DEBIASING SURGEON: FANTASTIC WEIGHTS AND HOW TO FIND THEM

Rémi Nahon **Ivan Luiz de Moura Matos** **Van-Tam Nguyen** **Enzo Tartaglione**
LTCI, Télécom Paris, Institut Polytechnique de Paris, France
{name.surname}@telecom-paris.fr

ABSTRACT

Nowadays an ever-growing concerning phenomenon, the emergence of algorithmic biases that can lead to unfair models, emerges. Several debiasing approaches have been proposed in the realm of deep learning, employing more or less sophisticated approaches to discourage these models from massively employing these biases. However, a question emerges: is this extra complexity really necessary? Is a vanilla-trained model already embodying some "unbiased sub-networks" that can be used in isolation and propose a solution without relying on the algorithmic biases? In this work, we show that such a sub-network typically exists, and can be extracted from a vanilla-trained model without requiring additional training. We further validate that such specific architecture is incapable of learning a specific bias, suggesting that there are possible architectural countermeasures to the problem of biases in deep neural networks.

***K*eywords** Debiasing · Pruning · Freezed model · Deep learning

1 Introduction

In the last decade, recent technical and technological advances enabled the large-scale deployability of deep learning (DL)-based approaches, impacting, among others, the computer vision community. The possibility of training systems in an end-to-end fashion enables access to non-trivial solutions to complex tasks, ushering in unprecedented breakthroughs and fundamentally reshaping the landscape of visual perception. The transformative impact of deep learning finds nowadays broad applicability in diverse real-world scenarios, including autonomous driving, medical imaging [42], augmented reality [38], and robotics [12]. As the computer vision community continues to harness the power of deep learning with scaling models and methods, combining for example language and vision models [62, 20], the boundaries of what is achievable in visual understanding are continually pushed, promising exciting avenues for innovation and discovery.

Unfortunately, from big power comes big responsibility: a big aspect to account for comes from the need to avoid the unintended over-reliance on spurious correlations or biases, naturally present in datasets [32]. This poses a practical significant challenge in DL real-world deployment. As an explicative case, in the context of image classification tasks such as detecting pedestrians, if environmental cues (e.g., the presence of a sidewalk/pedestrian crossing) become spuriously correlated with the target classes, neural networks may exploit such correlations as *shortcuts* for classification [23], thereby leading to performance degradation when presented with images containing different backgrounds (e.g., pedestrian crossing the road not on pedestrian crossing lanes). This leads to some potential threats in certain applications.

In 2021, the European Commission put forward the Artificial Intelligence Act (AI Act) [57], aiming to categorize and oversee artificial intelligence implementations according to their potential to cause harm [58, 31]. Similarly to the General Data Protection Regulation (GDPR) [50], the AI Act has the potential to establish a global benchmark [57]. Regardless, the EU's AI regulation is already garnering attention worldwide: for example, in 2021 Brazil's Congress approved a bill to establish a legal framework for artificial intelligence, pending approval by the country's Senate. Guaranteeing the avoidance of spurious biases that might undermine the safety, trust, and accountability of DL models is then not only becoming a matter of safety but will soon be a legal constraint.

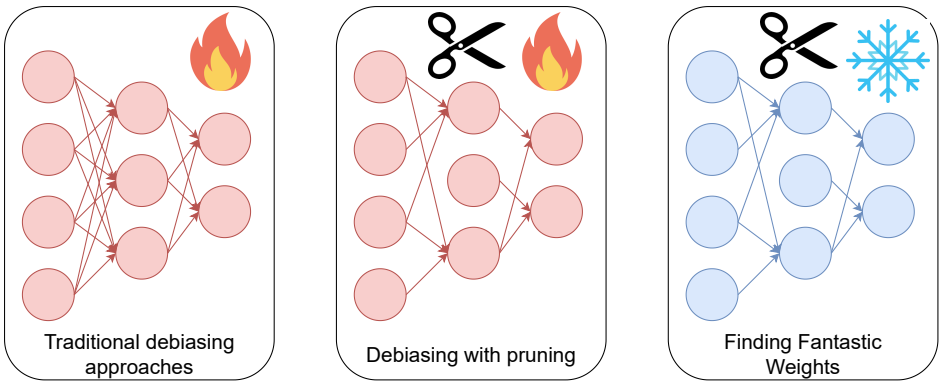


Figure 1: Despite other debiasing approaches implying training or fine-tuning the whole model, with Finding Fantastic Weights (FFW) we maintain the model's parameters frozen and remove the sub-network responsible for bias information propagation.

Multiple solutions are proposed in the last luster, all relying on relatively heavy re-training approaches. More specifically, in the realm of DL debiasing, we can identify three main lines of research: (i) supervised, where the labels of the bias are provided; (ii) bias-tailored, where proxy models can capture specific biases; (iii) unsupervised, where biases are guessed directed within the vanilla-trained model. All three approaches typically require heavy training procedures for either properly tuning the hyper-parameters of the models, or training multiple times the neural networks. This comes at another very relevant environmental cost, and besides it is not granted that such solutions can always apply to any architecture/task. Some recent studies hinder the possibility of reusing pre-trained models and removing bias sources with some adjustments [65, 66], but none of them provides a final solution yet.

In this work we show the existence of unbiased sub-networks in vanilla-trained models, providing also some analysis on the final performance. In a nutshell, we learn how to mask trained weights such that the information about the bias becomes non-extractable by the layer(s) entitled to solve the target task. Therefore, we **F**ind the **F**antastic **W**eights (FFW) constituting an unbiased sub-network that solves the tasks, without further fine-tuning (Fig. 1). Our results, besides providing a clear indication that such a sub-network exists in vanilla-trained overfitting models, contribute to the DL debiasing community, showing that these solutions are in principle achievable without relying on over-complex training schemes.

At a glance, this work proposes the following contributions:

- We draft a theory that shows how the performance on some given task might depend on biased features, suggesting that removing the bias source *not always* results in an enhanced performance (Sec. 3.1);
- To our knowledge, FFW is the first parameter selection strategy, from a vanilla-trained model and without the model's retraining, that provides some guarantees on the bias-related extractable information from the task classifier (Sec. 3.3);
- We show that the non-spurious correlations are learned even in vanilla-trained models, in later training stages (Sec. 4.5), suggesting that over-complicated learning setups are not needed for debiasing;
- We test on very common setups for the community, showing the existence of these sub-networks (Sec. 4.4). This opens the road to the development of the design of more energy-efficient debiasing strategies.

2 Related works

Spurious correlations. The risk of the insurgence of spurious correlations employed for a given target task is a broadly known and acknowledged issue in the DL community. This phenomenon is known as *shortcut learning* [23], where the DL algorithm relies on the simplest correlations to fit the training set, but do not generalize well. Some of these can range from the texture of images [24] to biases inherent in language [26], or even sensitive variables like ethnicity or gender [47, 21]. Such behavior raises practical concerns, particularly in applications where the reliability of deep networks is critical, such as healthcare, finance, and legal services [15].

Debiasing. Recent efforts aimed at training debiased networks resilient to spurious correlations can be broadly classified into two major categories.

Some approaches rely on some information related to the nature of the presence of biases, and we will name these as *supervised* approaches. Among these, we mention *data sanitizing* approaches, working as *pre-processing* approaches, employing for example GANs [13, 34] or style-transfer [24], *post-processing* methods attempting to correct biased behaviors [27, 33] and *in-processing* methods, trying to sanitize the model directly at training time [4, 55, 45].

Besides these, a new class of *unsupervised* approaches is rising [43], where in principle little or even no clue on the nature of the bias is provided. Specifically, we can recognize models pre-capturing texture biases [59, 25], leveraging groups imbalances [39, 16, 3] and assuming that the bias is learned in early training stages [45, 37].

Studying impacts of neural architectures. Recently a lot of attention has been devoted in studying the influence of a deep neural network's architecture on its generalizability. For example, Diffenderfer *et al*. [19] built a pruning algorithm inspired by the lottery ticket hypothesis [22] to craft compact and resilient network architectures. Similarly, Bai *et al*. [7] tackles a similar issue through the lenses of out-of-distribution performance. This class of approaches, although effective at the cost of enhanced computational complexity [54], does not entirely eradicate connections to spurious input attributes. On the other side of the coin, Zhang *et al*. and Zhao *et al*. suggest the efficacy of pruning weights associated with such attributes sided with some fine-tuning [64, 2], finding some counter-arguments in the literature [10] and even with works suggesting the need to employ corrective distillation terms [48]. The discussion around these empirical findings is still open.

In a recent work, it has been empirically showcased the presence of subnetworks within neural networks that exhibit reduced susceptibility to spurious features [64]. By capitalizing on the modular nature inherent in neural architectures [18], Zhang *et al*. were able to demonstrate that, in principle, it is possible to select, already at initialization, networks that will avoid the employment of biased features. However, this is not a sufficient condition to still guarantee the presence of such sub-networks in an already-trained model. In the era of foundation models, indeed, training a model from scratch can be very costly and impractical [63]. In this work we focus on this problem, characterizing the condition of extractability for such debiased sub-networks, drawing as well a link between bias removal and impact on the performance of the target task. We will sketch the theory motivating our approach, and we will study the problem of extracting these subnetworks, trained with vanilla strategies, *without requiring finetuning*.

