On The Fairness Impacts of Hardware Selection in Machine Learning

Sree Harsha Nelaturu [*1,2] **Nishaanth K Ravichandran** [*1] **Cuong Tran** [3,4] **Sara Hooker** [5] **Ferdinando Fioretto** [4]

Abstract

In the machine learning ecosystem hardware selection is often regarded as a mere utility, overshadowed by the spotlight on algorithms and data. This is especially relevant in contexts like machine learning as-a-service platforms, where users often lack control over the hardware used for model training and deployment. This paper investigates the influence of hardware on the delicate balance between model performance and fairness. We demonstrate that hardware choices can exacerbate existing disparities, and attribute these discrepancies to variations in gradient flows and loss surfaces across different demographic groups. Through both theoretical and empirical analysis, the paper not only identifies the underlying factors but also proposes an effective strategy for mitigating hardware-induced performance imbalances.

1. Introduction

The leap in capabilities of modern machine learning (ML) models has been powered primarily by the availability of large-scale datasets, gains in available compute, and the development of algorithms that can effectively use these resources (Radford et al., 2019; Brown et al., 2020). As ML-based systems become integral to decision-making processes that bear considerable social and economic consequences, questions about their ethical application inevitably surface. While an active area of research has been devoted to understanding algorithmic choices and their implications on fairness (Hooker et al., 2020; Quan et al., 2023; Caton & Haas, 2020) and robustness (Carlini & Wagner, 2017; Waqas et al., 2022) in neural networks, there has been limited work to date concerning the influence of hardware tooling on these critical aspects of model performance (Hooker, 2020; Zhuang et al., 2022; Jean-Paul et al., 2019).

This inquiry is especially pertinent as the ML hardware landscape undergoes substantial diversification, from successive generations of GPUs to custom deep-learning accelerators like TPUs (Jouppi et al., 2017).

This is significant, as ML models are frequently trained on ML services where there is limited choice in hardware, often geared towards increasingly specialized AI hardware, encouraging the use of a narrow range of ML frameworks (Mince et al., 2023). More importantly, recent studies have indicated that models trained on different hardware can exhibit varying levels of accuracy (Zhuang et al., 2022). A potential explanation is that hardware-induced nuances, such as precision discrepancies and threading behaviours, may lead iterative optimizers to different local minima during training (Hooker, 2020).

This paper further shows that these hardware-induced variations can disproportionately affect different groups, leading to a "rich get richer, poor get poorer" dynamic. This effect is depicted in Figure 1, which shows the variable impact of hardware changes on both a facial recognition task accuracy (left) and on an image classification task (right) across demographic groups or classes. Note that the only variable factor in this study is the underlying GPU adopted for training, while all other sources of software-related randomness are controlled. Remarkably, while the accuracy rates for majority groups (illustrated with lighter colors) remain relatively stable across different hardware configurations, the rates for minority groups (darker colors) exhibit considerable variability (left). This disparity also arises in balanced datasets (right).

Building on these observations, this work leverages a theoretical framework aimed at quantifying hardware-induced performance disparities and reveals, through an extensive empirical validations, that hardware choices systematically alter not just accuracy but also fairness. Our findings suggest that two key mechanisms contribute to these disparities: **(1)** variations in gradient flows across groups, and **(2)** differences in local loss surfaces. Informally, the former affects local optimality for groups, while the latter pertains to model separability. We analyze these contributing factors in detail, providing both theoretical and extensive

[*]Equal contribution [1]Cohere For AI Community [2]Saarland University [3]Dyania Health [4]University of Virginia [5]Cohere For AI. Correspondence to: Sara Hooker <sarahooker@cohere.com>, Ferdinando Fioretto <fioretto@virginia.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

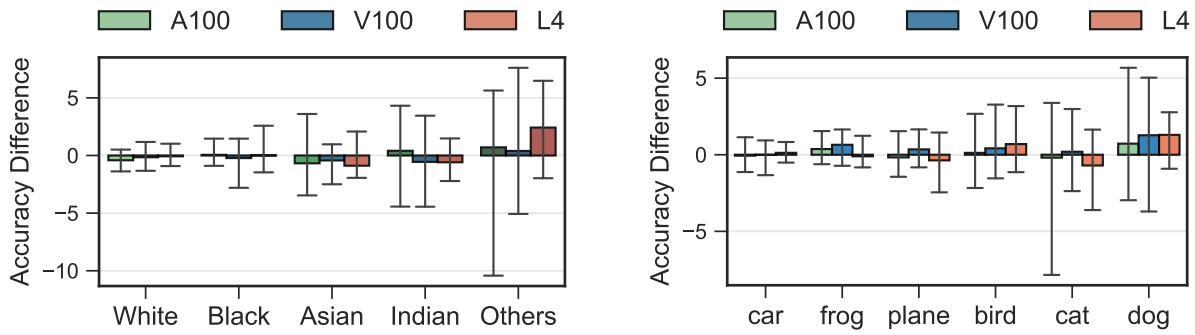


Figure 1: A model (ResNet34) with the same parameters (random seeds, epochs, batch-size) on different hardware can have vastly different performance results, especially for minority groups (dark colors). The reference hardware is T4. **Left**: UTK-Face, **Right**: CIFAR-10.

empirical experiments across various hardware configurations, networks, and datasets. Additionally, by recognizing these factors, we propose a simple yet effective technique that can be used to mitigate the disparate impacts caused by hardware tooling. The proposed method relies on an alteration to the training procedure to augment the training loss with the factors identified as responsible for unfairness to arise.

Our study stands out for its breadth, conducting experiments that cover a range of hardware architectures, datasets, and model types and the reported results highlight the critical influence of hardware on both performance and ethical dimensions of machine learning models.

2. Related Work

The intersection of hardware selection and fairness in ML is an emerging area of research that has received limited attention. For example, the stochastic effects introduced by software dependencies, such as compilers and deep learning libraries, have been recently shown to impact model performance (Hong et al., 2013; Pham et al., 2020). However, these studies have evaluated these effects within the constraints of specific setups, leaving a gap in understanding how hardware selection affect fairness in ML.

Research on ML fairness has predominantly focused on algorithmic aspects. The interplay between fairness and efficiency has been examined through the lens of model compression techniques like pruning and quantization (Xu & Hu, 2022; Ahia et al., 2021; Tran et al., 2022). Another possibly related line of work is that which looks at the relationship between fairness and privacy in ML systems (Bagdasaryan et al., 2019; Cummings et al., 2019; Tran et al., 2021a; Tran & Fioretto, 2023; Tran et al., 2021c; Zhu et al., 2022; Das et al., 2024). In particular, Differential Privacy (Dwork et al., 2006), an algorithmic property often employed to protect sensitive data in data analytics tasks, has been shown to conflict with fairness objectives. (Fioretto et al., 2022) surveys the recent progress in this area, exploring this tension, and suggesting that achieving both privacy and fairness is not solely a data-driven issue but may also require careful algorithmic design.

Finally, the influence of randomness introduced through algorithmic choices, including the impact of random seed, initialization, and data handling, has been a focal point of research. Summers & Dinneen (2021) benchmark the separate impact of choices of initialization, data shuffling and augmentation. Ko et al. (2023) evaluate how ensembling can mitigate unfair outcomes. Another body of scholarship has focused on sensitivity to non-stochastic factors including choice of activation function and depth of model (Snapp & Shamir, 2021; Shamir et al., 2020), hyperparameter choices (Lucic et al., 2018; Henderson et al., 2017; Kadlec et al., 2017; Bouthillier et al., 2021), the use of data parallelism (Shallue et al., 2019) and test set construction (Søgaard et al., 2021; Lazaridou et al., 2021; Melis et al., 2018). While these factors are critical in training phases, they do not study how hardware selection may influence the model outcomes.

Our analysis is inspired by the work of (Tran et al., 2021a), who study the disparate impact of learning under differential privacy. Our aim is to offer insights into how hardware choices can impact the balance between model performance and fairness.

3. Preliminaries

We consider a dataset D consisting of n datapoints $(\boldsymbol{x}_i, a_i, y_i)$, with $i \in [n]$, drawn i.i.d. from an unknown distribution Π. Therein, $\boldsymbol{x}_i \in \mathcal{X}$ is a feature vector, $a_i \in \mathcal{A}$

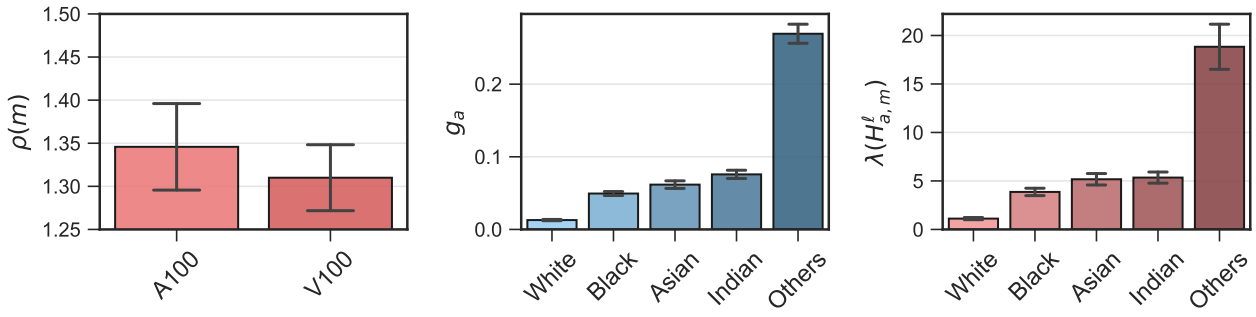


Figure 2: Illustration of the three main components in Theorem 4.1. **Left**: Difference in model parameter $\rho(m) = \max'_m \|\boldsymbol{\theta}^*_m - \boldsymbol{\theta}^*_{m'}\|_2$ when $m = T4$. **Middle**: Gradient flows $\|\boldsymbol{g}^{\ell}_a\|$ on T4 for five races. **Right**: Hessian max eigenvalues $\lambda(\boldsymbol{H}^{\ell}_{a,m})$ on T4 for five races.

with $\mathcal{A} = [g]$ (for some finite g) is a demographic group attribute, and $y_i \in \mathcal{Y}$ is a class label. For example, in a face recognition task, the training example feature $\boldsymbol{x}_i$ may describe a headshot of an individual, the protected attribute a_i the individual's gender or ethnicity, and y_i the identity of the individual. The goal is to learn a predictor $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, where $\boldsymbol{\theta}$ is a k-dimensional real-valued vector of parameters that minimizes the empirical risk function:

$$\overset{\star}{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; D) = \frac{1}{n}\sum_{i=1}^{n} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i), \qquad (1)$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a non-negative *loss function* that measures the model quality.

In this work, our focus is on analyzing the impact of different hardware, used when optimizing the above expression, in relation to the model fairness (as defined next). The paper uses $\overset{\star}{\boldsymbol{\theta}}_m$ to denote the parameters of a model training on hardware $m \in \mathcal{M}$, the set of all possible hardware.

Fairness. The fairness analysis focuses on the notion of *hardware sensitivity*, defined as the difference among the risk functions of some protected group a of models trained on a different hardware m' from a reference hardware m:

$$\Delta(a, m) = \max_{m' \in \mathcal{M}} |J(\overset{\star}{\boldsymbol{\theta}}_m; D_a) - J(\overset{\star}{\boldsymbol{\theta}}_{m'}; D_a)|. \qquad (2)$$

Therein, D_a denotes the subset of D containing samples $(\boldsymbol{x}_i, a_i, y_i)$ whose group membership $a_i = a$. Intuitively, the hardware sensitivity represents the change in loss (and thus, in accuracy) that a given group experiences as a result of hardware tooling. Fairness is measured in terms of the maximal *hardware loss difference*, also referred to as *fairness violation* across all groups:

$$\xi(D, m) = \max_{a, a' \in \mathcal{A}} |\Delta(a, m) - \Delta(a', m)|, \qquad (3)$$

defining the largest hardware sensitivity across all protected groups. A fair training method would aim at minimizing the hardware sensitivity across different hardware.

The goal of the paper is to shed light on why fairness issues arise when the only difference in training aspects of a model is the hardware in which it was trained.

