Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-26)
Survey Track

A Resource-Aware Taxonomy of AI Bias Mitigation Techniques

Daniela Loreti[1], **Roberta Calegari**[1] and **Michela Milano**[1,2]
[1]Alma Mater Studiorum–Università di Bologna, Italy
[2]Fondazione Bruno Kessler, Italy
{daniela.loreti, roberta.calegari, michela.milano}@unibo.it

Abstract

The literature on AI fairness has grown rapidly, proposing a large number of bias mitigation techniques that are commonly organized into pre-, in-, and post-processing methods. This pipeline-centric view offers an operational, lifecycle-based perspective on *where* mitigation can be applied. In deployment settings, however, practitioners also face an additional question: *whether* a mitigation family is applicable given the resources and access rights available in a concrete system. In this survey, we use *resources* broadly to denote data access/control, training capability, and deployment-time interface/decision control. Accordingly, we introduce a *resource-aware taxonomy* that complements existing taxonomies by classifying AI bias mitigation methods according to the conditions that make them practically implementable. We use this taxonomy to structure and reinterpret existing literature on the topic, highlighting which mitigation families remain feasible under resource constraints.

1 Introduction

Artificial Intelligence (AI) systems achieve state-of-the-art performance across many domains, yet can still produce discriminatory outcomes due to biased data, targets, or modeling choices [Caton and Haas, 2024]. This is especially concerning in high-stakes decisions, motivating sustained research and regulatory initiatives such as the EU AI Act [European Parliament, 2024]. This has produced a rapidly growing literature on fairness metrics and mitigation. Most surveys organize mitigation into pre-, in-, and post-processing [Caton and Haas, 2024; Hort *et al.*, 2024], i.e., by *where* interventions occur in the Machine Learning (ML) lifecycle. Domain-specific variants refine this view: recommender systems distinguish data-, ranking-, and re-ranking-based methods [Wang *et al.*, 2023], while Large Language Model (LLM) surveys separate in-training interventions from inference-time intra- and post-processing [Gallegos *et al.*, 2024]. Overall, these taxonomies offer an operational, lifecycle-based account of *where* fairness can be engineered. In deployment settings, however, practitioners must also answer a different operational question: *whether* a mitigation family is applicable at all, given the concrete resources and access rights available in the system. The feasibility of adopting a method is often governed by resource constraints such as: availability and modifiability of training data (including the possibility of collecting/generating new data), permissions and computational capacity to (re)train, and the degree of access to the deployed model and its interfaces (e.g., a trainable checkpoint versus a black-box API exposing only scores or labels). These conditions do not replace lifecycle-based views; rather, they make explicit an applicability dimension that refines them by indicating which lifecycle intervention points are realistically accessible under a given set of resource and access constraints. Motivated by this observation, we propose a *resource-based* taxonomy of AI bias mitigation techniques complementing existing pipeline-centric classifications by re-indexing the literature according to the resources that make interventions feasible. We use the term *resources* in a broad operational sense, encompassing (i) computational budget and training infrastructure, (ii) availability and modifiability of data (including the ability to collect or generate new data), and (iii) access to the trained model and/or training procedure (e.g., checkpoint access, interface outputs, or full training pipeline control). Concretely, we organize AI bias mitigation families along four resource axes: (i) **data resources** (feature control, instance control, or add-only data acquisition/generation), (ii) **training resources** (ability to run parameter updates, from full retraining to full/partial fine-tuning), (iii) **model access** (checkpoint vs. score- vs. decision-level interfaces, including downstream decision control), and (iv) **human oversight** (governance controls applicable across deployments). We further distill these axes into a screening flowchart that maps a concrete deployment setting to the mitigation families that are feasible under its constraints.

1.1 Problem Formulation and Notation

We model a deployed decision system as a predictor $f_\theta : \mathcal{X} \to \mathcal{S}$ parameterized by θ, where $\mathcal{X}$ is the input space and $\mathcal{S}$ denotes the output space exposed by the interface. Depending on the setting, $\mathcal{S}$ may contain (i) real-valued scores or probabilities $s \in \mathbb{R}$ (or $\mathbb{R}^k$ for multiclass), (ii) hard decisions/labels $\hat{y} \in \mathcal{Y}$, or (iii) structured outputs such as ranked lists or generated text. We assume supervised targets $y \in \mathcal{Y}$ are available during training or evaluation. Data are represented as a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where n denotes

the number of examples, each consisting of input features $x_i \in \mathcal{X}$ and a target label $y_i \in \mathcal{Y}$. When available, we denote by $a_i \in \mathcal{A}$ a protected attribute or group membership variable (e.g., gender, age bracket, sensitive group label), and we write $\mathcal{D} = \{(x_i, y_i, a_i)\}_{i=1}^n$ to make group metadata explicit. We define the training objective via a loss function $\ell : \mathcal{S} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, yielding a training objective

$$\mathcal{L}(\theta; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)\,, \tag{1}$$

possibly augmented with fairness terms. We denote by $\Omega_{\text{fair}}(\cdot)$ a generic fairness penalty and by $\mathcal{C}_{\text{fair}}(\cdot) \leq 0$ a generic set of fairness constraints (e.g., constraints on group disparities computed using a). A common fairness-aware training form is then either

$$\min_\theta \; \mathcal{L}(\theta; \mathcal{D}) + \lambda\, \Omega_{\text{fair}}(\theta; \mathcal{D}) \tag{2}$$

or

$$\min_\theta \; \mathcal{L}(\theta; \mathcal{D}) \quad \text{s.t.} \quad \mathcal{C}_{\text{fair}}(\theta; \mathcal{D}) \leq 0, \tag{3}$$

where $\lambda \geq 0$ controls the accuracy–fairness trade-off.

