MASTER'S THESIS 2022

Investigating and Mitigating Effects of Quantization on Algorithmic Bias

Oscar Andersson, William Isaksson

ISSN 1650-2884
LU-CS-EX: 2022-30

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-30

Investigating and Mitigating Effects of Quantization on Algorithmic Bias

Oscar Andersson, William Isaksson

Investigating and Mitigating Effects of Quantization on Algorithmic Bias

(A LaTeX class)

Oscar Andersson
os8218an-s@student.lu.se

William Isaksson
wi2628is-s@student.lu.se

June 13, 2022

Master's thesis work carried out at Arm.

Supervisors: Felix Johnny Thomasmathibalan,
FelixJohnny.Thomasmathibalan@arm.com
Axel Berg, Axel.Berg@arm.com
Flavius Gruian, flavius.gruian@cs.lth.se

Examiner: Jacek Malec, Jacek.Malec@cs.lth.se

Abstract

Quantizing neural networks is necessary for efficient inference on resource- constrained devices. In general, quantization slightly reduces the overall performance of a network. However, some sub-groups of a dataset might be impacted disproportionately. In this thesis, we investigate how quantization impacts algorithmic bias. We find that for hair color classification, the class-level bias is amplified, while on the attribute-level, the bias is rather unaffected for the attributes gender and age. We then show that model architecture and hyperparameters play a vital role in how a network is affected by quantization. Lastly, we propose two methods to mitigate the impact of quantization on the bias of a model.

Keywords: Machine Learning, Algorithmic Bias, Quantization

Acknowledgements

We would like to express our gratitude to our supervisors at Arm Lund, Axel Berg and Felix Johnny Thomasmathibalan, for their support, encouragement and valuable insights. For setting up and helping us get started, we also want to thank Fredrik Knutsson at Arm Lund. We also wish to thank Shounak Datta at Arm Austin, for helping us and for his valuable ideas, especially on fairness in machine learning. We would like to thank all the people we have met at Arm for making us feel welcome. We also want to thank our first supervisor at university, Jörn Janneck for helping us get started on the report. Finally, we want to thank our second supervisor Flavius Gruian for his persistent feedback and interest in the thesis.

Contents

Chapter 1
Introduction

The rapid development and increased amount of use cases of deep neural networks have revolutionized machine learning in many ways. As networks have become more complex and more capable, the demands on software and hardware have increased. Traditionally, computationally heavy applications such as neural networks would not be able to run on lightweight devices such as microcontrollers and would instead run remotely. It is convenient to run networks remotely as the lightweight edge device does not need to do the hard work. However, this requires the device to be connected to a server or cloud service and share possibly sensitive data. By running a trained network on an edge device instead, we can increase privacy and reduce the bandwidth and dependency of being connected to the cloud. A vital tool to achieve this is *model compression*. Model compression allows neural networks to run where computing resources face both strict size and power constraints, e.g., embedded systems, as it reduces memory and energy usage, inference latency, and computational overhead [18]. Model compression shrinks the model by removing weights or reducing the precision of the parameters in the model. However, model compression necessarily introduces a tradeoff between the degree of the compression and the size of the accuracy drop [13] [26].

An efficient way to compress a model is to quantize it. This can be done in slightly different ways. In this thesis, we use the popular open-source machine learning framework TensorFlow, where models use 32-bit floating-point precision [23]. Model quantization reduces the precision by quantizing weights, activations, inputs, and outputs to a lower bitwidth. If 8-bit integers are used, there is around a 4x size reduction of the model size. Also, this could allow fixed-point operations to be used rather than floating-point and would allow for the use of specialized hardware that utilizes single instructions and multiple data (SIMD) for accelerated inference. On compatible hardware, this could yield a larger than 3x speed-up on a quantized TensorFlow model [22].

Quantizing a trained network directly, a process called Post Training Quantization (PTQ), does not always preserve accuracy. When the accuracy drop is larger than desired, one can mitigate the accuracy drop by more specialized training with a process called quantization-aware training (QAT). Because many microcontrollers with Arm architecture require num-

bers to be represented in the int8 format, we restricted our experiments to 8-bit integer quantization in TensorFlow.

The starting point for this thesis is the work presented in [11]. In the paper, the authors investigate the effect of pruning on algorithmic bias, i.e., how the performance of different subgroups of the dataset is affected by pruning. The authors classify if a person is blonde or non-blonde on an imbalanced dataset called CelebA. Each image in the dataset is associated with a binary gender and age attribute. The authors claim that underrepresented sub-groups are responsible for a disproportional amount of the error introduced by compression. For instance, the relative increase in false positive rate (FPR) is 49.54% for males while only 6.32% for females. In the paper, the authors compress using pruning, which means removing insignificant weights from the network. In this thesis, we extend the work done in [11] to investigate how quantization affects the algorithmic bias of a model. Depending on the outcome of the investigation, we will investigate how we can mitigate any unwanted bias.

To measure bias, we introduce bias metrics that are based on the fairness criterion equalized odds [10], which requires the false positive rate (FPR) and false negative rate (FNR) to be equal across different sub-groups. The deviations from this ideal are measured as the difference between the best and worst sub-group or the variance of FPR and FNR across all sub-groups. If one were to force the model to perform equally across sub-groups, it could massively decrease the overall accuracy. This highlights a tradeoff between overall accuracy and algorithmic bias.

We also examine four different ways of tackling this tradeoff, starting with the choice of model architecture and hyperparameters. We then perform information retrieval through post-processing, and lastly, we examine QAT.

1.1 Related Work

On the topic of quantization, there exist a lot of related work, such as [19] [8] [18] [13]. The focus of most of these papers is to improve the accuracy of quantized networks or to reduce the number of bits being used.

Bias and fairness in machine learning (ML) is a field that has garnered much attention in recent years. The main focus of this area has been automated decision making, where ML directly affects human lives. [2] gives an introduction to and a good overview of this field. In facial recognition, [5] has investigated commercial gender classifiers from Microsoft, IBM and Face++ and compared performance across gender and skin tone. It was found that all classifiers had the largest error rate on females with dark skin and the lowest error rate on light males. However, the focus of this thesis has been bias in multiclass image classification, but no papers were found on this subject.

Mitigation of bias in ML has been investigated in [24], where the main focus is on dealing with imbalanced data. The paper presents two common techniques to re-balance the data; re-sampling and modifying the cost function during training. The second option we use to mitigate bias in this thesis. Another paper that has investigated bias is [1], which concludes that weight regularization produces models that are biased towards some classes. This paper was published towards the end of the work on this thesis, and we were not aware of its impact on bias during our experiments. Nonetheless, the conclusions from [1] are used to discuss some of our findings.

The intersection of compression of neural networks and fairness or bias is fairly unexplored. [11] investigated the effect of pruning on algorithmic bias. The paper concluded that pruning can amplify algorithmic bias and that underrepresented subgroups of the data are responsible for a disproportional amount of the total error. We investigate if the same conclusions hold for quantization in the binary and multiclass case. Another paper that has investigated the bias introduced by model compression is [3]. The paper introduced two ways of measuring bias between models. One of these metrics, combined error variance (CEV), we use throughout this thesis to compare the class-level bias between two models. Yet another paper that investigates the fairness effects of pruning is [14]. However, the paper introduces the Max-Min measure, which can capture bias or unfairness of a model. This measure we use to detect and measure bias in our experiments.

1.2 Research Questions

The research questions which we aim to answer in this thesis are:

- How does quantization affect algorithmic bias?
- If quantization increases algorithmic bias, how can we mitigate it?

1.3 Scientific Contribution

The contribution of this thesis is first and foremost to increase the knowledge of how quantization affects algorithmic bias and to quantify this bias using different metrics. Secondly, we investigate how different mitigation techniques impact these bias metrics as well as contribute a possibly novel post-processing mitigation technique that preserves performance while offering a different tradeoff between algorithmic bias and overall accuracy.

1.4 Division of Work

Oscar is responsible for the following sections:

- Artificial Neural Networks
- Quantization
- Approach
- Results

William is responsible for the following sections:

- Algorithmic Bias
- Post-processing
- Discussion

We have equal responsibility for:

- Abstract
- Introduction
- Conclusions

Chapter 2
Background

In this chapter, we start by introducing relevant concepts in artificial neural networks (ANNs). We then move on to quantization and introduce two quantization methods. Lastly, we introduce algorithmic bias and how it will be measured throughout the paper.

2.1 Artificial Neural Networks

Here we will cover some concepts and techniques used throughout the thesis. For a complete introduction to ANNs, we refer to [9].

2.1.1 Regularization

Regularization is commonly used to avoid overfitting on training data by restricting the flexibility of a model such that the training and validation performance is similar. We use weight regularization on some models in this thesis, as can be found in table 3.1. Weight regularization restricts the magnitude of the weights to be close to zero by penalizing weights with large magnitudes in the loss function. A problem with weight regularization is that it increases the class-level bias of a model, as demonstrated in [1].

2.1.2 Data Imbalance

Real-world datasets are rarely balanced, meaning that the frequencies of different classes are rarely equal. This can lead to models that are biased towards the larger classes and can lead to poor performance on small classes. There can also be imbalances within classes. For instance, one class might be dominated by one age group or gender.

To balance the classes in terms of the loss function, one can modify the training dataset or the loss function. Modifying the training dataset is usually referred to as re-sampling. To

balance the dataset, it removes samples from larger classes or adds duplicates to the smaller classes. However, this can lead to overfitting on the duplicates, or we can lose important information from the removed samples. Modifying the loss function refers to modifying the cost of misclassifying each class. For instance, to tackle the imbalance, we can modify the cost such that the smaller classes are more costly to misclassify than larger classes. In this thesis, we will refer to this technique as *class weighting*. The cost of misclassifying a sample of class i in the loss function is modified by a factor:

$$\mathbf{cost}_c = \frac{N_{max}}{N_c} \tag{2.1}$$

where N_c is the number of samples in class c, and N_{max} is the number of samples in the largest class. This means the loss for an observation o is:

$$-\sum_{c=1}^{M} cost_c * y_{o,c} \log(p_{o,c}) \tag{2.2}$$

Where y is a binary indicator of whether class label c is the correct classification for observation o, M is the number of classes, and $p_{o,c}$ is the output of the network for class c. The drawback of this technique is that it impacts the overall accuracy negatively [24]. Therefore, when training a model, one can choose whether to prioritize equal class-level performance or overall accuracy. This tradeoff will be referred to as the bias-accuracy tradeoff.

2.1.3 Feature Visualization

When analyzing the performance and behavior of a network, it is sometimes not enough to just analyze the outputs. In these cases, we may want to look at what happens inside the network and the intermediate values. The challenging part is to visualize these values as the output of a network's hidden layers is often multidimensional. A technique to visualize this is T-distributed Stochastic Neighbor Embedding (t-SNE) [16]. It uses dimensionality reduction techniques such as Principal Component Analysis (PCA) to project the data to a lower-dimensional space. In this thesis, we are interested in how the network changes when quantization is applied. t-SNE will be used as a tool to visualize this.

2.2 Quantization

We start by introducing quantization in general and how quantization is used in neural networks. Then, we move on to quantization in TensorFlow and how we will use this in our experiments.

2.2.1 Quantization in General

The most common formats for storing numbers in digital computers are floating-point numbers and integers. These numbers are represented using a collection of bits. The number of distinct values that can be represented is:

$$\#values = 2^{\#bits} \tag{2.3}$$

e.g., using 8 bits, one can represent $2^8 = 256$ different values.

Quantization is the process of mapping values from a large set into a smaller set. In this thesis, the larger set will be of the format floating-point 32-bit (FP32), and the smaller set will be of the format integer 8-bit (int8). From equation 2.3, one can find that this corresponds to a massive reduction in the number of values that can be represented after quantization (a reduction by a factor of 2^{32-8}), but at the same time a four-fold reduction in the number of bits needed to store these values.

A popular quantization function is:

$$Q(r) = Int(r/S) - Z \tag{2.4}$$

where Q is the quantization operator, r is the value to quantize (either an activation or a weight), S is a real-valued scaling factor, and Z is an integer zero point. This kind of quantization is called uniform since the spacing between the quantized values is uniformly spaced. Int is an operator that transforms the floating-point number to a nearby integer, often the rounding, flooring, or ceiling operation.

The equation for dequantization is as follows:

$$\tilde{r} = S(Q(r) - Z) \tag{2.5}$$

where $\tilde{r} \approx r$ due to the rounding operation [8].

A crucial factor in uniform quantization is choosing the scaling factor S, which is given by:

$$S = \frac{\beta - \alpha}{2^b - 1} \tag{2.6}$$

where b is the quantization bit width, and $[\alpha, \beta]$ denotes the clipping range, i.e., the range of values we want to quantize the most accurately within - values outside of the range are projected to either α or β. A clipping error is thus introduced. Values inside the max and min will be rounded to the closest quantized value. The error introduced is called rounding error. By increasing the scaling factor, we can increase the max and min and reduce the clipping error. However, by doing this, it is likely that the rounding error has increased instead.

2.2.2 Quantization in Neural Networks

In contrast to quantization in traditional signal processing, it is the change in the network outputs caused by the quantization of the network parameters rather than the quantization errors that are of interest. Modern deep neural networks tend to be over-parameterized, which means quantization can have little impact on the overall accuracy, such as top-1 and top-5 accuracy [8]. However, it is important to note that the model architecture plays a vital role, and different architectures are more or less robust to the noise introduced by quantization [19]. An architecture that can be impacted heavily is the common lightweight network MobileNetV1. 8-bit quantization can destroy the performance of the network [21] [25].