4. Fairness analysis in tooling

To gain insights into how tooling may introduce unfairness, we start by providing a useful bound for the hardware sensitivity of a given group. Its goal is to isolate key aspects of tooling that are responsible for the observed unfairness. The following assumes the loss function $\ell(\cdot)$ to be at least twice differentiable, which is the case for common ML loss functions, such as mean squared error or cross-entropy loss. We report proofs of all theorems in the Appendix.

Theorem 4.1. *Given reference hardware m, the **hardware sensitivity** $\Delta(a, m)$ of group $a \in \mathcal{A}$ is upper bounded by:*

$$\Delta(a, m) \leq \left\|\boldsymbol{g}^{\ell}_{a,m}\right\|_2 \times \rho(m) + \frac{1}{2}\lambda\left(\boldsymbol{H}^{\ell}_{a,m}\right) \times \rho(m)^2 + \mathcal{O}\left(\rho(m)^3\right), \qquad (4)$$

where $\rho(m) = \max_{m \in \mathcal{M}} \|\boldsymbol{\theta}^*_m - \boldsymbol{\theta}^*_{m'}\|_2$, $\boldsymbol{g}^{\ell}_{a,m} = \nabla_{\boldsymbol{\theta}} J(\overset{\star}{\boldsymbol{\theta}}_m ; D_a)$, and $\boldsymbol{H}^{\ell}_{a,m} = \nabla^2_{\boldsymbol{\theta}} J(\overset{\star}{\boldsymbol{\theta}}_m; D_a)$, with $\lambda(\Sigma)$ as the maximum eigenvalue of matrix Σ.

The upper bound is derived using a second-order Taylor expansion and Rayleigh quotient properties and is inspired by the analysis of Tran et al. (2021a;c).

Firstly, empirically, we find that this upper bound closely approximates the hardware sensitivity in practice, while also observing that the contribution of the third-order term in Equation (4) is negligible. This empirical validation is consistent with observations in existing literature (Vadera & Ameen, 2022; Gu & Guo, 2021).

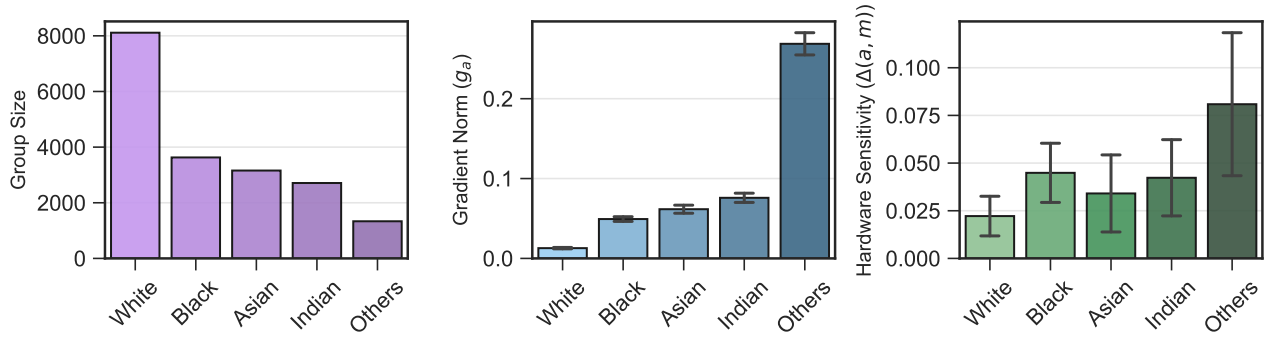


Figure 3: Illustration of Impact of group size on Gradient Norm Imbalance as shown in Theorem 4.3. **Left**: Group size used in training for five races. **Middle**: Gradient norms $\boldsymbol{g}_a$ averaged across three devices for five races and 10 seeds each. **Right**: Hardware Sensitivity; Notice higher sensitivity as the group size decreases.

Next, we note that the constant factor $\rho(m)$ is non-zero, as evidenced by Figure 2 (left). These two observations emphasize the presence of two key group-dependent terms in Equation (4) that modulate hardware sensitivity and form the crux of our fairness analysis. Specifically, they are **(1)** the norms of the gradients $\boldsymbol{g}^{\ell}_{a,m}$ (also called gradient flows) and **(2)** the maximum eigenvalue of the Hessian matrix $\boldsymbol{H}^{\ell}_{a,m}$ for a given group a and reference hardware m. Informally, the first term relates to the local optimality within each group, whereas the second term is indicative of the model's capacity to distinguish between different groups' data. Figure 2 provides an illustration of the disparity of these components across protected groups. We will subsequently demonstrate that these components serve as the primary sources of unfairness attributed to tooling.

The next sections analyze the effect of gradient flows and the Hessian to unfairness in the models trained on different hardware. This understanding, besides clarifying the roles of these components onto (un)fairness, will help us designing an effective mitigation technique, introduced in Section 7.

4.1. Group Gradient Flows

Theorem 4.1 illustrates that a key determinant of unfairness in hardware selection lies in the differences in gradient flows across groups. It points out that larger gradient flows within a group are associated with increased hardware sensitivity for that group.

The size of the training group, in particular, plays a significant role in determining the associated gradient flows. Theorem 4.2 explores this phenomenon in binary group settings, illustrating how differences in group sizes can lead to distinct gradient norms. Subsequently, Theorem 4.3 broadens the scope of this analysis to multi-group contexts, under mild assumptions.

Theorem 4.2. *Consider a local minima $\boldsymbol{\theta}^*_m$ of Equation (1) on a reference hardware $m \in \mathcal{M}$ and let the set of protected groups be $\mathcal{A} = \{a, b\}$. If $|D_a| > |D_b|$ then $\|\boldsymbol{g}_a\| < \|\boldsymbol{g}_b\|$.*

The proof is derived by leveraging the conditions for a local minimum and the proportional contributions of each group to the total gradient. This result explains why *smaller groups yield larger gradient norms, which consequently amplify sensitivity to stochasticity introduced by hardware*, as observed in our experimental results. We next generalize these insights to arbitrary group sets $\mathcal{A}$.

Theorem 4.3. *Consider a particular hardware $m \in \mathcal{M}$, suppose for any group $a, a' \in \mathcal{A}$ the angle between two gradient vectors $\overrightarrow{\boldsymbol{g}^{\ell}_{a,m}; \boldsymbol{g}^{\ell}_{a',m}}$ is smaller than $\frac{\pi}{2}$. Then if we $\underline{a} = \min_{a\in\mathcal{A}} |D_a|$, i.e., the group with least number of training samples then the following holds: $\|\boldsymbol{g}^{\ell}_{\underline{a},m}\| = \max_{a\in\mathcal{A}} \|\boldsymbol{g}^{\ell}_{a,m}\|$.*

Theorem 4.3 suggests that the group with the smallest number of training samples will exhibit the largest gradient norm upon convergence. The underlying assumption—that the angle between any pair of gradient vectors is less than $\frac{\pi}{2}$—essentially posits that the learning tasks across different groups are not highly dissimilar, which is often observed in practice (Guangyuan et al., 2022).

This influence of group size on gradient norm is exemplified in Figure 3. Within the UTK-Face dataset, the *White* category has the highest number of training samples, while *Others* has the fewest (left plot). Consequently, the majority group (*White*) exhibits the smallest gradient norm, while the group *Others* shows the largest (Figure 3 middle). As corroborated by Theorem 4.1, which establishes the link between gradient norm and hardware sensitivity (unfairness), the majority group manifests the least sensitivity, whereas the minority one has the highest. This is illustrated in the right subplot of Figure 3. Additional em-

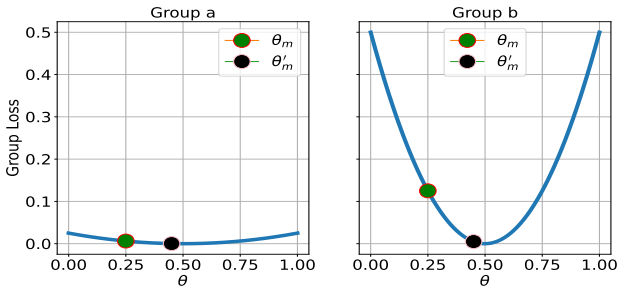


Figure 4: Illustration on the impact of group Hessians. Group 'a' has a smaller Hessian compared to Group 'b', resulting in lower sensitivity of the loss function for Group 'a'.

pirical evidence supporting the impact of group sizes on gradient norms is provided in Section 6.

4.2. Group Loss Landscape

While the previous section reviewed the influence of gradient flows on the unfairnes observed in tooling, Theorem 4.1 introduces another critical variable in determining hardware sensitivity: the *eigenvalues of the group Hessians* and suggests that groups with larger eigenvalues are more susceptible to higher hardware sensitivity. Intuitively, the eigenvalues associated to the group Hessian serves as an indicator for the *flatness* of the loss landscape around the optimal solution (Li et al., 2018), as well as for the model's generalization capability (Kaur et al., 2023).

For illustrative purposes, Figure 4 represents how differences in Hessians maximum eigenvalues impact hardware sensitivity. In this example, group a has a flatter loss landscape around the stationary point compared to group b due to its smaller Hessian norm. As a result, variations in model parameters $\boldsymbol{\theta}^*_m$ and $\boldsymbol{\theta}^*_{m'}$ across hardware m and m' lead to a much smaller change in the loss function for group a than for group b. This difference underscores the direct link between Hessian norm and the degree of hardware sensitivity experienced by each group.

The next result sheds light on the underlying reasons for the observed disparities in group-specific Hessians. Theorem 4.4 establishes a relationship between the maximum eigenvalues of the group Hessian and the average distance of samples within that group to the decision boundary.

Theorem 4.4. *Consider a model $f_{\boldsymbol{\theta}^*_m}$ trained using binary cross entropy on reference hardware m. Then, $\forall a \in \mathcal{A}$, the maximum eigenvalue of the group Hessian $\lambda(\boldsymbol{H}^\ell_a)$ is bounded by:*

$$\lambda(\boldsymbol{H}^\ell_a) \leq \frac{1}{|D_a|} \sum_{(\boldsymbol{x},y)\in D_a} \delta_{\boldsymbol{x}} \times \left\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})\right\|^2 + \left|f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x}) - y\right| \times \lambda\left(\nabla^2_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})\right),$$

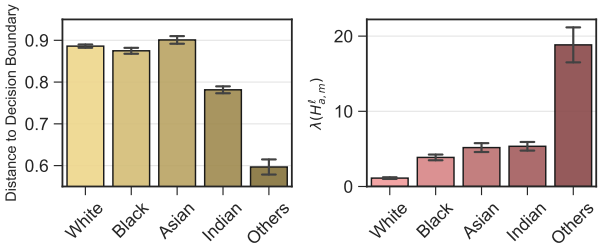


Figure 5: The relationship between Hessian norm and distance to the decision boundary.

where $\delta_{\boldsymbol{x}} = \left(f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})\right)\left(1 - f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})\right)$ is the distance to decision boundary and $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in [0,1]$ is the output obtained after the last (Sigmoid) layer.

This theorem relies on derivations of the Hessian associated with model loss function and Weyl inequality provided in Theorem 4.2. In other words, Theorem 4.4 shows that the maximum eigenvalue of the group-specific Hessian is directly linked to how close the samples from that group are to the decision boundary, as measured by the term $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})(1 - f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x}))$. Intuitively, this term is at its maximum when the classifier is most uncertain about its prediction, meaning when $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})$ is close to 0.5. Conversely, it reaches a minimum when the classifier is most certain, that is, when $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})$ approaches either 0 or 1. This relationship is further elaborated in the subsequent proposition.

Proposition 4.5. *Consider classifier $f_{\boldsymbol{\theta}*_m}(\boldsymbol{x})$ trained on hardware m. For a sample $\boldsymbol{x} \in D$, the term $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x})(1 - f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x}))$ is maximized when $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x}) = 0.5$ and minimized when $f_{\boldsymbol{\theta}^*_m}(\boldsymbol{x}) \in \{0,1\}$.*

An empirical illustration, shown in Figure 5, highlights the relationship between group Hessian eigenvalues and proximity to the decision boundary. Notice how samples from the *Others* group are closer to the decision boundary, indicating that they are less *separable* than those in other groups. As a result this group reports the largest eigenvalue of group Hessians. Similar observation on other datasets are discussed in the following section.

Having discussed the main reasons justifying unfairness in hardware selection, the next sections describes the empirical validation and discuss a possible mitigation solution.

5. Experimental Setup

We first review the experimental setup.