3 Method

3.1 Removing the bias impacts the performance

In this section, we will define the problem of learning a mapping between the input $\mathbf{x}$ and its target output $\hat{y}$ on a given dataset $\mathcal{D}$ (that is traditionally slit in train, validation, and test), under the assumption that there is an underlying bias $\hat{b}$ (namely, "spurious" correlations) associated to it. We can model this problem working with the random variables associated with these quantities: we define $\hat{Y}$ as the random variable associated with the target output, Y as the random variable associated with the output of the classifier $\mathcal{C}$, $\hat{B}$ as the random variable associated to the bias, and B the random variable associated to the extractable information regarding the bias, from $\mathcal{E}$. We can easily write the joint probability between $\hat{Y}$ and Y:

$$p\left(Y, \hat{Y}\right) = \frac{1}{N_C}\left[\delta_{y\hat{y}}(1-\varepsilon_{y\hat{y}}) + \bar{\delta}_{y\hat{y}}\frac{\varepsilon_{y\hat{y}}}{N_C-1}\right] \tag{1}$$

where $\varepsilon_{y\hat{y}}$ indicates the classification error on N_C classes, δ_{ab} is the Kronecker delta, and $\bar{\delta}_{ab} = 1-\delta_{ab}$. In debiasing problems, we can also write the joint probability between $\hat{Y}$ and $\hat{B}$ as

$$p\left(\hat{Y}, \hat{B}\right) = \frac{1}{N_C}\left[\delta_{\hat{y}\hat{b}}\rho + \bar{\delta}_{\hat{y}\hat{b}}\frac{1-\rho}{N_B-1}\right] \tag{2}$$

where N_B is the number of biases and ρ is the correlation between bias and target. For the sake of tractability, the bias is in (2) presented to be uniformly distributed across the classes, but in general this is not true (it is sufficient to add an index to the specific bias-target class pairing for ρ). In this case, if $\rho = \frac{1}{N_B}$, we know the dataset $\mathcal{D}$ is balanced wrt. the bias; otherwise, this could lead to the employment, from $\mathcal{C}$, of spurious features, which can imply generalization and fairness issues. Indeed, we can write the mutual information between the target class and the bias as

$$\mathcal{I}(\hat{B}, \hat{Y}) = \frac{N_B}{N_C}\left[\log(N_B) + \rho\log(\rho) + (1-\rho)\log\left(\frac{1-\rho}{N_B-1}\right)\right]. \tag{3}$$

The full derivation can be found in the Supp. Mat. Evidently, there is an implicit link between bias and target classes. Considering that we aim at learning a mapping between input and output, we can introduce the implicit parameter ϕ that encodes the tendency of our model to rely on biased features: we call this parameter *model's biasedness* and it

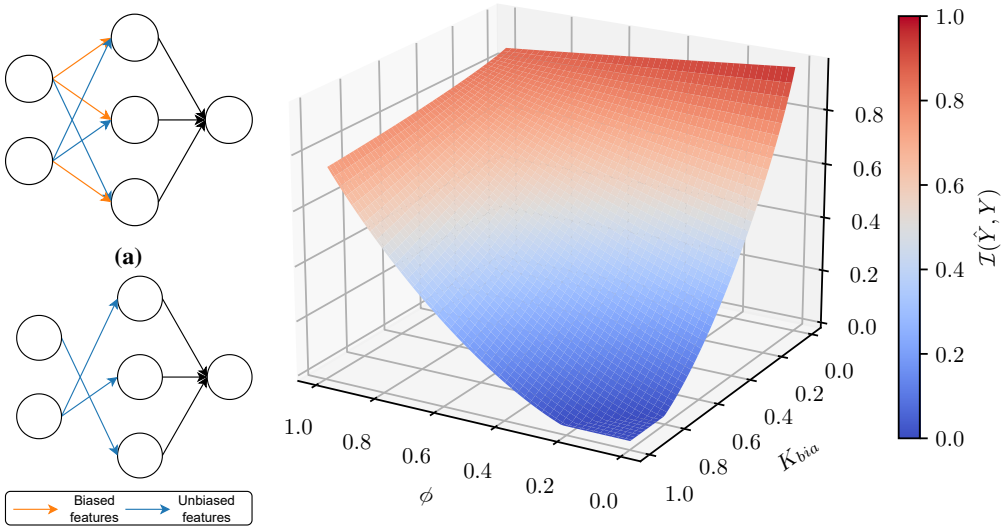


Figure 2: The vanilla model, where the model still employs information related to the bias ($\phi \neq 0$) (a), the model where the information of the bias is entirely removed ($\phi = 0$) (b), and the relationship between model biasedness ϕ and task biasedness K_{bia} (plot obtained with $N = 10$ and $\rho = 0.9$) (c).

contributes as well to the error ε. For tractability, we will now treat the problem assuming one bias source per target class (the same is extendible to having more bias sources per target class, as well as having one same bias per multiple target classes), and for simplicity, we have $N_C = N_B = N$. Besides, we will assume that all the sources of error are due to the presence of the bias and that they are uniform across classes. It is possible to write a joint probability distribution between $\hat{B}, \hat{Y}, Y$, and by marginalizing over $\hat{Y}$, we can obtain

$$\mathcal{I}(\hat{B}, Y) = \frac{\rho + \phi(1-\rho)}{\log(N)} \log\left[\rho + \phi(1-\rho)\right] + \frac{(1-\phi)(1-\rho)}{\log(N)} \log\left[\frac{(1-\phi)(1-\rho)}{N-1}\right] + 1. \tag{4}$$

The full derivation can be found in the Supp. Mat. During debiasing, what we would like to achieve, is to minimize such a quantity, which is achieved for $\phi \to 0$: on the contrary, we can evidently see that the model is completely biased when $\phi \to 1$.

Changing the biasedness also affects the predicted performance, and it is possible that by completely removing the information regarding the bias, the overall performance of the classifier will be harmed. Specifically, let us define the error disentangled from the bias as $\varepsilon_{\text{unb}} = 0$ and the error due to the lack of the bias features as

$$\varepsilon_{\text{bia}} = K_{\text{bia}}(1-\phi), \tag{5}$$

where K_{bia} is the inherent ground-truth dependence between the use of the biased feature and the target label, and is an implicit property of the learned task that we name *task's biasedness*. Specifically, K_{bia} indicates how much the target task depends on the bias to be solved. We can comprehensively write the mutual information between the target label and the model's prediction as

$$\begin{aligned}\mathcal{I}(\hat{Y}, Y) = f\left\{\frac{1}{N}\left[\rho(1-\varepsilon_{\text{bia}}) + (1-\phi)(1-\rho)(1-K_{\text{bia}})\right], N, N^2\right\} + \\ + f\left\{\frac{1}{N}\left[\frac{\phi(1-\rho)}{N-1}K_{\text{bia}} + \left(\frac{\rho^2(N-2) + \rho(1-\rho)(N-2)}{(N-2+\rho)}\right)\varepsilon_{\text{bia}}\right], N(N-1), N^2\right\},\end{aligned} \tag{6}$$

where $f(x, y, z) = xy\log(xz)$. The full derivation for (6) can be found in the Supp. Mat. Fig. 2c displays how the performance of a predictor drops as a function of both K_{bia} and ϕ. As we can observe, there is a clear dependence between the remotion of the bias features ($\phi \to 0$) and the final model's performance, depending on the dataset-related parameter K_{bia}; if K_{bia} is high, then the performance drops as ϕ approaches zero; on the contrary, if K_{bia} is low, the

performance increases as $\phi \to 0$. This tells us that debiasing, intended as fair use of features without biases, does not imply necessarily *removal* of these, but rather *finding a proper balancing* between the employed features. To achieve this, one possibility is, for example, to mask the information related to the bias, at the output of $\mathcal{E}$. Intuitively, if the information related to the bias can be disentangled from the target, this can be effectively achieved - however, if the target task intrinsically requires the employment of the information related to the bias, then its effective removal will harm the target task performance. In the next section, we will try to unveil what is the effect of removing the information related to the bias in a structured way.

3.2 Towards bias information removal

Following on the previous finding, we here analyze a naïve approach that allows in principle to solve the problem of removing biases in a DL model. From [49], we know that we can upper bound the mutual information between the extractable information about the bias at the output of $\mathcal{E}$ according to

$$\mathcal{I}(\hat{B}, B) \leq \log \frac{\exp(s^{\parallel})}{\frac{1}{N_B - 1} \sum_j \exp(s_j^{\perp})}, \tag{7}$$

where $s(\cdot)$ is a similarity measure, typically expressed as *cosine similarity*, in the feature space (in our case, the output of $\mathcal{E}$), between samples having same *bias aligned* ($s^{\parallel}$), and *bias misaligned* samples ($s^{\perp}$). What we aim to achieve through debiasing approaches is in general to minimize (7): we can achieve it by either minimizing the numerator (maintaining the denominator constant) or maximizing the quantity at the denominator (maintaining the quantity at the numerator constant). Some works in the literature attempted to tackle this problem directly, designing proper regularization or loss functions, requiring training of the model [55, 29, 8].