Our taxonomy characterizes *which transformations of this pipeline are actionable* under resource and access constraints. We use the following generic operators throughout the section: (i) a data transformation T applied to individual records, e.g., $x' = T(x)$ and $\mathcal{D}' = \{(T(x_i), y_i[, a_i])\}_{i=1}^n$; (ii) a (re)training or fine-tuning procedure $\mathcal{A}$ that updates parameters, yielding $\theta' = \mathcal{A}(\theta, \mathcal{D}')$ (with θ possibly random initialization or a pre-trained checkpoint); (iii) an output post-processing function g that maps scores to revised scores/decisions, $\hat{y} = g(f_\theta(x), a)$; and (iv) a decision-layer wrapper h that maps model outputs to final actions under constraints or policies, $\text{action} = h(\hat{y}, a)$. When only a released checkpoint is available, we refer to the accessible parameter vector as θ_0 and denote the updated checkpoint as θ', without assuming access to the original training code or configuration. This notation lets us characterize each family of interventions by its *primary operator*: a data/representation transform T (applied to records or features), a parameter-update procedure $\mathcal{A}$ (training or fine-tuning that maps $(\theta, \mathcal{D})$ to updated weights), an output mapping g acting on scores/probabilities, or a decision-layer policy h that post-processes model decisions downstream, depending on the available resources and access regime.

2 Resource-Aware Taxonomy

We introduce a *resource-aware* taxonomy of AI bias mitigation that makes explicit applicability conditions under resource and access constraints. Table 1 structures these conditions along four axes—**Data**, **Training**, **Model**, and **Human** resources—and indexes mitigation families via leaf categories (A–J), referenced throughout the paper.

2.1 Interventions through Data Resources

The **Data** column groups mitigation strategies that operate by manipulating the data resources available for training, adaptation, or evaluation. These methods are largely model-agnostic: they target the *inputs* available for learning rather than the learning algorithm itself. In our resource-aware perspective, applicability is determined by data access and control conditions: (i) whether features can be changed (**A**); (ii) whether existing rows are accessible and editable (**B/C**); and (iii) whether new rows can be added when the original ones are not editable or not available (**D**). In all cases, effectiveness ultimately depends on whether the resulting data resource can be operationalized downstream (e.g., via retraining/fine-tuning or input-pipeline updates).

Feature-level data interventions (Category A). Category A covers interventions that modify the *feature space* (i.e., *columns*): feature review [Limisiewicz and Mareček, 2022; Wang *et al.*, 2020], removal/blinding of sensitive attributes [Grgić-Hlača *et al.*, 2018; Feldman *et al.*, 2015], and mitigation of proxy variables [Panda *et al.*, 2022]. Technically, these methods apply a representation transform T to inputs, producing $x' = T(x)$ (and thus $\mathcal{D}' = \{(T(x_i), y_i[, a_i])\}_i$) via column deletion, re-encoding, or feature mapping, with the goal of reducing sensitive/proxy information before training or inference. They are applicable only when practitioners can control the feature representation presented to the model (at training time or in the deployed input pipeline). Importantly, Category A is *not* applicable when the feature interface is fixed by an external provider (e.g., a third-party training API that enforces a predefined schema). In such settings, feature-level mitigation requires renegotiating the interface or switching to an editable pipeline.

Instance-level data interventions (Category B). Category B includes interventions that operate on *existing instances* (i.e., *rows*) by *editing the original records*: cleaning [Thakur *et al.*, 2023; Raffel *et al.*, 2020], relabeling/perturbation [Luong *et al.*, 2011; Kamiran and Calders, 2012], and instance-level dataset editing/augmentation [Feng *et al.*, 2019; Maudslay *et al.*, 2019]. A key case is *counterfactual augmentation* [Kusner *et al.*, 2017; Lu *et al.*, 2020], which generates new samples but remains in Category B (intstead of pertaining to Category D) because it typically requires access to the original instances to construct plausible counterfactual variants (e.g., changing protected attributes while holding the remaining context fixed). Technically, these methods produce a revised dataset $\mathcal{D}' = T(\mathcal{D})$ by correcting, removing, rewriting, or augmenting rows based on the existing training set (and optionally group metadata a), with the intent of improving label quality, group coverage, and fairness-relevant signal. Operationally, Category B requires that the original dataset be accessible and that instances be *modifiable* (or at least usable as anchors for edits/augmentations). For mitigation at deployment time, it additionally requires a downstream mechanism to exploit $\mathcal{D}'$ (typically retraining or fine-tuning).

Reweighting/resampling (Category C). Category C captures instance-level methods that do *not* rewrite records, but change how existing instances influence learning through instance weighting [Petrović *et al.*, 2022; Han *et al.*, 2022] and stratified sampling [Ekstrand *et al.*, 2018; Iosifidis *et al.*, 2019]. Technically, they replace standard loss function with a weighted or resampled objective, e.g., $\min_\theta \sum_i w_i\, \ell(f_\theta(x_i), y_i)$, where $w_i \geq 0$ denotes the (possi-

Data Resources	Training Resources (Algorithm)	Model Access Resources	Human Oversight
Feature-level column modifications **A – Feature interventions**: • feature review • proxy removal/blinding **Instance-level row modifications** **B – Instance interventions**: • cleaning • relabeling • transformation/representation learning • counterfactual augmentation **C – Reweighting/resampling**: • instance weighting • stratified sampling **Add-only instance-level data (no access to existing rows)** **D – Data acquisition/generation**: • targeted collection • audit sets • synthetic data	**Full retrain** **E – Training from scratch with fairness objectives**: • constrained optimization • fairness regularizers • bandits-based learning **Fine-tuning (from a pre-trained checkpoint)** **F – Full fine-tuning (update all parameters)**: **G – Partial fine-tuning (update a subset of parameters)**: • head-only • adapters • LoRA • selective parameter updating	**White-box (model internals accessible)** **F/G – Model-only fine-tuning**: • pruning/weight masking • weight redistribution **Grey-box (only scores accessible)** **H – Output-only post-processing**: • thresholding• calibration **Black-box (only decisions accessible)** **I – Decision-layer control**: • re-ranking • decision-time constrained decisioning • decision/output rewriting	**J – Governance-only**: *(no downstream control, or no usable outputs)* monitoring, audits, and human oversight

Table 1: Four-axis resource-aware taxonomy of bias mitigation based on available resources and access rights: **Data Resources** (features vs. instances), **Training Resources (Algorithm)** (retraining vs. full/partial fine-tuning), **Model Access Resources** (white-/grey-/black-box interface regimes and operational levers), and **Human Oversight**.

bly group-dependent) importance assigned to instance i. Category C still presupposes access to the original dataset (since it reuses its instances), but it only requires control over the training/fine-tuning *data loader* (weights, sampling scheme), not the ability to edit feature values or labels. In many cases, it also requires group information a (or a proxy) to define weights or strata.