Hardware selection. The experiments use a variety of GPUs: Tesla T4 (NVIDIA, 2018a), Tesla V100 (NVIDIA, 2017a), Ampere A100 (NVIDIA, 2021), and Ada L4 GPU (NVIDIA, 2023). These GPUs differ in CUDA core count, total threads, and streaming multiprocessors (refer to Table

1 in the Appendix for details). Given the lack of detailed public information on the internal workings of these devices, we relied on generational differences as key indicators for our hardware selection, also considering availability and suitability while selecting the devices. In addition, we strictly control software related randomness and consider the fact that these GPUs show stochastic behavior in outputs due to varied floating-point processing as detailed in (Chou et al., 2020) and parallelization. For example, the T4 and L4 GPUs, being designed for inference, have lower memory requirements and distinct parallelization designs.

Controlling other sources of stochasticity. Our primary aim is to isolate the impact of hardware on model fairness and performance. This requires to control all other factors as much as possible. We ensure determinism by fixing the random seed for all python libraries, and ensuring consistent data-loading order and augmentations with FFCV-SSL (Bordes et al., 2023). This approach ensures that the same stochastic elements are present during training and inference. We also used the same library versions, including cudNN (Chetlur et al., 2014) with full-precision (FP32) except in the case of CelebA dataset where we used mixed-precision training. We compute hardware sensitivity by calculating differences for identical random seeds across various devices, and ensuring that all other sources of randomness are fixed, then average these over multiple seeds and report the mean and standard deviation. We also report mean and standard deviation of other metrics averaged across multiple random seeds. *This approach allows us to confidently attribute any observed variations in sensitivity or stochasticity specifically to the unique characteristics of the hardware platform.*

Datasets and architectures. Our experiments were conducted using three key datasets: CIFAR-10 (Krizhevsky, 2009), CelebA (Liu et al., 2015), and UTKFace (Zhang et al., 2017). CIFAR-10 is balanced, while UTKFace and CelebA are naturally imbalanced. To study class imbalance in CIFAR-10, we create a variant where class 8 (*Ship*) is reduced to 20% of its original size. For CelebA, we redefined the task into four classes based on *male* and *blond hair* attributes, creating an imbalanced multi-class dataset. In UTKFace, ethnicity labels are used for training. Additional details are provided in Appendix B.1.

The evaluations is performed across multiple hardware setups, hyper-parameters, datasets, and four architectures of increasing complexity: SmallCNN, ResNet18, ResNet34, and ResNet50 (He et al., 2015). More information regarding these architectures is provided in Appendix B.2 For all experiments We used SGD+momentum 0.99, with weight decay of $5e-4$, a three-phase one-cycle LR (Leslie, 2015) scheduler with a starting learning rate of 0.1. The batch size for CIFAR10, UTKFace and CelebA is set to 512, 128 (32 for ResNet50) and 200, respectively. Models trained on CIFAR10 and CelebA were trained for 15 epochs, and those trained on UTKFace for 20 epochs. Further analysis on model convergence and impact on hardware sensitivity are provided in Appendix C.

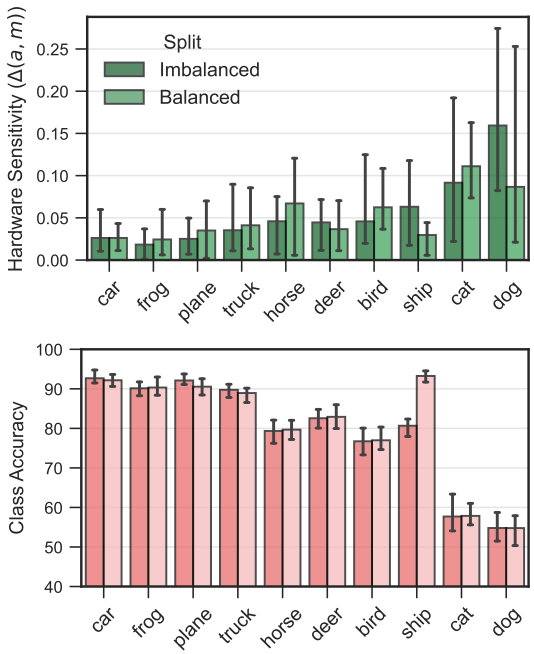


Figure 6: **Top:** Hardware sensitivity for CIFAR10 (ResNet34) Balanced (lighter) and Imbalanced (darker). **Bottom:** Class-wise accuracy. High Fairness violation are noted for the Imbalanced CIFAR10.

6. Experimental Results

Our fairness analysis relies on the notion of hardware sensitivity, as defined in Equation (2), as the maximum difference in class loss between a model trained on a reference hardware and models trained on various other hardware setups, while keeping all other parameters unchanged to control software-related randomness. The notion of hardware sensitivity is of large theoretical value as it helps us understand the impact of hardware on model performance. However, ultimately, we are interested in measuring the accuracy variations across classes, due to training a model on different hardware. Therefore, in this section, we will also examine how hardware variations contribute to differences in accuracy across different groups. When looking at hardware sensitivity, small values indicate more fair results.

In the models and architectures adopted, our analysis found notable fairness (hardware sensitivity) variations. Figure 6 illustrates this aspect for the CIFAR10 dataset.

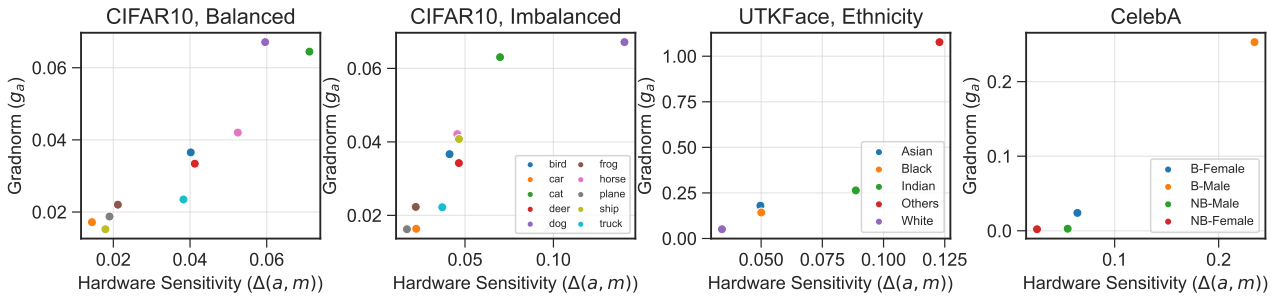


Figure 7: Correlation plot between Hardware sensitivity and gradient flows. **First:** Even with a perfectly balanced dataset, classes with higher gradient norm tend to have higher sensitivity in the change of hardware. **Second:** Class 8 (*Ship*) is imbalanced in this setting and sees a sharp increase in its gradient norm and sensitivity. **Third:**There is a strong correlation between the gradient norm of groups and the hardware sensitivity in the ascending order of imbalance for UTKFace. **Fourth:** A similar trend is also found for the CelebA classification task.

Firstly, observe that larger hardware sensitivity values for a class are associated with greater deviations in that class's accuracy. Next, also notice that classes showing smaller hardware sensitivity (indicative of greater fairness) tend to be those with higher overall accuracies. To gain a better understanding of these trends, let us examine the hardware sensitivity of class 8 (*Ship*) under both balanced and imbalanced scenarios, as depicted in Figure 6 (top). In the imbalanced setting, where class 8 (*Ship*) had five times fewer samples than other classes, there is a notable increase in hardware sensitivity from 0.017 to 0.046 compared to the balanced setting.

This pattern is not unique to one dataset. For instance, in the UTKFace dataset, the minority class *Others* exhibits significantly higher hardware sensitivity of 0.122 compared to 0.034 of the majority class *White*, this can be observed in Figure 8 (right). Similarly, in the CelebA dataset, the minority class *blond-male* has a hardware sensitivity of 0.23 which is more than 0.025 of the majority class *non-blond female* as can be seen in Figure 14 in the appendix. These observations support our hypothesis that hardware selection can disproportionately affect the performance of minority classes.

Additionally, the paper provides insights on the relation between model architecture and its size and hardware sensitivity. These results are reported in Appendix E and provide strong indication that hardware sensitivity, and thus unfairness, increase as the complexity of the model increases.

6.1. Gradient Flows

We now turn our attention to the influence of gradient flows on the disparities in accuracy resulting from hardware selection. As established in Theorem 4.1, the magnitude of gradient flow within a group is directly linked to its hardware sensitivity. The norm of the gradients, or the gradient flows, is indicative of the local optimality of the model for each group. Essentially, this term measures how sensitively the model responds to the specific characteristics within the data of each demographic group.

Larger gradient norm values suggest that the model is less optimized for that particular group, implying a greater potential for accuracy disparities due to hardware selection.

Figure 7 illustrates the relationship between the group gradient flows and the hardware sensitivity. Notice the strong correlation between a group's hardware sensitivity and its gradient flow, particularly under conditions of imbalance. In CIFAR10, in particular, unbalancing class *Ship* (five times fewer samples) results in an increase in the gradient norm from 0.015 to 0.041. This trend is also echoed in the UTKFace-ethnicity task, where the gradient norm of the majority class *White* 0.05 is significantly lower than 1.07 of the minority class *Others*. CelebA shows a similar pattern; the majority class *non-blond-female* exhibits a gradient norm of 0.002 which is much lesser than 0.25 of the minority class *blond-male*.

These observations highlight the impact of class imbalance on gradient norms and hardware sensitivity, reinforcing the idea that minority classes tend to exhibit higher sensitivity to hardware variations, which in turn can affect the accuracy of the model for these specific groups.

6.2. Distance to the Decision Boundary

Next, we look at the second factor to unfairness highlighted in Theorem 4.1: the effect of the maximum eigenvalue of the Hessian matrix of a group. Such values provide insight into the model's capacity to differentiate between the data of different groups. A larger maximum eigenvalue implies that the model's loss surface is more curved for the data of that particular group. This curvature is indicative of how

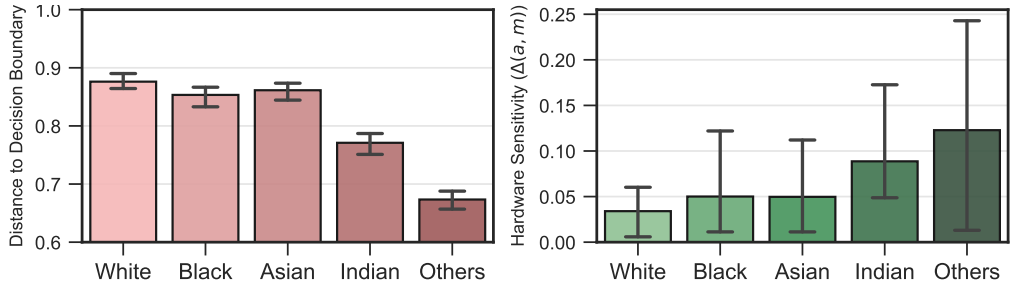


Figure 8: Relationship between distance to Decision Boundary and Hardware Sensitivity- UTKFace Ethnicity.

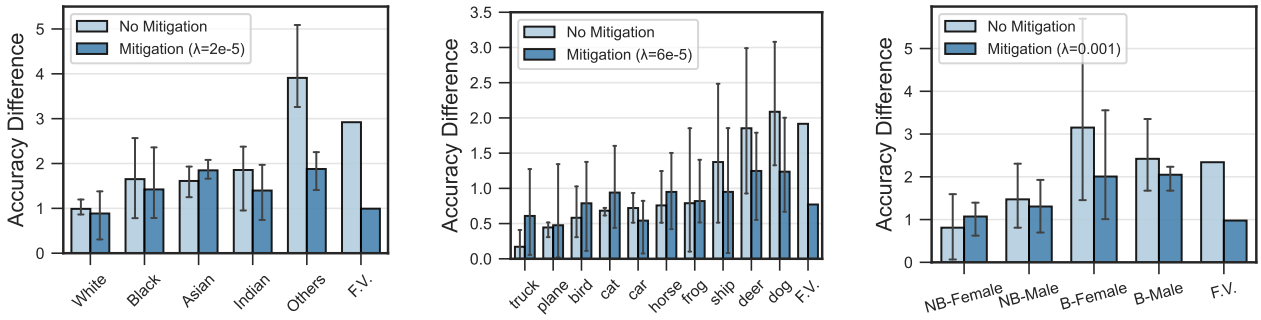


Figure 9: Accuracy difference across all hardware (analogous to hardware sensitivity) in percentage, for each class. **Left**: Mitigation applied to UTKFace ($\lambda = 2e-5$); **Middle**: Mitigation applied to CIFAR10 (Imbalanced) ($\lambda = 6e-5$); **Right**: Mitigation applied to CelebA Dataset ($\lambda = 0.001$). The last column in each subplot (named, F.V.) reports the fairness violation across all groups, which measures unfairness.

sensitive the model is to variations in the data belonging to that group. Theorem 4.4 further links this component with the distance to the boundary, and we show next how such notion connects to hardware sensitivity (unfairness).