One intriguing possibility, as already suggested in a recent work [63], is to employ *structured pruning* to completely mask all the layer's outputs that are responsible for biased information flow. More formally, let us name *bottleneck layer* the output of the encoder $\mathcal{E}$, producing a vector $\mathbf{z} \in \mathbb{R}^{N_{\text{bott}}}$, where N_{bott} is the output size of the bottleneck layer. Assuming that the information regarding the bias is encoded by a subset of these neurons, it is in principle possible to find a pruning mask $\mathcal{M}$ such that the output $\hat{\mathbf{z}} = \mathcal{E}(\mathbf{x}, \mathbf{w}) \odot \mathcal{M}$ does not encode the information related to the bias. This approach is in principle not always destined to succeed, and on the contrary, without an explicitly constrained training procedure that enforces the disentanglement between bias and target features, it is unlikely to be successful (as in Fig. 2a, it is not possible to mask neurons holding information of the bias and at the same time solve the task). It is indeed possible that the information about the bias is distributed across multiple dimensions of $\mathbf{z}$. Furthermore, given the non-linearity relationship inherently encoded in $\mathcal{E}$ between input and its output, any linear mapping, including approaches like the selection of Principal Components, is potentially unsuitable for the purpose. One more viable and flexible approach would be to perform *unstructured pruning*, sanitizing the DL model from the propagation and the computation of any information regarding the bias (as also visualized in Fig. 2b). There are indeed some weights encoding the information related to the bias, and there are *fantastic weights* that do not process such information. In the next section, we will discuss how to find them.

3.3 Finding the Fantastic Weights

In this section, we will provide the basics regarding how to find a mask $\mathcal{M}$ on a vanilla-trained model that removes the information related to biases. Overall, we will assume that a vanilla-trained DL model, composed of an encoder $\mathcal{E}(\cdot)$ and a task classifier head $\mathcal{C}(\cdot)$, is provided to us. We will not require any constraint regarding the way such a model is trained, and the parameters belonging to those two blocks will not be modified all along the process (apart from allowing their pruning).

Estimating bias leaking information. To estimate the amount of bias information leaking to the classifier $\mathcal{C}(\cdot)$, we will attach at the output of the encoder (the *bottleneck layer*) an auxiliary privacy head $\mathcal{P}(\cdot)$. We will design $\mathcal{P}(\cdot)$ such that it has the same number of parameters as $\mathcal{C}(\cdot)$: this head will be trained to estimate the *upper bound* on the possible information that can be used by the classifier about the bias. To this end, we can train $\mathcal{P}(\cdot)$ using a classification loss, like the categorical cross-entropy loss (CCE). Along the process, we will not allow the error signal to backpropagate through the encoder $\mathcal{E}(\cdot)$.

Metrics to remove the bias. We want to minimize the achievable performance of $\mathcal{P}(\cdot)$. Some works suggest that techniques like gradient inversion [36] are potentially employable in such a context; however, in this case, such an approach is not a good fit. More specifically, maximizing CCE does not really remove information, but simply minimizes the activation of the correct class (as also explained in some works like [43]). What we can on the contrary employ, is to minimize the mutual information between bias labels, making the output converge as much as possible to

Algorithm 1 Finding fantastic weights.

procedure FFW($\mathcal{E}$, $\mathcal{C}$, $\mathcal{D}_{\text{TRAIN}}$, $\mathcal{D}_{\text{VAL}}$, MODE)
  $m_i \leftarrow 1 \forall i$ ▷ Initialize gating weights
  $\tau \leftarrow 1$
  if mode = unstructured **then**
    Attach parameter mask estimator (8) to $\mathcal{E}$
  else
    Attach neuron mask estimator (9) to $\mathcal{E}$
  end if
  Initialize bias extraction head $\mathcal{P}$ and attach to the output of $\mathcal{E}$
  do
    do
      Train $\mathcal{P}$ on $\mathcal{D}_{\text{train}}$
      Optimize gating parameters m_i according to (10) on $\mathcal{D}_{\text{train}}$
    while Plateau in performance on $\mathcal{D}_{\text{val}}$
    $\tau \leftarrow 0.5 \cdot \tau$ ▷ Scale the temperature
  while performance of model with τ is the same as $\tau = 0$
end procedure

a uniform distribution. Such a term will not contribute to the update of $\mathcal{P}(\cdot)$ but it will reach the parameters in $\mathcal{E}(\cdot)$, to be then used to estimate the pruning masks.

Estimating the bias mask. In FFW we propose two possible approaches to find the pruning masks: one acts unstructuredly, while in the other we will behave in a structured manner. In an unstructured sub-network selection setup, every parameter in $\mathcal{E}(\cdot)$ will have a gating parameter $m_i \in \mathbb{R}$ associated with it, initialized to zero. The value of each parameter w_i will be replaced by

$$\hat{w}_i = w_i \cdot \left[\Theta(-m_i) \cdot 2\sigma\left(\frac{m_i}{\tau}\right) + \Theta(m_i)\right], \tag{8}$$

where τ is a temperature, $\sigma(\cdot)$ is the sigmoid function, and $\Theta(x)$ is the one-step function. To maintain the differentiability of our formulation, at the differentiation stage, we employ a straight-through estimator [9] for $\Theta(\cdot)$. As $\tau \to 0^+$, the value of (8) will become either w_i (unpruned) or 0 (pruned). In the structured variant, the whole unit will be pruned, and the j-th neuron's output z_j is masked, according to

$$\hat{z}_j = z_j \cdot \left[\Theta(-m_j) \cdot 2\sigma\left(\frac{m_j}{\tau}\right) + \Theta(m_j)\right]. \tag{9}$$

All the gating parameters are initialized to zero, such that we guarantee that, at initialization, the performance of the pruned model exactly matches the one of the vanilla. Then, the proxy parameters are updated according to two different loss terms:

- a task loss, to be kept as low as possible to maintain performance on the target task;
- an empirical mutual information loss, on the predictions of on the bias estimator head, to be minimized as well.

Given the above, we can formulate an objective function to be minimized:

$$J = \mathcal{L}(y, \hat{y}) + \gamma \mathcal{I}\left(b, \hat{b}\right), \tag{10}$$

where $\mathcal{I}\left(b, \hat{b}\right)$ is the empirical mutual information calculated for the given minibatch. The computation of the update for m_i follows standard back-propagation rules.

3.4 Overview of FFW

FFW is synthesized in Alg. 1. In our approach, we hypothesize that, given a sufficiently trained and parametrized network (namely, not in an under-fitting regime), a set of spurious and target features are learned and blended together. To this end, we target to learn a mask on the encoder $\mathcal{E}$ such that the information on the bias is filtered and not usable by the task classifier $\mathcal{C}$. We therefore initialize the gating parameters m_i to one and replace the parameters in $\mathcal{E}$ by the expression (8), where w_i is non-trainable but the only learnable parameter is m_i. The temperature τ is initialized to one,

Table 1: Results for FFW applied on Biased-MNIST ($\rho = 0.99$) with different γ on the validation set.

γ	2	5	10	20	50	200
Task	98.17 ± 0.27	98.07 ± 0.24	97.79 ± 0.30	97.52 ± 0.10	93.68 ± 3.77	23.31 ± 19.57
Bias	21.49 ± 5.68	15.94 ± 1.99	15.74 ± 3.26	17.10 ± 1.93	16.02 ± 0.58	10.61 ± 1.25
Sparsity	39.72 ± 0.03	41.49 ± 2.75	46.50 ± 0.01	46.90 ± 0.02	51.81 ± 0.02	52.13 ± 0.03

and a bias extraction head $\mathcal{P}$ is initialized and attached to the output of $\mathcal{E}$. Then, the bias extraction head $\mathcal{P}$ and the gating parameters are trained on the training set $\mathcal{D}_{\text{train}}$ until a plateau on a validation set $\mathcal{D}_{\text{val}}$ is reached. At this point, τ is rescaled by a factor 2 and the process is iterated until the performance of the model matches the one of the model obtained with $\tau = 0$, which is our stop criterion. In the next section, we will provide our empirical findings.

4 Experiments

4.1 Experimental setup

For our experiments, we employ architectures typically used for benchmarking bias in the three mainstream datasets Biased MNIST, CelebA, and Corrupted CIFAR10 (that will be detailed in Sec. 4.2). Specifically, for Biased MNIST, we employ the same fully convolutional network used for Rebias [5], composed of four convolutional layers with 7×7 kernels, with batch normalization layers. The training procedure to get the vanilla model is the same as in [53]. For the experiments on CelebA and Corrupted-CIFAR10, we used a pre-trained Resnet-18 on ImageNet, and we employed the same optimization strategy as in [46, 30]. For our FFW, we employ in all our experiments $\gamma = 10$ and we use a fixed learning rate for $\mathcal{P}$ of 0.1, until plateauing on the validation set. Noteworthily, we will employ two splits of the training set: one, biased, used for the vanilla training and indicated as $\mathcal{D}^b$, and another, unbiased and used by FFW to extract unbiased sub-networks, indicated as $\mathcal{D}^u$. In all our tables, we will report three values for our experiments: "Task" indicates the top-1 accuracy on the target task, "Bias" indicates the performance of $\mathcal{P}$ and acts as an upper bound on the information related to the bias information (as top-1 accuracy) usable by the task classifier $\mathcal{C}$, and the "Sparsity" of the extracted sub-network. Our code, provided in the Supp. Mat. will be open-sourced upon acceptance of the article.

