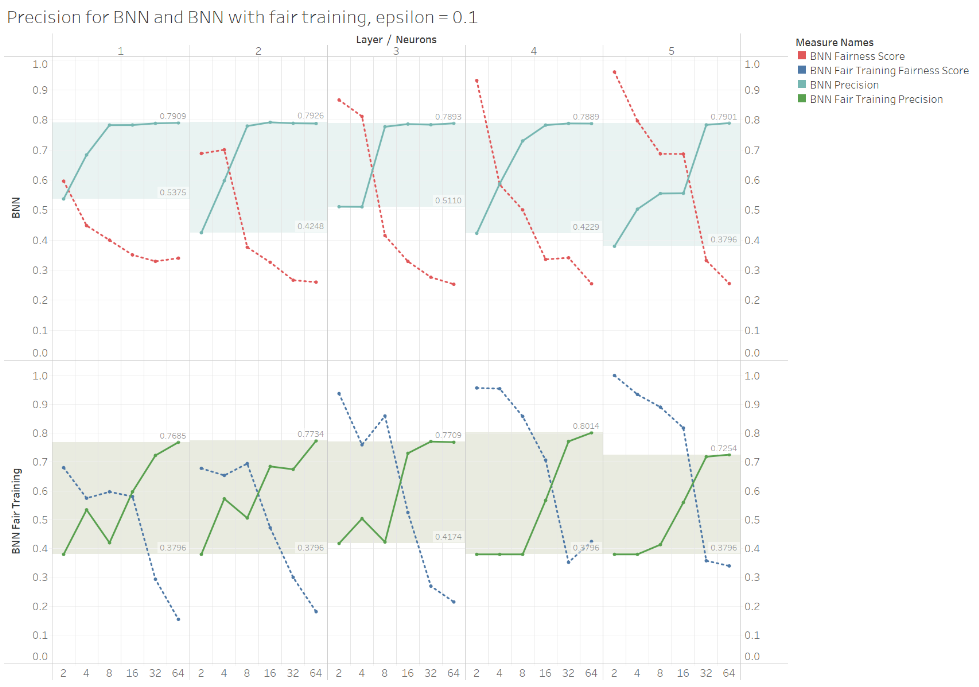


Figure A.10: Accuracy and Predictive Entropy metrics for $\epsilon_{fair} = 0.1$. Constant lines represent averages and bands represent min-max range.



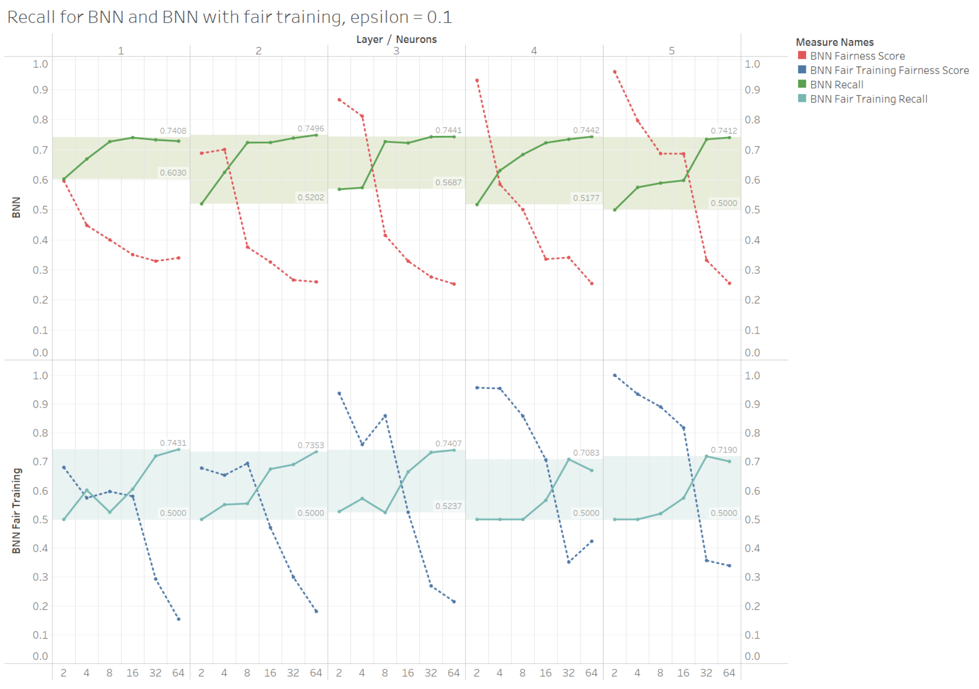


Figure A.11: Precision and Recall metrics for $\epsilon_{fair} = 0.1$. Constant lines represent averages and bands represent min-max range.