Figure 8 illustrates the relationship between the distance to decision boundary and hardware sensitivity for the UTKFace dataset. We adopt the definition of distance to the decision boundary from (Tran et al., 2021a). For each sample $\boldsymbol{x}$, this distance is computed as $\delta_{\boldsymbol{x}} = 1 - \sum_{i=1}^{|\mathcal{Y}|} p_i^2(\boldsymbol{x})$, where $p_i(\boldsymbol{x})$ represents the softmax probability distribution of $\boldsymbol{x}$ with values ranging between 0 and 1. Notably, the average distance to the decision boundary is a strong predictor of hardware sensitivity. In cases involving a minority class, such as *Others* in our dataset, this distance is significantly smaller (0.67) compared to other classes like *White* (0.875). We also note that other datasets follows similar trends: For example, on CelebA, the average distance to decision boundary is shorter (0.56) for *blond-male* compared to *non-blond-female* (0.91) as visible in Figure 15. These findings aligns with the theoretical implications presented in our paper.

7. Mitigation Solution

Given the influence of the groups' gradient flows and group Hessians on the model unfairness due to hardware selection, one intuitive approach to mitigate the observed effects is to equalize the gradient and Hessian values across groups during training. However, this approach is computationally intensive and often impractical, especially for large models, primarily due to the challenges in computing the Hessian matrix during backpropagation. To address this issue, we propose a more efficient mitigation strategy, underpinned by the observations provided in Theorem 4.4. This result elucidates the relationship between the group Hessian and the distance to the decision boundary. We leverage this insight and aim to align the average distance to the decision boundary among different groups.

We achieve this by augmenting the loss function with a component quantifying the disparity between the group-specific and batch-wide distances to the decision boundary.

$$\overset{\star}{\boldsymbol{\theta}}_F = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; D) + \lambda \sum_{a \in \mathcal{A}} (\delta_{\mathcal{D}_a} - \delta_{\mathcal{D}})^2, \quad (5)$$

where δ_S represents the average distance to the decision boundary of samples $\boldsymbol{x} \in S$ as described in the Section 6.2, and λ is a hyper-parameter which calibrates the level of

penalization associated with this term.

In our experiments, we implemented this mitigation strategy across various hardware setups and observed a significant reduction in fairness violation. While it is possible to optimize the choice of the value λ during the empirical risk process, e.g., using a Lagrangian dual approach as in (Fioretto et al., 2020a;b; Tran et al., 2021b), we found that even a traditional simple grid search allows us to yield an effective reduction in accuracy disparity.

Figure 9 illustrates the effectiveness of the proposed method on UTKFace (left), CIFAR10 (middle), and CelebA (right). The plots report the accuracy difference, in percentage, across all hardware, for each class, as well as the models fairness violations (FV). The latter captures our unfairness metric, and is measured as the maximal accuracy difference across all hardware configurations, as detailed in Equation (3).

The results show a marked decrease in maximum difference in accuracy within various groups. For example, for the UTKFace dataset (left), despite experiencing a slightly increase in the accuracy difference for the *Asian* group, we observe a three fold reduction in unfairness due to tooling. Similar effects are observed for the CIFAR10 dataset where, using $\lambda = 6e-5$, significantly reduces fairness violations, from 1.91 (pre-mitigation) to 0.77 (post-mitigation). Finally, for the ResNet50 model on CelebA (right), this mitigation reports a fairness violation reduction from 2.34 to 0.936 post-mitigation. It is to be noted that for each dataset, different λ values produced different reductions in accuracy difference.

These results highlight the effectiveness of our proposed method in addressing fairness concerns attributable to hardware variations.

8. Conclusion

This paper focused on an often overlooked aspect of responsible model: How variations in hardware can disproportionately affect different demographic groups, leading to a Matthew's effect in performance and fairness. We've presented a theoretical framework to quantitatively assess these hardware-induced disparities, pinpointing variations in gradient flows across groups and differences in local loss surfaces as primary factors contributing to these disparities. These findings have been validated by extensive empirical studies, carried out on multiple hardware platforms, datasets, and architectures.

The findings of this study are significant: the sensitivity of model performance to specific hardware choices can lead to unintended negative societal outcomes. For example, organizations that release their source codes and model parameters may attest to satisfactory performance levels for certain demographic groups, based on results from their chosen training hardware. However, this claimed performance could substantially degrade when the models are implemented on different hardware platforms. Our work thus serves as both a cautionary tale and a guide for responsible practices in reporting across diverse hardware settings.

Impact Statement

The analyses and solutions reported in this paper should not be intended as an endorsement for using the developed techniques to aid facial recognition systems. We hope this work creates further awareness of the unfairness caused by variations in data, model, and hardware setup.

Acknowledgments

This research is partly funded by NSF grants SaTC-2133169, RI-2232054, and CAREER-2143706. F. Fioretto is also supported by a Google Research Scholar Award and an Amazon Research Award. The views and conclusions of this work are those of the authors only. We would like to thank Cohere For AI for providing a generous amount of computing for conducting and analyzing our experiments.

References

Ahia, O., Kreutzer, J., and Hooker, S. The low-resource double bind: An empirical study of pruning for low-resource machine translation. *arXiv preprint arXiv:2110.03036*, 2021.

Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. Differential privacy has disparate impact on model accuracy. *Advances in neural information processing systems*, 32, 2019.

Bordes, F., Balestriero, R., and Vincent, P. Towards democratizing joint-embedding self-supervised learning, 2023. URL https://arxiv.org/abs/2303.01986.

Bouthillier, X., Delaunay, P., Bronzi, M., Trofimov, A., Nichyporuk, B., Szeto, J., Mohammadi Sepahvand, N., Raff, E., Madan, K., Voleti, V., Ebrahimi Kahou, S., Michalski, V., Arbel, T., Pal, C., Varoquaux, G., and Vincent, P. Accounting for variance in machine learning benchmarks. In Smola, A., Dimakis, A., and Stoica, I. (eds.), *Proceedings of Machine Learning and Systems*, volume 3, pp. 747–769, 2021. URL https://proceedings.mlsys.org/paper/2021/file/cfecdb276f634854f3ef915e2e980c31-Paper.pdf.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan,

J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. URL https://arxiv.org/abs/2005.14165.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.

Caton, S. and Haas, C. Fairness in machine learning: A survey. *ACM Computing Surveys*, 2020.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. cuDNN: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014.

Chou, Y., Ng, C., Cattell, S., Intan, J., Sinclair, M. D., Devietti, J., Rogers, T. G., and Aamodt, T. M. Deterministic atomic buffering. In *53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO*, 2020.

Cummings, R., Gupta, V., Kimpara, D., and Morgenstern, J. On the compatibility of privacy and fairness. In *Adjunct publication of the 27th conference on user modeling, adaptation and personalization*, pp. 309–315, 2019.

Das, S., Romanelli, M., and Fioretto, F. Disparate impact on group accuracy of linearization for private inference. In *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2402.03629.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of Theory of Cryptography Conference*, pp. 265–284. Springer, 2006.

Fioretto, F., Hentenryck, P. V., Mak, T. W., Tran, C., Baldo, F., and Lombardi, M. Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 118–135. Springer, 2020a.

Fioretto, F., Mak, T., and Van Hentenryck, P. Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):630–637, 2020b.

Fioretto, F., Tran, C., Hentenryck, P. V., and Zhu, K. Differential privacy and fairness in decisions and learning tasks: A survey. In *International Joint Conference on Artificial Intelligence*, pp. 5470–5477. ijcai.org, 2022. doi: 10.24963/ijcai.2022/766. URL https://doi.org/10.24963/ijcai.2022/766.

Gu, H. and Guo, X. An sde framework for adversarial training, with convergence and robustness analysis. *arXiv preprint arXiv:2105.08037*, 2021.

Guangyuan, S., Li, Q., Zhang, W., Chen, J., and Wu, X.-M. Recon: Reducing conflicting gradients from the root for multi-task learning. In *The Eleventh International Conference on Learning Representations*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Henderson, P., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560, 2017.

Hong, S.-Y., Koo, M.-S., Jang, J., Kim, J.-E. E., Park, H., Joh, M.-S., Kang, J.-H., and Oh, T.-J. An evaluation of the software system dependency of a global atmospheric model. *Monthly Weather Review*, 141(11):4165 – 4172, 2013. doi: 10.1175/MWR-D-12-00352.1.

Hooker, S. The hardware lottery. *Communications of the ACM*, 64:58 – 65, 2020.

Hooker, S., Moorosi, N., Clark, G., Bengio, S., and Denton, E. Characterising bias in compressed models, 2020.

Jean-Paul, S., Elseify, T., Obeid, I., and Picone, J. W. Issues in the reproducibility of deep learning results. *2019 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pp. 1–4, 2019.

Jouppi, N. P., Young, C., Patil, N., Patterson, D. A., Agrawal, G., Bajwa, R. S., Bates, S., Bhatia, S., Boden, N. J., Borchers, A., Boyle, R., luc Cantin, P., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., Ghaemmaghami, T., Gottipati, R., Gulland, W., Hagmann, R. B., Ho, C. R., Hogberg, D., Hu, J., Hundt, R., Hurt, D., Ibarz, J., Jaffey, A., Jaworski, A., Kaplan, A., Khaitan, H., Killebrew, D., Koch, A., Kumar, N., Lacy, S., Laudon, J., Law, J., Le, D., Leary, C., Liu, Z., Lucke, K. A., Lundin, A., MacKean, G., Maggiore, A., Mahony, M., Miller, K., Nagarajan, R., Narayanaswami, R., Ni, R., Nix, K., Norrie, T., Omernick, M., Penukonda, N., Phelps, A., Ross, J., Ross, M., Salek, A., Samadiani, E., Severn, C., Sizikov, G., Snelham, M., Souter, J., Steinberg, D., Swing, A., Tan, M., Thorson, G., Tian, B., Toma, H., Tuttle, E., Vasudevan, V., Walter, R., Wang, W., Wilcox, E., and Yoon, D. H. In-datacenter performance analysis of a tensor processing unit. *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12, 2017.

Kadlec, R., Bajgar, O., and Kleindienst, J. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 69–74, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2609.

Kaur, S., Cohen, J., and Lipton, Z. C. On the maximum hessian eigenvalue and generalization. In *Proceedings on*, pp. 51–65. PMLR, 2023.

Ko, W.-Y., D'souza, D., Nguyen, K., Balestriero, R., and Hooker, S. Fair-ensemble: When fairness naturally emerges from deep ensembling. *ArXiv*, abs/2303.00586, 2023. URL https://api.semanticscholar.org/CorpusID:257254836.

Krizhevsky, A. Learning multiple layers of features from tiny images. In *University of Toronto*, 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Lazaridou, A., Kuncoro, A., Gribovskaya, E., Agrawal, D., Liska, A., Terzi, T., Gimenez, M., de Masson d'Autume, C., Kocisky, T., Ruder, S., Yogatama, D., Cao, K., Young, S., and Blunsom, P. Mind the gap: Assessing temporal generalization in neural language models, 2021.

Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H., and Madry, A. ffcv. https://github.com/libffcv/ffcv/, 2022. commit xxxxxxx.

Leslie, N. Cyclical learning rates for training neural networks, 2015.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are gans created equal? a large-scale study, 2018.

maintainers, T. and contributors. Torchvision: Pytorch's computer vision library. https://github.com/pytorch/vision, 2016.

Melis, G., Dyer, C., and Blunsom, P. On the state of the art of evaluation in neural language models. *ArXiv*, abs/1707.05589, 2018.

Mince, F., Dinh, D., Kgomo, J., Thompson, N., and Hooker, S. The grand illusion: The myth of software portability and implications for ml progress, 2023.