Data acquisition/generation (Category D). Category D covers strategies that *add new data* without requiring access to, or modification of, the original instances: targeted collection for underserved groups [Garimella *et al.*, 2022; Borchers *et al.*, 2022], audit/challenge set construction [Adler *et al.*, 2018], and synthetic data generation [Xu *et al.*, 2018]. Technically, these approaches expand empirical coverage by adding new samples—collected or synthesized—yielding $\mathcal{D}' = \mathcal{D} \cup \Delta\mathcal{D}$ when integration with training data is possible, or (when training data are unavailable) producing an enhanced evaluation set $\mathcal{D}^{\text{audit}} = \mathcal{D}^{\text{audit}} \cup \Delta\mathcal{D}$ to support disparity estimation and governance actions. This category is central when the original dataset is inaccessible or immutable but new data can be obtained. As with other data-level strategies, its impact relies on a downstream mechanism to make the new data actionable, i.e., incorporation into (re)training or fine-tuning when allowed.

2.2 Interventions through Training Resources

The **Training** column groups mitigation strategies that intervene in the learning procedure by updating model parameters. These methods presuppose training resources: permission to run parameter updates, access to (and control over) the training/fine-tuning loop, and sufficient computational budget. They apply when fairness can be encoded in the optimization objective, rather than enforced only downstream.

Full retraining with fairness objectives (Category E). Category E includes methods that train a model *from scratch* under fairness objectives, e.g., constrained optimization [Cotter *et al.*, 2019; Blanzeisky and Cunningham, 2022; Ge *et al.*, 2021], fairness-aware regularization or loss shaping [Kamishima *et al.*, 2012; Jiang *et al.*, 2020], and bandit/RL-inspired formulations [Hossain *et al.*, 2021; Metevier *et al.*, 2019; Wang *et al.*, 2021]. Technically, they solve a fairness-augmented learning problem over the full training pipeline, for instance $\min_\theta \ \mathcal{L}(\theta;\mathcal{D})$ s.t. $\mathcal{C}_{\text{fair}}(\theta;\mathcal{D}) \le \varepsilon$, or $\min_\theta \ \mathcal{L}(\theta;\mathcal{D}) + \lambda\,\Omega_{\text{fair}}(\theta;\mathcal{D})$. This family offers maximum flexibility , but requires permitted and affordable full retraining (data, pipeline, compute).

Full fine-tuning of a pre-trained checkpoint (Category F). Category F covers fairness-aware *full* fine-tuning, where all parameters are updated starting from a pre-trained checkpoint [Garimella *et al.*, 2022; Islam *et al.*, 2021]. In this regime, the update procedure $\mathcal{A}$ consumes an *adaptation (fine-tuning) dataset*, which we denote by $\mathcal{D}_{\text{ft}}$; in our generic notation, this is simply the dataset input to the update, i.e., $\mathcal{D}' = \mathcal{D}_{\text{ft}}$. Technically, one initializes $\theta = \theta_0$ and optimizes $\mathcal{D}_{\text{ft}}$, e.g., $\theta^\star = \arg\min_\theta \ \mathcal{L}(\theta;\mathcal{D}_{\text{ft}}) + \lambda\,\Omega_{\text{fair}}(\theta;\mathcal{D}_{\text{ft}})$, updating the entire parameter vector. Operationally, it requires (i) access to a trainable checkpoint θ_0, (ii) suitable adaptation data $\mathcal{D}_{\text{ft}}$, and (iii) permission and compute to run fine-tuning.

Partial / parameter-efficient fine-tuning (Category G). Category G captures fine-tuning where only a subset of parameters is updated (e.g., head-only [Attanasio *et al.*, 2022], adapters [Lauscher *et al.*, 2021], LoRA [Zhou *et al.*, 2025], or selective updating [Gira *et al.*, 2022]). Technically, the backbone parameters θ_0 are kept fixed, and only a small trainable component θ is updated on $\mathcal{D}_{\text{ft}}$ under a (possibly) fairness-augmented objective. They reduce compute and deployment risk relative to full fine-tuning, but still require a trainable

checkpoint and the ability to run a fine-tuning procedure on available data.

2.3 Interventions through Model Resources

The **Model** column groups mitigation strategies determined by the deployment-time access regime. Here, the key resource is not the original training pipeline, but the *interface* and *controllability* available around the deployed system. We distinguish: (i) white-box access enabling checkpoint-based parameter updates, (ii) grey-box access enabling score-level post-processing, and (iii) black-box access where only decisions are available and mitigation relies on wrappers.

White-box: trainable checkpoint access (Category F/G). White-box deployment-time mitigation instantiates the same families introduced in the Algorithms axis, but executed *post-release* from checkpoint access rather than from an end-to-end retraining pipeline. Categories F/G apply when the model is available as a trainable checkpoint and its parameters can be updated (fully or partially) even if the original training code and configuration are unavailable. This regime includes post-release parameter editing techniques such as pruning/weight masking [Xue *et al.*, 2024; Hasan *et al.*, 2024] and weight redistribution [Zayed *et al.*, 2024]. Technically, mitigation starts from the released parameters θ and produces an updated checkpoint θ' by running an available parameter-update procedure on some adaptation data $\mathcal{D}_{\text{ft}}$, i.e., $\theta' \leftarrow \textsc{Update}(\theta, \mathcal{D}_{\text{ft}})$.

Grey-box: scores/probabilities accessible (Category H). Category H covers output-only post-processing when scores/probabilities are exposed, but model parameters cannot be updated. Technically, given a scoring function $s = f(x)$, these methods apply a deterministic or group-conditional mapping g (e.g., thresholds [Iosifidis *et al.*, 2019; Menon and Williamson, 2018] or calibration mappings [Hébert-Johnson *et al.*, 2018; Pleiss *et al.*, 2017]) to produce revised decisions $\hat{y} = g(s)$ or $\hat{y} = g(s, a)$ when group metadata a is available. The model f stays fixed, while decision rule is adjusted to reduce group disparities.