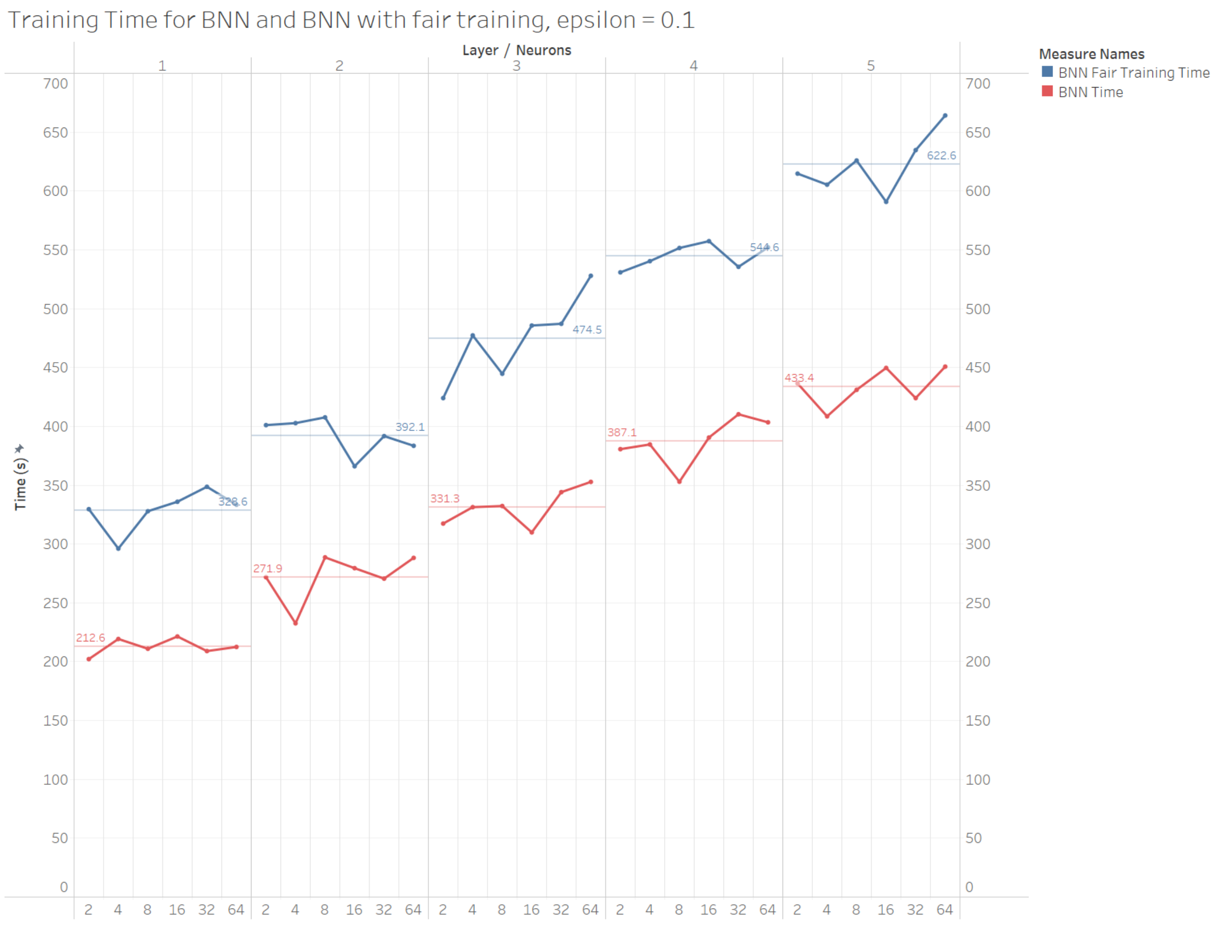
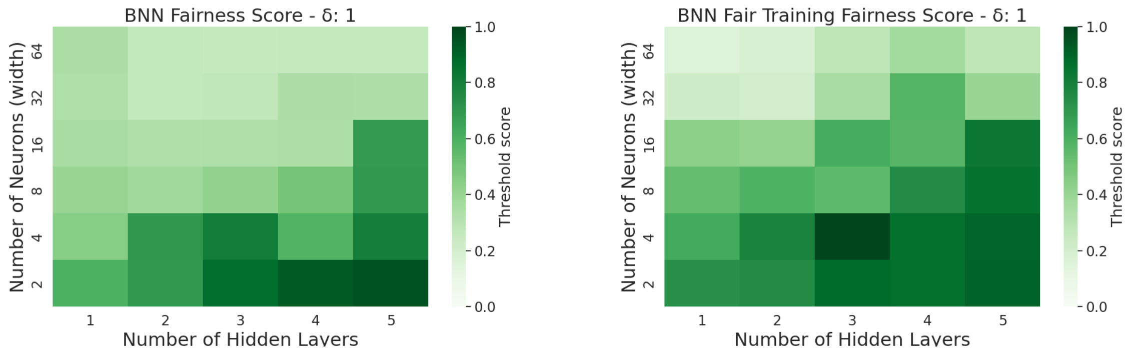
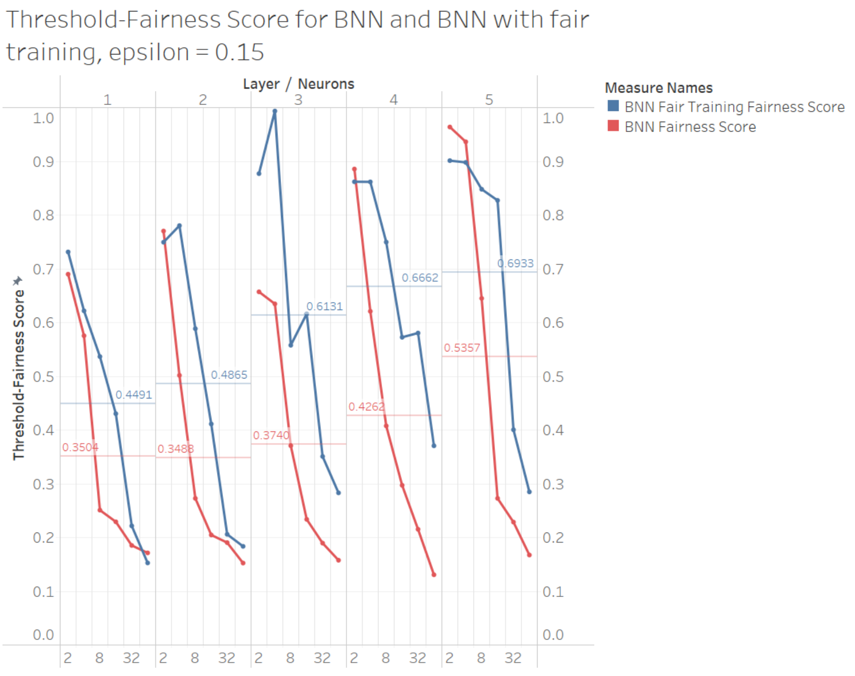