4.2 Datasets

Below, we provide an overview of the datasets used for quantitative evaluation, chosen to represent various biases, ranging from the straightforward Biased MNIST to datasets with multiple synthetic and real-world biases. As stated in 4.1, we chose datasets that provide bias labels and will train our gating weights on balanced, unbiased datasets.
Biased MNIST. Our first dataset, introduced by Bahng *et al.* [5], comprises 60k samples where MNIST handwritten digits are colored with a correlation ρ between color and digits. Each digit is assigned a specific color, and the samples receive a background color accordingly. We test four levels of color-digit correlation: ρ = {0.99, 0.995, 0.997, and 0.999}. The bias effect, namely the background color, is assessed by testing on an unbiased dataset with ρ = 0.1. As this dataset can be generated artificially by injecting a background color to black-and-white digit, for the following experiments, we built two 60k datasets for each experiment: the biased one $\mathcal{D}^b$ with a correlation ρ and an unbiased one that constitutes $\mathcal{D}^u$ that we split in three parts of proportions 60%-20%-20% to obtain our $\mathcal{D}^u_{\text{train}}$, $\mathcal{D}^u_{\text{val}}$, and $\mathcal{D}^u_{\text{test}}$.
CelebA. CelebA [41], a real-world dataset commonly used for debiasing evaluations, consists of 203k face images annotated with 40 attributes. Here, we focus on the classification of whether the individual has "Blond Hair" or not, where the primary bias stems from gender, with a strong tendency for "Females" to have the "Blond hair" attribute.
Corrupted CIFAR10. In the same process followed for Biased MNIST, Corrupted CIFAR10 consists in the injection of a specific correlation between an attribute (here a kind of image corruption, such as "motion blur", or "fog") and one of the ten classes of CIFAR10. The corruptions are taken from [28] and the specific corruption protocol followed here was taken from [8]. We tested their four levels of correlations: ρ = {0.95, 0.98, 0.99, 0.995}.

4.3 Preliminary analysis

We propose here a preliminary analysis where we analyze the impact of γ on both the target task and the bias. We conduct this analysis on Biased-MNIST in the unstructured pruning setup. The results are reported in Tab. 1 and they are averaged on three seeds. As expected, we observe that for extremely large values of γ the performance on the target task drops significantly, as the sparsity increases. However, we also notice that for values until $\gamma = 20$ the task accuracy remains high. The accuracy on the bias extraction remains consistently below 20% for $\gamma > 2$, which drives our selection to an intermediate value of $\gamma = 10$ for all the other experiments.

Table 2: Results for Biased-MNIST with different correlation levels.

Method	Metric	ρ 0.99	0.995	0.997	0.999
Vanilla	Task	88.46	74.23	46.04	10.02
	Bias	98.45	99.75	99.96	100.00
Rubi [11]	Task	93.6	43.0	90.4	13.7
EnD [56]	Task	96.0	93.9	83.7	52.3
BCon+BBal [30]	Task	98.1	97.7	97.3	94.0
ReBias [5]	Task	88.4	75.4	65.8	26.5
LearnedMixin [14]	Task	88.3	78.2	50.2	12.1
LfF [46]	Task	95.1	90.3	63.7	15.3
SoftCon [30]	Task	95.2	93.1	88.6	65.0
FFW Structured	Task	97.63	96.68	92.46	19.91
	Bias	16.89	19.03	22.59	17.57
	Sparsity	45.78	47.07	38.40	47.24
FFW	Task	97.79	97.30	89.36	22.29
	Bias	15.74	17.14	27.84	23.13
	Sparsity	46.50	44.62	36.47	43.11

Table 3: Results for CelebA with different attributes.

Method	Metric	Correlated attributes Gender-BlondHair	Gender-Glasses
Vanilla	Task	80.42	98.42
	Bias	72.22	50.00
EnD [56]	Task	86.9	-
LNL [35]	Task	80.1	-
DI [61]	Task	90.9	-
BCon+BBal [30]	Task	91.4	-
Group DRO [51]	Task	85.4	-
LfF [46]	Task	84.2	-
FFW	Task	87.08	98.58
	Bias	50.00	48.66
	Sparsity	49.11	46.58
FFW Structured	Task	86.95	98.28
	Bias	48.00	49.35
	Sparsity	49.11	46.43

4.4 Main results

In this section, we present and discuss the results obtained on the three datasets introduced in Sec. 4.2.

Tab. 2 reports the results obtained for Biased-MNIST at different proportions of bias-misaligned samples (indicated as ρ). For the lower values of ρ we can clearly notice that both FFW structured and unstructured are among other state-of-the-art approaches, suggesting that proper pruning of vanilla-trained models, without further retraining, can yield well-generalizing performance. For very high values of ρ, however, we observe that FFW is falling behind other approaches. This can be explained by the fact that since the correlation between bias and target is extremely high, the disentangled information from the target is not learned by the vanilla model, due to the scarcity of unbiased representants in the training set. In Sec. 4.5 we will ablate on the need for proper fitting of all the samples in the dataset.

Tab. 3 reports the results on the CelebA dataset for both the attributes "BlondHair" and "Glasses" assuming the gender being the bias. Regarding the first attribute, we observe that FFW stands among some debiasing algorithms, yielding performance around 87% for both the structured and the unstructured variants. Some approaches, like BCon+BBal [30] perform careful contrastive learning on the bias features weighted by the loss, and can achieve better

Table 4: Results of FFW method for debiasing Corrupted-CIFAR10.

Method	Metric	Proportion of bias-misaligned samples 0.5%	1%	2%	5%
Vanilla	Task	20.50 ± 0.45	23.90 ± 1.00	29.92 ± 0.10	43.03 ± 1.13
	Bias	89.67 ± 0.92	85.27 ± 1.33	82.18 ± 0.10	79.22 ± 0.45
EnD [55]	Task	19.38	23.12	34.07	36.57
HEX [60]	Task	13.87	14.81	15.20	16.04
ReBias [6]	Task	22.27	25.72	31.66	43.43
LfF [45]	Task	28.57	33.07	39.91	50.27
DFA [37]	Task	29.95	36.49	41.78	51.13
FFW	Task	51.57 ± 3.16	58.25 ± 1.46	58.8 ± 0.71	61.48 ± 0.44
	Bias	10.03 ± 0.35	9.77 ± 0.06	10.85 ± 0.44	10.47 ± 0.51
	Sparsity	6.93 ± 0.60	2.73 ± 0.21	3.19 ± 0.02	5.32 ± 0.12
FFW Structured	Task	52.02 ± 3.24	57.92 ± 0.89	59.20 ± 1.26	60.97 ± 0.40
	Bias	10.65 ± 0.93	10.38 ± 0.43	10.43 ± 0.19	10.40 ± 0.50
	Sparsity	5.26 ± 3.39	2.33 ± 0.88	3.14 ± 0.06	5.31 ± 0.14

Table 5: Results for FFW Biased-MNIST with $\rho = 0.99$ with different extraction times. Note that one full training consists of 600 batches (therefore the three first columns represent respectively 10, 100, and 300 iterations).

	Accuracy	Epochs 1/60	1/6	1/2	1	10	40
Vanilla	Task	9.65 ± 0.24	10.06 ± 0.16	9.97 ± 0.01	9.96 ± 0.01	16.68 ± 3.25	89.34 ± 0.45
	Bias	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	98.69 ± 0.36
FFW	Task	9.71 ± 0.68	11.16 ± 0.33	10.30 ± 0.911	12.42 ± 2.86	46.39 ± 18.70	97.96 ± 0.30
	Bias	9.98 ± 0.08	10.03 ± 0.00	10.02 ± 0.02	13.48 ± 6.03	23.81 ± 10.65	15.81 ± 1.64
	Sparsity	50.34 ± 0.13	51.41 ± 0.52	50.86 ± 0.36	51.19 ± 0.28	48.26 ± 2.48	42.01 ± 5.40

performance; however, notably the vanilla model, properly pruned, ranks among the firsts. Interestingly we have reported the performance for the attribute "Glasses", uncommon for the debiasing community, to show that, despite dataset imbalances, some attributes are naturally ignored by vanilla training (the bias extraction on the vanilla model is already at random guess).

Finally, Tab. 4 reports the results on Corrupted-CIFAR10. In this case, we have averaged our performance on three seeds, motivated by a surprising result: the performance of the vanilla model pruned through FFW surpasses state-of-the-art approaches, by a consistent margin: for 0.5% of proportion for bias-misaligned samples, FFW beats other techniques by approximately 20%. We hypothesize that the corruptions present in this dataset are easy to remove by pruning; still, it is surprising that the vanilla model, properly pruned and without extra fine-tuning, can reach such performance.