2.2.3 Quantization in TensorFlow

In TensorFlow, one can quantize a network in two different ways; post-training quantization (PTQ) and quantization-aware training (QAT). See figure 2.1 for a flowchart of the two methods. In our experiments, we mainly focus on PTQ but will evaluate QAT as well. We quantize the networks per layer (per-tensor) in both cases, as this is the default in TensorFlow. Below, we present the techniques and how they work.

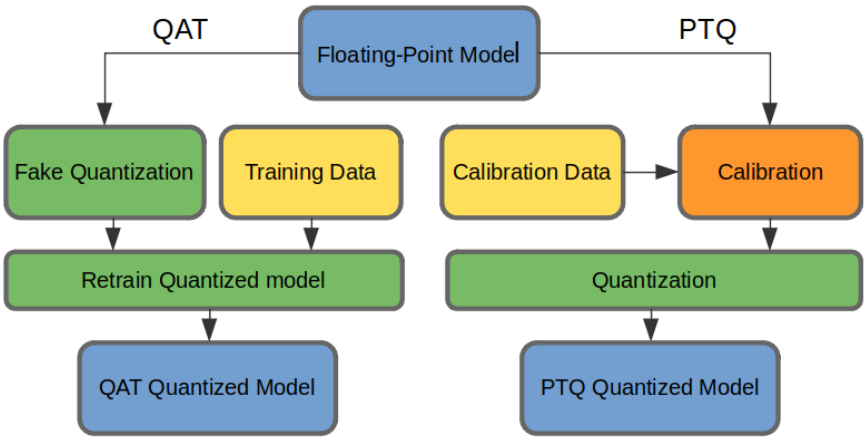


Figure 2.1: Flowchart for QAT and PTQ respectively

Post-training Quantization

Post-training quantization (PTQ) quantizes a trained network from 32-bit floating-point weights to lower precision, e.g., int8. This does not require any additional training, and while quantization aware training (QAT) usually outperforms PTQ, this usually is sufficient for achieving an accuracy close to that of the original network [19].

To quantize the activation functions of the network to 8 bits, a *representative dataset* is needed to calibrate the clipping ranges of the activation functions by running a few inference cycles on it. The range of the activation function will be the minimum and maximum value that is seen during this process. The representative dataset usually consists of roughly 100-500 unlabeled training samples [22].

Quantization-aware Training

A flaw of PTQ is that it naively quantizes a network's weights and activation functions to a lower bitwitdh without any knowledge about the introduced noise. Therefore, there is no guarantee that performance is preserved after quantization. In cases when PTQ introduces too much noise, quantization-aware training (QAT) is a better option than PTQ. QAT simulates the noise induced by quantization during training and finds weights and activation ranges suitable for quantization. The procedure of QAT is often to fine-tune a pre-trained network with QAT rather than training from scratch. When fine-tuned, TensorFlow can apply quantization to the network without any calibration data. If done correctly, the quantized inference accuracy should be close to the simulated quantization accuracy [13].

Compression Identified Exemplars

Compression identified exemplars (CIEs) are the examples where the model's prediction changed after compression [11]. This metric quantifies the impact of compression in terms of how much the model has changed. We will present CIEs as their fraction of the test dataset.

2.3 Algorithmic Bias

As machine learning algorithms have found their way in to decision-making, such as hiring processes and loan applications, it has become inevitable to look beyond the accuracy when evaluating the performance of a model [2]. One way to do this is to consider *Model fairness*. Model fairness measures how fair the model is across its different sub-groups given some fairness criterion. Sub-groups of interest are often decided by *protected attributes*. These attributes are often sensitive, such as gender, race, and religion. A common fairness criterion is for instance *equal opportunity*, i.e., that all sub-groups should have an equal opportunity to be positively classified [10]. This could for instance mean that people of different genders should have the same chance of getting a loan, in this case regardless of conditionals, such as salary differences between males and females.

The field of model fairness is almost always related to automated decision-making, as it has clear impacts on human lives. However, the principles of this field can be applied in cases where the fairness considerations are not obvious. For instance, in this thesis we will investigate several hair color classifiers. In this case, the potential impacts of faulty classifications on human lives are hard to find. As such, model fairness is not necessarily the correct term to use. Instead, a broader view of *algorithmic bias* can be used. This entails everything that model fairness does except the explicit fairness consequences. In this thesis, we consider two forms of algorithmic bias; class-level bias, i.e., different performance for different classes, and bias on protected attributes, that is for a given class, how does the performance differ across a protected attribute.

Below, we introduce the performance measures on which our fairness criterion is based and explain how they will be interpreted in this thesis. Then, we introduce our metrics of algorithmic bias. These are then used to compare the level of algorithmic bias between models.

2.3.1 Performance Measures for Multiclass Classification

When evaluating the performance of a classifier, it is common to use precision and true positive rate (TPR) to capture class-level performance on the test dataset. In the multiclass case, TPR, otherwise known as class-level accuracy or recall, measures what fraction of a class that was classified correctly. For fairness evaluation, it might be more apt to use the false negative rate (FNR) instead of TPR, as this makes relative comparisons easier to interpret. Precision on the other hand, measures the certainty of our predictions.

However, when considering algorithmic bias, we are only interested in the performance of a model, independent of priors. This is because we consider the performance on all classes

equally important. Furthermore, precision - in opposition to TPR - is dependent on priors which makes it unsuitable for measuring bias.

Another metric that is commonly used in fairness is the false positive rate (FPR). However, this is also dependent on priors. To get around this, we will instead use a balanced FPR, i.e. the FPR for every class assuming equal priors. Below, we will present how the balanced FPR is calculated for a class i:

$$\mathbf{FPR}_i = \frac{1}{N-1}\sum_{j\neq i}^{N}\frac{FP_{ji}}{P_j} = \frac{1}{N-1}\sum_{j\neq i}^{N} FNR_{ij} \tag{2.7}$$

where N is the number of classes, FP_{ji} is the number of samples of class j that have been misclassified as class i, and P_j is the number of positives in class j. Equivalently the fraction in the summation can be written as FNR_{ij}, i.e., the contribution of class i to the FNR of class j.

We consider FNR the most important class-wise performance measure in most applications as this determines the class accuracy. However, FPR is also relevant to consider, and in some cases, high FPR on a class can be very problematic will be discussed in section 2.3.6.

2.3.2 What Is Algorithmic Bias?

In model fairness literature, it is common to use *fairness criteria*. These criteria need to be fulfilled for the model to be considered fair. We will use a variant of the fairness criterion *equalized odds* as a way of quantifying bias. Equalized odds requires the FPR and FNR to be equal across all sub-groups of the dataset, but we will use the balanced FPR instead. This choice of fairness criterion was natural to us, as we want the model performance independent of priors to be as even as possible across sub-groups. In this thesis, we will consider two different types of sub-groups: classes and protected attributes. For an interested reader, there are other fairness criteria that can be found in [2] [10].

2.3.3 Class-level Bias Versus Protected-Attribute Bias

Algorithmic bias can be viewed both at a class-level and a protected-attribute-level. At the class-level, we can compare how different classes differ in terms of balanced FPR and FNR. For protected attributes, we can look at one or more attributes at a time and compare the differences in FPR and FNR inside one class.

2.3.4 Bias Metrics

When training multiple models and evaluating the amount of bias, it is relevant to consider both the average performance, but also the worst case. Below, we present two bias metrics based on balanced FPR and FNR that we use to measure class-level bias.

Max-Min

A simple, but useful bias metric, is the Max-Min metric. It is defined as the difference in a performance measure between the best and worst performing class, e.g., the difference

between the maximum and minimum balanced FPR. This indicates how biased the model is at its worst. Formally, it is defined as:

$$\textbf{Max-Min} = \max_{i \in C} e_i - \min_{i \in C} e_i, \tag{2.8}$$

where C is the set of classes, and e_i is the FPR or FNR of class i [14].

Variance

A drawback of Max-Min is that it does not capture what happens to the classes in between the max and the min. The variance does this by measuring the bias of the entire model for a given performance measure. In terms of variance, the most unbiased model would be a model where the performance measure is the same for all classes, giving a variance of 0. The most biased model (in terms of variance) would be a model where half of the classes have the value 1 of a performance measure, and half the classes are at 0, which corresponds to a variance of 0.25. Formally, variance is defined as:

$$V = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2, \tag{2.9}$$

where N is the number of samples, x_i is the i:th observation, and μ is the average of all observations [4]. As far as we know, this is a novel way of measuring class-level algorithmic bias.

2.3.5 Measuring Changes in Algorithmic Bias

Bias metrics attempt to capture how biased a given model is. To accurately capture how bias changes is not entirely trivial. One must decide if it is the change of bias or the bias of the final model that the metrics should be applied to. For example, a biased change of FNR or FPR in bias metrics could lead to a less biased model using the same metrics. In this thesis, we will measure the difference in bias between the final model and the baseline model, using Max-Min, variance, and combined error variance (CEV).

Max-Min

We will measure the difference in Max-Min by using the increase of the Max-Min between the quantized model and the baseline model. Formally, we define this as:

$$\Delta\textbf{Max-Min} = \textbf{Max-Min}_{quant} - \textbf{Max-Min}_{fp} \tag{2.10}$$

where $\textbf{Max-Min}_{quant}$ is the Max-Min of the quantized model, and $\textbf{Max-Min}_{fp}$ is the Max-Min of the floating-point model.

Variance

We will present the variance increase between two models in terms of the ratio between the increase and the total possible variance increase in percentage points, i.e.:

$$V_{score} = 100 \cdot \frac{\Delta V}{0.25} \tag{2.11}$$

where ΔV is the variance increase after compression. The value 0.25 in the denominator is the maximum possible variance for values between 0 and 1. This measures how much the variance of the performance measure has increased relative to how much it could maximally increase. This choice is made to increase readability.

Combined Error Variance

Combined error variance (CEV) is a metric proposed in [3] that measures fairness between two models. The fairness measurement can be described as the tendency of the model to sacrifice the performance of one class for another. A feature of CEV is that it by default compares two models by combining FPR and FNR into one number. Formally, it is defined as:

$$\delta X_{ie} = \frac{X_{ie} - \hat{X_{ie}}}{\hat{X_{ie}}} \tag{2.12}$$

$$\delta X_{\mu e} = \frac{1}{n} \sum_{i=0}^{n} (\delta X_{ie}) \tag{2.13}$$

$$cev = \frac{1}{n} \sum_{i=0}^{n} (dist((\delta X_{\mu pos}, \delta X_{\mu neg}), (\delta X_{ipos}, \delta X_{ineg})))^2 \tag{2.14}$$

where X_{ie} is a pair of FPR and FNR for class i for the model to compare with (in our case the quantized model), and $\hat{X_{ie}}$ for the original model. e tells if it is the positive or negative rate. δX_{ie} is the normalized change in FPR/FNR and is used to compute the average movement of all classes, $\delta X_{\mu e}$. The CEV is then computed by taking the average distance between all such positive and negative pairs. The dist function refers to the euclidean distance between the points.

2.3.6 Ethical Considerations

The ethical considerations of ML and intelligent systems are many, especially in systems where human lives are affected. In some applications, such as cancer detection, the FNR might be more relevant than the FPR. The consequences of classifying someone as not having cancer when they have cancer are more severe than the opposite scenario. An example where the FPR is more important than the FNR is facial recognition for unlocking mobile phones, at least when the FNR and FPR are reasonably small. It is better that the correct person cannot unlock their phone from time to time than that the wrong person manages to unlock someone else's phone. Also, in facial recognition, there are sometimes performance differences between protected attributes. [5] show that commercial gender classifiers produced by IBM, Microsoft, and Face++ have vastly different error rates across genders and skin tones. For Microsoft, the classifier had a 20.8% error rate on dark females while it had a 0.0% error rate on light males, and for IBM the corresponding numbers were 34.7% against 0.3%. The authors also state that "Inclusive benchmark datasets and subgroup accuracy reports will be necessary to increase transparency and accountability in artificial intelligence."

Ethical concerns of algorithmic bias in multiclass image classifiers are not as easy to come up with. However, there are situations where it is undesirable to have significant performance differences across different classes, and it is problematic to deploy models that are biased

towards some protected attribute. For instance, in [17], a classifier is trained to classify five different types of skin cancer. While this is not the case, it would be awful if the classifier had high overall accuracy but poor performance on a rare but dangerous type of cancer. It is also undesirable that the classifier would perform differently on, e.g., different skin tones.

The use of image classification itself can also be problematic in itself. For instance, a facial recognition system that can recognize all citizens of a country could be used to catch crime suspects and reduce crime overall, but at the same time reduce the privacy of citizens and could also be misused.

Chapter 3
Approach

Our work consists of a literature study and experiments. The experiments were done in two phases; a first phase exploring what effects quantization has on algorithmic bias, and a second phase trying to mitigate these potential undesired effects.

3.1 Literature Study

The first stage of the thesis was to conduct a literature study. As there is a lack of papers on the impact of quantization on bias, the starting point of this study was [11], which investigated the impact of pruning on algorithmic bias. The experiments in the paper were used as a reference for our initial experiments. The further aims of the literature study were to understand how to quantize in TensorFlow, find metrics that could quantify algorithmic bias, and to find a suitable dataset and model architecture where algorithmic bias is sufficiently affected by quantization.