Figure A.12: Training time for $\epsilon_{fair} = 0.1$. Constant lines represent averages and bands represent min-max range.





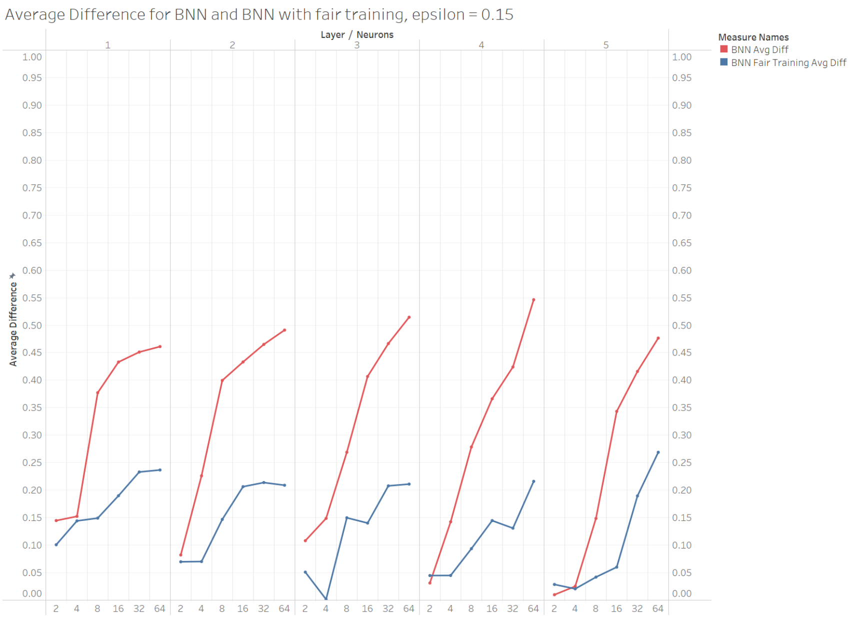
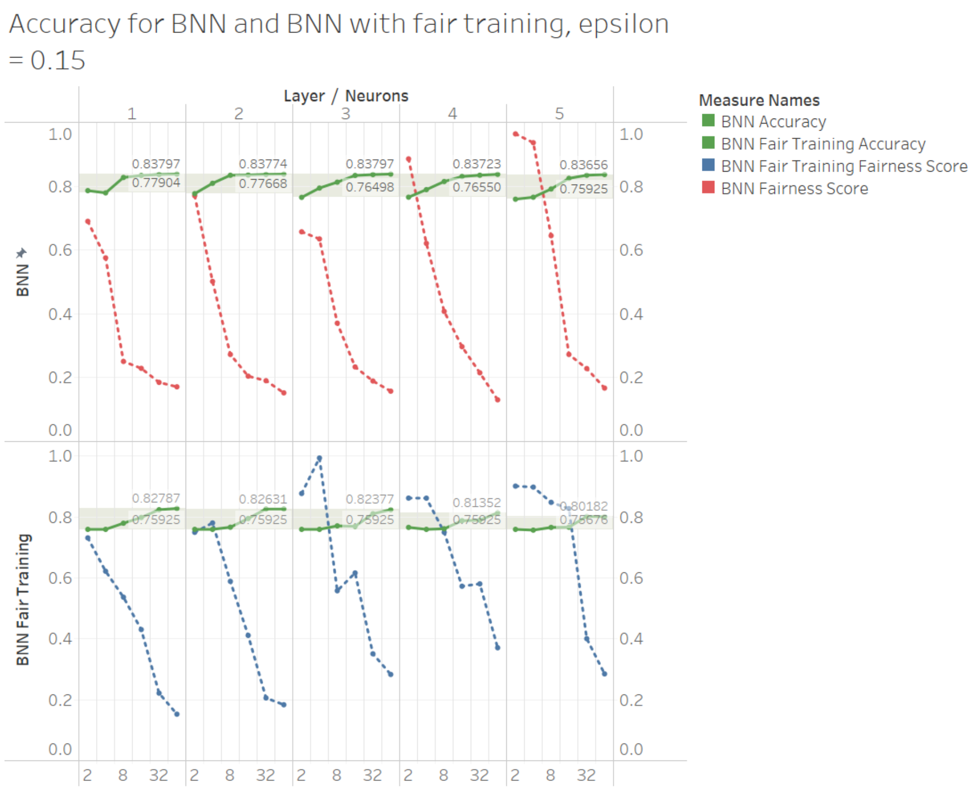


Figure A.13: Fairness metrics for $\epsilon_{fair} = 0.15$. Constant lines represent averages and bands represent min-max range.



