Report for RNF-Fairness

实验设计

模型结构

- **第一阶段：Net（MLP模型）：**
  - **输入层：** 接收138维特征`input_size=138`
  - **隐藏层：** 包含两个全连接层`fc1`和`fc2`，每层输出50维，并使用`ReLU`激活函数和`Dropout`防止过拟合
  - **输出层：** 映射到2个类别`num_classes=2`
  - **前向传播：** `fc1 → ReLU → Dropout → fc2 → ReLU → Dropout → fc3`
- **第二阶段：FC（分类头）：**
  - 用于第二阶段，接收Net的隐藏表示（50维的`fc1 → ReLU → Dropout`传播结果）作为输入
  - **前向传播：** `fc2 → ReLU → Dropout → fc3`

公平性指标

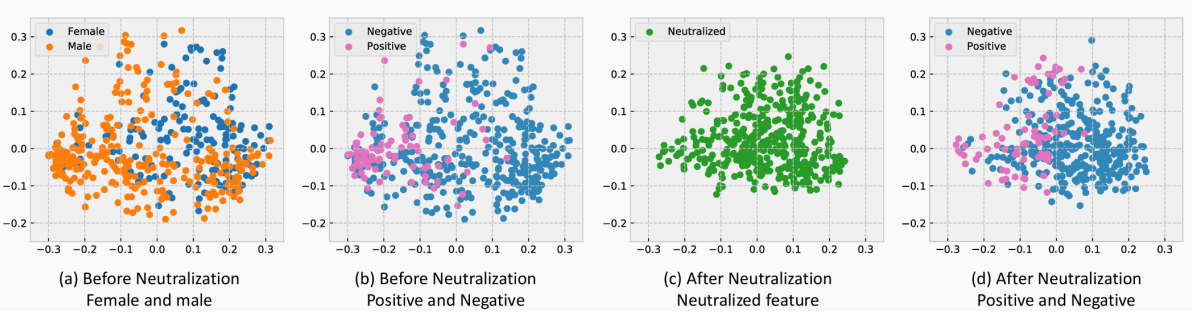
DP： 衡量敏感属性组（a=0和a=1）在正类预测概率上的绝对差异： 理想值为0，表示两个组的正类预测概率相等。

`DP = P(y=1 | a=0) - P(y=1 | a=1)`

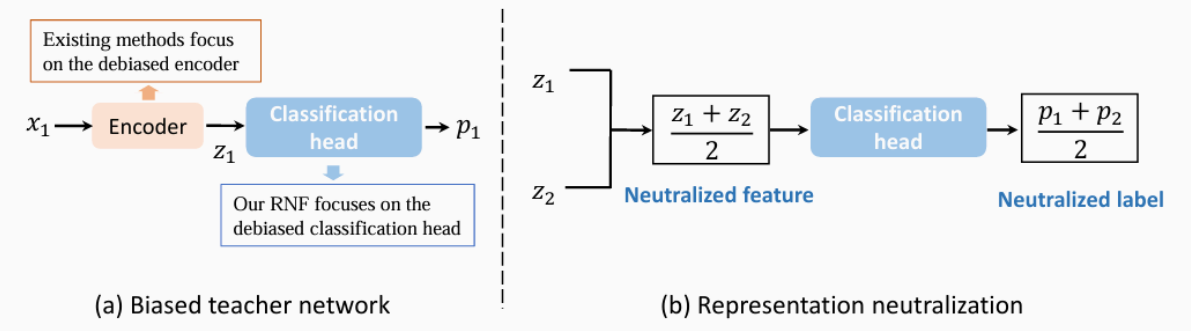
DI： 衡量敏感属性组正类预测概率的比率： 理想值为1，表示两个组的正类预测概率相同。

`DI = P(y=1 | a=1) / P(y=1 | a=0)`

`feature_neutralization`



(a) Before Neutralization Female and male　(b) Before Neutralization Positive and Negative　(c) After Neutralization Neutralized feature　(d) After Neutralization Positive and Negative



(a) Biased teacher network　(b) Representation neutralization

功能概述

第二阶段训练的核心，旨在通过对隐藏层表示和预测概率进行中和化处理，减少模型对敏感属性的依赖，从而提高公平性。其核心思想是通过**混合当前样本与具有相同标签但不同敏感属性的样本的表示，生成对敏感属性不敏感的表示。**