4.5 Importance of fitting the training set

As an ablation study, we propose to extract unbiased subnetworks from a vanilla-trained model from models trained for a different amount of epochs. For this study, we select Biased-MNIST with $\rho = 0.99$. We report the results in Tab. 5, averaging the results on three seeds. For this setup, we evidently observe that, even training less than an epoch (as little as 1/60 of an epoch) the vanilla model perfectly fits the bias, and the disentangled features are completely not learned, until the completion of the first epoch. Progressively though, non-biased features are learned by the model, and the accuracy of the target task increases. This suggests that a simple approach as FFW can work on vanilla-trained models if they are sufficiently trained (and fitting) on the target task.

5 Conclusion

In this work, we have analyzed the problem of finding, from vanilla-trained models on known biased setups, debiased sub-networks that can both solve the target task and not rely on the information related to the bias itself. Such an analysis is very important for the debiasing community for both the increasing concerns related to trustworthiness in

AI systems and efficiency in treating models already trained. Indeed, the typical approach followed by the debiasing community is to fine-tune the whole model to remove such a source of bias, showcasing enhanced performance in unbiased environments.

Our first contribution to this work lies in the fact that removing the bias does not necessarily improve the final task's performance. Indeed, in the case such a feature identified as "bias" is inherently necessary to solve a target task, its removal leads to a performance drop. We have derived a theoretical framework that also suggests such a behavior.

Secondly, we have proposed FFW, a technique that, without requiring fine-tuning the vanilla model, is able to surgically remove parameters from the model, unveiling the existence of an unbiased sub-network. We have proposed both an unstructured and a structured variant of such an approach, which also provides guarantees on the biased information employable for solving the target task. On three common benchmarks, we have observed that such sub-networks exist, leading to performance comparable with other state-of-the-art approaches. One major finding is that they are even structured, potentially leading also to computational gains. This finding bridges the sparsity and debiasing communities, opening the road to the design of more energy-efficient debiasing approaches.

Acknowledgements

Part of this work was funded by the French National Research Agency (ANR-22-PEFT-0007) as part of France 2030 and the NF-FITNESS project, and by Hi!PARIS Center on Data Analytics and Artificial Intelligence. Besides, this work was also funded by the European Union's Horizon Europe research and innovation program under grant agreement No. 101120237 (ELIAS).

A More experiments

A.1 Testing our method on multiple biases: Multi-Color MNIST

Table 6: Test accuracy on four subsets of Multi-Color MNIST. The "Unbiased" one is the average of the four. A stands for Aligned and C for Conflicting (regarding the bias-alignement of the samples in each subset).

Method	**Sparsity**	**Bias acc. [%] (↓)**		**Task acc. [%] (↑)**				
		Left	Right	A_L / A_R	A_L / C_R	C_L / A_R	C_L / C_R	Unbiased
Vanilla	0.00	96.38	64.7	100.0	97.7	30.37	7.68	58.9
LfF [46]	0.00	-	-	99.6	4.7	98.6	5.1	52.0
EIIL [17]	0.00	-	-	100.0	97.2	70.8	10.9	69.7
PGI [1]	0.00	-	-	98.6	82.6	26.6	9.5	54.3
DebiAN [40]	0.00	-	-	100.0	95.6	76.5	16.0	72.0
VCBA [44]	0.00	-	-	100.0	90.9	77.5	24.1	**73.1**
FFW	16.95	20.27	16.55	34.57	35.17	39.86	**35.85**	36.37

To complexify the challenge posed by Biased MNIST, Li *et al.* proposed a bi-colored version in [40] to assess models' performance on multiple biases simultaneously. In this version, the left and right sides of the background have different colors, each correlated to the target with ρ_L and ρ_R respectively. We adopt their setup with $\rho_L = 0.99$ and $\rho_R = 0.95$. We applied FFW to this dataset to obtain the results presented in Tab. 6. If FFW yields worst results on average than the vanilla network, we can notice that it is the pruning led to the best results on the worst sub-group: the one with bias-conflicting backgrounds on both sides. On this one, we get an increase of 11%. We can also notice that the results are comparable on every subset, meaning that the network is nos its choices on the background color anymore.

B Visualizations

B.1 Is FFW helping the network focus on the right features ?

For Fig.3, we applied the Grad-Cam visualization method from [52] to FFW on Biased-MNIST. The raw sample (on the left) leads to high activation around the digit for the biased model (in the middle) but after pruning with FFW, the higher activations are placed on the digit.

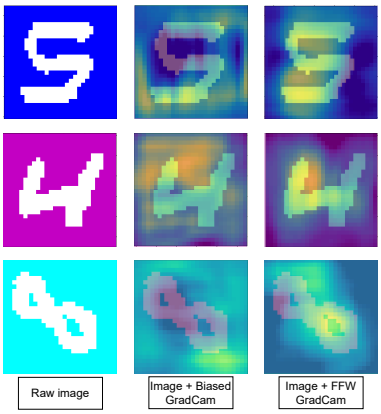
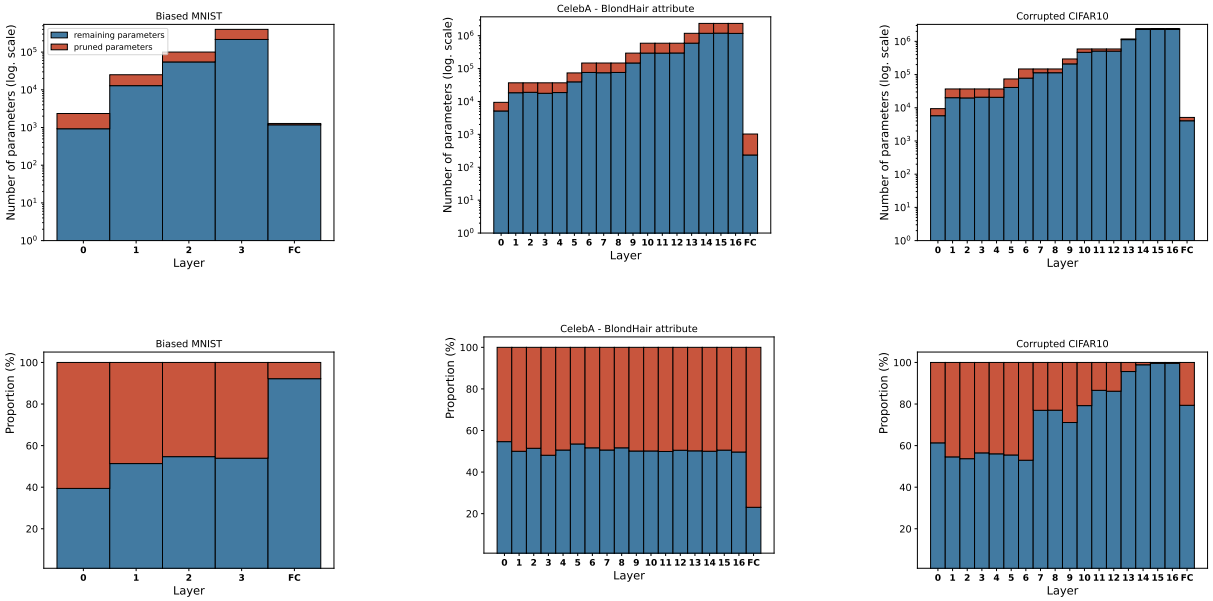


Figure 3: Grad-Cam visualization of the effects of FFW on Biased-MNIST with $\rho = 0.997$.

B.2 Pruning distribution across the networks.



(a) Biased MNIST on a small CNN.

(b) CelebA on ResNet18.

(c) Corrupted CIFAR10 on ResNet18.

Figure 4: Absolute number (top row) and proportions (bottom row) of pruned parameters after applying FFW to Biased MNIST, CelebA, and Corrupted CIFAR10.

Fig. 4 shows the proportions of pruning in multiple networks for multiple datasets and two networks. It should be noted that while the network's pruning is spread over all layers for CelebA, it is focused on the first few layers for Corrupted Cifar10, potentially indicating the higher simplicity of the bias (a simple filter applied to the images) for that dataset.

C More ablations

C.1 Variations on the Mutual Information Minimization

In Fig. 5 we can visualize the results of Tab. 1, showing that the Task Accuracy can be maintained high while minimizing the Private Accuracy for γ ranging from 5 to 50 on that specific dataset.

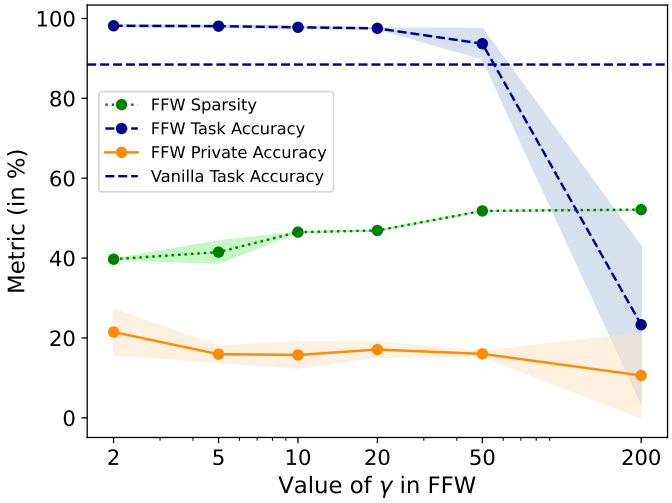


Figure 5: Results for FFW applied on Biased-MNIST ($\rho = 0.99$) with different γ on the validation set.