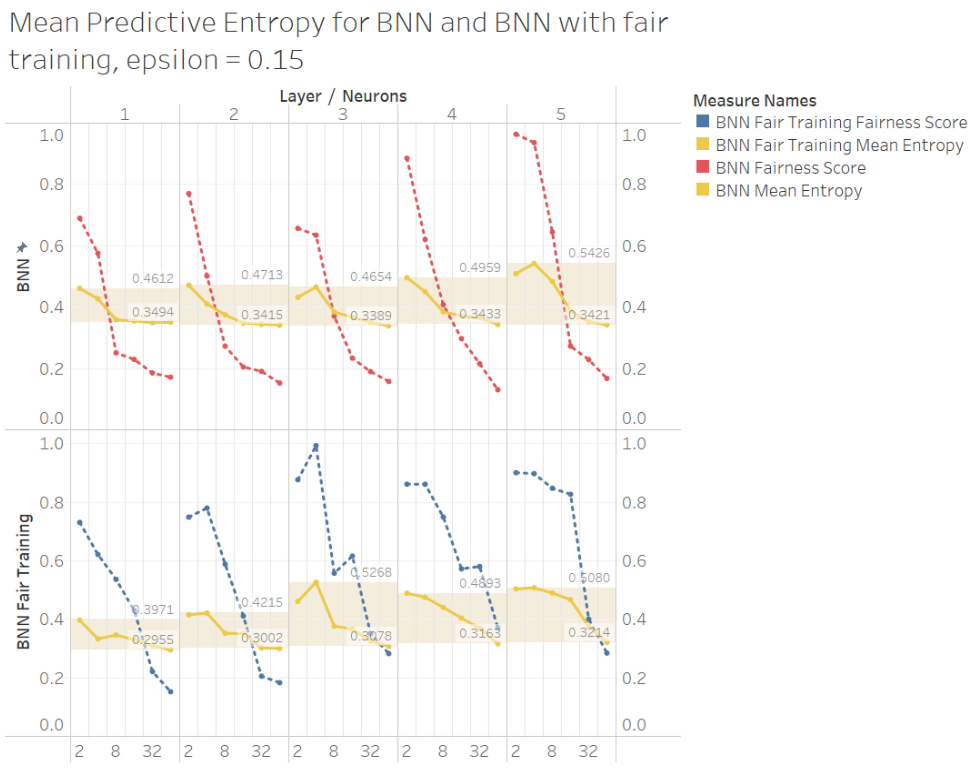
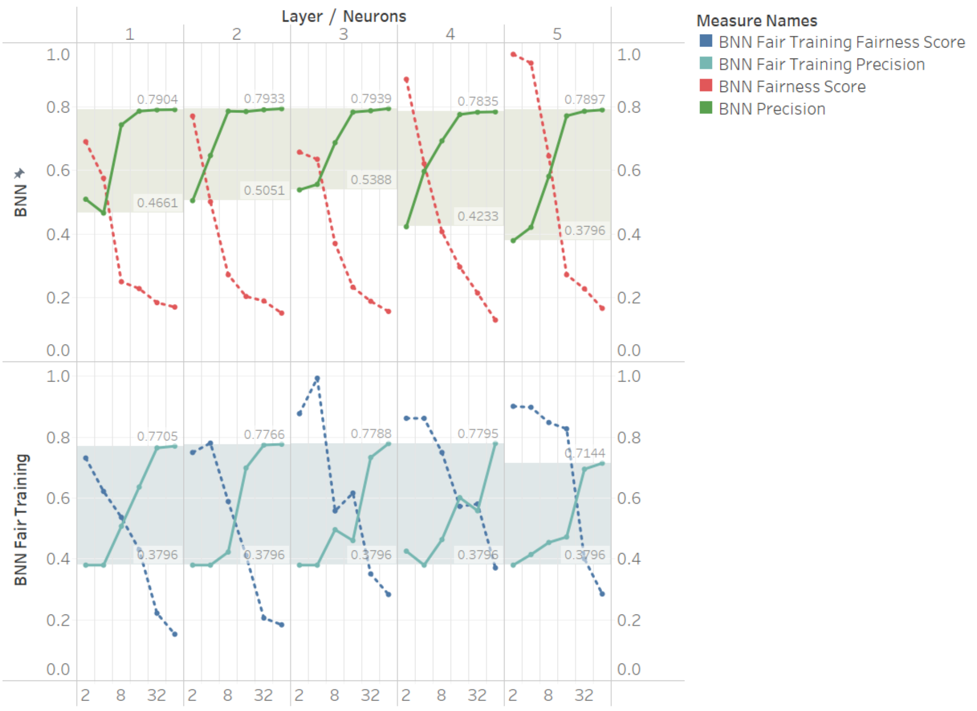


Figure A.14: Accuracy and Predictive Entropy metrics for $\epsilon_{fair} = 0.15$. Constant lines represent averages and bands represent min-max range.

