MoE Expert Models: A Comprehensive Analysis of Three Specialized Fairness-Aware Neural Networks

Fairness Machine Learning Project

October 9, 2025

1 Problem Formulation

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ denote a set of platforms (clients). Each platform c_i holds a private local dataset D_i that cannot be accessed by other platforms:

$$D_i = \{(x_{i,j}, s_{i,j}, y_{i,j})\}_{j=1}^{|D_i|}, \tag{1}$$

where $x_{i,j} \in \mathbb{R}^d$ is the feature vector of applicant j on platform i (including personal attributes and possibly representations of unstructured data), $s_{i,j} \in \{0, 1\}$ is a binary sensitive attribute (e.g., 0 = female, 1 = male), and $y_{i,j} \in \{0, 1\}$ is the target label indicating whether the applicant receives a positive decision.

We study fairness with respect to s and focus on both result fairness and procedural fairness. Let f_θ be a predictive model with parameters θ. For an input x, the model outputs class probabilities $p = f_\theta(x) \in [0, 1]^2$ and a predicted label $\hat{y} = \arg\max_c p_c$. Let $h_\theta(x)$ be the hidden representation and let $A_\theta(x) \in \mathbb{R}^d$ denote a normalized attribution ("attention") vector computed via a post-hoc interpreter (e.g., SHAP/IG/GradientSHAP), normalized so that $A_\theta(x) \geq 0$ and $\sum_{k=1}^{d} A_\theta(x)_k = 1$.

Result Fairness Metrics. We use Demographic Parity (DP) and Equal Opportunity (EO) gaps:

$$\mathrm{DP} = |\mathbb{P}(\hat{y} = 1 \mid s = 1) - \mathbb{P}(\hat{y} = 1 \mid s = 0)|, \tag{2}$$

$$\mathrm{EO} = |\mathbb{P}(\hat{y} = 1 \mid s = 1, y = 1) - \mathbb{P}(\hat{y} = 1 \mid s = 0, y = 1)|. \tag{3}$$

Procedural Fairness Metrics. We consider three complementary notions derived from explanations and attributions. Let $\mathrm{EQ}(x) \in \mathbb{R}_{\geq 0}$ denote a scalar explanation quality for instance x produced by a fixed explainer. Define a top-K rule: let τ_K be the K-th percentile of $\{\mathrm{EQ}(x_j)\}_{j=1}^{n}$ over the evaluation set and

$$\hat{q}_j = \mathbb{1}[\mathrm{EQ}(x_j) \geq \tau_K]. \tag{4}$$

Let $G_s^K = \{j : s_j = s,\ \hat{q}_j = 1\}$ be the indices of top-K high-quality explanations within group s.

- **ΔREF** (Ratio-based Explanation Fairness): difference in high-quality rates between groups

$$\Delta\mathrm{REF} = |\mathbb{P}(\hat{q} = 1 \mid s = 0) - \mathbb{P}(\hat{q} = 1 \mid s = 1)|. \tag{5}$$

  Smaller is better; 0 indicates equal access to high-quality explanations.

- **ΔVEF** (Value-based Explanation Fairness): difference in average explanation quality among top-K within each group

$$\Delta\mathrm{VEF} = \left| \frac{1}{|G_0^K|} \sum_{j \in G_0^K} \mathrm{EQ}(x_j) \; - \frac{1}{|G_1^K|} \sum_{j \in G_1^K} \mathrm{EQ}(x_j) \right|. \tag{6}$$

  Smaller is better; 0 indicates equal quality among the top-K across groups.

- **ATT_JSD**: Jensen–Shannon divergence between the group-level attribution distributions

$$\mathrm{ATT\_JSD} = \mathrm{JSD}(\bar{A}_0, \bar{A}_1) = \tfrac{1}{2}\,\mathrm{KL}(\bar{A}_0 \| M) + \tfrac{1}{2}\,\mathrm{KL}(\bar{A}_1 \| M), \tag{7}$$

$$M = \tfrac{1}{2}(\bar{A}_0 + \bar{A}_1). \tag{8}$$

Problem Definition. Given platforms $\mathcal{C}$ with local datasets $\{D_i\}_{i=1}^N$, learn a global model f_θ that maximizes predictive utility while mitigating result fairness gaps (DP/EO) and procedural fairness disparities (REF/VEF/ATT_JSD), under data privacy constraints (no raw data sharing across platforms). A generic objective is