3.2 First Phase: Exploration

In the first phase, we investigate our first research question: how does quantization affect algorithmic bias. As we did not know how quantization would affect algorithmic bias and knew from [11] that pruning could amplify bias according to the authors, we started by trying to reproduce their results, but with post-training quantization instead of pruning. However, we found that the impact of quantization on algorithmic bias was negligible. The impact of quantization was measured by comparing the change of FNR and normal FPR of the quantized and baseline models within a given protected attribute. In this way, we can see if, for instance, males or females are affected differently by quantization.

Because of this, we moved on to the smaller models in the MobileNet family [20] [12] and trained them on multiclass classification tasks. These networks are known to be sensitive to

quantization and should therefore yield more interesting results [21] [25]. The transition to multiclass classification was done to open up more degrees of freedom for bias to occur. At the class-level, the impact of quantization was quantified by measuring the change in bias metrics between the quantized model and the baseline model. For the protected attributes, the accuracy difference between protected attributes and the changes in FNR and balanced FPR were used to measure the bias. We found here that the MobileNetV3Small architecture was the most sensitive to quantization.

The models were trained to do hair color classification on the CelebA dataset. This dataset was chosen because of its size and presence of protected attributes. The dataset is also unbalanced, which lets us analyze whether underrepresented classes and groups are disproportionately affected by quantization, as was the hypothesis in the paper that was used as a starting point for this thesis [11].

3.3 Second Phase: Mitigation

The second phase is heavily based on observations made during the first phase. Since the MobileNetV3Small architecture was the most sensitive to quantization, we focused on mitigating the impact of quantization on this network. First, we attempted to mitigate the effects by adding two different hyperparameters when training our model. In one of the models, we added L1-regularization to the last layer. This choice was made because we had observed in the first phase that this type of regularization was less sensitive to quantization in terms of accuracy. In the other model, we added class weighting so that all classes are equally important in the loss function 2.1.2. The hypothesis was that this should improve class-level performance, which could affect how the model would be impacted by quantization.

Second, we turned to quantization-aware training. As mentioned in 2.2.3, QAT is a good option when PTQ leads to a large accuracy drop. As it turns out, QAT on MobileNetV3Small specifically is not supported by default in TensorFlow Model Optimization. Despite our best efforts, we could not do it manually either. Instead, we chose another version called MobileNetV3Small minimalistic, which is supported by QAT. We perform both PTQ and QAT on this network to be able to compare them.

Finally, when looking at the t-SNE plots of the baseline MobileNetV3Small setup in figure 4.5, we realized that the performance on some classes after quantization was degraded more than what was reasonable, considering how separated the classes were after quantization. This led us to believe that there is a pattern in how some examples change after quantization. By training a small classifier to predict what prediction the floating-point model made, using only the output from the quantized model, one should then be able to get better performance if there is such a pattern. In other words, this is a post-processing method used to mimic the classifications of the floating-point model, using only the output of the quantized model.

3.4 Implementation

Below, we present the implementation details of our experiments. We start by presenting the frameworks we use and how we pre-process the dataset. We then present the hyperparameters

of the models we have trained, and how we quantize them. Finally, we present how we have implemented feature visualization and our post-processing method.

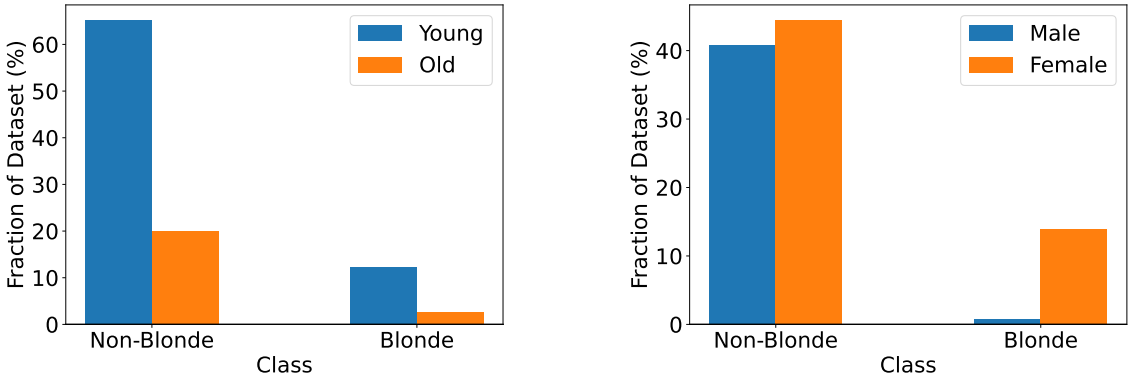
3.4.1 Frameworks

For training and evaluation, we use TensorFlow 2.8 along with Keras. For quantization and converting the models to compressed models, we use TensorFlow Lite. For QAT, we use TensorFlow Model Optimization Toolkit.

3.4.2 Datasets

For all setups, we use the cropped and aligned CelebA dataset downloaded from [15]. The dataset contains roughly 200,000, 178×218 images of celebrities' faces, along with 40 binary attributes associated with each image. A few example images are presented in figure 3.2. We pre-process and label images with the provided attributes file. We split the data into three sets; training, validation, and test, and use a split of 60% for training and 20% for validation and test, respectively. This choice was done rather arbitrarily but gives us a lot of training data, as well as reliable validation and test results.

For reproducing the experiments in [11] we label images as blonde or non-blonde. The distribution of the classes and sub-groups in the dataset are presented in figure 3.1. Note the imbalance between the blonde and non-blonde class, where non-blonde constitutes more than 80% of the dataset. Note also the imbalance in protected attributes, e.g., young compared to old for non-blonde, and male compared to female for blonde.



(a) Class frequencies of ages.

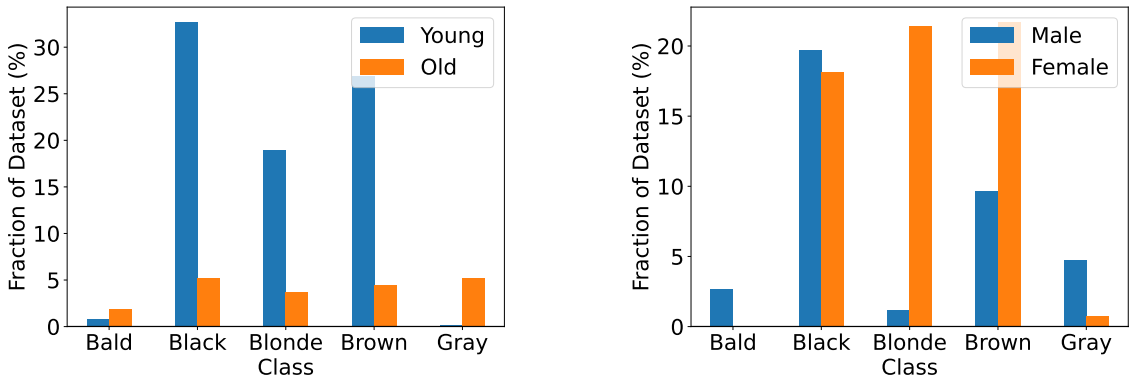
(b) Class frequencies of genders.

Figure 3.1: Class frequencies of sub-groups in the CelebA dataset.

We also construct a multiclass dataset out of CelebA consisting of five classes of hair color. Samples that are labeled as having multiple hair colors are removed. This resulted in a remaining dataset with around 125,000 images. The distribution of the classes and sub-groups are presented in figure 3.2 and 3.3. There are two extremely underrepresented classes, bald and gray. Bald and gray are also more frequent in old males than in any other sub-group.

(a) Bald, 2.72% **(b)** Black, 37.85% **(c)** Blonde, 22.62% **(d)** Brown, 31.36% **(e)** Gray, 5.46%

Figure 3.2: Class frequencies and example images from the CelebA dataset.



(a) Class frequencies of ages.

(b) Class frequencies of genders.

Figure 3.3: Class frequencies of sub-groups.

3.4.3 Training of Models

For all of our setups, a subset of the hyperparameters do not change. In all cases, we use Stochastic Gradient Descent (SGD), initialized with a learning rate of 0.01 and an exponential learning rate schedule as the optimizer. We also use a batch size of 256 and obtain the best models using a model checkpoint that monitors the validation performance during training. For detailed hyperparameters, see table 3.1.

All of our MobileNet models initialize their weights to ImageNet weights to decrease training time and time spent on hyperparameter tuning. For L1-regularization, we only apply it to the final dense layer of the model, and in cases where we use image augmentation, we only do a simple horizontal flip. When we use class weighting, we modify the cost associated with each class such that the sum of misclassifying every sample of a class should be equal for all classes. We want to emphasize that these hyperparameters were not chosen because they are optimal but were tried to be kept as simple as possible.

Table 3.1: Hyperparameters for different setups.

Model	Parameters	Epochs	Regularization	Weight init.
ResNet-18	11.2M	21	L1 ($\alpha = 0.001$)	Random
MobileNetV2	2.2M	20	L1 ($\alpha = 0.01$)	ImageNet
MobileNetV3Small	0.94M	20		ImageNet
MobileNetV3Small L1	0.94M	20	L1 ($\alpha = 0.01$)	ImageNet
MobileNetV3Small Class Weighting	0.94M	20		ImageNet
MobileNetV3Small Minimalistic	0.44M	15	Image augmentation	ImageNet

3.4.4 Post-training Quantization

When the models have been trained, the next step is to quantize the models. As calibration data, we randomly select 100 images from the validation dataset and quantize the models. This workflow is visualized on the right-hand side of figure 2.1. The resulting tflite models are evaluated on the same test dataset as the floating-point models.

3.4.5 Quantization-aware Training

When the floating-point models have been trained, it is time to simulate the quantization. This is done by annotating all the layers in the network that should be quantized. We then train for another 15 epochs with the same hyperparameters as before. When this training is done, we can quantize the model. In contrast to PTQ, we do not need a calibration dataset since the ranges have already been calibrated during QAT. This workflow is visualized on the left-hand side of figure 2.1. The quantized tflite models are evaluated just like in the PTQ case.

3.4.6 Feature Visualization

Recall from section 2.1.3 that the network could be analyzed from within by looking at intermediate values of the network. We do this by excluding the output layer of the models to construct "new" models that output intermediate values. The new models are then quantized. To get an overall picture of the internal representation of the classes in the model, we pass 1,000 images through the floating-point and quantized models. From this, we get two matrices with the 1,000 intermediate output vectors of the quantized model. Each output has more than 500 dimensions, and we reduce dimensionality to two using the t-SNE function from Scikit-Learn. We then plot all these outputs in two dimensions.

3.4.7 Post-processing

In figure 3.4 you can find the workflow for training the post-processing of the output of our quantized network. When implementing the classifier used to post-process the output of the quantized network, we used a small XGBoost classifier with a size of 10 kB because this had the best performance. XGBoost is one of the most popular machine learning methods for tabular data [7]. More information on XGBoost can be found in [6]. Important to note is

that this classifier was not quantized to int8 but used floating-point numbers. The classifier was trained both on validation data and the training data used to train the neural network, respectively, because we were worried that using previously used training data would lead to overfitting. However, no evidence of overfitting was found, leading us to use training data for training the post-processing classifier. Using 5,000 examples chosen randomly, the XGBoost classifier was trained to predict the initial prediction of the floating-point model, using the output of the quantized model as input. The experiment was then repeated using a balanced dataset with 1,000 examples from each class.

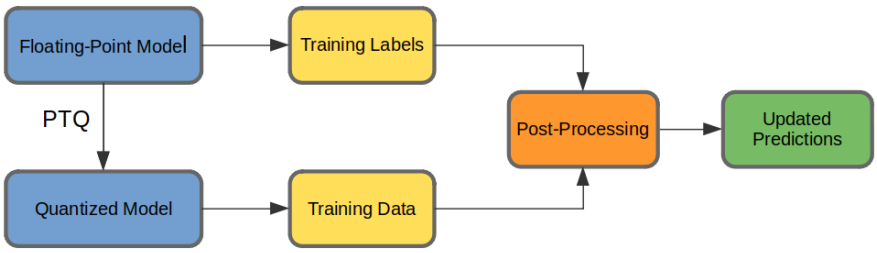


Figure 3.4: Flowchart for our post-processing method

Chapter 4

Results

In this chapter, we present the results of our experiments. We start with the binary case, and move on to multiclass classification and compare performance of different architectures and hyperparameters. Lastly, we present the results of our mitigation attempts and how the bias on protected attributes is affected.

4.1 ResNet-18 CelebA

The baseline floating-point ResNet-18 models achieved an average accuracy of 95.3% on the test dataset. The quantized models also achieved an average accuracy of 95.3% on the test dataset. The CIE fraction is 0.29%.

In Table 4.1, we present FPR and FNR overall and for the different sub-groups: Male (M), Female (F), Young (Y), Old (O), Male Young (MY), Male Old (MO), Female Young (FY) and Female Old (FO). We present these metrics for the floating-point model (baseline), and absolute and relative change between the floating-point and quantized model.

Note the initial bias, where for instance males (M) have a FNR of 61%, while females (F) have a FNR of 15.54%. Note that all absolute changes in FNR are negative while all changes in FPR are positive.

Table 4.1: Sub-group comparison between baseline and quantized ResNet-18 models on CelebA for sub-groups: Male (M), Female (F), Young (Y), Old (O), Male Young (MY), Male Old (MO), Female Young (FY) and Female Old (FO).