Precision for BNN and BNN with fair training, epsilon = 0.15



Recall for BNN and BNN with fair training, epsilon = 0.15

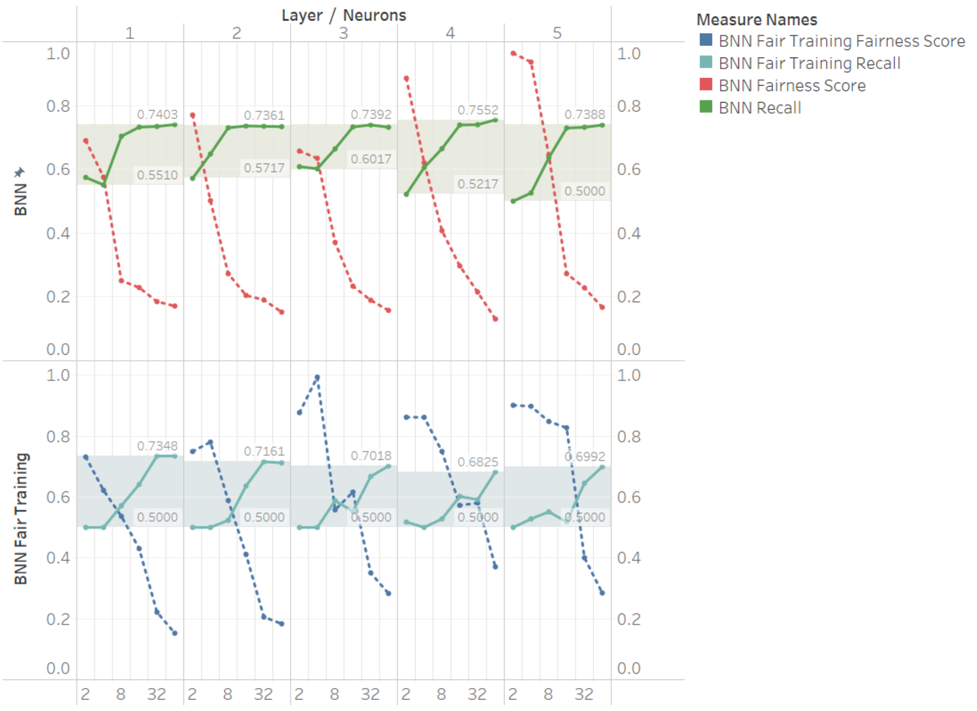


Figure A.15: Precision and Recall metrics for $\epsilon_{fair} = 0.15$. Constant lines represent averages and bands represent min-max range.

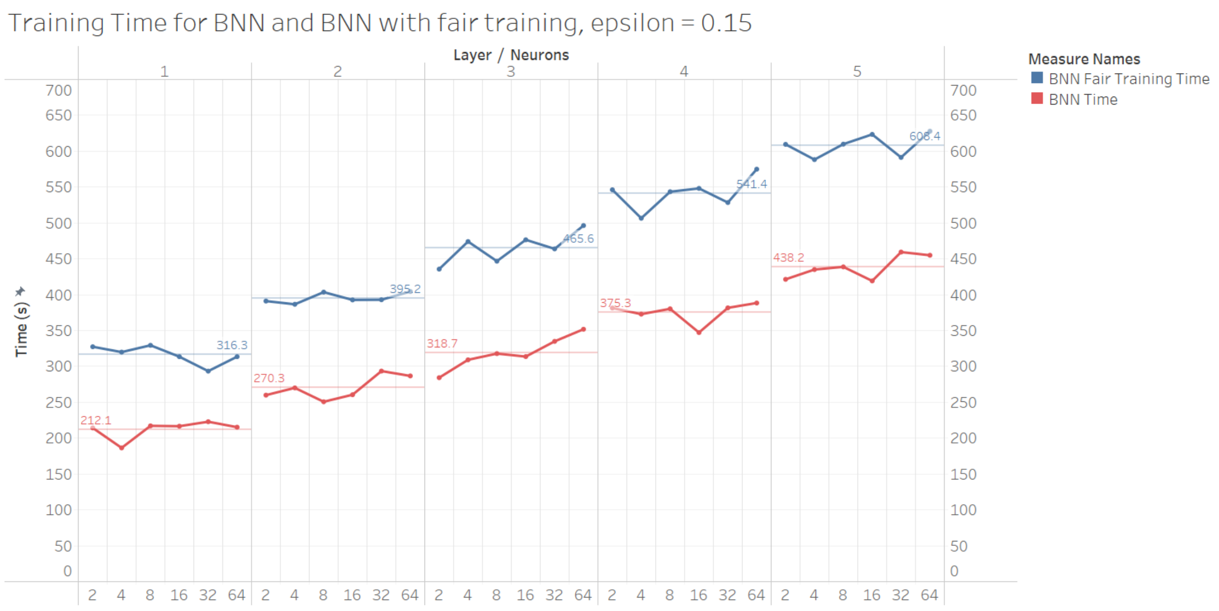
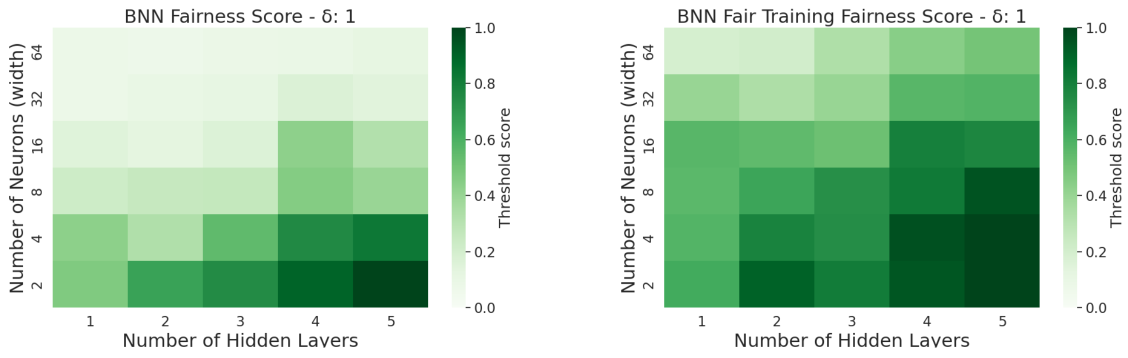
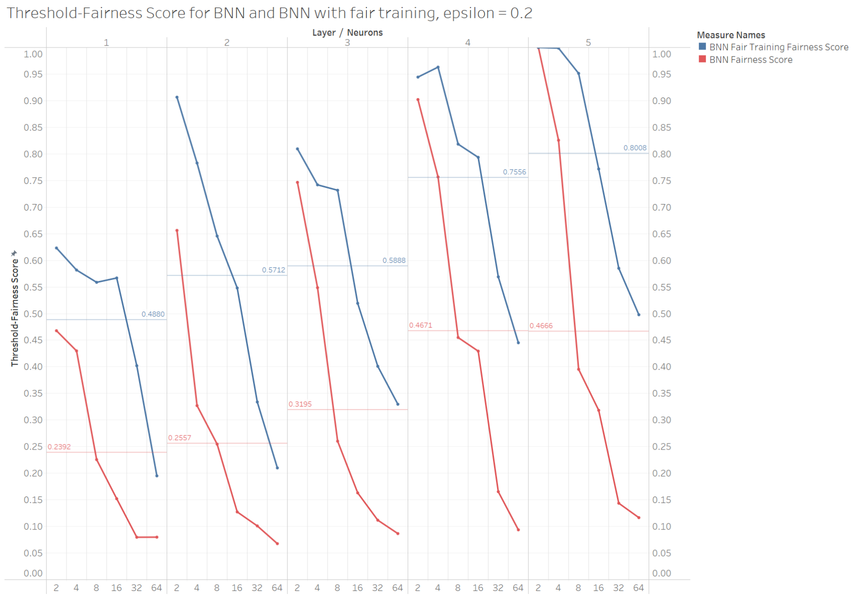