Black-box with decision-layer control (Category I). Category I applies when only final decisions (labels/rankings/generations) are accessible, but an external decision layer can be introduced downstream. Technically, mitigation is implemented by a wrapper h that transforms the model output into the final action, i.e., action $= h(\hat{y})$ (or list-level transformations in ranking settings), enforcing constraints or policies after inference. This includes re-ranking [Serbos *et al.*, 2017; Kaya *et al.*, 2020], constrained decision logic [Kim *et al.*, 2018], and decision/output rewriting modules [Nabi and Shpitser, 2018; Tokpo and Calders, 2022]. The defining resource is *downstream decision control*, not access to model internals.

2.4 Human Oversight

Beyond technical levers on data, training, and deployment interfaces, many real-world settings rely on *human oversight* as an operational control layer to manage residual fairness risks. This category captures governance mechanisms that remain applicable even when no technical mitigation is feasible or permitted, and can also be layered on top of any technical path to provide monitoring, escalation, and accountability.

Black-box without decision control (Category J). Category J covers settings where technical intervention is not feasible: outputs are not accessible in a usable form, or only labels are returned and no wrapper can be introduced. Technically, mitigation is implemented through non-ML controls (monitoring signals, audit protocols, human review thresholds, escalation/recourse workflows) rather than transformations of data, parameters, or decision functions. Accordingly, the only viable measures are governance-only controls such as monitoring, audits and human oversight.

3 Resource-Constrained Mitigation Paths

Figure 1 operationalizes Table 1 as a *path-based* selection procedure: instead of classifying methods only by lifecycle stage, it makes explicit the *resource gates* that determine which interventions are reachable and which end-to-end mitigation pipelines are feasible. Overall, Fig. 1 frames AI bias mitigation as a constraint-driven navigation problem: selecting a mitigation family requires verifying that resource gates are satisfied and upstream actions can be operationalized by reachable downstream stages. The flow also encodes a *dominance relation* between paths: even if less restrictive capabilities are available, a user may deliberately follow a more restrictive path. For example, if training capability is available (Q1=*yes*), a user may bypass parameter updates and rely on deployment-time options (Q2/Q3) to reduce cost, risk, or operational disruption.

3.1 Flowchart Gates and Path Semantics

The flowchart is organized as a sequence of *resource gates* that progressively narrow which mitigation families are actionable. Gates 1 and 3 capture **training resources** (permission, infrastructure, compute), Gate 2 captures **data resources** (access and controllability), and Gate 4 captures **model access** (interface observability and decision-layer control). Training is split because it is both the *enabler* that operationalizes upstream data interventions and the *choice* of how far parameter updates can go once training is feasible.

Gate 1 ◇ : training capability (enabler). Q1 asks whether the user has authorization, infrastructure, and compute to run parameter updates (training or fine-tuning). If *no*, all paths that require consuming data through parameter updates are infeasible, and selection is driven solely by deployment-time access (Categories **H/I/J**). If *yes*, both upstream data interventions and training-stage interventions become reachable.

Gate 2 ◇ : data access and controllability. Conditioned on Q1=*yes*, Q4–Q8 refine what can be done on the data resource: (i) whether features can be changed (**A**); (ii) whether existing rows are accessible and editable (**B/C**); and (iii) whether new rows can be added when original rows are not available (**D**). These choices determine which upstream (preprocessing) families are feasible *for* training.

Gate 3 ◇ : training scope (choice). Given training capability, Q9 selects the maximum scope of parameter updates:

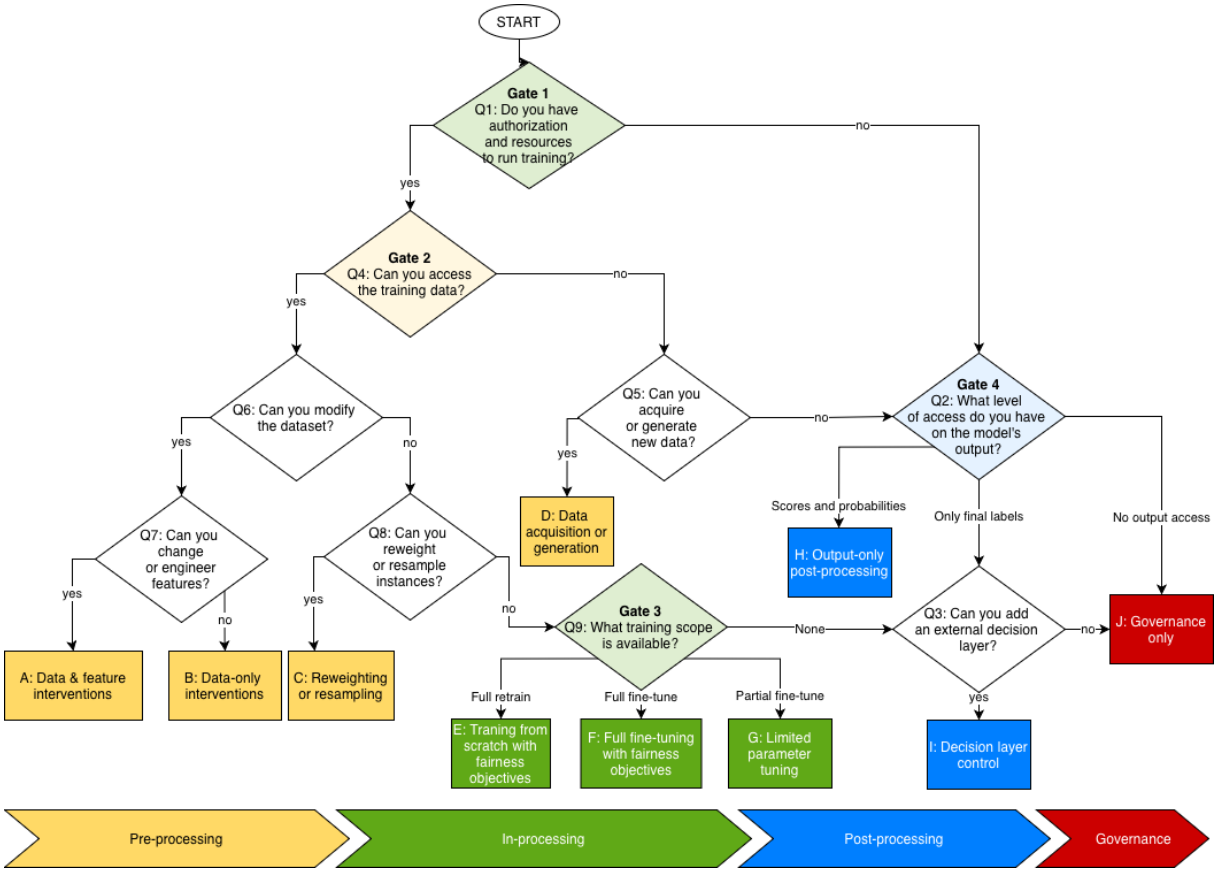

Figure 1: Resource-aware selection flow for bias mitigation. The flow screens which mitigation families are actionable given (i) training capability (permission/compute to update parameters), (ii) data access and control (feature/row editability vs. add-only), and (iii) model interface access (scores vs. decisions, with or without downstream decision control). Leaf nodes map to categories **A**–**J** in Table 1. Colors indicate the closest lifecycle placement (yellow: pre-processing; green: in-processing; blue: post-processing; red: governance).

full retraining from scratch (**E**), full fine-tuning from a checkpoint (**F**), or partial/parameter-efficient fine-tuning (**G**). This gate captures *how much* model access and compute are available once training is feasible.

Gate 4 ◇ : deployment-time interface and decision control. When Q1=*no* (or when deployment-time adjustments are desired in addition to training), Q2–Q3 determine which interface-level interventions are possible: score-level post-processing (**H**) requires access to scores/probabilities, while decision-layer control (**I**) requires permission to introduce a downstream wrapper/policy layer. If neither is possible, the remaining option is governance-only controls (**J**).

3.2 Dependencies: when a Leaf is Only a First Step

The color coding in Fig. 1 reflects lifecycle placement (yellow: pre-processing; green: in-processing; blue: post-processing; red: governance), but feasibility in practice is governed by *dependencies* between stages. In particular, some leaves are intrinsically *upstream* (they only modify inputs to training), while others are *deployment-time* controls that can be applied independently of training.

Pre-processing is upstream and must be operationalized downstream. Categories **A**–**D** act on the *data resource* (features and/or instances). By construction, they do not change the behavior of a fixed deployed model unless the modified data are subsequently *consumed* by a parameter-update execution (training or fine-tuning). Therefore selecting **A/B/C/D** implicitly requires Q1=*yes* and must be followed by *some* training execution—which may coincide with the original training pipeline, or with an explicitly fairness-aware update (**E/F/G**).

In-processing implements parameter updates and can be preceded by data interventions. Categories **E/F/G** are training-stage mitigation families that directly update model parameters and therefore constitute the main *technical endpoint* when Q1=*yes*. Their feasibility is gated by training prerequisites: (i) authorization and compute to run optimization, (ii) a trainable starting point (random initialization for **E**, checkpoint access for **F/G**), and (iii) an available training/adaptation dataset. Accordingly, yellow-to-green composition is the canonical mitigation pipeline: **A**–**D** can be used to reshape the effective data resource, and **E/F/G** operationalize that change by encoding it into updated parameters.

Post-processing is deployment-time and may follow training or stand alone. Categories **H** and **I** are post-processing

families applied at deployment time and are *not* dependent on retraining. They can be used (i) as the only technical mitigation when Q1=*no*, or (ii) as an additional alignment layer after **E/F/G** when further correction is beneficial. Their feasibility is instead determined by the deployment interface and controllability gates: **H** requires access to scores/probabilities (Q2), enabling score-to-decision mappings such as thresholding/calibration; **I** requires downstream decision control (Q3), enabling an external policy/wrapper layer (e.g., re-ranking or constrained decisioning) when only final decisions are observable. If neither score access nor decision-layer control is available, post-processing is infeasible and the technical path collapses to governance-only controls (**J**).

Governance is a persistent overlay rather than a technical dependency. Category **J** is not required by any specific technical path, but can accompany any deployment as a cross-cutting safety layer. In high-stakes settings, **J** is often adopted regardless of whether mitigation is implemented via training-stage updates (**E/F/G**) and/or post-processing controls (**H/I**).

3.3 Canonical Mitigation Paths

To make the dependency structure explicit, we represent mitigation as compositions of four blocks: **Data** (upstream data interventions), **Train** (parameter updates), **Post** (deployment-time controls), and **Gov** (human oversight). We define:

$$\textbf{Data} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\},$$
$$\textbf{Train} \in \{\text{standard training}\} \cup \textbf{Train}_{\text{fair}}, \textbf{Train}_{\text{fair}} \in \{\mathbf{E}, \mathbf{F}, \mathbf{G}\},$$
$$\textbf{Post} \in \{\mathbf{H}, \mathbf{I}\}, \qquad \textbf{Gov} \equiv \mathbf{J}.$$

Path 1: data-first (upstream mitigation). A data-first path starts by modifying the data resource and is actionable for deployed mitigation only if followed by parameter updates:

$$\textbf{Data} \rightarrow \textbf{Train} \rightarrow [\textbf{Post}] \rightarrow [\textbf{Gov}].$$

This path requires Q1=*yes* because **Data** must be *consumed* by training to affect the deployed model. Within **Data**, applicability is gated by access/control: **A** requires feature editability; **B/C** require access to existing rows; **D** is add-only and remains feasible when existing rows are not accessible/editable.

Path 2: training-first (parameter updates). A training-first path enforces mitigation directly during parameter updates, optionally with fixed data:

$$\textbf{Train}_{\text{fair}} \rightarrow [\textbf{Post}] \rightarrow [\textbf{Gov}],$$

or, if preceded by upstream data changes,

$$[\textbf{Data}] \rightarrow \textbf{Train}_{\text{fair}} \rightarrow [\textbf{Post}] \rightarrow [\textbf{Gov}].$$

This path requires Q1=*yes*. The choice among **E/F/G** is governed by the maximal feasible update scope (Q9): retraining from scratch, full fine-tuning, or partial/parameter-efficient fine-tuning.