Table 7: Results for Biased-MNIST ($\rho = 0.99$ with different pruning strategies.

Strategy	Sparsity	Accuracy	
		Task	Bias
Vanilla Model	0	88.46 ± 0.63	98.45 ± 0.20
Magnitude Pruning	0.5	88.06 ± 0.8	99.04 ± 0.17
	0.75	78.16 ± 4.59	99.58 ± 0.14
	0.9	31.14 ± 9.96	98.30 ± 0.23
	0.95	10.50 ± 1.16	77.45 ± 11.29
	0.975	9.52 ± 0.53	40.98 ± 7.18
FFW Structured	0.46	97.63 ± 1.02	16.89 ± 3.15
FFW	0.47	97.79 ± 0.30	15.64 ± 3.26

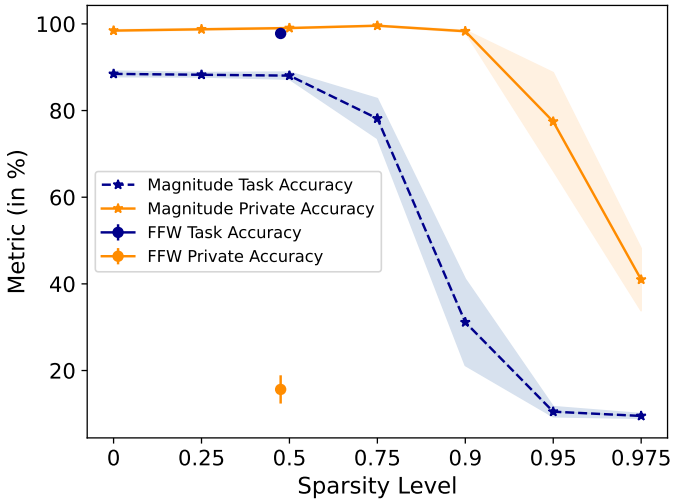


Figure 6: Results of Magnitude Pruning on Biased MNIST ($\rho = 0.99$) at different levels of sparsity compared to FFW.

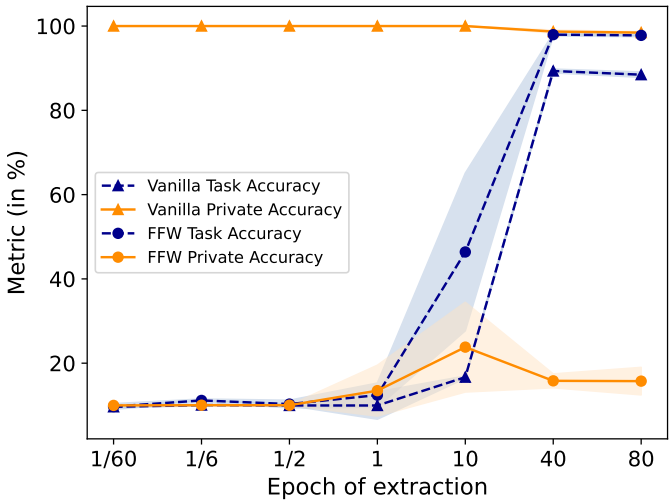


Figure 7: Results of FFW on Biased MNIST ($\rho = 0.99$) applied after different biased training durations.

C.2 Comparison of Pruning Strategies

In Fig. 6, we can compare FFW to a vanilla magnitude pruning approach. While the magnitude pruning approach destroys both the Task and the Bias information while pruning, which is clear in the fact the both curves drop as sparsity increases, FFW does the opposite. At its 46% of sparsity, the pruning leads to a Private Accuracy close to random guess while increasing the Task Accuracy.

C.3 Effect of the extraction point

Fig. 7 helps us visualize the results of Tab.5. Indeed it shows that until the completion of at least a few epochs, the model fits only the bias and therefore, it is impossible to extract a debiased subnetwork by our method. However, as the network learns unbiased features, our method keeps minimizing the propagation of biased information but starts progressively extracting subnetworks that are efficient on the target task.

D Details for Section 3

In this section, we will provide all the derivations for Sec. 3 in the main paper. more specifically, Sec. D.1 will propose the derivation for (3), Sec. D.2 will discuss the joint probability that will be employed to derive (4) in Sec. D.3, and (6) in Sec. D.4.

D.1 Derivation for (3)

Given the joint probability as in (2), we can easily express the mutual information between $\hat{B}$ and $\hat{Y}$ as

$$\begin{aligned}
\mathcal{I}(\hat{B},\hat{Y}) =& \sum_{i,j} p(\hat{b}_j,\hat{y}_i)\log_2\frac{p(\hat{b}_j,\hat{y}_i)}{p(\hat{b}_j)p(\hat{y}_i)} \\
=& \frac{N_B}{N_C}\rho\log_2\left[\frac{\rho N_C N_B}{N_C}\right] + \frac{N_B(N_B-1)}{N_C(N_B-1)}(1-\rho)\log_2\left[\frac{(1-\rho)N_C N_B}{N_B-1}\right] \\
=& \frac{N_B}{N_C}\left\{\rho\log_2(N_B)+\rho\log_2(\rho)+(1-\rho)\log_2(N_B)+(1-\rho)\log_2\left(\frac{1-\rho}{N_B-1}\right)\right\} \\
=& \frac{N_B}{N_C}\left\{\log_2(N_B)+\rho\log_2(\rho)+(1-\rho)\log_2\left(\frac{1-\rho}{N_B-1}\right)\right\} \qquad (11)
\end{aligned}$$

finding back (3).

D.2 Joint probability between $\hat{B}$, $\hat{Y}$, Y

A clear dependency between ρ and (3), as already showcased in Sec. D.1, exists. This measure is applied to ground-truth labels, investigating the common information between them (and for instance, the information that is possible to disentangle). Nonetheless, in the more general case, the trained model (whose output is modelizable as the random variable Y) is not a perfect learner, having $H(Y|\hat{Y}) \neq 0$. The model, in this case, does not correctly classify the target for two reasons.

1. It gets confused by the bias features, and it tends to learn to classify samples based on them. We model this tendency of learning biased features with ϕ, which we call *biasedness*. The higher the biasedness is, the more the model relies on features that we desire to suppress, inducing bias in the model and for instance error in the model.
2. Some extra error ε, non-directly related to the bias features, which can be caused, for example, by stochastic unbiased effects, to underfit, or to other high-order dependencies between data. This contribution is already visible in (1).

We can write the discrete joint probability for $\hat{B}, \hat{Y}, Y$, composed of the following terms.

- When target, bias, and prediction are aligned, the bias is aligned with the target class and correctly classified. Considering that we are not perfect learners, we introduce the error term ε.
- When the target and bias are misaligned and the prediction is correct, it means that the model has learned the correct feature and the bias is being contrasted. This effect is due to the dual effect of both the model's biasedness ϕ and the inherent ground-truth dependence between the use of the biased feature and the target label K_{bia}.
- When target and bias are not aligned, but the prediction is incorrect and bias and output are aligned, it means that the model has learned the bias, introducing the error we target to minimize in this work.
- In all the other cases, the error of the model is due to higher-order dependencies, not directly related to the biasedness ϕ.

Under the assumption $N_B = N_C = N$, we can write the joint distribution

$$\begin{aligned} p(\hat{B}, \hat{Y}, Y) = &\frac{1}{N} \cdot \left[\delta_{\hat{b}\hat{y}y} \rho(1-\varepsilon) + \delta_{\hat{y}y} \bar{\delta}_{\hat{b}y} \bar{\delta}_{\hat{b}\hat{y}} \frac{(1-\phi)(1-\rho)}{N-1} (1 - K_{\text{bia}}) + \right. \\ &+ \bar{\delta}_{\hat{y}y} \delta_{\hat{b}y} \bar{\delta}_{\hat{b}\hat{y}} \frac{\phi(1-\rho)}{N-1} K_{\text{bia}} + \bar{\delta}_{\hat{y}y} \bar{\delta}_{\hat{b}y} \delta_{\hat{b}\hat{y}} \frac{\varepsilon\rho^2}{N-2+\rho} + \\ &\left. + \bar{\delta}_{\hat{y}y} \bar{\delta}_{\hat{b}y} \bar{\delta}_{\hat{b}\hat{y}} \frac{\varepsilon\rho(1-\rho)}{(N-1)(N-2+\rho)} \right]. \end{aligned} \tag{12}$$

D.3 Derivation for (4)