Figure A.16: Training time for $\epsilon_{fair} = 0.15$. Constant lines represent averages and bands represent min-max range.





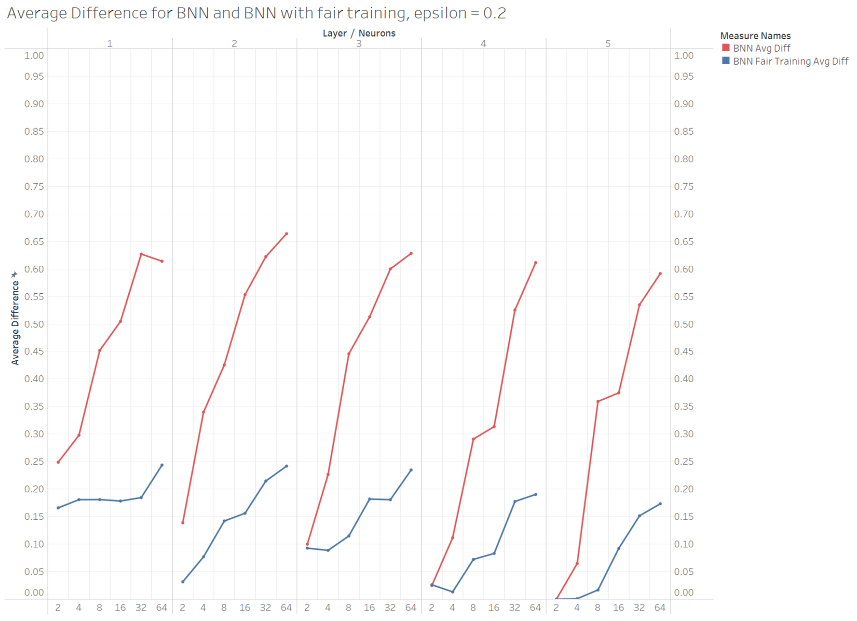


Figure A.17: Fairness metrics for $\epsilon_{fair} = 0.2$. Constant lines represent averages and bands represent min-max range.