Path 3: post-processing-first (deployment-time). When training is not available, or when deployment-time adjustments are preferred, mitigation can start at deployment:

$$\textbf{Post} \rightarrow [\textbf{Gov}].$$

Feasibility is entirely interface-driven: **H** requires score access; **I** requires decision-layer control. If neither holds, technical post-processing is infeasible and only **Gov** remains.

Governance overlay. Human oversight (**J**) is not a technical prerequisite for any path, but can accompany any deployment as a persistent safety layer that can be applied to any technical path:

$$\textbf{Gov}.$$

Composition within the Data block. When **Data** is feasible, interventions are composable: a common order is **A** (stabilize the feature schema) $\rightarrow$ **B** (row edits/augmentation) and/or **C** (reweighting/resampling), optionally complemented by **D** (add-only acquisition/generation) when new instances can be introduced.

3.4 Infeasible Paths and Why They Fail

The same resource gates in Fig. 1 also explain which sequences are infeasible or internally inconsistent.

Pre-processing without downstream parameter updates. Any pipeline that stops at **A/B/C/D** without a subsequent parameter-update execution cannot change deployed model behavior, because the new data resource is never *consumed* by retraining or fine-tuning. Such interventions may still support auditing, documentation, and governance, but they are not technical mitigation of an already-deployed model unless followed by **E/F/G** (or an equivalent training execution).

Training-stage mitigation without training capability. Categories **E/F/G** are infeasible when Q1=*no* by definition: they require authorization, infrastructure, and compute to run parameter updates.

Post-processing without the required deployment interface. Category **H** is infeasible when scores/probabilities are not observable (only final decisions are returned). Category **I** is infeasible when an external decision layer cannot be introduced (no downstream wrapper/policy control). If neither condition holds (no usable scores and no decision-layer control), the only reachable mitigation family is **J**.

Data paths with mismatched controllability assumptions. Category **A** is infeasible when the feature contract is fixed (no ability to change the input schema or representation presented to the model). Categories **B/C** presuppose access to the original rows and the ability to edit them (Category **B**) or to control their influence via weights/sampling (Category **C**); when original instances are not accessible or not modifiable, the only data-side option is Category **D** (add-only acquisition/generation), provided new data can be obtained and operationalized downstream.

4 Reframing Existing Taxonomies

To position our resource-aware taxonomy relative to prior work, we select four representative surveys covering the dominant taxonomy patterns: the classical pre-/in-/post-processing view in general ML [Caton and Haas, 2024] and in classifier-focused surveys [Hort *et al.*, 2024], a recommender-system taxonomy [Wang *et al.*, 2023], and an LLM-specific taxonomy emphasizing inference-time levers [Gallegos *et al.*, 2024]. While not exhaustive, this set captures the main ways the literature organizes mitigation and provides a compact basis for comparison. Most prior surveys classify mitigation primarily by *lifecycle placement*: *pre-processing* modifies

Our Category	[Caton and Haas, 2024]	[Hort *et al.*, 2024]	[Gallegos *et al.*, 2024]	[Wang *et al.*, 2023]
A – Feature-level data interventions	*Pre-processing:* variable blinding, relabeling and perturbation, transformation	*Pre-processing:* relabeling and perturbation, latent variables	*Pre-processing:* dataset filtering, projection-based mitigation	*Data-oriented*
B – Instance-level data interventions	*Pre-processing:* relabeling and perturbation, causal methods, adversarial learning, transformation	*Pre-processing:* relabeling and perturbation, representation learning	*Pre-processing:* data filtering, data balancing, interpolation, selective replacement, projection-based mitigation	*Data-oriented*
C – Reweighting / resampling	*Pre-processing:* reweighting, resampling, adversarial learning	*Pre-processing:* sampling	*Pre-processing:* instance reweighting	*Data-oriented*
D – Data acquisition / generation	(*)*Pre-processing:* resampling, adversarial learning	*Pre-processing:* synthetic instance generation (subset of sampling); *Post-processing:* input correction	*Pre-processing:* data generation	*Data-oriented*
E – Full retraining	*In-processing:* regularization, adversarial learning, bandits, constraint optimization	*In-processing:* regularization, constraints, adversarial learning, compositional models, adjusted learning	*In-training:* architecture and loss function modification	*Ranking*
F – Full fine-tuning	*In-processing:* adversarial learning	*In-processing:* adjusted learning; *Post-processing:* classifier correction	*Pre-processing:* instruction tuning, projection-based bias mitigation	*Ranking*
G – Partial fine-tuning	Not covered	(**)*In-processing:* hyperparameter tuning (subset of adjusted learning); *Post-processing:* classifier correction	*In-training:* selective parameter updating, filtering model parameters	*Ranking*
H – Grey-box: scores/probabilities	*Post-processing:* calibration, thresholding, transformation	*Post-processing:* output correction	*Intra-processing:* decoding strategy modification	*Re-ranking*
I – Black-box: decision-layer control	Not covered	Not covered	*Intra-processing:* modular debiasing networks; *Post-processing:* rewriting	*Re-ranking*
J – Black-box: no decision-layer control	Not covered	Not covered	Not covered	Not covered

Table 2: Crosswalk between our resource-aware leaf categories (A–J) and four representative pipeline-centric taxonomies. Each cell reports the closest macro category and method-family labels; "Not covered" indicates out-of-scope categories. (*) Not explicitly separated, typically subsumed under data interventions. (**) No explicit full/partial fine-tuning split.