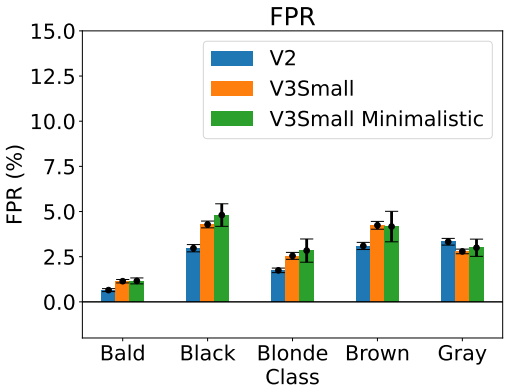
Model	Metric	Overall	M	F	Y	O	MY	MO	FY	FO
Baseline	FPR	2.35%	0.56%	4%	2.28%	2.58%	0.48%	0.69%	3.46%	8.05%
	FNR	18.13%	61%	15.54%	17.40%	21.43%	61.62%	59.74%	15.24%	16.96%
Absolute change										
Quantized	FPR	0.05%	0.03%	0.08%	0.04%	0.09%	0.02%	0.05%	0.06%	0.22%
	FNR	−0.34%	−0.82%	−0.31%	−0.30%	−0.53%	−0.71%	−1.04%	−0.28%	−0.47%
Relative increase										
Quantized	FPR	2.27%	**5**%	1.92%	1.8%	3.61%	3.18%	**7.19**%	1.67%	2.72%
	FNR	−1.89%	−1.34%	−2.02%	−1.74%	−2.46%	−1.15%	−1.75%	−1.85%	−2.75%

4.2 Architecture

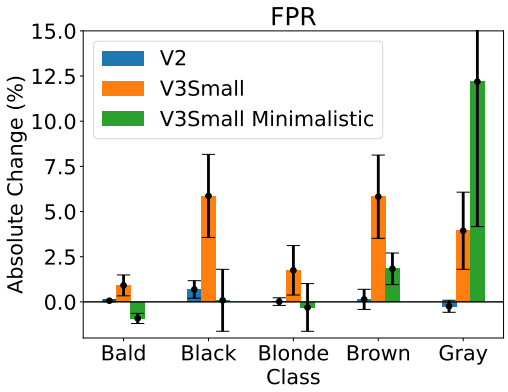
We compare three different architectures: MobileNetV2, MobileNetV3Small and MobileNetV3Small minimalistic. We visualize the differences in terms of FPR and FNR plots in figure 4.1 and 4.2. We also present the bias metrics for the three models in table 4.2.

4.2.1 FPR

Figure 4.1 shows that there initially, are some biases with black and brown having the highest FPR, and bald having the lowest. After quantization, V2 barely changes compared to V3Small and V3Small minimalistic. For V3Small, black and brown increases the most in FPR. For V3Small minimalistic, gray FPR increases with around 12.5%, while bald FPR decreases.



(a) Average FPR with 95% confidence intervals for floating-point models.



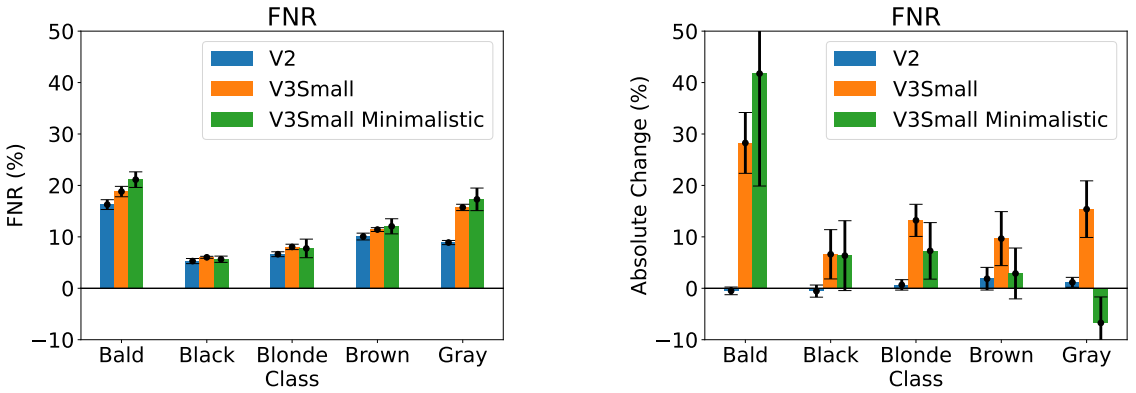
(b) Average absolute change in FPR with 95% confidence intervals.

Figure 4.1: Initial FPRs and changes for for MobileNet: V2, V3Small and V3Small minimalistic.

4.2.2 FNR

In figure 4.2, we see that for FNR, there is some initial bias, where bald FNR is the highest and black FNR the lowest. One can also observe that the Max-Min is the largest for V3Small minimalistic between black and bald.

As for FPR, V2 barely changes in terms of FNR, compared to V3Small and V3Small minimalistic. For V3Small, observe the difference in absolute change between bald and the other classes. The same holds for V3Small minimalistic, where the FNR for bald increases by 42%, while gray FNR decreases.



(a) Average FNR with 95% confidence intervals for floating-point models.

(b) Average absolute change in FNR with 95% confidence intervals.

Figure 4.2: Initial FNRs and changes for MobileNet: V2, V3Small and V3Small minimalistic.

4.2.3 Bias Metrics

In table 4.2, the bias metrics of the three architectures are presented. V2 is relatively unaffected by the quantization with only 0.58% accuracy degradation and 3.15% CIE fraction. Also, the bias metrics are close to 0, except FPR max-min increase that is 0.27%, which is still small compared to the other architectures.

V3Small is the most impacted architecture in terms of accuracy degradation and CIE fraction despite being larger than V3Small minimalistic. However, V3Small minimalistic is the architecture where the bias metrics are the largest, e.g. FNR max-min increase of 36.83% compared to 21.66% of V3Small and 0.04% of V2.

Table 4.2: Bias comparison for MobileNetV2, MobileNetV3Small and MobileNetV3Small minimalistic. The highest accuracy, and lowest CIE fraction and bias metrics values are highlighted in bold.

Metric	Model	Value
Floating-point accuracy	V2	**92.42**%
	V3Small	90.95%
	V3Small Minimalistic	90.81%
Quantized accuracy	V2	**91.84**%
	V3Small	80.83%
	V3Small Minimalistic	85.10%
CIE fraction	V2	**3.15**%
	V3Small	17.43%
	V3Small Minimalistic	11.6%
FPR variance increase	V2	**0.01**%
	V3Small	0.35%
	V3Small Minimalistic	0.98%
FNR variance increase	V2	**0.00**%
	V3Small	4.65%
	V3Small Minimalistic	14.62%
FPR Max-Min increase	V2	**0.27**%
	V3Small	4.95%
	V3Small Minimalistic	11.3%
FNR Max-Min increase	V2	**0.04**%
	V3Small	21.66%
	V3Small Minimalistic	36.83%
CEV	V2	**0.0114**
	V3Small	0.0941
	V3Small Minimalistic	0.647

4.3 Mitigation

For mitigation of bias, we present the results in three sub-sections, starting with the impact of hyperparameters. We then present the results of our proposed post-processing and QAT.

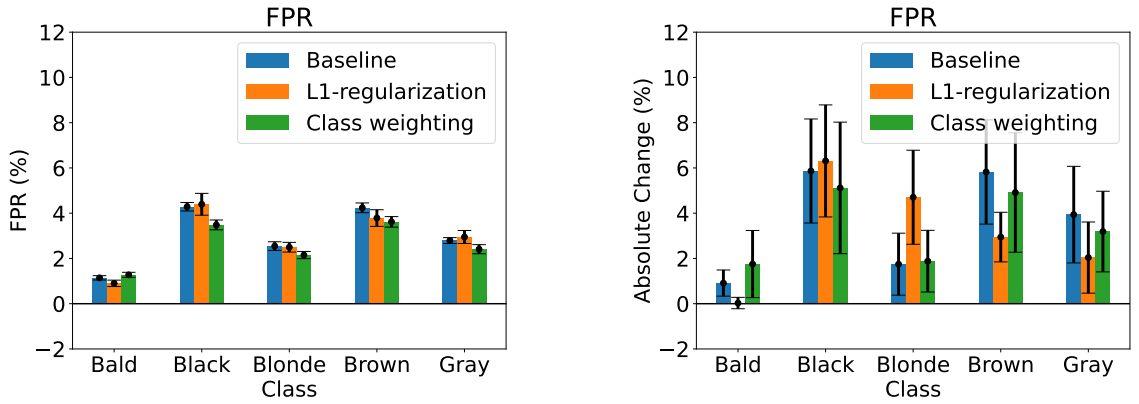
4.3.1 Impact of Hyperparameters

We compare the performance of MobileNetV3Small with three different hyperparameter settings; baseline, L1-regularization and class weighting. We visualize the differences in terms of FPR and FNR plots in 4.3, 4.4. We also present the bias metrics for the three settings in 4.3, and present a representative t-SNE plot for each setting.

FPR

From 4.3, we see that class weighting is the most unbiased model, meaning that the FPR of all classes are the most equal.

When quantizing, we see that different settings are affected differently, where mainly L1-regularization differs from other two settings. Baseline and class weighting are affected similarly, but notice that the changes are closer to being equal for class weighting than for the baseline.



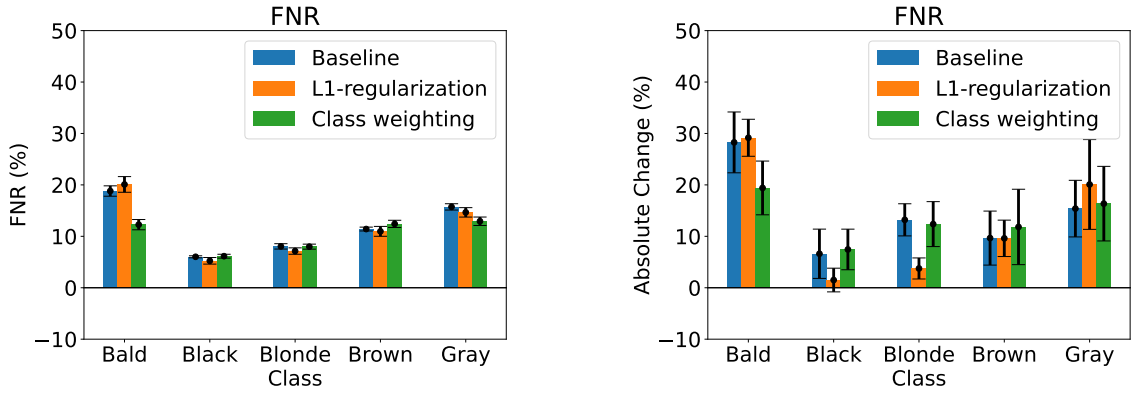
(a) Average FPR with 95% confidence intervals for floating-point models.

(b) Average absolute change in FPR with 95% confidence intervals.

Figure 4.3: Initial FPRs and changes on MobileNetV3Small for three different hyperparameter settings; baseline, L1-regularization and class weighting.

FNR

In figure 4.4, it is notable that the initial FNR of bald for class weighting is much lower than for the other two settings. This also holds for the absolute change in FNR, where bald increases around 10% less for class weighting. Other noticeable things are the small increases in FNR of black and blonde for L1-regularization.



(a) Average FNR with 95% confidence intervals for floating-point models.

(b) Average absolute change in FNR with 95% confidence intervals.

Figure 4.4: Initial FNRs and changes on MobileNetV3Small for three different hyperparameter settings; baseline, L1-regularization and class weighting.

Bias Metrics

In table 4.3, the bias metrics of the three different settings are presented. The highest accuracy, lowest CIE fraction and bias metrics values are highlighted in bold. L1-regularization has the highest accuracy for the floating-point models, and is least affected in terms of accuracy and CIE fraction when quantizing. It is noticeable that L1-regularization loses almost half as much accuracy as the other settings. Class weighting and the baseline are affected similarly in terms of accuracy degradation and CIE fraction.

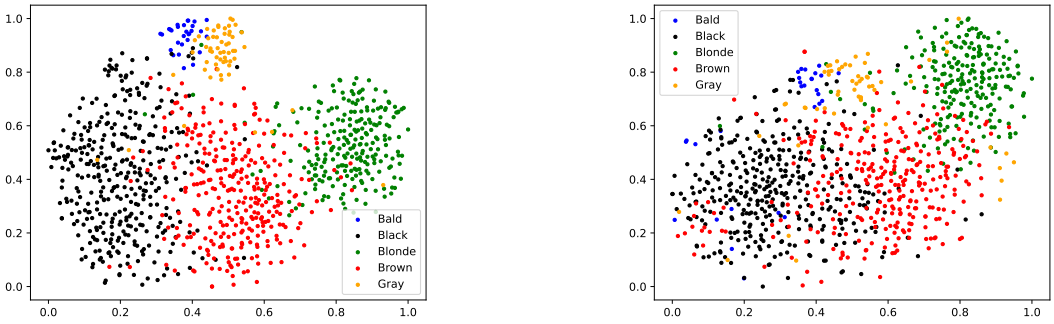
In terms of bias metrics, L1-regularization has the largest FPR and FNR Max-Min increase as well as FNR variance increase. Class weighting on the other hand, has the smallest increase in all bias metrics.

Table 4.3: Bias comparison for MobileNetV3Small for three different hyperparameter settings; baseline, L1-regularization and class weighting.