$$\min_\theta \; \mathbb{E}_{(x,y)\sim D}\big[\mathcal{L}_{\mathrm{CE}}(f_\theta(x), y)\big] + \lambda_{dp}\,\mathrm{DP} + \lambda_{eo}\,\mathrm{EO} + \lambda_{ref}\,\Delta\mathrm{REF} + \lambda_{vef}\,\Delta\mathrm{VEF} + \lambda_{att}\,\mathrm{ATT\_JSD}, \tag{9}$$

where $D = \bigcup_{i=1}^N D_i$ conceptually denotes the union distribution (implemented in practice via local training/aggregation), and $\lambda_\bullet \geq 0$ control the utility–fairness trade-off. Lower values of DP/EO/REF/VEF/ATT_JSD indicate better fairness.

2 Introduction

This document provides a detailed mathematical and architectural analysis of the three expert models implemented in the Mixture of Experts (MoE) framework for fairness-aware machine learning. The system consists of three specialized neural network experts, each designed to address different aspects of fairness in machine learning: utility optimization, result-driven fairness, and procedural fairness.

3 Architecture Overview

All three experts inherit from a common `ExpertBase` class and share a fundamental Multi-Layer Perceptron (MLP) backbone. The base architecture consists of:

- Input layer: d_{input} dimensions
- Hidden layer: $d_{hidden} = 8$ dimensions (default)
- Output layer: 2 dimensions (binary classification)
- Activation: LeakyReLU
- Dropout: 0.3 (applied to hidden layer)

The forward pass of each expert follows the pattern:

$$h = \text{MLP}_{hidden}(x) \in \mathbb{R}^{d_{hidden}} \tag{10}$$

$$\ell = \text{MLP}_{output}(h) \in \mathbb{R}^2 \tag{11}$$

$$p = \text{softmax}(\ell) \in \mathbb{R}^2 \tag{12}$$

where h represents the hidden representation, ℓ the logits, and p the output probabilities.

4 Expert 1: Utility-Focused Expert

4.1 Objective

Expert 1 is designed to maximize predictive accuracy without explicit fairness constraints. It serves as a baseline utility-focused model.

4.2 Architecture

Expert 1 inherits directly from `ExpertBase` without additional components. The loss function is the standard cross-entropy loss:

$$\mathcal{L}_{E1} = \text{CrossEntropy}(p, y) = -\sum_{i=1}^{n} \sum_{c=0}^{1} y_{i,c} \log(p_{i,c}) \tag{13}$$

where $y_{i,c}$ is the one-hot encoded true label for sample i and class c, and $p_{i,c}$ is the predicted probability.

4.3 Mathematical Formulation

The complete loss computation is:

$$\mathcal{L}_{E1} = \mathcal{L}_{CE} \tag{14}$$

$$\mathcal{L}_{CE} = -\frac{1}{n} \sum_{i=1}^{n} \log(p_{i,y_i}) \tag{15}$$

where y_i is the true class label for sample i.

5 Expert 2: Result-Driven Fairness Expert

5.1 Objective

Expert 2 addresses result-driven fairness by incorporating both representation alignment and demographic parity/equal opportunity constraints.

5.2 Architecture

Expert 2 extends `ExpertBase` with two additional hyperparameters:

- λ_{rep}: Weight for representation alignment loss
- λ_{fair}: Weight for fairness loss

5.3 Mathematical Formulation

The total loss function combines three components:

$$\mathcal{L}_{E2} = \mathcal{L}_{CE} + \lambda_{rep} \cdot \mathcal{L}_{rep} + \lambda_{fair} \cdot \mathcal{L}_{fair} \tag{16}$$

5.3.1 Representation Alignment Loss

The representation alignment loss minimizes the distance between group mean embeddings:

$$\mu_0 = \frac{1}{n_0} \sum_{i:s_i=0} h_i \tag{17}$$

$$\mu_1 = \frac{1}{n_1} \sum_{i:s_i=1} h_i \tag{18}$$

$$\mathcal{L}_{rep} = ||\mu_0 - \mu_1||_2^2 \tag{19}$$

where $s_i \in \{0, 1\}$ is the sensitive attribute for sample i, and n_0, n_1 are the counts of samples in each group.