Let the joint probability as in (12). We can marginalize on $\hat{Y}$ by summing all the N_C (in our simplified case, N) biases per given target class and prediction of the model:

$$p(\hat{B}, Y) = \frac{1}{N} \left\{ \delta_{\hat{b}y} \left[\rho(1-\varepsilon) + \phi(1-\rho) \right] + \bar{\delta}_{\hat{b}y} \left[\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right] \right\}. \tag{13}$$

From this, following the definition of mutual information, we can write

$$\begin{aligned} \mathcal{I}(\hat{B}, Y) = &\sum_{i,j} p(\hat{b}_j, y_i) \log_2 \frac{p(\hat{b}_j, y_i)}{p(\hat{b}_j) p(y_i)} \\ = &\frac{1}{N} \left\{ N \cdot \left[\rho(1-\varepsilon) + \phi(1-\rho) \right] \cdot \log_2 (N \cdot (\rho(1-\varepsilon) + \phi(1-\rho))) + \right. \\ &+ (N^2 - N) \cdot \left[\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right] \cdot \\ &\left. \cdot \log_2 \left[N \cdot \left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right) \right] \right\} \end{aligned}$$

$$
\begin{aligned}
&= [\rho(1-\varepsilon) + \phi(1-\rho)] \cdot [\log_2 N + \log_2(\rho(1-\varepsilon) + \phi(1-\rho)] + \\
&\quad + \left[(1-\phi)(1-\rho) + \frac{(N-1)\rho\varepsilon}{N-2+\rho}\right] \cdot \\
&\quad \cdot \left[\log_2 N + \log_2\left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho}\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \log_2 \left[\rho(1-\varepsilon) + \phi(1-\rho)\right]^{\rho(1-\varepsilon)+\phi(1-\rho)} \\
&\quad + \log_2 \left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho}\right)^{(1-\phi)(1-\rho)+\frac{(N-1)\rho\varepsilon}{N-2+\rho}} \\
&\quad + \log_2 N \left[\rho(1-\varepsilon) + \phi(1-\rho) + \right. \\
&\quad \left. + \ (1-\phi)(1-\rho) + \frac{(N-1)\rho\varepsilon}{N-2+\rho}\right] \\
&= \log_2 \left[\rho(1-\varepsilon) + \phi(1-\rho)\right]^{\rho(1-\varepsilon)+\phi(1-\rho)} \\
&\quad + \log_2 \left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho}\right)^{(1-\phi)(1-\rho)+\frac{(N-1)\rho\varepsilon}{N-2+\rho}} \\
&\quad + \log_2 N \left[1 - \rho\varepsilon + \frac{(N-1)\rho\varepsilon}{N-2+\rho}\right].
\end{aligned} \tag{14}
$$

From here, we can easily obtain the normalized mutual information by scaling down the results of a factor $\log_2(N)$. Under the assumption that $\varepsilon = 0$, we obtain

$$
\begin{aligned}
\mathcal{I}(\hat{B}, Y) =& \frac{1}{\log_2(N)} \left\{(\rho + \phi(1-\rho)) \log_2 \left[\rho + \phi(1-\rho)\right] + \right. \\
&\left. + [(1-\phi)(1-\rho)] \log_2 \left[\frac{(1-\phi)(1-\rho)}{N-1}\right] + 1 \right\} \\
=& \frac{\rho + \phi(1-\rho)}{\log_2(N)} \log_2 \left[\rho + \phi(1-\rho)\right] + \frac{(1-\phi)(1-\rho)}{\log_2(N)} \log_2 \left[\frac{(1-\phi)(1-\rho)}{N-1}\right] + 1,
\end{aligned} \tag{15}
$$

finding back (4).

D.4 Derivation for (6)

Similarly to the approach taken in Sec. D.3, from the joint probability as in (12), we marginalize, but this time on $\hat{B}$, by summing all the N_B (in our simplified case, N) biases per given target class and prediction of the model. Under the assumption that $\varepsilon = \varepsilon_{\text{bia}}$, we have:

$$
\begin{aligned}
p(\hat{Y}, Y) =& \frac{1}{N} \left\{\delta_{\hat{y}y} \left[\rho(1-\varepsilon_{\text{bia}}) + (1-\phi)(1-\rho)(1-K_{\text{bia}})\right] + \right. \\
&\left. + \bar{\delta}_{\hat{y}y} \left[\frac{\phi(1-\rho)}{N-1} K_{\text{bia}} + \frac{\varepsilon_{\text{bia}}\rho^2}{N-2+\rho} + \frac{N-2}{N-2+\rho} \frac{\varepsilon_{\text{bia}}\rho(1-\rho)}{N-1}\right]\right\}
\end{aligned} \tag{16}
$$

Also in this case, following the definition of mutual information, we can write

$$\begin{aligned}
\mathcal{I}(\hat{Y},Y) =& \sum_{i,j} p(\hat{y}_j, y_i) \log_2 \frac{p(\hat{y}_j, y_i)}{p(\hat{y}_j)p(y_i)} \\
=& \frac{1}{N} \{N \cdot [\rho(1-\varepsilon) + \phi(1-\rho)] \cdot \log_2(N \cdot (\rho(1-\varepsilon) + \phi(1-\rho))) + \\
&+ (N^2 - N) \cdot \left[\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right] \cdot \\
&\log_2 \left[N \cdot \left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right) \right] \Big\} \\
=& [\rho(1-\varepsilon) + \phi(1-\rho)] \cdot [\log_2 N + \log_2(\rho(1-\varepsilon) + \phi(1-\rho)] + \\
&+ \left[(1-\phi)(1-\rho) + \frac{(N-1)\rho\varepsilon}{N-2+\rho} \right] \cdot \\
&\cdot \left[\log_2 N + \log_2 \left(\frac{(1-\phi)(1-\rho)}{N-1} + \frac{\rho\varepsilon}{N-2+\rho} \right) \right].
\end{aligned}$$

Here, defining

$$f(x,y,z) = xy \log_2(xz), \tag{17}$$

we can write

$$\begin{aligned}
\mathcal{I}(\hat{Y},Y) = f\left\{ \frac{1}{N} \left[\rho(1-\varepsilon_{\text{bia}}) + (1-\phi)(1-\rho)(1-K_{\text{bia}}) \right], N, N^2 \right\} + \\
f\left\{ \frac{1}{N} \left[\frac{\phi(1-\rho)}{N-1} K_{\text{bia}} + \left(\frac{\rho^2(N-2) + \rho(1-\rho)(N-2)}{(N-2+\rho)} \right) \varepsilon_{\text{bia}} \right], N(N-1), N^2 \right\},
\end{aligned}$$

finding back (6).

References

[1] Ahmed, F., Bengio, Y., van Seijen, H., Courville, A.C.: Systematic generalisation with group invariant predictions. In: International Conference on Learning Representations (2021)

[2] et al., J.Z.: REST: Enhancing Group Robustness in DNNs Through Reweighted Sparse Training. Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2023)

[3] Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D.: Invariant risk minimization. arXiv preprint arXiv:1907.02893 (2019)

[4] Bahng, H., Chun, S., Yun, S., Choo, J., Oh, S.J.: Learning de-biased representations with biased representations. In: International Conference on Machine Learning. pp. 528–539. PMLR (2020)

[5] Bahng, H., Chun, S., Yun, S., Choo, J., Oh, S.J.: Learning De-biased Representations with Biased Representations (Jun 2020), `http://arxiv.org/abs/1910.02806`, arXiv:1910.02806 [cs, stat]

[6] Bahng, H., Chun, S., Yun, S., Choo, J., Oh, S.J.: Learning de-biased representations with biased representations. In: International Conference on Machine Learning (ICML) (2020)

[7] Bai, H., Zhou, F., Hong, L., Ye, N., Chan, S.H.G., Li, Z.: Nas-ood: Neural architecture search for out-of-distribution generalization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8320–8329 (2021)

[8] Barbano, C.A., Dufumier, B., Tartaglione, E., Grangetto, M., Gori, P.: Unbiased supervised contrastive learning. In: ICLR (2023)

[9] Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)

[10] Blakeney, C., Huish, N., Yan, Y., Zong, Z.: Simon says: Evaluating and mitigating bias in pruned neural networks with knowledge distillation. arXiv preprint arXiv:2106.07849 (2021)

[11] Cadene, R., Dancette, C., Ben-younes, H., Cord, M., Parikh, D.: RUBi: Reducing Unimodal Biases in Visual Question Answering (Mar 2020), `http://arxiv.org/abs/1906.10169`, arXiv:1906.10169 [cs]

[12] Cebollada, S., Payá, L., Flores, M., Peidró, A., Reinoso, O.: A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data. Expert Systems with Applications **167**, 114195 (2021)

[13] Choi, Y., Uh, Y., Yoo, J., Ha, J.W.: Stargan v2: Diverse image synthesis for multiple domains. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8188–8197 (2020)

[14] Clark, C., Yatskar, M., Zettlemoyer, L.: Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases (Sep 2019), http://arxiv.org/abs/1909.03683, arXiv:1909.03683 [cs]

[15] Corbett-Davies, S., Goel, S.: The measure and mismeasure of fairness: A critical review of fair machine learning. arXiv preprint arXiv:1808.00023 (2018)