Metric	Model	Value
Floating-point accuracy	Baseline	90.95%
	L1	**91.62**%
	Class weighting	90.94%
Quantized accuracy	Baseline	80.83%
	L1	**85.30**%
	Class weighting	80.19%
CIE fraction	Baseline	17.43%
	L1	**11.98**%
	Class weighting	17.96%
FPR variance increase	Baseline	0.35%
	L1	0.35%
	Class weighting	**0.18**%
FNR variance increase	Baseline	4.65%
	L1	8.73%
	Class weighting	**1.36**%
FPR Max-Min increase	Baseline	4.95%
	L1	6.28%
	Class weighting	**3.23**%
FNR Max-Min increase	Baseline	21.66%
	L1	27.66%
	Class weighting	**11.3**%
CEV	Baseline	0.0941
	L1	0.208
	Class weighting	**0.054**

Feature Visualization

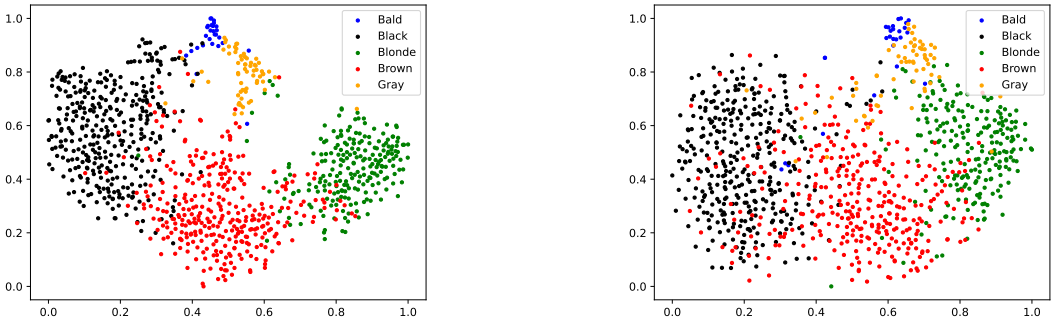
Following are three t-SNE plots that are representative of this model. These plots mainly give some intuition on how networks and classes are impacted by quantization. Note that before quantization the classes are rather separated, while after quantization the classes have diffused and are harder to separate. It can be seen that some classes are affected more than others by quantization. For instance in figure 4.5, before quantization the bald class is well separated from the other classes, but after quantization many examples have been displaced and are now surrounded by black examples and are classified as such. Note also that when using class weighting, the bald class is not as affected, as seen in figure 4.7.



(a) t-SNE of floating-point model.

(b) t-SNE of quantized model.

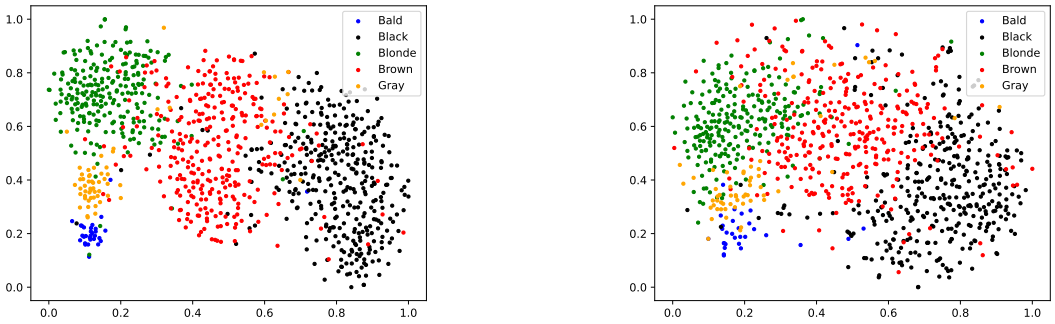
Figure 4.5: t-SNE of MobileNetV3Small baseline.



(a) t-SNE of floating-point model.

(b) t-SNE of quantized model.

Figure 4.6: t-SNE of MobileNetV3Small with L1-regularization.



(a) t-SNE of floating-point model.

(b) t-SNE of quantized model.

Figure 4.7: t-SNE for MobileNetV3Small with class weighting.

4.3.2 Post-processing

In table 4.4 we compare the results of PTQ for a setup with the results of PTQ after post-processing. The first unbalanced post-processing algorithm is trained on 5000 examples with the same distribution of classes as the network was trained on, while the balanced post-processing algorithm was trained on a balanced dataset using 1000 examples from each class. Note that the average accuracy after post-processing is increased for both methods but that the unbalanced method improved accuracy the most. Also note that the unbalanced method increased bias on all setups except V3small with L1 regularization. Particularly noteworthy is how the balanced post-processing method drastically reduces or eliminates the increase in bias from quantization on all setups. In figure 4.8, a few CIEs for the balanced post-processing are shown. In all images, the floating-point model classified correctly, the quantized model incorrect and the post-processing managed to retrieve the correct prediction. In figure 4.9, examples of the opposite scenario are shown, i.e., the floating-point model classified incorrect, the quantized model correct and the post-processing incorrect.

Table 4.4: Bias comparison for post-processing of different MobileNet setups.

	Model	Floating-point model		
Floating-point accuracy	V2	92.42%		
	V3 baseline	90.95%		
	V3 L1	91.62%		
	V3 class weighting	90.94%		
Metric	Model	PTQ	Post-processing	Balanced post-processing
Quantized accuracy	V2	91.84%	92.20%	92.22%
	V3 baseline	80.83%	82.88%	81.30%
	V3 L1	85.30%	86.78%	85.98%
	V3 class weighting	80.19%	83.90%	82.06%
CIE fraction	V2	3.15%	1.91%	1.89%
	V3 baseline	17.43%	15.29%	16.94%
	V3 L1	11.98%	10.38%	11.24%
	V3 class weighting	17.96%	14.16%	15.83%
FPR variance increase	V2	0.01%	0.01%	0.00%
	V3 baseline	0.35%	0.32%	0.00%
	V3 L1	0.35%	0.16%	0.02%
	V3 class weighting	0.18%	0.24%	0.00%
FNR variance increase	V2	0.00%	0.18%	−0.07%
	V3 baseline	4.65%	7.81%	0.42%
	V3 L1	8.73%	4.42%	0.31%
	V3 class weighting	1.36%	3.79%	0.22%
FPR Max-Min increase	V2	0.27%	0.22%	0.05%
	V3 baseline	4.95%	4.94%	0.03%
	V3 L1	6.28%	3.38%	0.08%
	V3 class weighting	3.23%	4.99%	0.15%
FNR Max-Min increase	V2	0.04%	1.55%	−0.65%
	V3 baseline	21.66%	25.88%	2.24%
	V3 L1	27.66%	17.88%	1.93%
	V3 class weighting	11.3%	20.29%	1.65%
CEV	V2	0.0114	0.0014	0.0013
	V3 baseline	0.0941	0.155	0.0772
	V3 L1	0.208	0.056	0.0342
	V3 class weighting	0.054	0.27	0.069

(a) Floating-point: Bald, Quantized: Black, Post-processing: Bald

(b) Floating-point: Black, Quantized: Gray, Post-processing: Black

(c) Floating-point: Blonde, Quantized: Brown, Post-processing: Blonde

(d) Floating-point: Brown, Quantized: Black, Post-processing: Brown

(e) Floating-point: Gray, Quantized: Bald, Post-processing: Gray

(f) Floating-point: Bald, Quantized: Gray, Post-processing: Bald

Figure 4.8: Example of CIEs that post-processing managed to classify correct.

(a) Floating-point: Bald, Quantized: Gray, Post-processing: Bald

(b) Floating-point: Brown, Quantized: Black, Post-processing: Brown

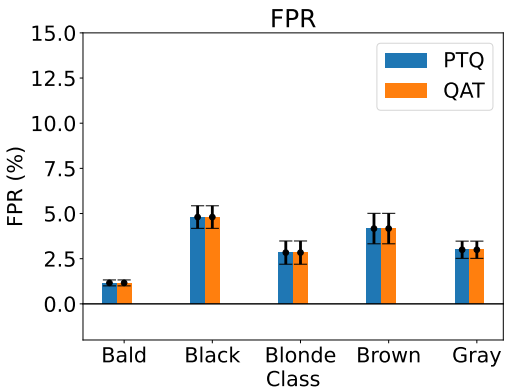
(c) Floating-point: Blonde, Quantized: Brown, Post-processing: Blonde

Figure 4.9: Example of CIEs where the quantized model was correct, but the post-processing classified the images incorrect.

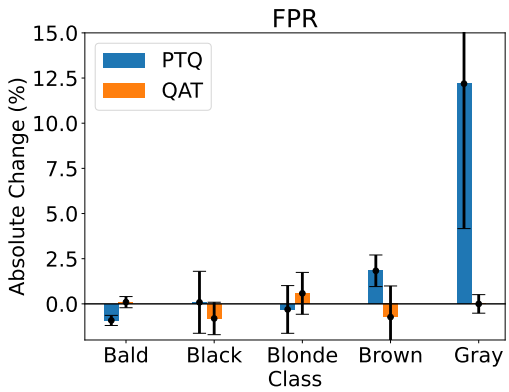
4.3.3 Quantization-aware Training

Below are the class-level changes in FNR and FPR with 95% confidence intervals for PTQ and QAT. Here PTQ and QAT share the same floating-point model. Note the massive difference in the change in class-level bias between the two methods. The only change for QAT that is significantly different from zero is that the FNR on the bald class decreases.

FPR



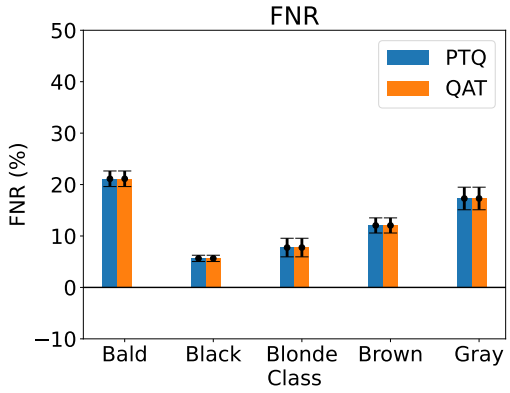
(a) Average FPR with 95% confidence intervals for floating-point model.



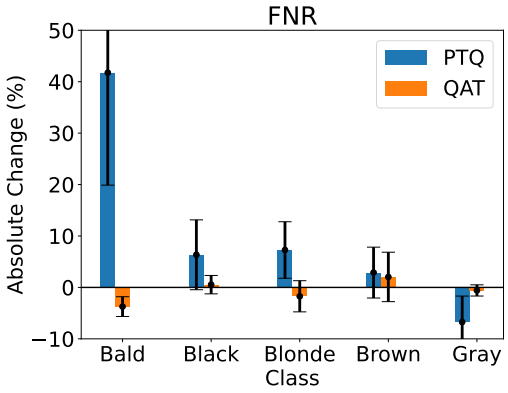
(b) Average absolute change in FPR with 95% confidence intervals.

Figure 4.10: Initial FPRs and changes on MobileNetV3Small minimalistic for PTQ and QAT.

FNR



(a) Average FNR with 95% confidence intervals for floating-point model.



(b) Average absolute change in FNR with 95% confidence intervals.

Figure 4.11: Initial FNRs and changes on MobileNetV3Small minimalistic for PTQ and QAT.

Bias Metrics

In this comparison of bias metrics between PTQ and QAT, we find that the class-level bias massively increased for PTQ, while it decreased for QAT.

Table 4.5: Bias comparison between PTQ and QAT on MobileNetV3Small minimalistic.

Metric	Model	Value
Floating-point accuracy	Baseline	90.81%
Quantized accuracy	PTQ	85.10%
	QAT	**90.49%**
CIE fraction	PTQ	11.6%
	QAT	**6.27%**
FPR variance increase	PTQ	0.98%
	QAT	**−0.03%**
FNR variance increase	PTQ	14.62%
	QAT	**−0.33%**
FPR Max-Min increase	PTQ	11.3%
	QAT	**−0.9%**
FNR Max-Min increase	PTQ	36.83%
	QAT	**−4.11%**
CEV	PTQ	0.647
	QAT	**0.023**

4.4 Protected Attributes

In this section, we start by presenting the accuracy differences and how the change between protected attributes. We then look protected-attribute bias per class for MobileNetV3Small which was the only significantly biased model. In section 6, we present the results for MobileNetV2 and MobileNetV3Small minimalistic with QAT.

For accuracy change per protected attribute, we evaluated five different models; MobileNet: V2, V3Small baseline, V3Small L1-regularization, V3Small class weighting and V3-Small minimalistic with QAT. Figure 4.12 shows the difference in accuracy between males and females, and old and young people respectively. Interesting to note here is that all floating-point models have a bias towards females and young people, which can be seen by the negative differences. Figure 4.13 shows how these differences change after quantization. For gender, we observe that the difference significantly increases in MobileNetV3Small baseline and L1-regularization. For age, we see that all three MobileNetV3Small models increase the difference.