生成中和化表示：

- 对每个样本，函数通过线性插值生成中和化的隐藏表示和概率：
  - **表示中和化：** 以不同比例（`50%~90%`）混合当前样本的隐藏表示和匹配样本的隐藏表示，用于第二阶段的训练。
  - **概率中和化：** 以`50%`的比例混合当前样本的预测概率和匹配样本的预测概率，用于第二阶段的训练。
- 多种混合比例的目的是在第二阶段训练中引入多样性，引导分类头在不同程度的中和化表示上都能生成一致的预测，从而增强模型的公平性。

两阶段训练流程

- **第一阶段（第1-9轮）：** 训使用交叉熵损失函数进行二分类任务的训练，优化模型的预测准确性。
- **第二阶段（第10-14轮）：** 冻结`Net`模型的参数，仅训练一个独立的分类头，通过特征中和化`feature neutralization`减少对敏感属性的依赖，旨在提高模型的公平性。

stage2

1. **冻结Net模型：** 设置`Net`参数的`requires_grad=False`，确保其权重不再更新。
2. **前向传播：** 通过`Net`模型获取隐藏表示和预测概率。
3. **特征中和化：** 使用`feature_neutralization`函数，生成中和化表示（`50%~90%`）和中和化概率（`50%`）。
4. **分类头预测：** 使用FC分类头对`50%`中和化表示（`neutralization_repre_5`）进行预测，得到预测概率（`pred_neutra`）。
5. **损失计算：**
   - **MSE损失：** 使用均方误差`MSE`损失，使`FC`分类头的预测概率接近中和化概率（`neutralization_probability5`）
   - **作用**：尽量减少acc的损失，保证预测结果正确率

     We use the knowledge distillation loss. In particular, the mean squared error (MSE) loss is used as a distance-based metric to measure the similarity between model prediction and the supervision signal.

     $$\mathcal{L}_{\text{MSE}} = (\hat{y}_i - y)^2 = \{c(\frac{1}{2}z_1 + \frac{1}{2}z_2) - (\frac{1}{2}p_1 + \frac{1}{2}p_2)\}^2. \quad (1)$$

     where, c is the classification head to project representations to softmax prediction probability.

```
pred, representation = model(sent)  # Get Net's logits and
representation
pred_softmax = softmax(pred / temperature)  # Softened probabilities
# Perform feature neutralization
neutra_repre_5, neutra_repre_6, neutra_repre_7, neutra_repre_8,
neutra_repre_9, neutra_probability5 = feature_neutralization(
    representation, pred_softmax, label, sensitive_label, hidden_dim,
device
)
pred_neutra = classification_head(neutra_repre_5)  # Predict with 50%
neutralized representation
pred_neutra = softmax(pred_neutra)  # Normalize probabilities
loss = F.mse_loss(pred_neutra, neutra_probability5)  # MSE loss
```

- **正则化项**：计算FC分类头在其他中和化表示（`60%~90%`）上的预测概率与`50%`中和化表示预测概率的绝对差
- **作用**：通过诱导模型在不同中和化表示中的预测结果一致，使得模型公平性提升

Smoothing Neutralization. To further enforce the model to ignore sensitive attributes, we construct augmented training samples using a hyper-parameter λ to control the degree of neutralization of the samples $\{z_1, p_1, y\}$ and $\{z_2, p_2, y\}$. The augmented neutralized sample is given by $z = \lambda z_1 + (1-\lambda)z_2$, $\lambda \in [\frac{1}{2}, 1)$. We encourage the classification head to give similar prediction scores for the augmented and the neutralized sample (with $\lambda = \frac{1}{2}$). The regularization loss is given by:

$$\mathcal{L}_{\text{Smooth}} = \sum_{\lambda \in [\frac{1}{2}, 1)} |c(\lambda z_1 + (1-\lambda)z_2) - c(\frac{1}{2}z_1 + \frac{1}{2}z_2)|_1. \tag{3}$$

By varying λ we control the degree of sensitive information for the augmented samples. It is utilized to penalize the large changes in softmax probability when we move along the interpolation between two samples. We linearly combine the MSE loss in Eq. (1) with the regularization term as follows:

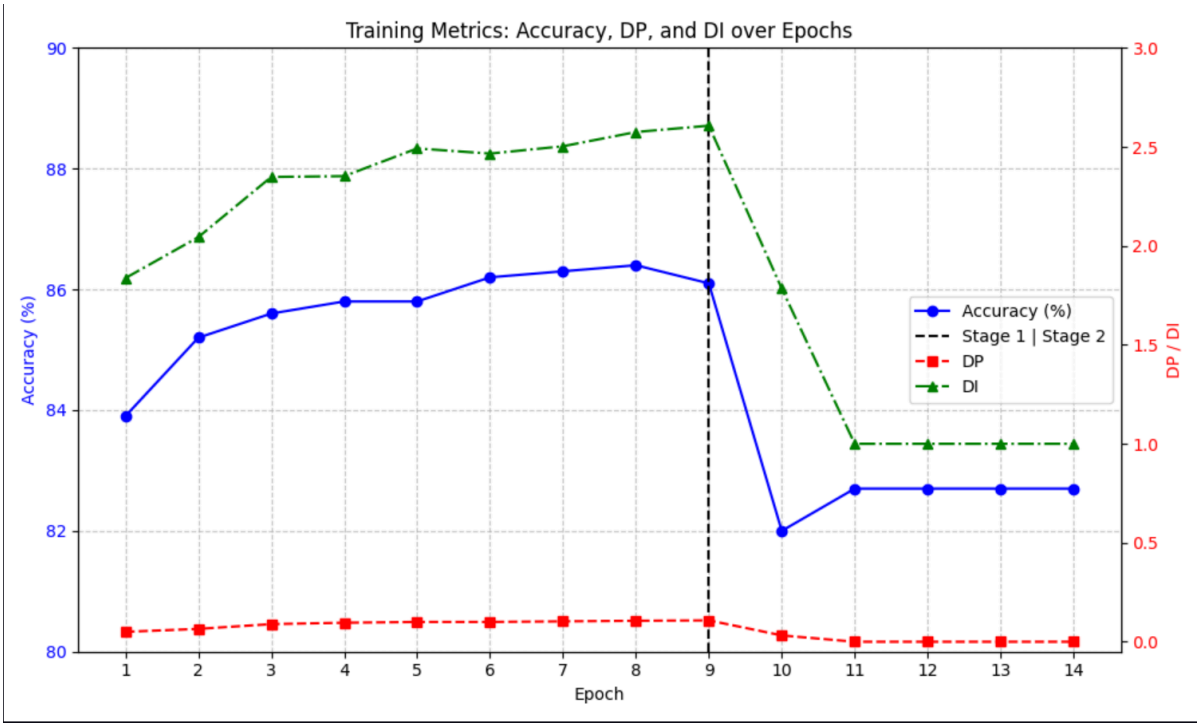
$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \alpha \mathcal{L}_{\text{Smooth}}. \tag{4}$$

```
    # Regularization term to ensure consistency across neutralized
representations
    augmented_list = [neutra_repre_6, neutra_repre_7, neutra_repre_8,
neutra_repre_9]
    # augmented_list = [neutra_repre_9]
    difference_sum = 0
    for aug_repre in augmented_list:
        pred_augmented = classification_head(aug_repre)
        pred_augmented = softmax(pred_augmented)
        difference_sum += torch.abs(pred_augmented - pred_neutra).sum()

    loss += alpha * difference_sum  # Add regularization term
```

6. **反向传播**：清零FC分类头的梯度，计算损失的梯度，更新分类头参数。
7. **评估**：使用Net的隐藏表示和FC分类头计算最终预测，收集预测标签、真实标签和敏感属性，计算损失、准确率、DP和DI。

结果分析



- **第一阶段（第1-9轮）**：专注于优化分类准确率，导致模型可能学习到与敏感属性相关的模式，从而增加偏见（DP升高，DI偏离1）。
- **第二阶段（第10-14轮）**：特征中和化显著提高了公平性（DP=0, DI=1），但同时损失了`5%`左右的准确率。

Fairness-Accuracy trade-off?

- 通过实验结果，我们通过手段去除信息中的敏感信息来提高Fairness，但往往以部分acc的损失为代价
- 我们希望在尽量不损失acc的前提条件下，尽量提升fairness，针对此想法进行了一些拓展的工作

优化措施

`AdversarialClassifier`

新增的对抗性分类器，旨在第一阶段减少 Net 模型隐藏表示中的敏感属性信息。

```
class AdversarialClassifier(nn.Module):
    def __init__(self, hidden_size):
        super(AdversarialClassifier, self).__init__()
        self.fc = nn.Linear(hidden_size, 1)  # 预测敏感属性
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out = self.fc(x)
        out = self.sigmoid(out)
        return out
```