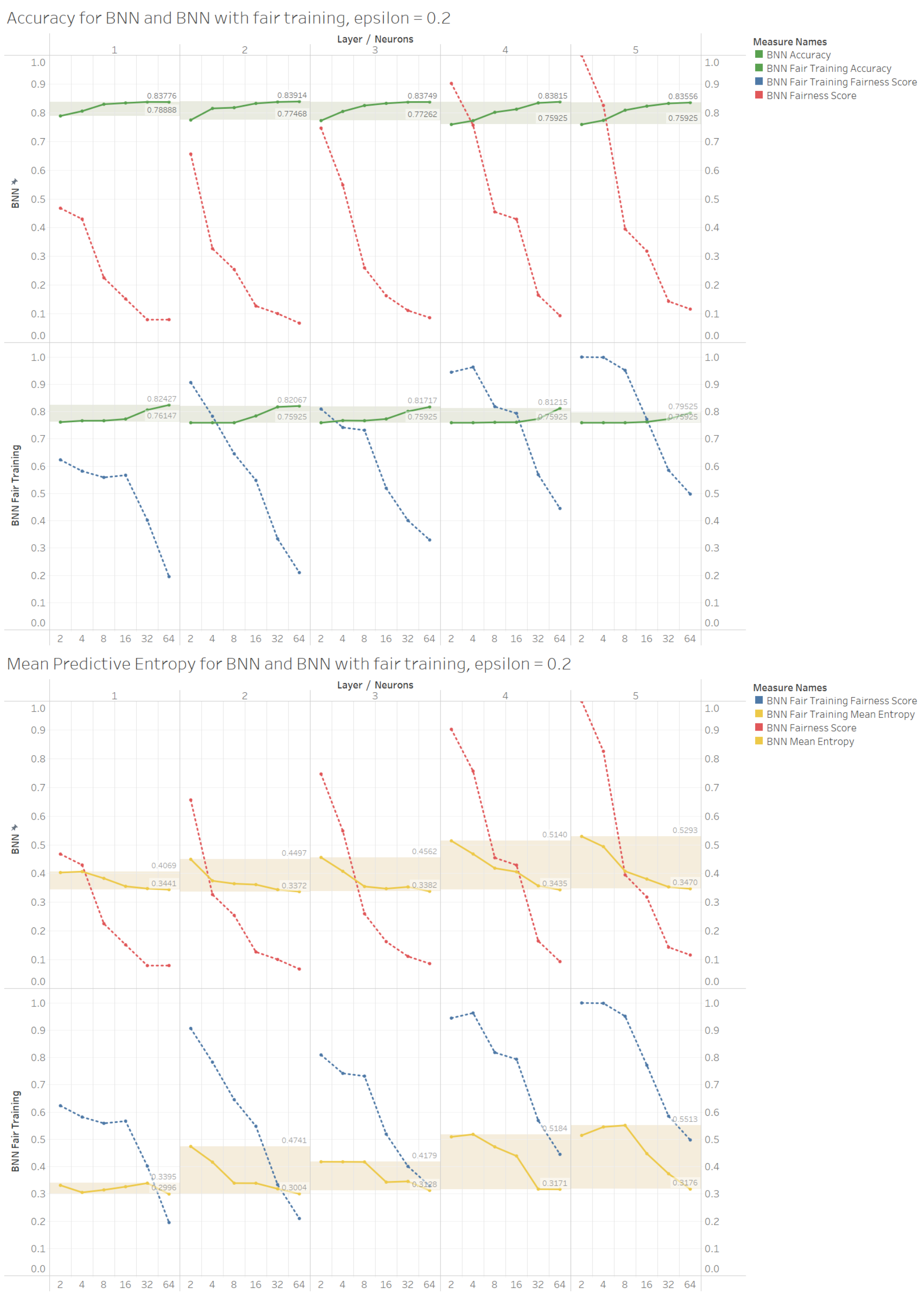
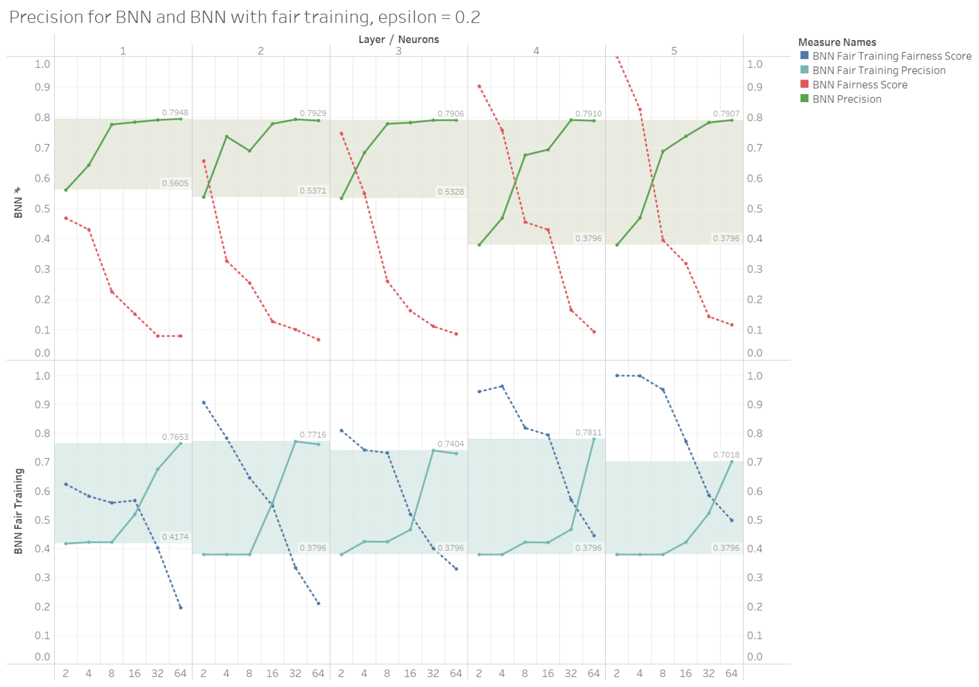


Figure A.18: Accuracy and Predictive Entropy metrics for $\epsilon_{fair} = 0.2$. Constant lines represent averages and bands represent min-max range.