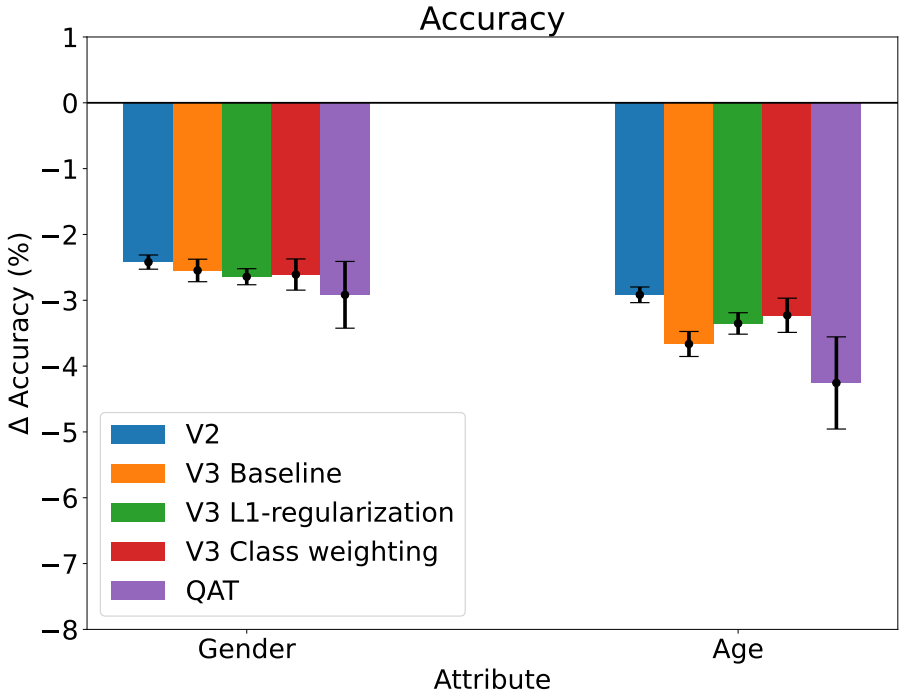


Figure 4.12: Average Δ accuracy with 95% confidence intervals.

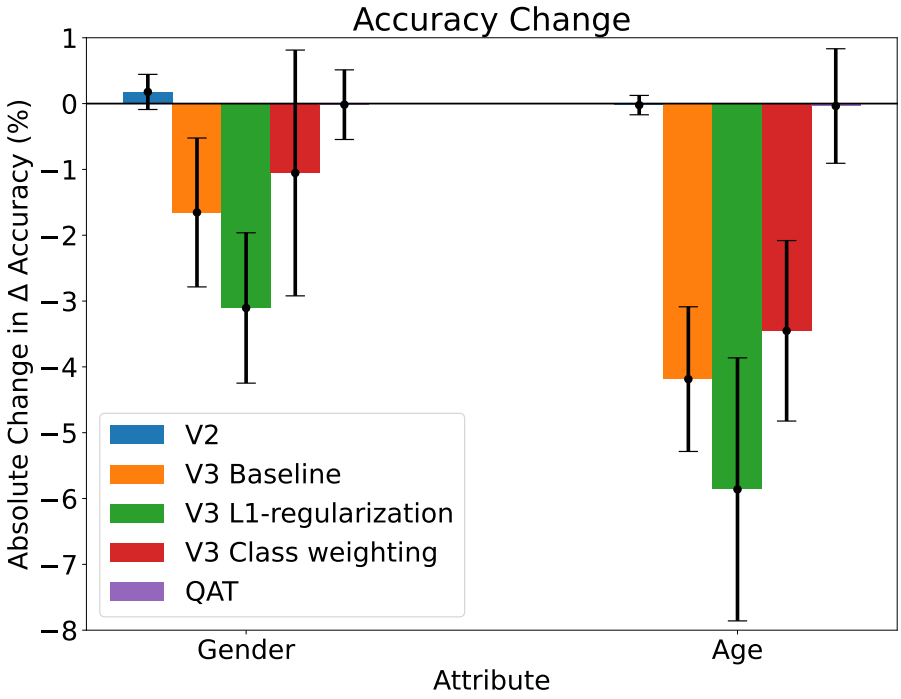
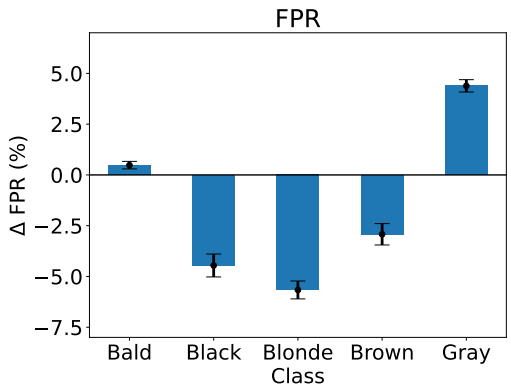


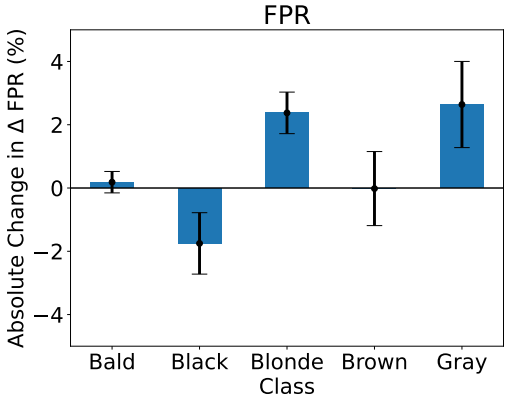
Figure 4.13: Average absolute change in Δ accuracy with 95% confidence intervals.

Age

For the protected attribute age, we compare FPR and FNR between old and young, i.e. the difference in FPR/FNR between old and young. As there are very few young people with gray hair, the results for gray should be analyzed with caution. In terms of FPR, we observe that the difference between old and young for black is increased after quantization. We also notice that the difference decreases for blonde. In terms of FNR, we observe that difference between old and young FNR on blonde increases, i.e. old has a much larger FNR, and thus becomes more biased in that aspect.

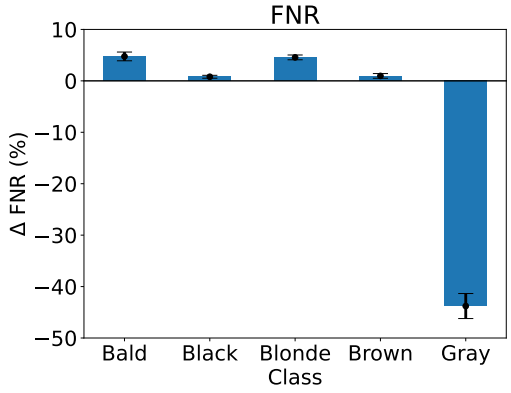


(a) Average Δ FPR with 95% confidence intervals.

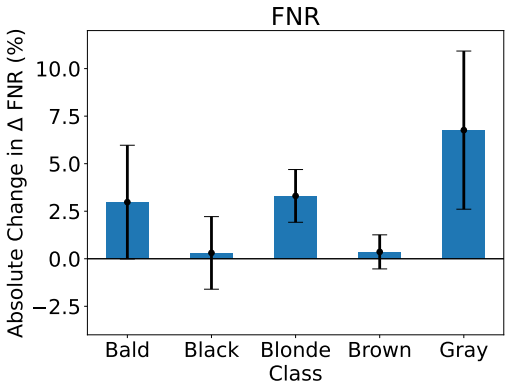


(b) Average absolute change in Δ FPR with 95% confidence intervals.

Figure 4.14: Initial Δ FPRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.



(a) Average Δ FNR with 95% confidence intervals.

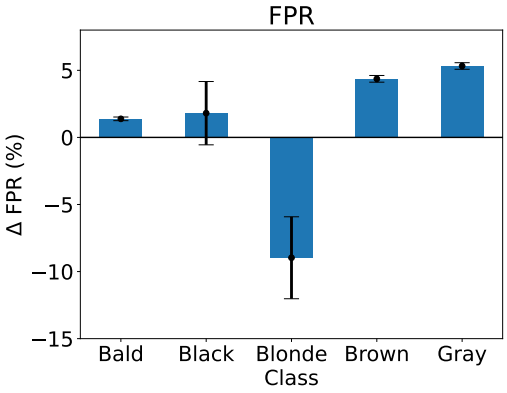


(b) Average absolute change in Δ FNR with 95% confidence intervals.

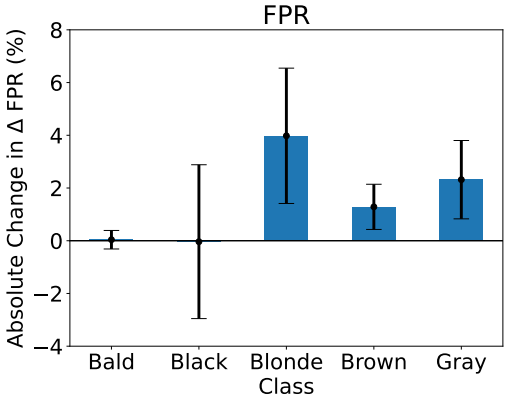
Figure 4.15: Initial Δ FNRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.

Gender

For the protected attribute gender, we compare FPR and FNR between male and female, i.e. the difference in FPR/FNR between male and female. As there are very few females people that are bald, the results for bald should be analyzed with caution. In terms of FPR, we observe that there is a significant change for blonde, brown and gray. In terms of FNR, the only significant change is for brown.

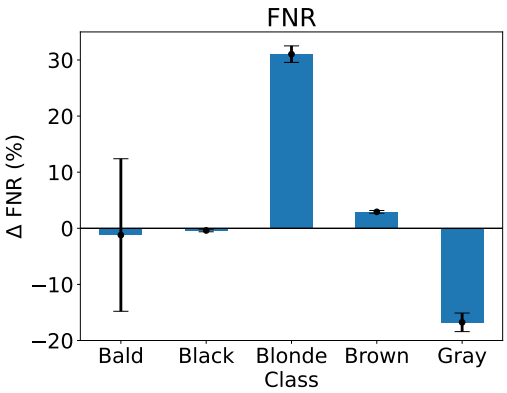


(a) Average Δ FPR with 95% confidence intervals.

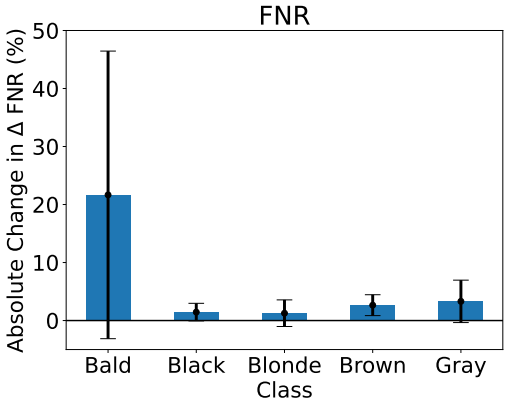


(b) Average absolute change in Δ FPR with 95% confidence intervals.

Figure 4.16: Initial Δ FPRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.



(a) Average Δ FNR with 95% confidence intervals.



(b) Average absolute change in Δ FNR with 95% confidence intervals.

Figure 4.17: Initial Δ FNRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.

Chapter 5
Discussion

The discussion is separated into protected-attribute and class-level bias. For protected attributes, we discuss the results of our ResNet-18 binary classifier and the MobileNets, which we analyzed from a protected-attribute perspective. On the class-level, we discuss the impacts of using different architectures and hyperparameters. Last, we comment on the results of our mitigation attempts.

5.1 Protected Attributes

In this section the effect of quantization on the tendency of one age group or gender to have better performance than the other is discussed for hair color classification.

5.1.1 ResNet-18 Binary Classification

Looking at table 4.1 we find similar effects of quantization to those of [11] for 95 % pruning, the difference mainly being the smaller scale of the effects for quantization. For instance, the Male Old and Male relative increases in FPR are the largest in both results. However, some FNR and FPR are improved when quantizing, which is not the case in [11]. Even though the relative changes appear to amplify bias, the absolute changes are minimal. This speaks to the robustness of ResNet-18 under quantization. This robustness makes the impact of quantization on algorithmic bias negligible.

5.1.2 MobileNet Multiclass Classification

From figures 4.12 and 4.13, we observe that MobileNetV3Small models tend to increase the accuracy difference between protected attributes. It is also interesting that the size of the change in Δaccuracy seems to align with the FNR bias metrics in 4.3, where L1-regularization

has the most biased changes and class weighting the least. We argue that this increased bias is mainly a consequence of the class-level impact of quantization and the distribution of protected attributes across the different classes. For instance, both males and old people are frequent in classes bald and gray, which are affected the most in terms of FNR. Also, as we will discuss below, there is not much increase in bias inside classes, i.e., the performance differences between protected attributes do not generally increase.

For class-level protected attribute analysis, sub-groups gray young and bald female will not be considered because their supports are very small, and the quality of the data is poor in these sub-groups.

The initial class-level performance of a network on a protected attribute is - not surprisingly - correlated with the distribution of class frequencies for that protected attribute. This can be seen, e.g, by comparing figure 3.3 with figures A.3 a),b),A.1a),b). For instance, blonde males are very underrepresented, and this is reflected in that they have higher FNR than blonde females. Similarly, for old, the categories black, blonde, and brown are underrepresented, which is also reflected in a higher FNR for these classes. Interestingly, the performance gap for different genders on blonde is much larger than the performance gaps for different ages on black, brown, and blonde despite being similarly underrepresented. A specific reason on the blonde class for this could be the different amounts of hair between females and males, making classification harder than that between young and old. In other words, because blonde males are so underrepresented compared to females, while black or brown males are not, the network might have developed a very simple solution to blonde classification that just sums up the number of blonde pixels.

Looking at results in 4.4, the quantization changes for FNR in absolute terms appear to be similar over gender and age. We do not find a strong correlation with initial FNR/FPR values. An example is blonde males, which initially have around 30% higher FNR than females, but this difference does not increase in a statistically significant way. This is interesting since the difference is so big initially. Generally, we do not observe a statistically significant impact on the difference in FNR across a protected attribute. The only cases where we have a significant impact are blonde on age and brown on gender for MobileNetV3Small. However, for the changes in FPR after quantization, we have mixed results. For MobileNetV2, there are no interesting changes to discuss. For MobileNetV3Small, the picture is more complicated. Over age, the difference in FPR for black is amplified. However, for blonde, the opposite is true. Over gender, the difference in FPR is amplified for brown and gray, while it decreases for blonde.