data, labels, or representations before training; *in-processing* modifies learning (e.g., objectives, constraints, regularization, adversarial training); and *post-processing* transforms outputs or decisions after training [Caton and Haas, 2024; Hort *et al.*, 2024]. Domain- and model-specific surveys refine this pipeline view to match their target systems. For recommender systems, mitigation is often reorganized into *data-oriented*, *ranking*, and *re-ranking* families that mirror the multi-stage serving pipeline [Wang *et al.*, 2023]. For LLMs, surveys additionally separate *in-training* updates (pre-training and fine-tuning) from inference-time behavior control (*intra-processing*) and output rewriting/filtering (*post-processing*) [Gallegos *et al.*, 2024]. Our resource-aware view keeps these broad families, but re-indexes them by *conditions of applicability* in deployment—i.e., which resources and control regimes are actually available (data access and controllability, training capability and update scope, and deployment-time interface/decision control). This shift is useful because the pre-/in-/post-processing labels can be *too tight* in practice: they sometimes mix *timing* (when an intervention is applied) with *what is modified* (data vs. model vs. outputs). As a result, techniques that are "pre-processing" by *object* may be labeled "post-processing" by *timing* when they are executed at the end of the pipeline. [Hort *et al.*, 2024] make this explicit by grouping under "post-processing" not only output-side corrections, but also (i) *input correction* methods that modify *test-time* inputs (and are discussed as comparable to pre-processing on training data), and (ii) *classifier correction* methods that adapt an already trained model. Conversely, LLM taxonomies may label inference-time mechanisms as "intra-processing" even when operationally they amount to decision-layer control without score access (e.g., *modular debiasing networks* and output *rewriting*), which aligns with our Category I because the gating condition is downstream controllability rather than probability access. Table 2 therefore reports the closest terminology used by each survey for each of our leaf categories (A–J). When a survey does not cover a category, we mark it as *Not covered* to preserve scope differences. When a pipeline-centric label in the source survey spans multiple operational regimes (as in the "post-processing" category of [Hort *et al.*, 2024]), we prioritize the method's *operational preconditions* in the mapping: which resource is required (training data, checkpoint access, or deployment interface) and what control is available (edit vs. add-only data, full vs. partial parameter updates, score access vs. decision-layer control). In this sense, the crosswalk is not a redefinition of prior taxonomies, but a deployment-faithful framing that explains why lifecycle labels may split across multiple resource-aware leaves.

5 Conclusions

We proposed a *resource-aware* taxonomy of AI bias mitigation that complements lifecycle-based views by making explicit the *applicability conditions* induced by real resource and access constraints. The taxonomy indexes mitigation into families across data resources, training resources, model access, and governance. We further provide a screening flowchart and a crosswalk to representative pipeline-centric taxonomies, linking *where-to-intervene* labels to the *access regimes* that determine what can be implemented in practice.

References

[Adler *et al.*, 2018] Philip Adler, Casey Falk, Sorelle Friedler, Tionney Nix, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 54(1), 2018.

[Attanasio *et al.*, 2022] Giuseppe Attanasio, Debora Nozza, Dirk Hovy, and Elena Baralis. Entropy-based attention regularization frees unintended bias mitigation from lists. In *Annual Meeting of ACL*, pages 1105–1119, 2022.

[Blanzeisky and Cunningham, 2022] William Blanzeisky and Pádraig Cunningham. Using Pareto simulated annealing to address algorithmic bias in machine learning. *Know. Eng. Review*, 37, 2022.

[Borchers *et al.*, 2022] Conrad Borchers, Dalia Gala, Benjamin Gilburt, Eduard Oravkin, Wilfried Bounsi, Yuki Asano, and Hannah Kirk. Looking for a handsome carpenter! debiasing GPT-3 job advertisements. In *4th Workshop on Gender Bias in Natural Language Processing*, pages 212–224, 2022.

[Caton and Haas, 2024] Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM CSUR*, 56(7), April 2024.

[Cotter *et al.*, 2019] Andrew Cotter, Heinrich Jiang, Maya Gupta, Serena Wang, Taman Narayan, Seungil You, and Karthik Sridharan. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of ML Research*, 20, 2019.

[Ekstrand *et al.*, 2018] Michael Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. All the cool kids, how do they fit in? Popularity and demographic biases in recommender evaluation and effectiveness. In *Conference on fairness, accountability and transparency*, pages 172–186. PMLR, 2018.

[European Parliament, 2024] European Parliament. Regulation (eu) 2024/1689 (Artificial Intelligence Act), 2024.

[Feldman *et al.*, 2015] Michael Feldman, Sorelle Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[Feng *et al.*, 2019] Rui Feng, Yang Yang, Yuehan Lyu, Chenhao Tan, Yizhou Sun, and Chunping Wang. Learning fair representations via an adversarial framework, 2019.

[Gallegos *et al.*, 2024] Isabel Gallegos, Ryan Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen Ahmed. Bias and fairness in large language models: A survey. *Computational Linguistics*, 50(3):1097–1179, 2024.

[Garimella *et al.*, 2022] Aparna Garimella, Rada Mihalcea, and Akhash Amarnath. Demographic-aware language model fine-tuning as a bias mitigation technique. pages 311–319, 2022.

[Ge *et al.*, 2021] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. Towards long-term fairness in recommendation. In *14th ACM int. conference on web search and data mining*, pages 445–453, 2021.

[Gira *et al.*, 2022] Michael Gira, Ruisu Zhang, and Kangwook Lee. Debiasing pre-trained language models via efficient fine-tuning. In *Proceedings of the second workshop on language technology for equality, diversity and inclusion*, pages 59–69, 2022.

[Grgić-Hlača *et al.*, 2018] Nina Grgić-Hlača, Muhammad Bilal Zafar, Krishna Gummadi, and Adrian Weller. Beyond Distributive Fairness in Algorithmic Decision Making: Feature Selection for Procedurally Fair Learning. 32(1), 2018.

[Han *et al.*, 2022] Xudong Han, Timothy Baldwin, and Trevor Cohn. Balancing out bias: Achieving fairness through balanced training. In *Conf. on empirical methods in NLP*, pages 11335–11350, 2022.

[Hasan *et al.*, 2024] Adib Hasan, Ileana Rugina, and Alex Wang. Pruning for protection: Increasing jailbreak resistance in aligned llms without fine-tuning. *arXiv:2401.10862*, 2024.

[Hébert-Johnson *et al.*, 2018] Ursula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. Multicalibration: Calibration for the (computationally-identifiable) masses. In *Int. Conf. on ML*, pages 1939–1948. PMLR, 2018.

[Hort *et al.*, 2024] Max Hort, Zhenpeng Chen, Jie Zhang, Mark Harman, and Federica Sarro. Bias mitigation for machine learning classifiers: A comprehensive survey. *ACM Journal on Responsible Computing*, 1(2):1–52, 2024.