[16] Creager, E., Jacobsen, J.H., Zemel, R.: Environment inference for invariant learning. In: International Conference on Machine Learning. pp. 2189–2200. PMLR (2021)

[17] Creager, E., Jacobsen, J.H., Zemel, R.: Environment inference for invariant learning (2020). https://doi.org/10.48550/ARXIV.2010.07249, https://arxiv.org/abs/2010.07249

[18] Csordás, R., van Steenkiste, S., Schmidhuber, J.: Are neural nets modular? inspecting functional modularity through differentiable weight masks. arXiv preprint arXiv:2010.02066 (2020)

[19] Diffenderfer, J., Bartoldson, B., Chaganti, S., Zhang, J., Kailkhura, B.: A winning hand: Compressing deep networks can improve out-of-distribution robustness. Advances in Neural Information Processing Systems **34**, 664–676 (2021)

[20] Fan, L., Krishnan, D., Isola, P., Katabi, D., Tian, Y.: Improving clip training with language rewrites. Advances in Neural Information Processing Systems **36** (2024)

[21] Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 259–268 (2015)

[22] Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)

[23] Geirhos, R., Jacobsen, J.H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., Wichmann, F.A.: Shortcut learning in deep neural networks. Nature Machine Intelligence **2**(11), 665–673 (2020)

[24] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F.A., Brendel, W.: Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231 (2018)

[25] Goel, K., Gu, A., Li, Y., Ré, C.: Model patching: Closing the subgroup performance gap with data augmentation. arXiv preprint arXiv:2008.06775 (2020)

[26] Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S.R., Smith, N.A.: Annotation artifacts in natural language inference data. arXiv preprint arXiv:1803.02324 (2018)

[27] Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning. Advances in neural information processing systems **29** (2016)

[28] Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019)

[29] Hong, Y., Yang, E.: Unbiased classification through bias-contrastive and bias-balanced learning. Advances in Neural Information Processing Systems **34**, 26449–26461 (2021)

[30] Hong, Y., Yang, E.: Unbiased Classification through Bias-Contrastive and Bias-Balanced Learning. In: Advances in Neural Information Processing Systems. vol. 34, pp. 26449–26461. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/hash/de8aa43e5d5fa8536cf23e54244476fa-Abstract.html

[31] Hupont, I., Micheli, M., Delipetrev, B., Gómez, E., Garrido, J.S.: Documenting high-risk ai: a european regulatory perspective. Computer **56**(5), 18–27 (2023)

[32] Izmailov, P., Kirichenko, P., Gruver, N., Wilson, A.G.: On feature learning in the presence of spurious correlations. Advances in Neural Information Processing Systems **35**, 38516–38532 (2022)

[33] Kamiran, F., Karim, A., Zhang, X.: Decision theory for discrimination-aware classification. In: 2012 IEEE 12th international conference on data mining. pp. 924–929. IEEE (2012)

[34] Kang, L., Riba, P., Rusinol, M., Fornes, A., Villegas, M.: Content and style aware generation of text-line images for handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(12), 8846–8860 (2021)

[35] Kim, B., Kim, H., Kim, K., Kim, S., Kim, J.: Learning not to learn: Training deep neural networks with biased data (2018). https://doi.org/10.48550/ARXIV.1812.10352, https://arxiv.org/abs/1812.10352

[36] Kim, B., Kim, H., Kim, K., Kim, S., Kim, J.: Learning not to learn: Training deep neural networks with biased data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9012–9020 (2019)

[37] Lee, J., Kim, E., Lee, J., Lee, J., Choo, J.: Learning debiased representation via disentangled feature augmentation. Advances in Neural Information Processing Systems **34**, 25123–25133 (2021)

[38] Li, X., Tian, Y., Zhang, F., Quan, S., Xu, Y.: Object detection in the context of mobile augmented reality. In: 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR). pp. 156–163. IEEE (2020)

[39] Li, Z., Hoogs, A., Xu, C.: Discover and mitigate unknown biases with debiasing alternate networks. In: European Conference on Computer Vision. pp. 270–288. Springer (2022)

[40] Li, Z., Hoogs, A., Xu, C.: Discover and Mitigate Unknown Biases with Debiasing Alternate Networks (Sep 2022), http://arxiv.org/abs/2207.10077, arXiv:2207.10077 [cs]

[41] Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild (2014). https://doi.org/10.48550/ARXIV.1411.7766, https://arxiv.org/abs/1411.7766

[42] Ma, X., Ouyang, W., Simonelli, A., Ricci, E.: 3d object detection from images for autonomous driving: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)

[43] Nahon, R., Nguyen, V.T., Tartaglione, E.: Mining bias-target alignment from voronoi cells. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 4923–4932 (2023). https://doi.org/10.1109/ICCV51070.2023.00456

[44] Nahon, R., Nguyen, V.T., Tartaglione, E.: Mining bias-target alignment from voronoi cells (2023)

[45] Nam, J., Cha, H., Ahn, S., Lee, J., Shin, J.: Learning from failure: De-biasing classifier from biased classifier. Advances in Neural Information Processing Systems **33**, 20673–20684 (2020)

[46] Nam, J., Cha, H., Ahn, S., Lee, J., Shin, J.: Learning from Failure: De-biasing Classifier from Biased Classifier. In: Advances in Neural Information Processing Systems. vol. 33, pp. 20673–20684. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/hash/eddc3427c5d77843c2253f1e799fe933-Abstract.html

[47] Narayanan, A.: Translation tutorial: 21 fairness definitions and their politics. In: Proc. Conf. Fairness Accountability Transp., New York, USA. vol. 1170, p. 3 (2018)

[48] O'Neill, J., Dutta, S., Assem, H.: Self-distilled pruning of deep neural networks. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 655–670. Springer (2022)

[49] Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., Tucker, G.: On variational bounds of mutual information. In: International Conference on Machine Learning. pp. 5171–5180. PMLR (2019)

[50] Regulation, G.D.P.: General data protection regulation (gdpr). Intersoft Consulting, Accessed in October **24**(1) (2018)

[51] Sagawa, S., Koh, P.W., Hashimoto, T.B., Liang, P.: Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization (Apr 2020), http://arxiv.org/abs/1911.08731, arXiv:1911.08731 [cs, stat]

[52] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision **128**(2), 336–359 (Oct 2019). https://doi.org/10.1007/s11263-019-01228-7, http://dx.doi.org/10.1007/s11263-019-01228-7

[53] Tartaglione, E.: Information Removal at the bottleneck in Deep Neural Networks (Sep 2022), http://arxiv.org/abs/2210.00891, arXiv:2210.00891 [cs]

[54] Tartaglione, E.: The rise of the lottery heroes: why zero-shot pruning is hard. In: 2022 IEEE International Conference on Image Processing (ICIP). pp. 2361–2365. IEEE (2022)

[55] Tartaglione, E., Barbano, C.A., Grangetto, M.: End: Entangling and disentangling deep representations for bias correction. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 13508–13517 (2021)

[56] Tartaglione, E., Barbano, C.A., Grangetto, M.: End: Entangling and disentangling deep representations for bias correction. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13503–13512 (2021). https://doi.org/10.1109/CVPR46437.2021.01330

[57] The eu artificial intelligence act (2024), https://artificialintelligenceact.eu/

[58] Veale, M., Zuiderveen Borgesius, F.: Demystifying the draft eu artificial intelligence act—analysing the good, the bad, and the unclear elements of the proposed approach. Computer Law Review International **22**(4), 97–112 (2021)

[59] Wang, H., He, Z., Lipton, Z.C., Xing, E.P.: Learning robust representations by projecting superficial statistics out. arXiv preprint arXiv:1903.06256 (2019)

[60] Wang, H., He, Z., Lipton, Z.L., Xing, E.P.: Learning robust representations by projecting superficial statistics out. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=rJEjjoR9K7

[61] Wang, Z., Qinami, K., Karakozis, I.C., Genova, K., Nair, P., Hata, K., Russakovsky, O.: Towards fairness in visual recognition: Effective strategies for bias mitigation (2019). https://doi.org/10.48550/ARXIV.1911.11834, https://arxiv.org/abs/1911.11834

[62] Xu, P., Zhu, X., Clifton, D.A.: Multimodal learning with transformers: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023)

[63] Zayed, A., Mordido, G., Shabanian, S., Baldini, I., Chandar, S.: Fairness-aware structured pruning in transformers. arXiv preprint arXiv:2312.15398 (2023)

[64] Zhang, D., Ahuja, K., Xu, Y., Wang, Y., Courville, A.: Can subnetwork structure be the key to out-of-distribution generalization? In: International Conference on Machine Learning. pp. 12356–12367. PMLR (2021)

[65] Zhang, M., Sohoni, N.S., Zhang, H.R., Finn, C., Ré, C.: Correct-n-contrast: A contrastive approach for improving robustness to spurious correlations. arXiv preprint arXiv:2203.01517 (2022)

[66] Zhao, B., Chen, C., Ju, Q., Xia, S.: Learning debiased models with dynamic gradient alignment and bias-conflicting sample mining. arXiv preprint arXiv:2111.13108 (2021)