For QAT, there is only one significant but small change. This is for the black FPR over age. In our opinion, the most interesting result is that the differences in average FNR (which we consider the most important performance measure) barely change or do so randomly. This is surprising considering that the class-level changes in FNR are substantial on, for instance, MobileNetV3Small. This discrepancy is difficult to reason about, but it seems that quantization affects members within a class more or less the same in terms of FNR. It is worth noting that bias can still be amplified in the worst case.

As mentioned in 2.3.6, there are also cases where differences in FPR could be problematic in the multiclass case. Unlike the FNR, the differences in FPR are more impacted by quantization. The direction of this impact seems to go both ways as, in some cases, bias is amplified or reduced, as mentioned above. In terms of FPR, we argue that PTQ *can* amplify algorithmic bias over protected attributes.

5.2 Class-Level Algorithmic Bias

In this section, the effects of quantization on class-level algorithmic bias is discussed.

5.2.1 Architecture

We have found that the effect of post-training quantization on fairness varies a lot depending on what model architecture one uses. For an overparameterized model like ResNet-18, we found that the impact of quantization was minimal for a binary classification task 4.1. If we instead look at the less overparameterized MobileNet models in 4.2 we find more interesting effects 4.2.

First, we observe that the impact of quantization on bias seems to decrease with the size of a model, where the model sizes and hyperparameters can be found in 3.1. Looking at MobileNetV2, it is seen that accuracy slightly decreases after quantization, while most bias metrics are relatively unchanged compared to the other models. V3Small is affected massively in terms of both the increase in bias and the decrease in accuracy. We attribute this in part to the smaller model size of V3Small, but there could be other architecture-specific reasons for this.

Comparing V3Small to V3Small minimalistic, we do not see the same pattern as between V2 and V3Small. While the bias increase of V3Small minimalistic is larger than V3Small, the accuracy drop is only half that of V3Small. This is probably due to architectural differences, but the image augmentation of V3Small minimalistic could also play a role.

5.2.2 Mitigation

In this thesis, we have investigated three ways of mitigating bias in image classification for a given architecture. The first way is to choose the right hyperparameters. As we will discuss, the choice of hyperparameters can significantly change how robust the network is to quantization as well as how algorithmic bias is impacted. Second, we discuss the results of QAT on MobilenetV3Small. Third, we discuss our proposed simple but, as far as we know, novel post-processing method of mitigating bias by training a small (10 kB) XGBoost-classifier that predicts the initial prediction of the baseline model from the output of the quantized model, as explained in 3.4.7.

Impact of Hyperparameters

We find in table 4.3 that hyperparameters play a vital role in the way that quantization impacts model accuracy and bias. Comparing L1-regularization with the baseline model, we observe a bias-accuracy tradeoff. As mentioned in section 2.1.1, regularization often leads to a more biased model and the model with L1-regularization is indeed more biased initially as can be mainly seen in 4.4, but also 4.3. The bias increase for L1-regularization is also larger than for the baseline in all aspects except FPR variance increase, but the accuracy drop is half as large.

For the class weighted model, we move in the opposite direction in terms of the bias-accuracy tradeoff, observing basically the opposite results to that of L1-regularization. For

the floating-point models, we find that the class-level bias is smaller than for the baseline model, as can be seen in figure 4.4 and 4.3. As for the changes after quantization, looking at table 4.3, we see that the accuracy drop is larger than for the baseline but has a smaller increase in bias.

When quantizing, we find that the absolute change in FNR/FPR is correlated with their initial values, see figure 4.4 and 4.3. For instance, the class with the largest FPR, black, increases the most while the class with the smallest FPR, bald, increases the least. Similarly, for FNR, the class with the largest FNR, bald, increases the most, while the class with the smallest FNR, black, increases the least. With a few exceptions, the same holds within a specific class for different hyperparameters. For instance, the initial FNR on bald using class weighting is lower than when using L1-regularization, and this order is preserved while considering the changes in FNR instead.

To summarize, we have seen that different hyperparameters lead to a different bias-accuracy tradeoff and that this tradeoff seems to transfer to the changes after quantization. If this is correct, problems with bias when quantizing should be reduced by decreasing the bias of the initial model.

Quantization-aware Training

Looking at table 4.5, we see that the decrease in accuracy is much smaller for QAT than PTQ. This is not surprising, considering that QAT literally entails training a model to be robust to quantization. However, it is interesting that QAT decreases bias in the model in terms of FPR/FNR variance and Max-Min, although these changes are tiny. This makes QAT the go-to method for mitigating bias as accuracy is barely affected, while bias is not impacted, at least not in this case. Also, since QAT is a flexible method, one can use other mitigation methods to mitigate this if bias is increased using ordinary QAT, such as class weighting. As such, the only drawback of QAT seems to be the cost and data needed to do QAT.

Post-processing

After looking at figures 4.5, 4.6, 4.7, and other models, we reasoned that there were information to be retrieved after quantization. For example, bald FNR when using L1-regularization is degraded by 30%, which is more than seems reasonable when looking at the corresponding t-SNE plot.

Our original XGBoost classifier managed to retrieve several percentage points in accuracy, as can be seen in 4.4. However, in all models but the L1-regularized MobileNetV3Small, the bias increased. When we instead trained the classifier with a balanced dataset, the accuracy improved less, but the bias barely changed from the floating-point model. The balanced dataset contained around 1,000 examples of each class, making this somewhat cheap in terms of data. The model is 10 kB large, so there is some added memory requirement.

Overall, this makes this a promising technique in principle, especially when QAT is not possible or is deemed to be too expensive. The XGBoost classifier in this case is not using int8 precision, so the performance might have been somewhat worse if it did. Also, the choice of algorithm here is not restricted to XGBoost but can be any classifier as long as it is good enough at predicting the output of the original model. This technique can also be combined with other mitigation methods like class weighting.

Chapter 6
Conclusions

Recall our research questions from section 1.2. **How does quantization affect algorithmic bias?** We conclude that for the task of hair color classification, quantization can impact algorithmic bias on the class-level depending on architecture. However, as for protected attributes, we find that the change in bias is minimal when class-level impacts are taken into account. We also conclude that the quantized change in bias-accuracy tradeoff correlates with the initial tradeoff, even if the initial discrepancies are tiny. This implies that the choice of hyperparameters also impact the change in bias-accuracy tradeoff after quantization.

If quantization increases algorithmic bias, how can we mitigate it? We find QAT the best method for mitigating bias while maintaining high overall accuracy. In case QAT is deemed to be too expensive, a post-processing classifier like ours can be used. We also conclude that a good way to mitigate bias is to mitigate biases during the training phase with class weighting or similar techniques.

References

[1] Randall Balestriero, Leon Bottou, and Yann LeCun. *The Effects of Regularization and Data Augmentation are Class Dependent*. arXiv, 2022. DOI: 10.48550/ARXIV.2204.03632.

[2] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. http://www.fairmlbook.org. 2019.

[3] Cody Blakeney et al. *Measure Twice, Cut Once: Quantifying Bias and Fairness in Deep Neural Networks*. arXiv, 2021. DOI: 10.48550/ARXIV.2110.04397.

[4] Gunnar Blom et al. *Sannolikhetsteori och statistikteori med tillämpningar*. Studentlitteratur, 2017. ISBN: 978-91-44-12356-1.

[5] Joy Buolamwini and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. Proceedings of Machine Learning Research. PMLR, 23–24 Feb 2018, pp. 77–91.

[6] Tianqi Chen and Carlos Guestrin. "XGBoost". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Aug. 2016. DOI: 10.1145/2939672.2939785.

[7] F. Chollet. *Deep Learning with Python, Second Edition*. Manning, 2021. ISBN: 9781617296864. URL: https://books.google.co.uk/books?id=XHpKEAAAQBAJ.

[8] Amir Gholami et al. *A Survey of Quantization Methods for Efficient Neural Network Inference*. 2021. arXiv: 2103.13630 [cs.CV].

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

[10] Moritz Hardt, Eric Price, and Nathan Srebro. *Equality of Opportunity in Supervised Learning*. arXiv, 2016. DOI: 10.48550/ARXIV.1610.02413.

[11] Sara Hooker et al. *Characterising Bias in Compressed Models*. 2020. arXiv: 2010.03058 [cs.LG].

[12] Andrew Howard et al. *Searching for MobileNetV3*. arXiv, 2019. DOI: 10.48550/ARXIV.1905.02244.

[13] Benoit Jacob et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. arXiv, 2017. DOI: 10.48550/ARXIV.1712.05877.

[14] Vinu Joseph et al. *Going Beyond Classification Accuracy Metrics in Model Compression*. 2021. arXiv: 2012.01604 [cs.CV].

[15] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015, pp. 3730–3738.

[16] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605.

[17] Roman C. Maron et al. "Systematic outperformance of 112 dermatologists in multiclass skin cancer image classification by convolutional neural networks". In: *European Journal of Cancer* 119 (2019), pp. 57–65. ISSN: 0959-8049. DOI: https://doi.org/10.1016/j.ejca.2019.06.013.

[18] Gaurav Menghani. *Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better*. 2021. arXiv: 2106.08962 [cs.LG].

[19] Markus Nagel et al. *A White Paper on Neural Network Quantization*. 2021. arXiv: 2106.08295 [cs.LG].

[20] Mark Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: (2018). DOI: 10.48550/ARXIV.1801.04381.

[21] Tao Sheng et al. "A Quantization-Friendly Separable Convolution for MobileNets". In: *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)* (Mar. 2018). DOI: 10.1109/emc2.2018.00011.

[22] TensorFlow. *Post-training quantization*. Accessed 2022-05-09 15:41:00. URL: https://www.tensorflow.org/lite/performance/post_training_quantization.

[23] TensorFlow. *TensorFlow Model Optimization*. Accessed 2022-05-02 08:34:00. URL: https://www.tensorflow.org/lite/performance/model_optimization.

[24] Shoujin Wang et al. "Training deep neural networks on imbalanced data sets". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 4368–4374. DOI: 10.1109/IJCNN.2016.7727770.

[25] Stone Yun and Alexander Wong. *Do All MobileNets Quantize Poorly? Gaining Insights into the Effect of Quantization on Depthwise Separable Convolutional Networks Through the Eyes of Multi-scale Distributional Dynamics*. 2021. arXiv: 2104.11849 [cs.CV].

[26] Michael Zhu and Suyog Gupta. *To prune, or not to prune: exploring the efficacy of pruning for model compression*. arXiv, 2017. DOI: 10.48550/ARXIV.1710.01878.

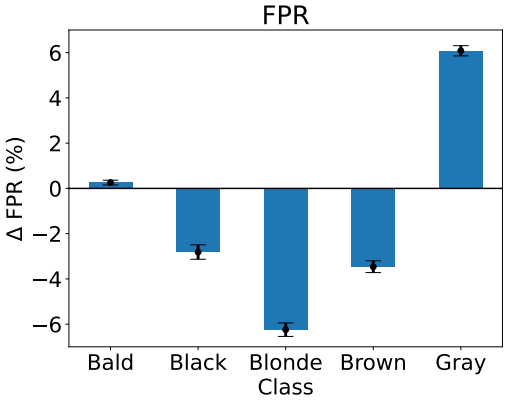
Appendices

Appendix A

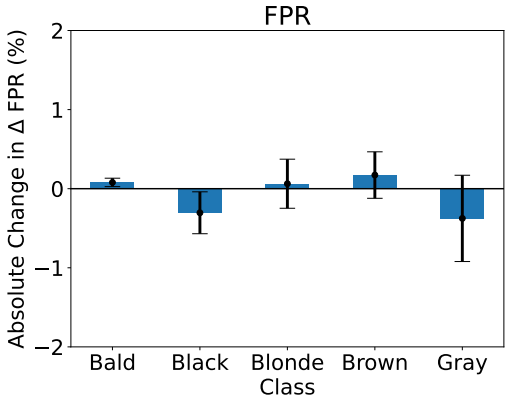
A.1 Protected Attributes

A.1.1 MobileNetV2

Age

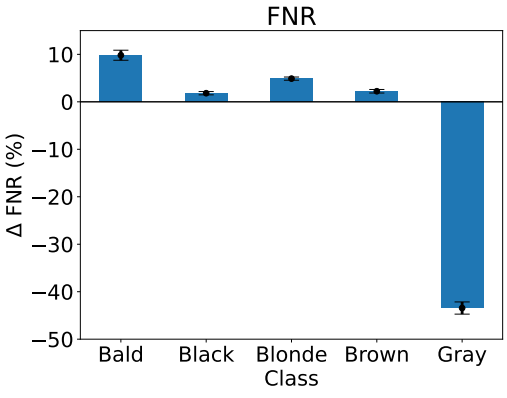


(a) Average Δ FPR with 95% confidence intervals.

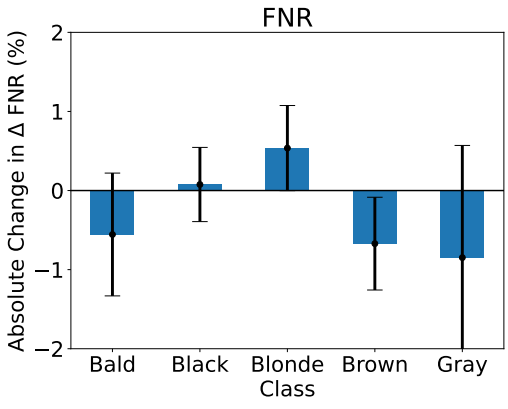


(b) Average absolute change in Δ FPR with 95% confidence intervals.