[Hossain *et al.*, 2021] Safwan Hossain, Evi Micha, and Nisarg Shah. Fair algorithms for multi-agent multi-armed bandits. volume 34, pages 24005–24017, 2021.

[Iosifidis *et al.*, 2019] Vasileios Iosifidis, Besnik Fetahu, and Eirini Ntoutsi. FAE: A Fairness-Aware Ensemble Framework. In *2019 IEEE international conference on big data (big data)*, pages 1375–1380. IEEE, 2019.

[Islam *et al.*, 2021] Rashidul Islam, Kamrun Naher Keya, Ziqian Zeng, Shimei Pan, and James Foulds. Debiasing career recommendations with neural fair collaborative filtering. In *Web Conference 2021*, pages 3779–3790, 2021.

[Jiang *et al.*, 2020] Ray Jiang, Aldo Pacchiano, Tom Stepleton, Heinrich Jiang, and Silvia Chiappa. Wasserstein fair classification. In *Conf. Uncertainty in AI 2020*. PMLR, 2020.

[Kamiran and Calders, 2012] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and information systems*, 33(1):1–33, 2012.

[Kamishima *et al.*, 2012] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Joint*

European conf. on ML and knowledge discovery in databases, pages 35–50. 2012.

[Kaya *et al.*, 2020] Mesut Kaya, Derek Bridge, and Nava Tintarev. Ensuring fairness in group recommendations by rank-sensitive balancing of relevance. In *RecSys 2020*, pages 101–110, 2020.

[Kim *et al.*, 2018] Michael Kim, Omer Reingold, and Guy Rothblum. Fairness through computationally-bounded awareness. volume 31, 2018.

[Kusner *et al.*, 2017] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual Fairness, 2017.

[Lauscher *et al.*, 2021] Anne Lauscher, Tobias Lueken, and Goran Glavaš. Sustainable modular debiasing of language models. *arXiv:2109.03646*, 2021.

[Limisiewicz and Mareček, 2022] Tomasz Limisiewicz and David Mareček. Don't forget about pronouns: Removing gender bias in language models without losing factual gender information. In *arXiv:2206.10744*, 2022.

[Lu *et al.*, 2020] Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural nlp. *LNCS*, 12300 LNCS:189–202, 2020.

[Luong *et al.*, 2011] Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-NN as an implementation of situation testing for discrimination discovery and prevention. In *17th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 502–510, 2011.

[Maudslay *et al.*, 2019] Rowan Hall Maudslay, Hila Gonen, Ryan Cotterell, and Simone Teufel. It's all in the name: Mitigating gender bias with name-based counterfactual data substitution. *arXiv:1909.00871*, 2019.

[Menon and Williamson, 2018] Aditya Krishna Menon and Robert Williamson. The cost of fairness in binary classification. In *Conference on Fairness, accountability and transparency*, pages 107–118. PMLR, 2018.

[Metevier *et al.*, 2019] Blossom Metevier, Stephen Giguere, Sarah Brockman, Ari Kobren, Yuriy Brun, Emma Brunskill, and Philip Thomas. Offline contextual bandits with high probability fairness guarantees. *Advances in neural information processing systems*, 32, 2019.

[Nabi and Shpitser, 2018] Razieh Nabi and Ilya Shpitser. Fair Inference on Outcomes. In *AAAI conference on artificial intelligence*, volume 32, 2018.

[Panda *et al.*, 2022] Swetasudha Panda, Ari Kobren, Michael Wick, and Qinlan Shen. Don't just clean it, proxy clean it: Mitigating bias by proxy in pre-trained models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5073–5085, 2022.

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

[Pleiss *et al.*, 2017] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Weinberger. On fairness and calibration. *Advances in neural information processing systems*, 30, 2017.

[Raffel *et al.*, 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of ML research*, 21(140):1–67, 2020.

[Serbos *et al.*, 2017] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Fairness in package-to-group recommendations. In *26th international conference on world wide web*, pages 371–379, 2017.

[Thakur *et al.*, 2023] Himanshu Thakur, Atishay Jain, Praneetha Vaddamanu, Paul Pu Liang, and Louis-Philippe Morency. Language models get a gender makeover: Mitigating gender bias with few-shot data interventions. *arXiv:2306.04597*, 2023.

[Tokpo and Calders, 2022] Ewoenam Kwaku Tokpo and Toon Calders. Text style transfer for bias mitigation using masked language modeling. *arXiv:2201.08643*, 2022.

[Wang *et al.*, 2020] Serena Wang, Wenshuo Guo, Harikrishna Narasimhan, Andrew Cotter, Maya Gupta, and Michael Jordan. Robust optimization for fairness with noisy protected groups. volume 33, pages 5190–5203, 2020.

[Wang *et al.*, 2021] Lequn Wang, Yiwei Bai, Wen Sun, and Thorsten Joachims. Fairness of exposure in stochastic bandits. In *Int. Conf. on ML*, pages 10686–10696. PMLR, 2021.

[Wang *et al.*, 2023] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. A survey on the fairness of recommender systems. *ACM Transactions on Information Systems*, 41(3):1–43, 2023.

[Xu *et al.*, 2018] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. FairGAN: Fairness-aware generative adversarial networks. In *IEEE Int. Conf. on Big Data, Big Data 2018*, IEEE Int. Conf. on Big Data, Big Data 2018, pages 570–575, 2018.

[Xue *et al.*, 2024] Yuyang Xue, Junyu Yan, Raman Dutt, Fasih Haider, Jingshuai Liu, Steven McDonagh, and Sotirios A Tsaftaris. BMFT: Achieving Fairness via Bias-Based Weight Masking Fine-Tuning. In *Workshop on Fairness of AI in Medical Imaging*, pages 98–108. 2024.

[Zayed *et al.*, 2024] Abdelrahman Zayed, Gonçalo Mordido, Samira Shabanian, Ioana Baldini, and Sarath Chandar. Fairness-Aware Structured Pruning in Transformers. 38(20):22484–22492, 2024.

[Zhou *et al.*, 2025] Songqi Zhou, Zeyuan Liu, and Benben Jiang. FairNet: Dynamic Fairness Correction without Performance Loss via Contrastive Conditional LoRA, 2025.