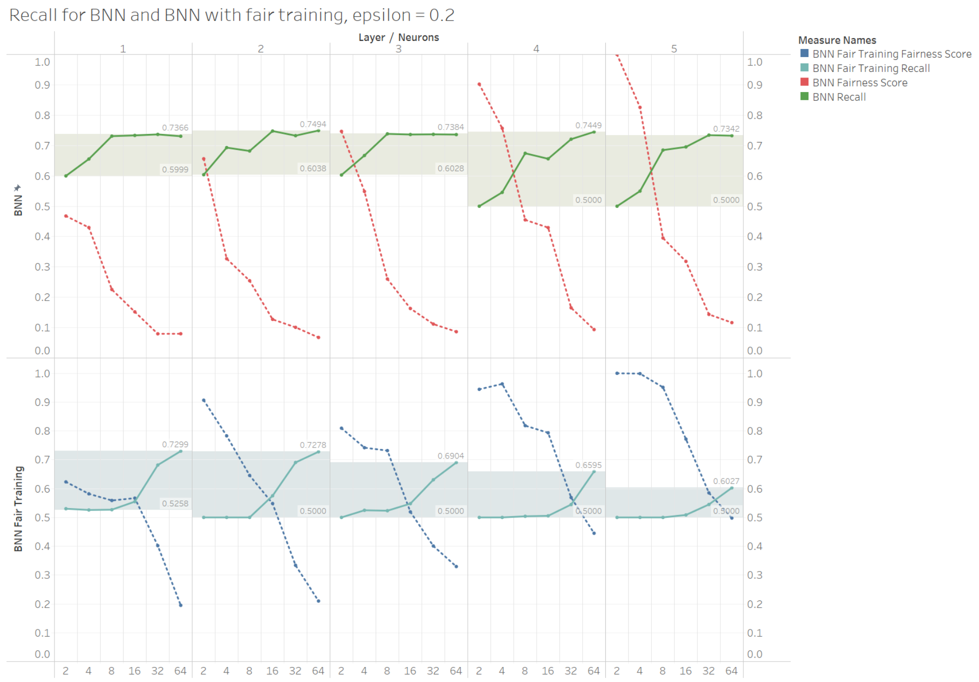


Figure A.19: Precision and Recall metrics for $\epsilon_{fair} = 0.2$. Constant lines represent averages and bands represent min-max range.

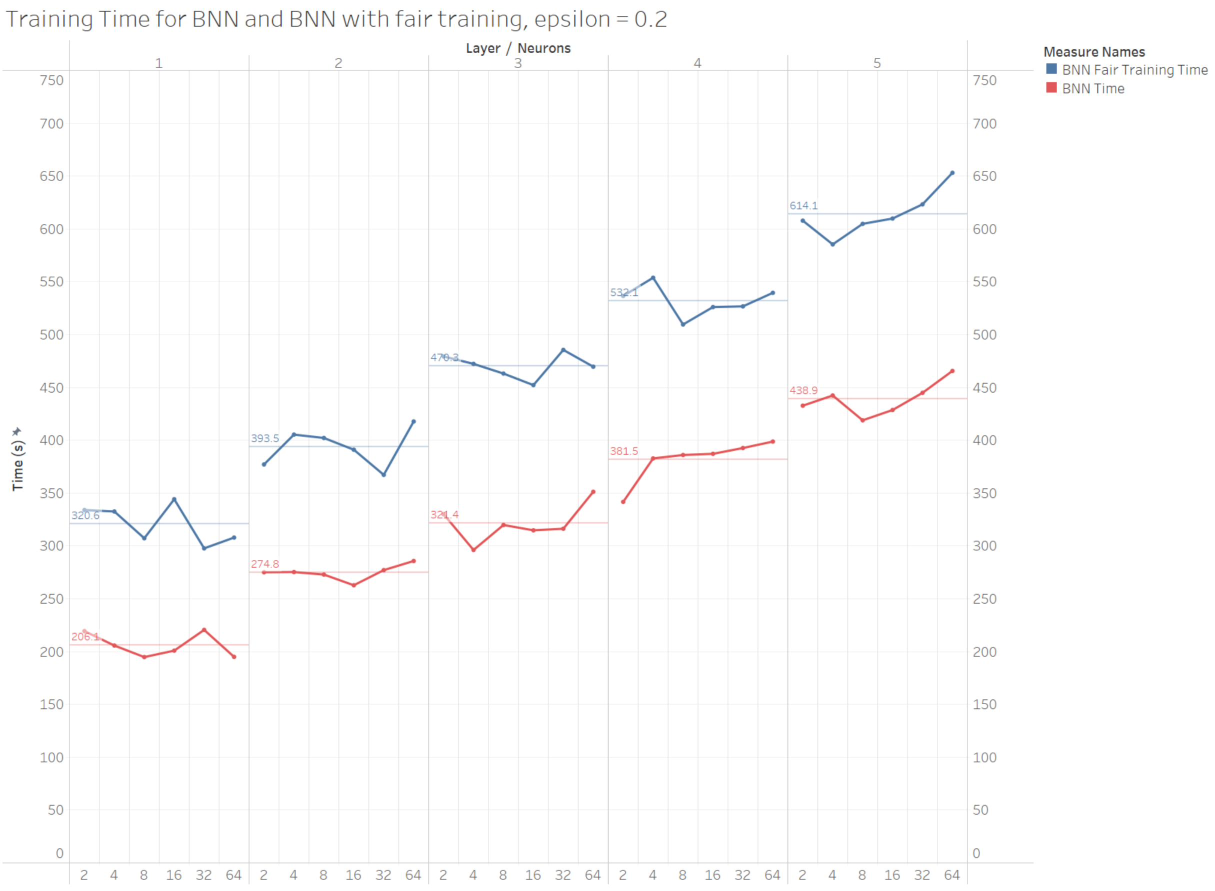


Figure A.20: Training time for $\epsilon_{fair} = 0.2$. Constant lines represent averages and bands represent min-max range.