Figure A.1: Initial Δ FPRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.



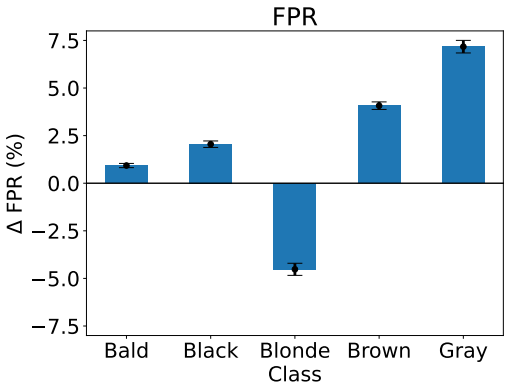
(a) Average Δ FNR with 95% confidence intervals.



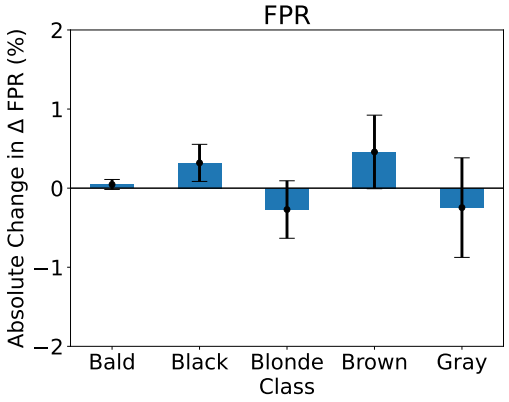
(b) Average absolute change in Δ FNR with 95% confidence intervals.

Figure A.2: Initial Δ FNRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.

Gender

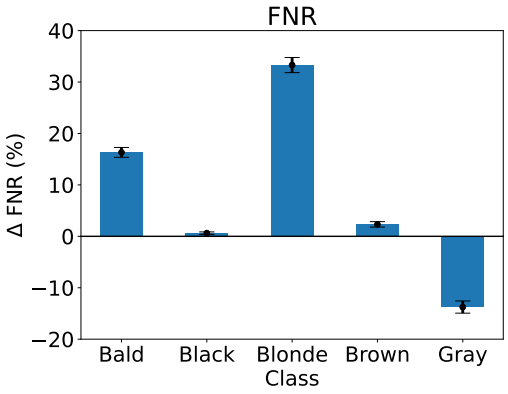


(a) Average Δ FPR with 95% confidence intervals.

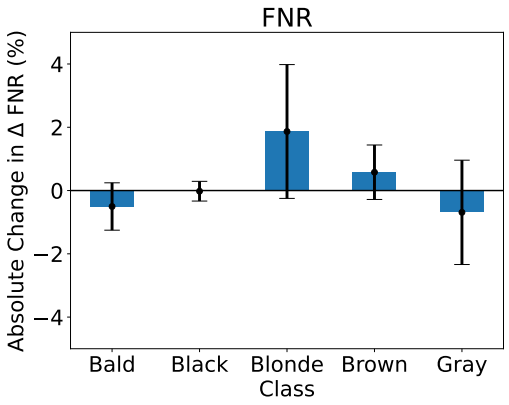


(b) Average absolute change in Δ FPR with 95% confidence intervals.

Figure A.3: Initial Δ FPRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.



(a) Average Δ FNR with 95% confidence intervals.

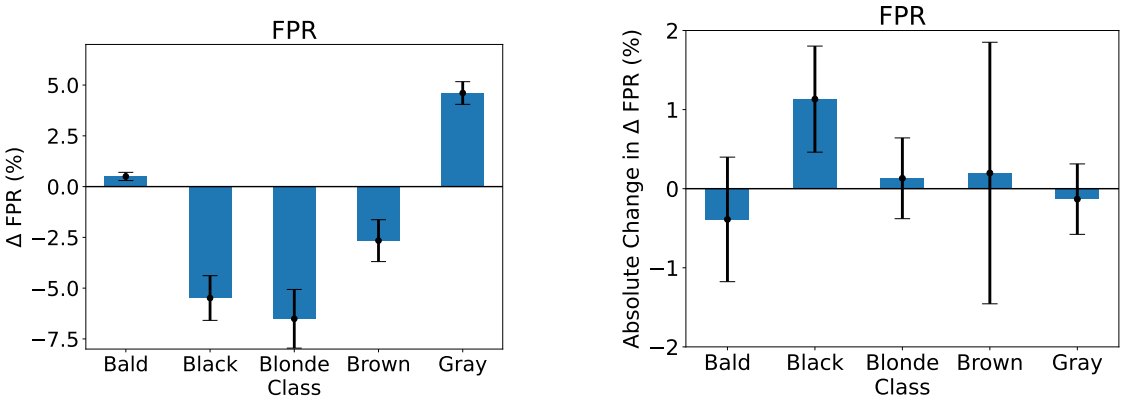


(b) Average absolute change in Δ FNR with 95% confidence intervals.

Figure A.4: Initial Δ FNRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.

A.1.2 Quantization-aware Training

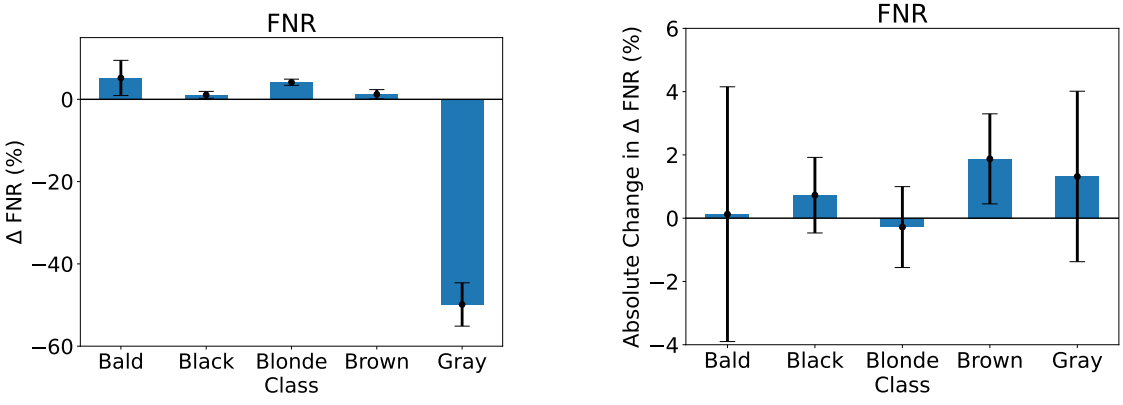
Age



(a) Average Δ FPR with 95% confidence intervals.

(b) Average absolute change in Δ FNR with 95% confidence intervals.

Figure A.5: Initial Δ FPRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.

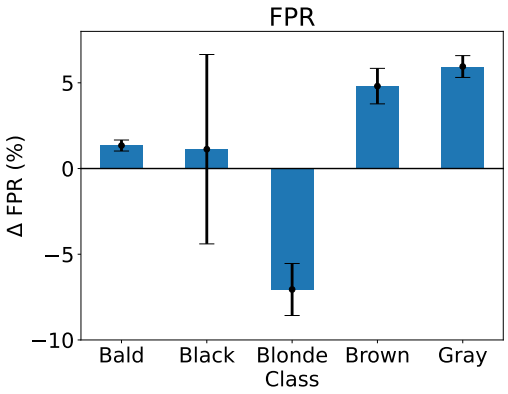


(a) Average Δ FNR with 95% confidence intervals.

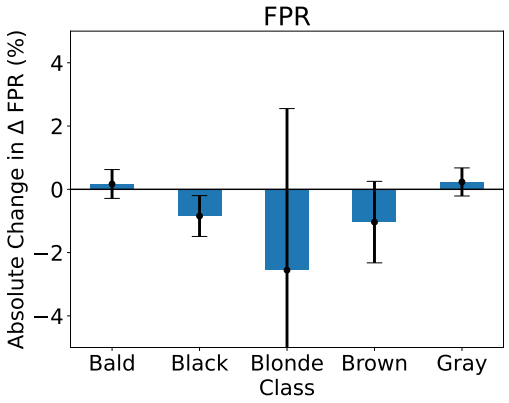
(b) Absolute change in Δ FNR with 95% confidence intervals.

Figure A.6: Initial Δ FNRs and changes between old and young. Note that the scale of the vertical axis is different between the two plots.

Gender

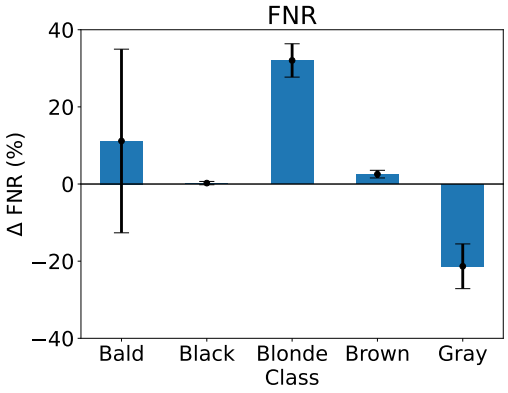


(a) Average Δ FPR with 95% confidence intervals.

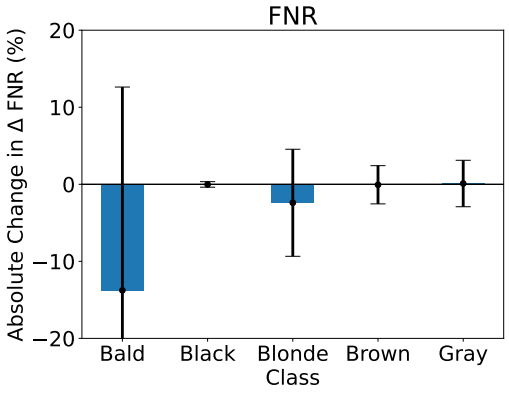


(b) Average absolute change in Δ FPR with 95% confidence intervals .

Figure A.7: Initial Δ FPRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.



(a) Average Δ FNR with 95% confidence intervals.



(b) Average absolute change in Δ FNR with 95% confidence intervals.

Figure A.8: Initial Δ FNRs and changes between male and female. Note that the scale of the vertical axis is different between the two plots.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-06-03

EXAMENSARBETE Investigating and Mitigating Effects of Quantization on Algorithmic Bias
STUDENTER Oscar Andersson, William Isaksson
HANDLEDARE Flavius Gruian (LTH), Axel Berg (Arm), Felix Johnny Thomasmathibalan (Arm)
EXAMINATOR Jacek Malec (LTH)

Påverkan av kvantisering på algoritmiskt bias i deep learning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Oscar Andersson, William Isaksson**

Kvantisering av neurala nätverk gör det möjligt att använda sig av maskininlärning där det annars inte vore möjligt som t.ex. mikroprocessorer. I detta arbete undersöker vi hur kvantiseringen påverkar prestanda olika för olika grupper av datan. Vi föreslår också hur man kan lindra dessa icke önskvärda effekter.

Deep learning har blivit en allt mer populär typ av maskininlärning. Deep learning utgörs av artificiella neurala nätverk som är inspirerade av den mänskliga hjärnan på en ytlig nivå. Dessa nätverk kan lära sig mönster genom att träna på väldigt stora mängder data som t.ex. bilder. De kan t.ex. lära sig att avgöra vilken hårfärg en person har. Ett sådant nätverk lär sig att avgöra hårfärg enbart genom att titta på bilder med associerad hårfärg. Den kommer lära sig trender i dessa bilder, som kanske inte är önskvärda. T.ex. om nästan alla bilder på blonda personer består av kvinnor, riskerar nätverket att lära sig att män inte kan vara blonda. Detta är ett exempel på algoritmiskt bias. Igenkänning av hårfärg är inte särskilt problematiskt, men i andra fall, som t.ex. igenkänning av olika typer av cancer, kan sådant algoritmiskt bias vara problematiskt.

Deep learning har traditionellt sett varit resurskrävande, men tack vare bättre processorer och bättre mjukvara, kan man idag använda deep learning även i resursbegränsade miljöer, så som en smartwatch eller till och med en mikrovågsugn. Neurala nätverk använder sig av en stor mängd parametrar (eller sparade nummer), som tar upp mycket datorminne och datorkraft. Dessa nummer tar upp olika mycket minne beroende på dess precision, t.ex. hur många decimaler som sparats. Att reducera precisionen av ett nätverk kallas kvantisering av nätverket. Kvantisering är i vissa fall nödvändigt för att ett nätverk ska kunna användas i en digital enhet. Kvantisering kan påverka prestandan på vissa grupper oproportionerligt mycket, t.ex. skulle nätverket kunna bli mycket sämre på att avgöra om en person är brunett, men i övrigt vara opåverkat.

I detta examensarbete har vi undersökt hur prestandan på olika hårfärger påverkas olika av kvantisering. Vi undersöker också om prestandan för en hårfärg påverkas olika mellan män och kvinnor samt gamla och unga personer. Detta undersöks på fyra nätverk och vi kommer fram till att underrepresenterade härfärger påverkas mer negativt än andra hårfärger. Vi undersöker också hur olika nätverk och olika inställningar av nätverk påverkas olika av kvantisering.

För att lindra dessa icke önskevärda effekter, föreslår vi ett par strategier. Vår första strategi är att se till datan som nätverket lär sig av är balanserad. Vår andra strategi ser till att nätverket påverkas mindre av kvantisering och således påverkar algoritmiskt bias mindre också. Det tredje alternativet handlar om att kompensera för förändringarna i det kvantiserade nätverket.