NVIDIA. Nvidia tesla v100 gpu architecture, 2017a. URL https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

NVIDIA. Nvidia volta architecture white paper, 2017b. URL https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

NVIDIA. Nvidia turing gpu architecture, 2018a. URL https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

NVIDIA. Nvidia turing architecture white paper, 2018b. URL https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

NVIDIA. Nvidia ampere architecture white paper, 2021. URL https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

NVIDIA. Nvidia ada architecture white paper, 2023. URL https://www.nvidia.com/en-us/technologies/ada-architecture/.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:202786778.

Pham, H. V., Qian, S., Wang, J., Lutellier, T., Rosenthal, J., Tan, L., Yu, Y., and Nagappan, N. Problems and opportunities in training deep learning software systems: An analysis of variance. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, pp. 771–783, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367684. doi: 10.1145/3324884.3416545.

Quan, T., Zhu, F., Liu, Q., and Li, F. Learning fair representations for accuracy parity. *Engineering Applications of Artificial Intelligence*, 119:105819, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Shallue, C. J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., and Dahl, G. E. Measuring the effects of data parallelism on neural network training, 2019.

Shamir, G. I., Lin, D., and Coviello, L. Smooth activations and reproducibility in deep networks, 2020.

Snapp, R. R. and Shamir, G. I. Synthesizing irreproducibility in deep networks. *CoRR*, abs/2102.10696, 2021.

Søgaard, A., Ebert, S., Bastings, J., and Filippova, K. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1823–1832, Online, April 2021. Association for Computational Linguistics.

Summers, C. and Dinneen, M. J. Nondeterminism and instability in neural network optimization, 2021.

Tran, C. and Fioretto, F. On the fairness impacts of private ensembles models. In *International Joint Conference on Artificial Intelligence*, pp. 510–518. ijcai.org, 2023. doi: 10.24963/ijcai.2023/57. URL https://doi.org/10.24963/ijcai.2023/57.

Tran, C., Dinh, M., and Fioretto, F. Differentially private empirical risk minimization under the fairness lens. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27555–27565. Curran Associates, Inc., 2021a. URL https://openreview.net/forum?id=7EFdodSWee4.

Tran, C., Fioretto, F., and Hentenryck, P. V. Differentially private and fair deep learning: A lagrangian dual approach. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 9932–9939. AAAI Press, 2021b.

Tran, C., Fioretto, F., Van Hentenryck, P., and Yao, Z. Decision making with differential privacy under a fairness lens. In *International Joint Conference on Artificial Intelligence*, pp. 560–566. ijcai.org, 2021c. doi: 10.24963/ijcai.2021/78. URL https://doi.org/10.24963/ijcai.2021/78.

Tran, C., Fioretto, F., Kim, J.-E., and Naidu, R. Pruning has a disparate impact on model accuracy. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022. URL https://openreview.net/forum?id=11nMVZK0WYM.

Vadera, S. and Ameen, S. Methods for pruning deep neural networks. *IEEE Access*, 10:63280–63300, 2022.

Waqas, A., Farooq, H., Bouaynaya, N. C., and Rasool, G. Exploring robust architectures for deep artificial neural networks. *Communications Engineering*, 1(1):46, 2022.

Xu, G. and Hu, Q. Can model compression improve nlp fairness. *arXiv preprint arXiv:2201.08542*, 2022.

Zhang, Z., Song, Y., and Qi, H. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

Zhao, H. and Gordon, G. Inherent tradeoffs in learning fair representations. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/b4189d9de0fb2b9cce090bd1a15e3420-Paper.pdf.

Zhu, K., Fioretto, F., and Hentenryck, P. V. Post-processing of differentially private data: A fairness perspective. In *International Joint Conference on Artificial Intelligence*, pp. 4029–4035. ijcai.org, 2022. doi: 10.24963/ijcai.2022/559. URL https://doi.org/10.24963/ijcai.2022/559.

Zhuang, D., Zhang, X., Song, S., and Hooker, S. Randomness in neural network training: Characterizing the impact of tooling. *Proceedings of Machine Learning and Systems*, 4:316–336, 2022.

SUPPLEMENTAL MATERIAL

A. Missing proofs

Theorem A.1. *Given a reference hardware m, the* hardware sensitivity *of a group $a \in \mathcal{A}$ is upper bounded by*[1]*:*

$$\Delta(a,m) \leq \left\|\boldsymbol{g}^{\ell}_{a,m}\right\| \times \max_{m' \in \mathcal{M}} \left\|\overset{\star}{\boldsymbol{\theta}}_m - \overset{\star}{\boldsymbol{\theta}}_{m'}\right\| + \frac{1}{2}\lambda\left(\boldsymbol{H}^{\ell}_{a,m}\right) \times \max_{m' \in \mathcal{M}} \left\|\overset{\star}{\boldsymbol{\theta}}_m - \overset{\star}{\boldsymbol{\theta}}_{m'}\right\|^2 + \mathcal{O}\left(\max_{m' \in \mathcal{M}} \left\|\overset{\star}{\boldsymbol{\theta}}_m - \overset{\star}{\boldsymbol{\theta}}_{m'}\right\|^3\right), \quad (6)$$

where $\boldsymbol{g}^{\ell}_{a,m} = \nabla_{\boldsymbol{\theta}} J(\overset{\star}{\boldsymbol{\theta}}_m; D_a)$ is the vector of gradients associated with the loss function ℓ evaluated at $\overset{\star}{\boldsymbol{\theta}}_m$ and computed using group data D_a, $\boldsymbol{H}^{\ell}_{a,m} = \nabla^2_{\boldsymbol{\theta}} J(\overset{\star}{\boldsymbol{\theta}}_m; D_a)$ is the Hessian matrix of the loss function ℓ, at the optimal parameters vector $\overset{\star}{\boldsymbol{\theta}}_m$, computed using the group data D_a (henceforth simply referred to as group hessian*), and $\lambda(\Sigma)$ is the maximum eigenvalue of a matrix Σ.*

Proof. Using a second order Taylor expansion around $\boldsymbol{\theta}^*_m$, the change in loss function of one particular group a when it is trained on another hardware $m' \in \mathcal{M}$ can be approximated as:

$$\begin{aligned} J(\boldsymbol{\theta}^*_{m'}; D_a) - J(\boldsymbol{\theta}^*_m; D_a) &= J\left(\boldsymbol{\theta}^*_m; D_a\right) + \left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right)^{\top} \nabla_{\theta} J\left(\boldsymbol{\theta}^*_m; D_a\right) \\ &\quad + \frac{1}{2}\left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right)^{\top} \boldsymbol{H}^{\ell}_a \left(\boldsymbol{\theta} *_{m'} - \boldsymbol{\theta}^*_m\right) + \mathcal{O}\left(\max_{m' \in \mathcal{M}} \left\|\overset{\star}{\boldsymbol{\theta}}_m - \overset{\star}{\boldsymbol{\theta}}_{m'}\right\|^3\right) - J\left(\boldsymbol{\theta}^*_m; D_a\right) \\ &= \left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right)^{\top} \boldsymbol{g}^{\ell}_a + \frac{1}{2}\left(\boldsymbol{\theta}^*_m - \boldsymbol{\theta}^*_m\right)^{\top} \boldsymbol{H}^{\ell}_a \left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right) + \mathcal{O}\left(\max_{m' \in \mathcal{M}} \left\|\overset{\star}{\boldsymbol{\theta}}_m - \overset{\star}{\boldsymbol{\theta}}_{m'}\right\|^3\right) \quad (7) \end{aligned}$$

The above, follows from the loss $\ell(\cdot)$ being at least twice differentiable, by assumption.

By Cauchy-Schwarz inequality, it follows that:

$$\left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right)^{\top} \boldsymbol{g}^{\ell}_a \leq \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\| \times \left\|\boldsymbol{g}^{\ell}_a\right\|. \quad (8)$$

In addition, due to the property of Rayleigh quotient we have:

$$\frac{1}{2}\left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right)^{\top} \boldsymbol{H}^{\ell}_a \left(\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right) \leq \frac{1}{2}\lambda\left(\boldsymbol{H}^{\ell}_a\right) \times \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\|^2. \quad (9)$$

Combine Equation 7, Equation 8 and Equation 9 together we obtain the following upper bound:

$$J(\boldsymbol{\theta}^*_{m'}; D_a) - J(\boldsymbol{\theta}^*_m; D_a) \leq \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\| \times \left\|\boldsymbol{g}^{\ell}_a\right\| + \frac{1}{2}\lambda\left(\boldsymbol{H}^{\ell}_a\right) \times \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\|^2.$$

By the definition of hardware sensitivity it follows that:

$$\Delta(a,m) = \max_{m' \in \mathcal{M}} \left|J(\boldsymbol{\theta}^*_{m'}; D_a) - J(\boldsymbol{\theta}^*_m; D_a)\right| \leq \max_{m' \in \mathcal{M}} \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\| \times \left\|\boldsymbol{g}^{\ell}_a\right\| + \frac{1}{2}\lambda\left(\boldsymbol{H}^{\ell}_a\right) \times \max_{m' \in \mathcal{M}} \left\|\boldsymbol{\theta}^*_{m'} - \boldsymbol{\theta}^*_m\right\|^2.$$

□

Theorem A.2. *Consider a particular hardware $m \in \mathcal{M}$, suppose for any group $a, a' \in \mathcal{A}$ the angle between two gradient vectors $\overrightarrow{\boldsymbol{g}^{\ell}_{a,m}}; \overrightarrow{\boldsymbol{g}^{\ell}_{a',m}}$ is smaller than $\frac{\pi}{2}$. Then if we $\underline{a} = \min_{a \in \mathcal{A}} |D_a|$, i.e., the group with least number of training samples then the following holds: $\left\|\boldsymbol{g}^{\ell}_{\underline{a},m}\right\| = \max_{a \in \mathcal{A}} \left\|\boldsymbol{g}^{\ell}_{a,m}\right\|$*

Proof. For notational convenience, denote $\boldsymbol{g}^{\ell}_m$ to be the gradient at convergence point over the whole dataset D. By the assumption, the gradient descent converges it follows that:

[1] With a slight abuse of notation, the results refer to $\bar{\boldsymbol{\theta}}$ as the homonymous vector which is extended with $k - \bar{k}$ zeros.

$$\boldsymbol{g}_m^\ell = \sum_{a \in \mathcal{A}} \frac{|D_a|}{|D|} \boldsymbol{g}_{a,m}^\ell = \mathbf{0}^T. \tag{10}$$

Consider the most minority group $\underline{a}$ (i.e, $|D_{\underline{a}}| = \arg\min_{a \in \mathcal{A}} |D_a|$), it follows from the above equation that:

$$\boldsymbol{g}_{\underline{a},m}^\ell = -\sum_{a \neq \underline{a}} \frac{|D_a|}{|D_{\underline{a}}|} \boldsymbol{g}_{a,m}^\ell$$

Taking the squared norm of vector on both sides of the previous equation, we have:

$$\|\boldsymbol{g}_{\underline{a},m}^\ell\|_2^2 = \left| \sum_{a \neq \underline{a}} \frac{|D_a|}{|D_{\underline{a}}|} \boldsymbol{g}_{a,m}^\ell \right|_2^2 = \sum_{a \neq \underline{a}} \|\boldsymbol{g}_{a,m}^\ell\|_2^2 + 2 \sum_{a \neq a' \neq \underline{a}} (\boldsymbol{g}_{a,m}^\ell)^T \boldsymbol{g}_{a',m}^\ell \tag{11}$$

By the assumption that the angle between two gradient vectors of two arbitrary groups is less than $\frac{\pi}{2}$ hence $(\boldsymbol{g}_{a,m}^\ell)^T \boldsymbol{g}_{a',m}^\ell \geq 0$. Thus it follows that:

$$\|\boldsymbol{g}_{\underline{a},m}^\ell\|_2^2 \geq \sum_{a \neq \underline{a}} \|\boldsymbol{g}_{a,m}^\ell\|_2^2 \geq \max_a \|\boldsymbol{g}_{a,m}^\ell\|^2 \tag{12}$$

Hence the smallest minority group will present the largest gradient norm.