5.3.2 Fairness Loss

The fairness loss combines demographic parity (DP) and equal opportunity (EO) gaps:

$$\mathcal{L}_{fair} = \frac{\mathcal{L}_{DP} + \mathcal{L}_{EO}}{2} \tag{20}$$

Demographic Parity Gap:

$$p_0 = \frac{1}{n_0} \sum_{i:s_i=0} p_{i,1} \tag{21}$$

$$p_1 = \frac{1}{n_1} \sum_{i:s_i=1} p_{i,1} \tag{22}$$

$$\mathcal{L}_{DP} = |p_0 - p_1| \tag{23}$$

Equal Opportunity Gap:

$$p_0^{pos} = \frac{1}{n_0^{pos}} \sum_{i:s_i=0,y_i=1} p_{i,1} \tag{24}$$

$$p_1^{pos} = \frac{1}{n_1^{pos}} \sum_{i:s_i=1,y_i=1} p_{i,1} \tag{25}$$

$$\mathcal{L}_{EO} = |p_0^{pos} - p_1^{pos}| \tag{26}$$

where $p_{i,1}$ is the probability of positive class for sample i, and n_0^{pos}, n_1^{pos} are the counts of positive samples in each group.

6 Expert 3: Procedural Fairness Expert

6.1 Objective

Expert 3 addresses procedural fairness through attention alignment and adversarial debiasing techniques.

6.2 Architecture

Expert 3 extends `ExpertBase` with:

- $\lambda_{attention}$: Weight for attention fairness loss
- λ_{adv}: Weight for adversarial debiasing loss
- Gradient Reversal Layer (GRL) for adversarial training
- Two adversarial networks: one for hidden representations, one for logits

6.3 Mathematical Formulation

The total loss function is:

$$\mathcal{L}_{E3} = \mathcal{L}_{CE} + \lambda_{attention} \cdot \mathcal{L}_{attention} + \lambda_{adv} \cdot \mathcal{L}_{adv} \tag{27}$$

6.3.1 Attention Fairness Loss

The attention fairness loss uses Jensen-Shannon Divergence (JSD) to measure the difference in attention distributions between groups:

$$\mathcal{L}_{attention} = \text{JSD}(A_0, A_1) \tag{28}$$

$$\text{JSD}(A_0, A_1) = \frac{1}{2}\text{KL}(A_0||M) + \frac{1}{2}\text{KL}(A_1||M) \tag{29}$$

where $M = \frac{1}{2}(A_0 + A_1)$ is the average distribution, and A_0, A_1 are the mean attention weights for each group.

The attention weights are extracted using interpretability methods (SHAP, Integrated Gradients, LIME, or Gradient SHAP) and normalized:

$$A_{i,j} = \frac{|\text{attribution}_{i,j}|}{\sum_{k=1}^{d} |\text{attribution}_{i,k}| + \epsilon} \tag{30}$$

6.3.2 Adversarial Debiasing Loss

The adversarial loss prevents the model from learning sensitive attribute information:

$$\mathcal{L}_{adv} = \frac{\mathcal{L}_{adv}^{hidden} + \mathcal{L}_{adv}^{logits}}{2} \tag{31}$$

Hidden Representation Adversarial Loss:

$$h_{adv} = \text{GRL}(h) \tag{32}$$

$$\ell_{adv}^{hidden} = \text{MLP}_{adv}^{hidden}(h_{adv}) \tag{33}$$

$$\mathcal{L}_{adv}^{hidden} = \text{CrossEntropy}(\ell_{adv}^{hidden}, s) \tag{34}$$

Logits Adversarial Loss:

$$p_{adv} = \text{GRL}(p) \tag{35}$$

$$\ell_{adv}^{logits} = \text{MLP}_{adv}^{logits}(p_{adv}) \tag{36}$$

$$\mathcal{L}_{adv}^{logits} = \text{CrossEntropy}(\ell_{adv}^{logits}, s) \tag{37}$$

6.3.3 Gradient Reversal Layer

The GRL implements adversarial training by reversing gradients during backpropagation:

$$\text{GRL}(x) = x \quad \text{(forward pass)} \tag{38}$$

$$\frac{\partial \text{GRL}(x)}{\partial x} = -\lambda \quad \text{(backward pass)} \tag{39}$$

7 Implementation Details

7.1 MLP Backbone

The shared MLP backbone uses the following architecture:

- Linear layer: $d_{input} \rightarrow d_{hidden}$
- LeakyReLU activation
- Dropout (0.3)
- Linear layer: $d_{hidden} \rightarrow 2$

7.2 Adversarial Networks

Expert 3 includes two adversarial networks:

Hidden Adversarial Network:

- Input: d_{hidden} dimensions
- Hidden: $\max(4, d_{hidden}/2)$ dimensions
- Output: 2 dimensions (sensitive attribute prediction)

Logits Adversarial Network:

- Input: 2 dimensions (logits)
- Hidden: 8 dimensions
- Output: 2 dimensions (sensitive attribute prediction)

8 Training and Optimization

Each expert is trained independently with its specialized loss function. The training process involves:

1. Forward pass through the MLP backbone
2. Computation of specialized loss components
3. Backpropagation with appropriate gradient modifications (e.g., GRL for Expert 3)
4. Parameter updates using standard optimization algorithms

9 Gating Network Architecture

9.1 Overview

The Gating Network is responsible for selecting which expert to use for each input sample. It uses a REINFORCE-based policy gradient approach to learn optimal expert selection.

9.2 Architecture

The gating network consists of:

- Input layer: $d_{input} + 3 \cdot d_{classes} + 3 + 1$ dimensions
- Hidden layer: 16 dimensions (default)
- Output layer: 3 dimensions (one for each expert)
- Activation: ReLU
- Temperature parameter: $\tau = 1.0$ for softmax scaling

9.3 State Representation

The gating network receives an augmented state vector:

$$s = [x, p_1, p_2, p_3, \text{conf}_1, \text{conf}_2, \text{conf}_3, \text{disagree}] \tag{40}$$

where:

$$\text{conf}_i = \max(p_i) \quad \text{(confidence of expert } i\text{)} \tag{41}$$

$$\text{disagree} = \frac{1}{3}\left(\|p_1 - p_2\|_1 + \|p_1 - p_3\|_1 + \|p_2 - p_3\|_1\right) \tag{42}$$

9.4 Policy Formulation

The gating network outputs a probability distribution over experts:

$$\ell = \text{MLP}(s) \in \mathbb{R}^3 \tag{43}$$

$$\pi(a|s) = \text{softmax}(\ell/\tau) \in \mathbb{R}^3 \tag{44}$$

9.5 Action Sampling

Expert selection follows a categorical distribution:

$$a \sim \text{Categorical}(\pi(a|s)) \tag{45}$$

$$\log \pi(a|s) = \ell_a - \log \sum_{j=1}^{3} \exp(\ell_j/\tau) \tag{46}$$

9.6 Mixture Output

The final prediction is a weighted combination of expert outputs:

$$p_{final} = \sum_{i=1}^{3} \pi_i \cdot p_i \tag{47}$$

where π_i is the gating probability for expert i and p_i is the output probability from expert i.

9.7 Expert Routing

During training, each sample is routed to its selected expert:

$$p_{selected} = \begin{cases} p_1 & \text{if } a = 1 \\ p_2 & \text{if } a = 2 \\ p_3 & \text{if } a = 3 \end{cases} \tag{48}$$

During evaluation, the mixture output is used for consistent performance measurement.

10 Training Algorithm

10.1 Two-Phase Training

The MoE system uses a two-phase training approach:

10.1.1 Phase 1: Expert Pretraining

Each expert is pretrained independently for $\frac{T}{2}$ epochs:

$$\mathcal{L}_{total} = \mathcal{L}_{E1} + \mathcal{L}_{E2} + \mathcal{L}_{E3} \tag{49}$$

$$\mathcal{L}_{E1} = \text{CrossEntropy}(p_1, y) \tag{50}$$

$$\mathcal{L}_{E2} = \text{CrossEntropy}(p_2, y) + \lambda_{rep}\mathcal{L}_{rep} + \lambda_{fair}\mathcal{L}_{fair} \tag{51}$$

$$\mathcal{L}_{E3} = \text{CrossEntropy}(p_3, y) + \lambda_{attention}\mathcal{L}_{attention} + \lambda_{adv}\mathcal{L}_{adv} \tag{52}$$

10.1.2 Phase 2: Gate Training

The gating network is trained using REINFORCE for $\frac{T}{2}$ epochs.