□

Theorem A.3. *Let $f_{\boldsymbol{\theta}_m^*}$ be a binary classifier trained using a binary cross entropy loss on one reference hardware m. For any group $a \in \mathcal{A}$, the maximum eigenvalue of the group Hessian $\lambda(\boldsymbol{H}_a^\ell)$ can be upper bounded by:*

$$\lambda(\boldsymbol{H}_a^\ell) \leq \frac{1}{|D_a|} \sum_{(\boldsymbol{x},y) \in D_a} \underbrace{\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)\left(1 - f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)}_{\textit{Closeness to decision boundary}} \times \left\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right\|^2 + \underbrace{\left|f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}) - y\right|}_{\textit{Error}} \times \lambda\left(\nabla_{\boldsymbol{\theta}}^2 f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right). \tag{13}$$

Proof. First notice that an upper bound for the Hessian loss computed on a group $a \in \mathcal{A}$ can be derived as:

$$\lambda(\boldsymbol{H}_a^\ell) = \lambda\left(\frac{1}{|D_a|} \sum_{(\boldsymbol{x},y) \in D_a} \boldsymbol{H}_{\boldsymbol{x}}^\ell\right) \leq \frac{1}{|D_a|} \sum_{(\boldsymbol{x},y) \in D_a} \lambda\left(\boldsymbol{H}_{\boldsymbol{x}}^\ell\right) \tag{14}$$

where $\boldsymbol{H}_{\boldsymbol{x}}^\ell$ represents the Hessian loss associated with a sample $\boldsymbol{x} \in D_a$ from group a. The above follows Weily's inequality which states that for any two symmetric matrices A and B, $\lambda(A+B) \leq \lambda(A) + \lambda(B)$.

Next, we will derive an upper bound on the Hessian loss associated to a sample $\boldsymbol{x}$. First, based on the chain rule a closed form expression for the Hessian loss associated to a sample $\boldsymbol{x}$ can be written as follows:

$$\boldsymbol{H}_{\boldsymbol{x}}^\ell = \nabla_f^2 \ell\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}), y\right)\left[\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)^\top\right] + \nabla_f \ell\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}), y\right) \nabla_{\boldsymbol{\theta}}^2 f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}). \tag{15}$$

The next follows from that

$$\nabla_f \ell\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}), y\right) = \left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}) - y\right),$$
$$\nabla_f^2 \ell\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}), y\right) = f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(1 - f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right).$$

Applying the Weily inequality again on the R.H.S. of Equation 15, we obtain:

$$\begin{aligned}\lambda(\boldsymbol{H}_{\boldsymbol{x}}^{\ell}) &\leq f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)\times\left\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right\|^2+\lambda\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})-y\right)\times\nabla^2_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\\ &\leq f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)\times\left\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right\|^2+\left|f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})-y\right|\lambda\left(\nabla^2_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)\end{aligned} \tag{16}$$

The statement of Theorem A.3 is obtained combining Equations 16 with 14. □

Proposition A.4. *Consider a binary classifier $f_{\boldsymbol{\theta}*_m}(\boldsymbol{x})$ trained on one reference hardware m. For a given sample $\boldsymbol{x} \in D$, the term $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})(1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x}))$ is maximized when $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})=0.5$ and minimized when $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\in\{0,1\}$.*

Proof. First, notice that $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\in[0,1]$, as it represents the soft prediction (that returned by the last layer of the network), thus $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\geq f^2_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})$. It follows that:

$$f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)=f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})-f^2_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\geq 0. \tag{17}$$

In the above, it is easy to observe that the equality holds when either $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})=0$ or $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})=1$.

Next, by the Jensen inequality, it follows that:

$$f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\left(1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)\leq\frac{\left(f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})+1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})\right)^2}{4}=\frac{1}{4}. \tag{18}$$

The above holds when $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})=1-f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})$, in other words, when $f_{\boldsymbol{\theta}_m^*}(\boldsymbol{x})=0.5$. Notice that, in the case of binary classifier, this refers to the case when the sample $\boldsymbol{x}$ lies on the decision boundary. □

B. Choice of Hardware

We report experiments across widely adopted GPU types: Tesla T4 (NVIDIA, 2018a), Tesla V100 (NVIDIA, 2017a), Ampere A100 (NVIDIA, 2021) and Ada L4 GPU (NVIDIA, 2023). We choose this hardware because it represents a valuable variety of different design choices at a system level, and is also widely adopted across research and industry. Below we include additional context about how the design of this hardware differs.

Feature/Specification	**Tesla V100** (NVIDIA, 2017a)	**Ampere A100** (NVIDIA, 2021)	**Tesla T4** (NVIDIA, 2018a)	**Ada L4** (NVIDIA, 2023)
Hardware Architecture	Volta	Ampere	Turing	Ada Lovelace
CUDA Cores	5,120	6,912	2,560	7,424
Streaming Multiprocessors	80	108	40	58
Total Threads	163,840	221,184	81,920	–
Tensor Cores	640	432 (improved)	320	232 (4th Gen)
Memory	16GB/32GB HBM2	40GB/80GB HBM2	16GB GDDR6	24GB GDDR6 w/ ECC
Memory Bandwidth	Up to 900 GB/s	Up to 2,000 GB/s	Up to 320 GB/s	300 GB/s
FP32 Performance	15.7 TFLOPS	19.5 TFLOPS	8.1 TFLOPS	30.3 TFLOPS
Interconnect	NVLink 2.0, 300 GB/s	NVLink 3.0, 600 GB/s	PCIe Gen 3 (32GB/s), No NVLink	–
TDP	300W	400W (variant-dependent)	70W	72 Watts

Table 1: Comparison of the feature design and system specifications of the hardware evaluated across all experiements.

NVIDIA V100 GPU The NVIDIA V100 GPU, built upon the Volta microarchitecture (NVIDIA, 2017b), introduced Tensor Cores as a notable innovation. Tensor Cores are specialized units that perform Fused-Multiply Add (FMA) operations, enabling the multiplication of two FP16 4x4 matrices with the addition of a third FP16 or FP32 matrix.

NVIDIA Tesla T4 GPU The T4 is based upon the Turing Microarchitecture (NVIDIA, 2018b), presented second-generation Tensor Cores capable of conducting FMA operations on INT8 and INT4 matrices. Despite both the T4 and the V100 sharing the same CUDA Core version and an equal number of CUDA cores per Streaming Multiprocessor (SM), the Tesla T4 GPU is specifically designed for inference workloads. Consequently, it incorporates only half the number of CUDA cores, SM Units, and Tensor Cores compared to the V100 GPU. Additionally, it utilizes slower GDDR6 memory, resulting in reduced memory bandwidth, but it is much more efficient in power consumption terms making it desirable for inference workloads.

The generous memory of the V100 GPU relative to the T4 leads to increased speed of processing of the V100 GPU. This is because tensor cores are relatively fast, and typically the delay in processing is attributable to waiting for inputs from memory to arrive. With smaller memory, this means more retrieval trips.

NVIDIA A100 GPU The A100 GPU is based on the Ampere microarchitecture (NVIDIA, 2021) and presents significant improvements relative to the V100 and T4. The A100 features faster up to 80 GB HBM2e memory, compared to the V100's upper limit of 20 GB HBM2 memory. The A100 provides more memory capacity and higher memory bandwidth, which allows for handling larger datasets and more complex models.

Although the number of Tensor cores per group was reduced from 8 to 4, compared to the Turing V100 GPU, these Third-generation Tensor Cores exhibit twice the speed of their predecessors and support newer data types, including FP64, TF32, and BF16. Furthermore, the Ampere architecture increased the number of CUDA cores and SM Units to 6,912 and 108, respectively, resulting in a notable 35% increase in the number of threads, compared to the V100 GPU, that can be processed in parallel.

NVIDIA L4 GPU The Ada Lovelace Microarchitecture (NVIDIA, 2023), was designed on TSMC's 5nm Process Node, leading to an increased Performance Per Watt Metric. This allowed Nvidia to pack in more CUDA Cores within a single Streaming Multiprocessor (SM), leading to an increased FLOPS Throughput. The L4 GPU, of the Ada Lovelace Microarchitecture, is an inference-friendly GPU, Leading to TDP being fixed at 72 Watts, allowing High Energy Efficiency. To reduce cost, Nvidia opted for the GDDR6 Memory instead of the High Performance HBM, found in their flagship GPUs. The Fourth-generation Tensor Cores support new Datatypes like FP8 (With Sparsity), allowing a much higher throughput. Also, the number of Tensor Cores per SM has been increased. L4 is a substantial improvement from the previous inference-friendly GPU T4.

B.1. Datasets

CIFAR10 and CIFAR100 (Krizhevsky, 2009) are datasets which contain colored natural images of size 32 x 32. In CIFAR10, there are 10 classes of objects with a total of 60000 images (50000 train - 5000 per class, 10000 test – 1000 per class).

Imbalanced versions For our experiments we benchmark two versions of the CIFAR10 dataset, a *Balanced* version which is the original dataset described above and an *Imbalanced* version. The 'Imbalanced' version is a modified version of the original where the class 8 (Ship - CIFAR10) has been reduced to 20% of their original size. The other classes are not modified.

UTKFace The UTKFace (Zhang et al., 2017) is a large scale dataset of face images. This dataset has 20,000 images with annotations for age, gender and ethnicity and images taken in a variety of conditions and image resolutions. It is naturally imbalanced with respect to ethnicity, which provides a challenging and informative setting for our experiments. In this paper, we investigated classification using the ethnicity annotation. The task we perform is image classification – there are 5 class labels: Asian, Indian, White, Black and Others. This is a useful task as it allows us to investigate the disparate effect of tooling on a task where the dataset is naturally imbalanced and highlights a sensitive use case involving protected attributes.

CelebA The CelebA (Liu et al., 2015) is an image dataset that consists of around 202,599 face images with 40 associated attribute annotations. For this task, we aim to classify face images into 4 distinct classes: 'Blond Male', 'Blond Female', 'Non-Blond Male' and 'Non-Blond Female.' This is also a naturally imbalanced task, with 'Blond Male' being the minority class. Here, gender is a protected attribute and our goal is to understand how hardware amplifies the bias.

B.2. Architectures

SmallCNN - We use a custom Convolutional Neural Network with 5 convolutional layers, 3 linear layers and one Max-Pooling layer with stride = 2. Using SmallCNN as the base architecture enabled us to explore an extensive ablation grid while making effective use of computational resources.

ResNet18, ResNet34 and ResNet50 (He et al., 2015) - includes residual blocks and has become an architecture of choice for developing computer vision applications. We evaluate two variants, namely ResNet18 and ResNet34 and ResNet50 with 18, 34 and 50 layers respectively. The versions used in this code were the default implementations available in the torchvision library maintainers & contributors (2016).

Controlling model stochasticity. Stochasticity is typically introduced into deep neural network optimization by factors including algorithmic choices, hardware and software (Zhuang et al., 2022). Our goal is to precisely measure the impact of tooling on the fairness and performance of the model. Hence, we seek to control stochasticity introduced by algorithmic factors, to disambiguate the impact of noise introduced by hardware.

The stochasticity arising from algorithmic factors was controlled as follows: the experimental setup maintained a fixed random seed across all Python libraries, including PyTorch (Paszke et al., 2019) 2.0, ensuring consistency. We ensure that the data loading order and augmentation properties were controlled using a fixed seed through FFCV-SSL (Bordes et al., 2023), a fork of FFCV (Leclerc et al., 2022).

A critical part of our analysis requires that there is a fair comparison between different hardware platforms used for both training and inference. To ensure consistent experimental configuration across hardware platforms, we fix the parameters related to the training harness for a given dataset and model. It includes but is not limited to batch size, learning rate, initialization, and optimizer. The models trained on UTKFace and CIFAR-10 for both settings were in full-precision (FP32) for both training and inference. Models trained on the CelebA dataset, we employed mixed-precision training due to memory and time constraints. For these experiments, we use float16 as the intermediate data type. The inference, however, takes place in full precision (FP32). Reported metrics were averaged across runs gathered from approximately five random seeds.

While the theoretical analysis focuses on the notion of disparate impacts under the lens of hardware sensitivity with respect to the risk functions, the empirical results which we report are differences in the accuracy of the resulting models across different hardware. This way the empirical results thus reflect the setting commonly adopted when measuring accuracy parity Zhao & Gordon (2019) across groups. In addition, we also report metrics on gradient norm, Hessian's max eigenvalue, and the average distance from the decision boundary for various groups in the datasets which highlights optimization differences amplified by tooling, which could lead to an increase in hardware sensitivity and shown in the paper, unfairness.

C. Model Convergence

We report training loss curves and test accuracy for each dataset adopted. The results are shown in Figure 10. Notice that, in our experiments, in the paper, we identified a stable training configuration for each dataset, where further training led to overfitting. For comparison, here, we report results on a ResNet34 model trained for 100 epochs. The learning rate scheduler, `OneCycleLR`, is dependent on the total number of epochs specified initially, resulting in faster performance improvement with fewer epochs, which is the reason for the different slopes in the curves. Notice, that beyond the purple line that we call the "significant epoch" there is diminishing returns in terms of improvement in accuracy, and for datasets like CelebA and UTKFace, over-fitting is observed.

Recall that our analysis focused on hardware sensitivity to measure unfairness. As shown in Figure 11, our analysis remains unaltered, regardless of the number of training epoch adopted. While for larger number of epochs the sensitivity values decrease slightly in some datasets, the overall trends and the phenomenon is analogous to what we observed in the paper, even with improved test accuracy. This supports our argument presented in the rebuttal.

As a secondary point, it is also important to note also that our training was limited by processing costs and capabilities of the slowest GPU, making computational efficiency a crucial factor due to the extensive nature of our experiments.

D. Fairness/Accuracy Tradeoffs

We report additional results on the fairness-accuracy trade-offs resulting from the proposed mitigation mechanism. Recall that our mitigation mechanism augments the loss function with a component quantifying the disparity between the group-specific and batch-wide distances to the decision boundary:

$$\overset{\star}{\boldsymbol{\theta}}_F = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; D) + \lambda \sum_{a \in \mathcal{A}} (\delta_{\mathcal{D}_a} - \delta_{\mathcal{D}})^2, \tag{19}$$

where δ_S represents the average distance to the decision boundary of samples $\boldsymbol{x} \in S$, and λ is a hyper-parameter which calibrates the level of penalization.

In the following experiment, we report results on the largest accuracy difference across hardware at the varying of the hyper-parameter λ chosen within values $\{0.01, 0.001, 0.0001, 1e-5, 2e-5, 4e-5, 6e-5, 8e-5, 1e-6\}$ for CIFAR10

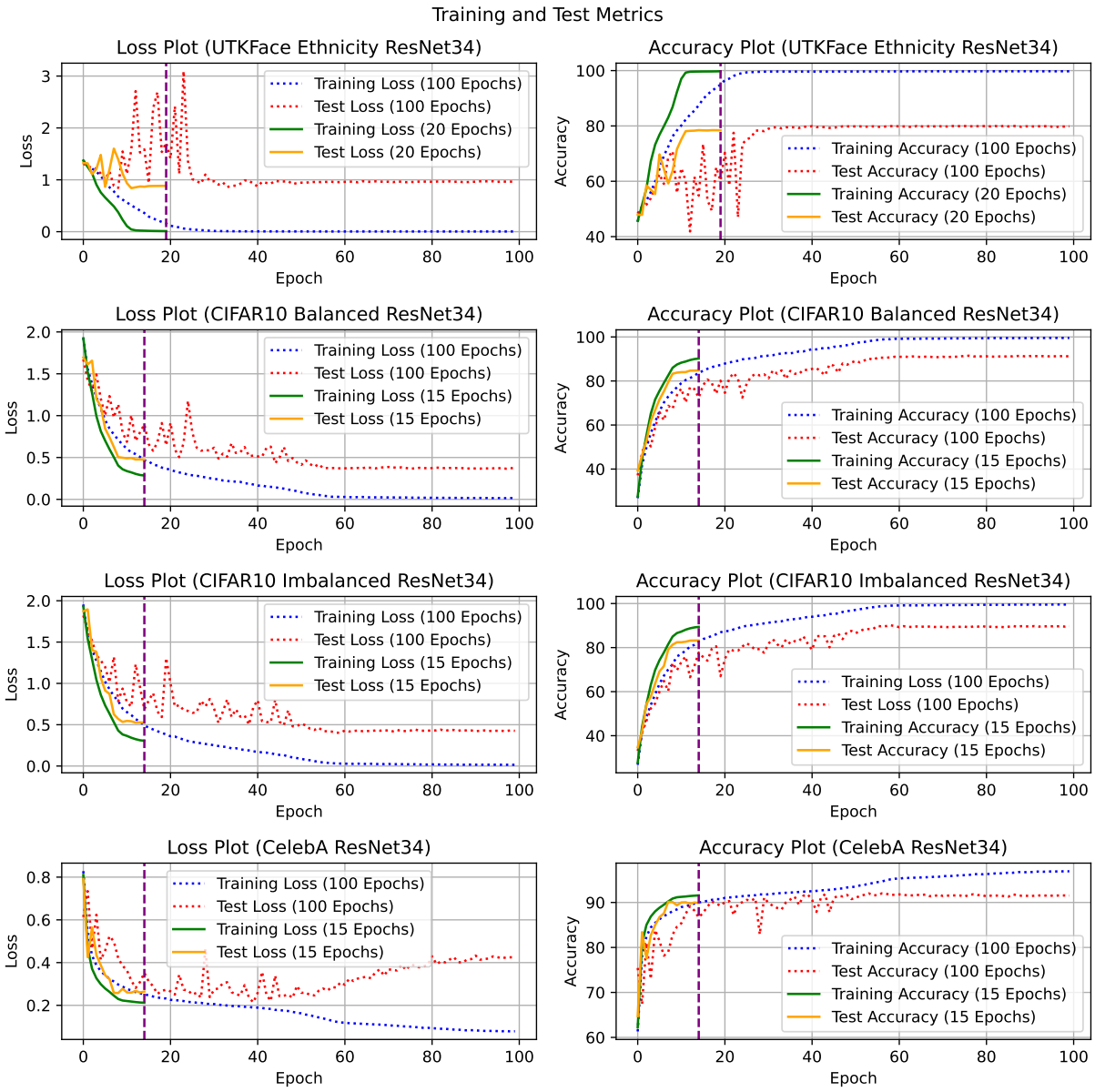


Figure 10: Training and test loss (left) and training and test accuracy (right) for models trained with 100, 20, or 15 epochs. First row: UTKFace; Second row: CIFAR10 balanced; Third row: CIFAR10 unbalanced; Fourth row: CelebA; In each plot, the purple dotted vertical line indicates the training epoch used in the main paper for that specific dataset. Beyond this epoch, continued training leads to no significant impact to hardware sensitivity.

and UTKFace.

The results are illustrated in Figure 12 for CIFAR-10 and UTKFace, where we only report a few parameters representative of the major change among the various class accuracy difference across various hardware.

We observed that setting the parameter $\lambda = 4e - 5$ results in a significant reduction in unfairness when compared to the unmitigated solution. Unfairness here is measured by the maximal accuracy difference across all hardware configurations (which is the **fairness violation** referred to as **F.V.** in the figure and displayed as the last column) detailed in equation 3 of the main paper. However, a smaller value of $\lambda = 2e - 5$ leads to an even greater reduction in unfairness (F.V.), despite increasing the accuracy difference (hardware sensitivity) for the "Asian" group.

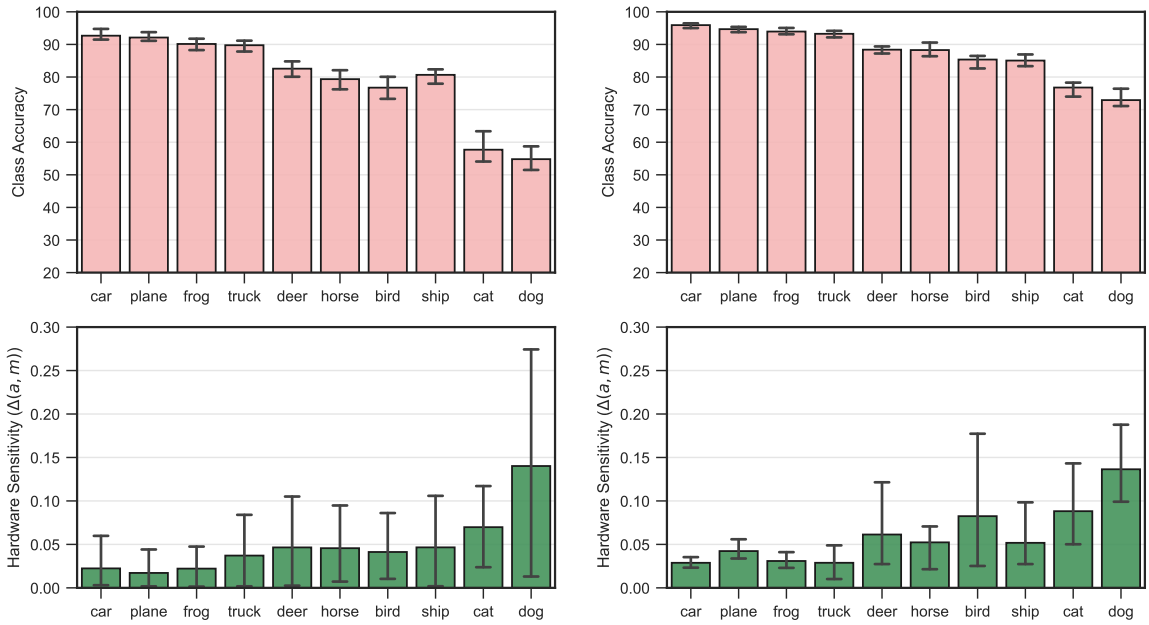


Figure 11: Class accuracy (top) and Hardware sensitivity (bottom) — our fairness metric — for CIFAR-10 (unbalanced). The left plot report models trained with 15 epochs and the right plots models trained over 100 epochs. Notice that, while some accuracy difference is observable for cat and dog classes, the hardware sensitivity trends are not affected.

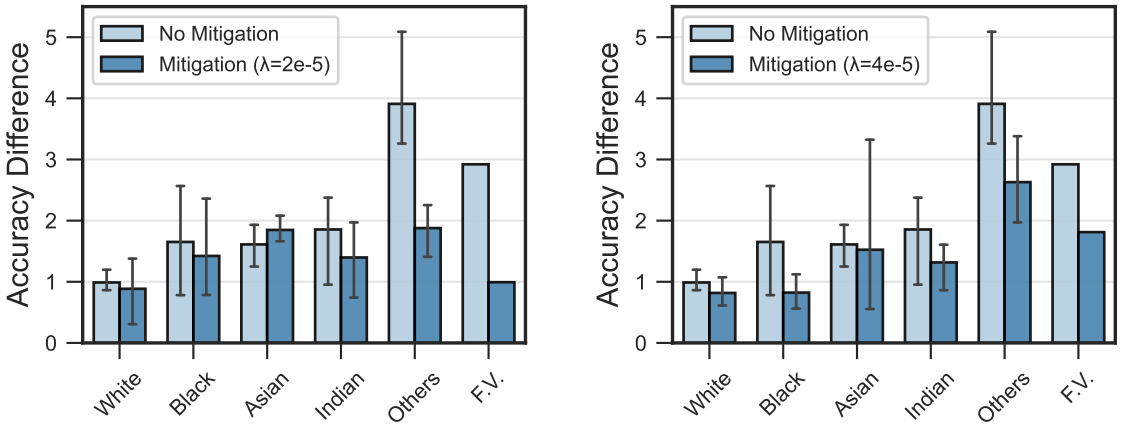
Similar effects are observed for the CIFAR dataset. Using $\lambda = 6e - 5$ significantly reduces fairness violation (F.V.), decreasing it from 1.91 (unmitigated solution) to 0.77 (post-mitigation). However, this setting tends to increase hardware sensitivity for certain classes (e.g., bird, cat, and horse). On the other hand, a smaller value of $\lambda = 1e - 6$ results in the most substantial decrease in hardware sensitivity for classes such as deer and dog. Nevertheless, this value does not reduce unfairness (F.V.) as effectively as $\lambda = 6e - 5$.

Therefore, as stated in the paper, while it is possible to optimize the choice of the value λ during the empirical risk process, e.g., using a Lagrangian dual approach as in (Fioretto et al., 2020a;b), we found that even a traditional simple grid search allows us to find good λ values yielding an effective reduction in accuracy disparity.