10.2 REINFORCE Algorithm

The gate training uses policy gradient with the following components:

10.2.1 Reward Formulation

The reward function balances utility improvement and fairness improvement:

$$R = (U_2 - U_1) - (F_2 - F_1) \tag{53}$$

where:

$$U = \frac{\text{Accuracy} + \text{F1} + \text{AUC}}{3} \tag{54}$$

$$F = \frac{F_{result} + F_{procedure}}{2} \tag{55}$$

$$F_{result} = \frac{\text{DP} + \text{EO}}{2} \tag{56}$$

$$F_{procedure} = \frac{\text{REF} + \text{VEF} + \text{ATT}}{3} \tag{57}$$

10.2.2 Baseline and Advantage

A moving average baseline is used to reduce variance:

$$b_{t+1} = \alpha \cdot b_t + (1 - \alpha) \cdot R_t \tag{58}$$

$$A_t = R_t - b_t \tag{59}$$

10.2.3 Policy Loss

The policy loss includes entropy regularization and load balancing:

$$\mathcal{L}_{gate} = -\mathbb{E}[\log \pi(a|s) \cdot A] - \lambda_{entropy}\mathcal{H}(\pi) + \lambda_{lb}\mathcal{L}_{lb} \tag{60}$$

10.2.4 Entropy Regularization

The entropy term encourages exploration:

$$\mathcal{H}(\pi) = -\sum_{i=1}^{3} \pi_i \log(\pi_i + \epsilon) \tag{61}$$

where ϵ is a small constant to avoid numerical issues.

10.3 Load Balancing

To ensure balanced expert usage, a KL divergence penalty is applied:

$$q_i = \frac{\text{usage\_ma}_i + \epsilon}{\sum_{j=1}^{3}(\text{usage\_ma}_j + \epsilon)} \tag{62}$$

$$\mathcal{L}_{lb} = \text{KL}(q \| \text{Uniform}(1/3)) \tag{63}$$

where usage_ma_i is updated using exponential moving average:

$$\text{usage\_ma}_i^{(t+1)} = \beta \cdot \text{usage\_ma}_i^{(t)} + (1 - \beta) \cdot \pi_i^{(t)} \tag{64}$$

10.4 Expert Fine-tuning

During gate training, experts are periodically fine-tuned to maintain their performance:

$$\mathcal{L}_{fine-tune} = \mathcal{L}_{E1} + \mathcal{L}_{E2} + \mathcal{L}_{E3} \tag{65}$$

This fine-tuning occurs every 100 epochs to prevent expert degradation during gate training.

11 Evaluation Metrics

11.1 Utility Metrics

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \not\Vdash[\hat{y}_i = y_i] \tag{66}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{67}$$

$$\text{AUC} = \int_0^1 \text{TPR}(t) \cdot \text{FPR}'(t)\, dt \tag{68}$$

11.2 Result Fairness Metrics

$$\text{DP Gap} = |\mathbb{E}[\hat{y}|s=0] - \mathbb{E}[\hat{y}|s=1]| \tag{69}$$

$$\text{EO Gap} = |\mathbb{E}[\hat{y}|s=0, y=1] - \mathbb{E}[\hat{y}|s=1, y=1]| \tag{70}$$

11.3 Procedural Fairness Metrics

$$\text{REF} = \text{Representation Fairness} \tag{71}$$

$$\text{VEF} = \text{Value Fairness} \tag{72}$$

$$\text{ATT} = \text{Attention JSD between groups} \tag{73}$$

11.4 Final Score

The overall performance is evaluated using:

$$\text{Final Score} = U - F_{result} - F_{procedure} \tag{74}$$

12 Implementation Details

12.1 Caching System

The system supports expert and gate caching:

- Expert weights are cached after pretraining
- Gate weights are cached after training
- Cache directory: `weights/moe_experts/`

12.2 Optimization

- Expert optimizer: Adam with learning rate lr and weight decay
- Gate optimizer: Adam with learning rate $gate\_lr$ and no weight decay
- Momentum for baseline: $\alpha = 0.9$
- Momentum for usage tracking: $\beta = 0.9$

12.3 Training Hyperparameters

- Total epochs: T (default: 200)
- Expert pretraining: $T/2$ epochs
- Gate training: $T/2$ epochs
- Learning rates: $lr = 10^{-3}$, $gate\_lr = 10^{-3}$
- Regularization: $\lambda_{entropy} = 10^{-3}$, $\lambda_{lb} = 10^{-3}$

13 Conclusion

The three expert models provide complementary approaches to fairness in machine learning:

- **Expert 1**: Pure utility optimization without fairness constraints
- **Expert 2**: Result-driven fairness through representation alignment and statistical parity
- **Expert 3**: Procedural fairness through attention alignment and adversarial debiasing

The gating network uses REINFORCE to learn optimal expert selection, balancing utility and fairness improvements. This multi-expert framework allows for flexible combination of different fairness approaches, enabling the system to adapt to various fairness requirements and trade-offs between accuracy and fairness.