E. Model Architecture/Size

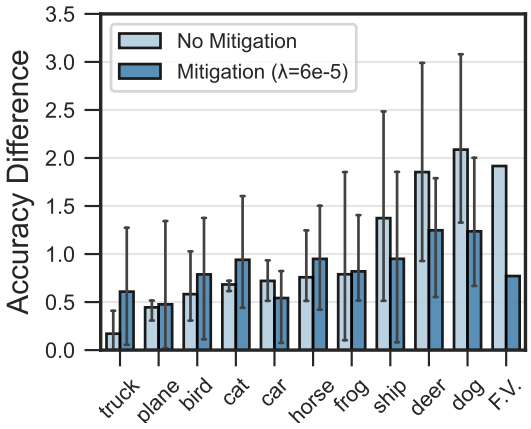
We report a summary of our observations related to model capacity and its relation to hardware sensitivity. Figure 13 reports the class accuracy (first and third rows) and Hardware sensitivity (second and fourth rows) for UTKFace Ethnicity (top rows) and CIFAR10 (Imbalanced) (bottom rows). **Unfairness** is reported in the last histogram of the green plots (second and fourth rows), and indicates the maximal pairwise class difference in accuracy across hardware, consistently with the notion adopted in the paper and above (named F.V) The three columns display results obtained using three different architectures of increasing complexity: A small CNN (527,754 parameters) , a ResNet18 (11.4M parameters) and a ResNet34 (21.7M parameters).

Firstly, notice that unfairness due to hardware tooling is widespread across all datasets and architectures tested. Next, notice that the smallest architecture (left) exhibits high hardware sensitivity for certain classes, such as ship, dog, and cat in CIFAR10 and others in UTKFace. This is likely due to under-fitting. The ResNet18 (middle) has a moderate impact on reducing hardware sensitivity. However, transitioning to an ever deeper model, ResNet34 (right) results in significant increase in fairness violations — correspondingly hardware sensitivity increases sharply for minority classes. This exacerbates the overall fairness violation of the model. Specifically, the fairness violation increases by approximately 68% for ResNet34 (0.27) compared to ResNet18 (0.16) and SmallCNN (0.17) in CIFAR10, and nearly doubles for ResNet34 (0.294) versus ResNet18 (0.14) and SmallCNN (0.12) in UTKFace Ethnicity.

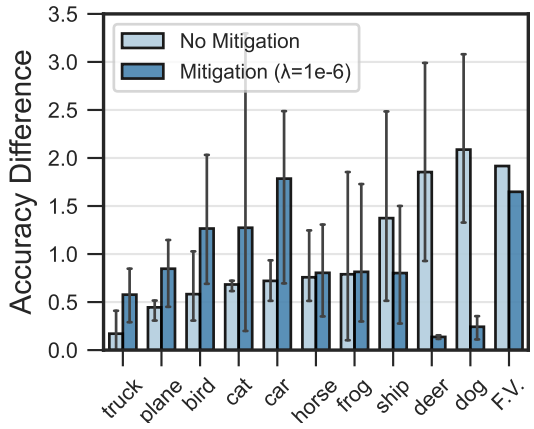


(a) UTKFace: $\lambda = 2e-5$

(b) UTKFace: $\lambda = 4e-5$



(c) CIFAR-10: $\lambda = 6e-5$



(d) CIFAR-10: $\lambda = 1e-6$

Figure 12: **UTKFace and CIFAR-10:** Mitigation solution using different values of λ. The y-axis describes the accuracy difference across all hardware (analogous to hardware sensitivity) for each class (x-axis). The last column of the x-axis (named, F.V) reports the fairness violation across all groups, which measures unfairness.

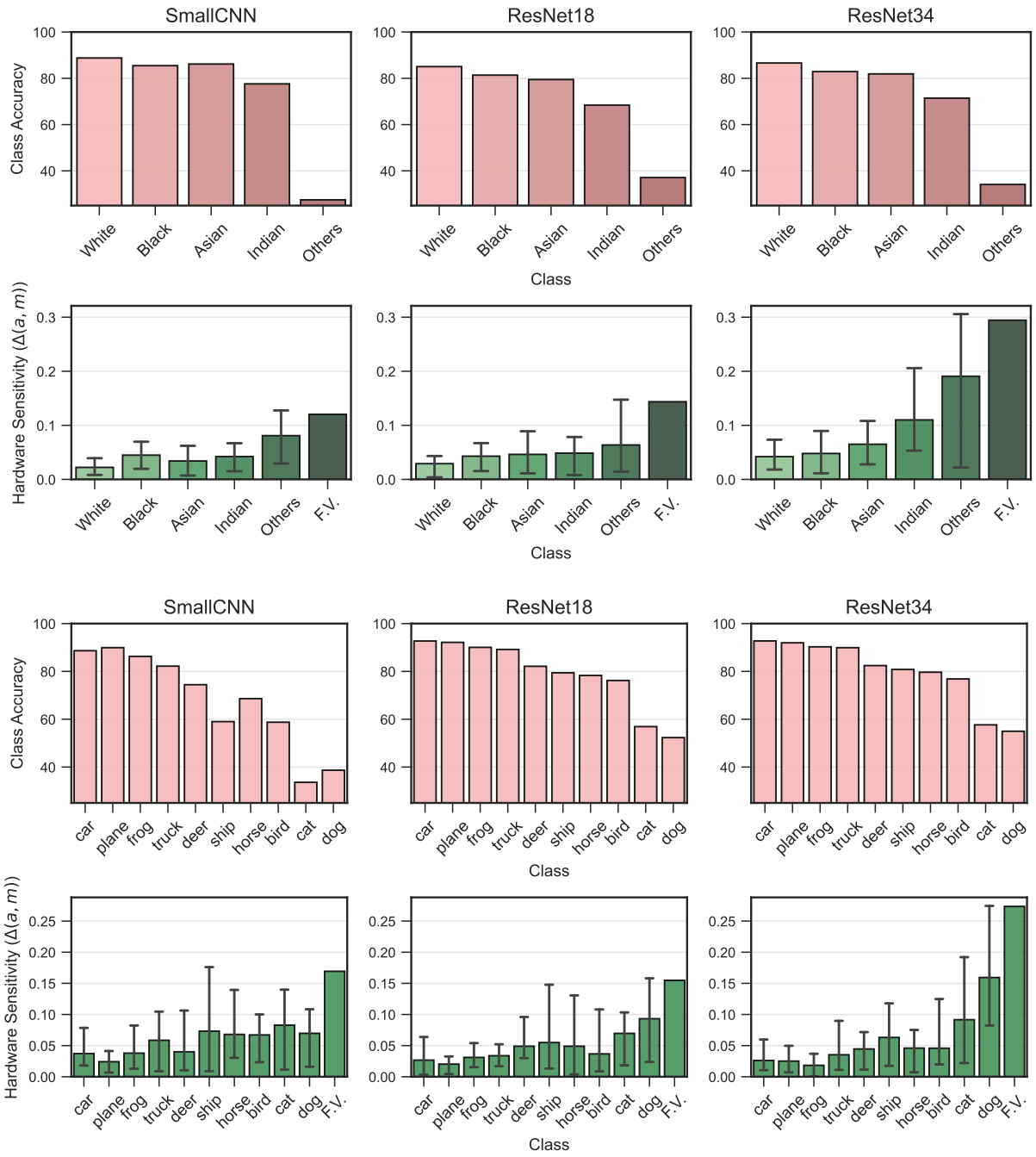


Figure 13: **Left to right:** Hardware Sensitivity Across Architectures. Notice as we increase the model size, the Fairness impact due to hardware choice increases. **Top:** Class-wise accuracy and Hardware Sensitivity for UTKFace Ethnicity. **Bottom:** For CIFAR-10.

F. Additional plots for CelebA

In this section, we report figures related to our experiments on the CelebA dataset. We find the trends to be similar to the other datasets, and support our observations further. Below we provide figures related to hardware sensitivity, distance to decision boundary and mitigation experiments on CelebA.

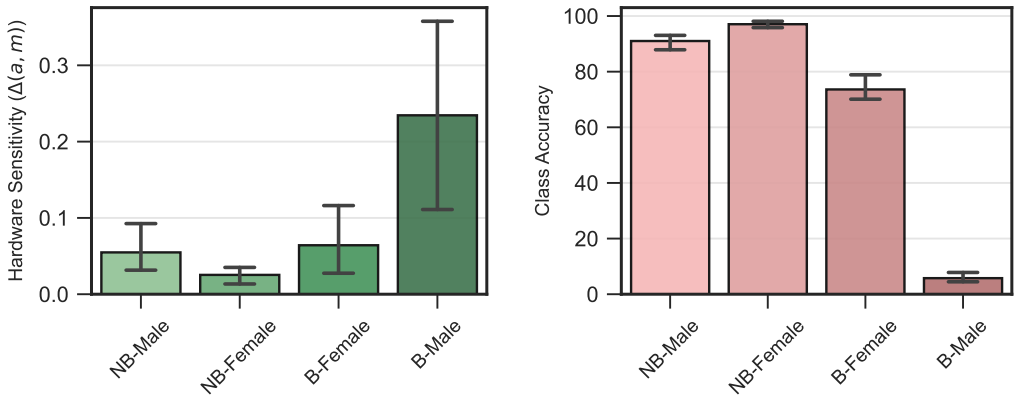


Figure 14: **Left:** Hardware Sensitivity for CelebA (ResNet34). **Right:** Class-wise accuracy

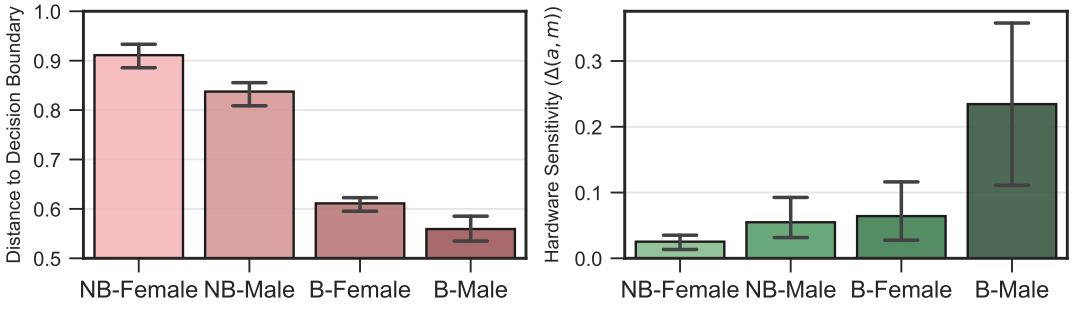


Figure 15: **Left:** Distance-to-decision Boundary for CelebA on ResNet34 **Right:** Hardware Sensitivity

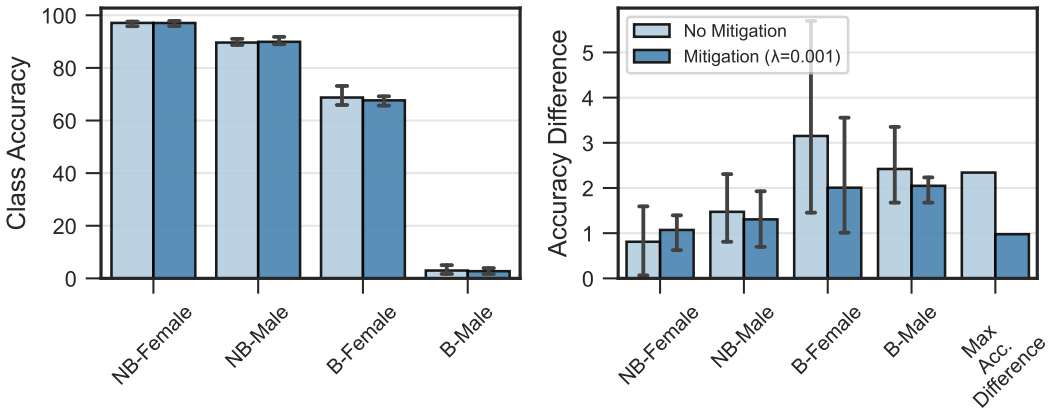


Figure 16: **Left:** Class-wise Accuracy for groups pre and post mitigation. **Right :** Accuracy Difference for groups pre and post mitigation. Notice the Maximum Accuracy Difference between the maximum and minimum accuracy within groups is generally reduced post-mitigation averaged across hardware. CelebA on ResNet50.