- 输入为Net 模型的隐藏表示，网络为单层全连接，输出敏感属性的预测概率。
- 预测隐藏表示中的敏感属性信息，通过对抗性训练和梯度反转层（`GRL`），使 Net 的隐藏表示难以被`AdversarialClassifier`用来预测敏感属性，从而减少隐藏表示中的偏见信息。

- 使用`AdversarialClassifier`的预测概率和真实敏感信息标签计算`loss`，但这部分`loss`反向传播到`Net`时进行梯度反转。

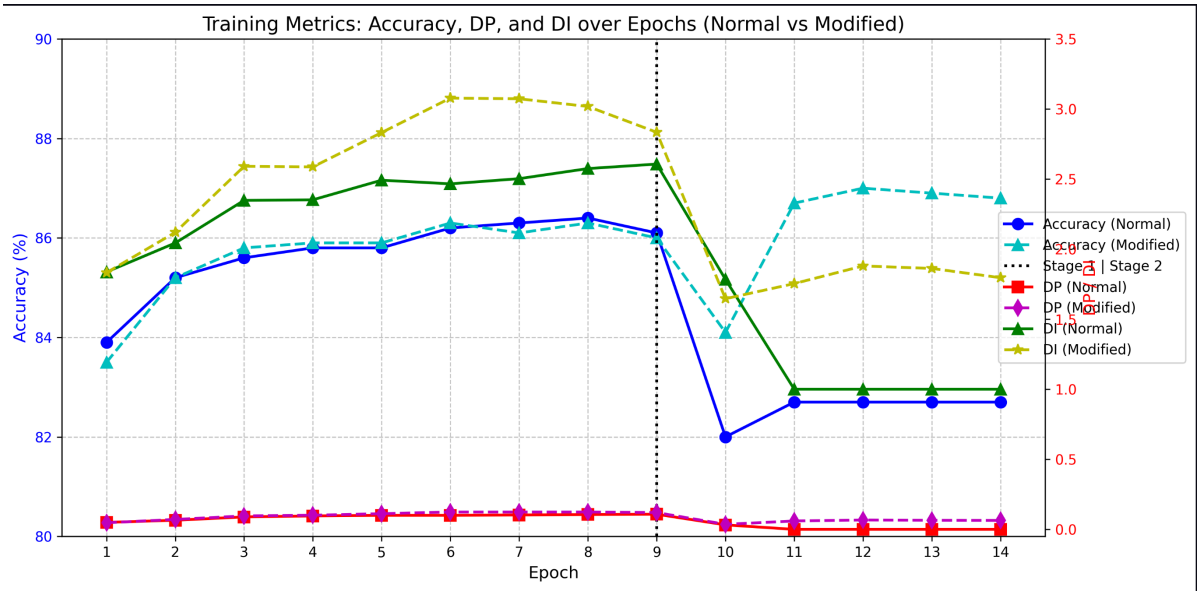
`feature_neutralization`近邻配对

- 原文中，在所有与当前样本具有相同标签但不同敏感属性的样本之中，**随机**选择一个和当前样本进行配对
- 我们考虑到，这种做法虽然模糊了敏感信息，但是很大程度上也中和掉了隐空间之中的其他特征的有效信息，可能导致acc降低
- 我们将其优化为，在所有与当前样本具有相同标签但不同敏感属性的样本之中，选择**在隐空间与当前样本欧拉距离最近**的进行配对
- 目的是去除模糊敏感信息的同时尽可能保留有效信息不被模糊掉

```
        # Select candidate group based on label and sensitive attribute
        if y == 0 and a == 0:
            candidate_group = category1_bias2 if category1_bias2 else
category1_bias1  # Prefer y=0, a=1
        elif y == 0 and a == 1:
            candidate_group = category1_bias1 if category1_bias1 else
category1_bias2  # Prefer y=0, a=0
        elif y == 1 and a == 0:
            candidate_group = category2_bias2 if category2_bias2 else
category2_bias1  # Prefer y=1, a=1
        elif y == 1 and a == 1:
            candidate_group = category2_bias1 if category2_bias1 else
category2_bias2  # Prefer y=1, a=0


        # Compute Euclidean distances to all candidates
        distances = [torch.norm(current_r - candidate[0]).item() for candidate in
candidate_group]
        # Select candidate with minimum distance
        min_distance_idx = np.argmin(distances)
        neutralization_sample = candidate_group[min_distance_idx]
```

实验结果



- 公平性 `fairness` 指标的优化力度减弱了，无法像之前做到绝对公平（`DP=0 DI=1`），可能是保留了和敏感信息相关的特征信息
- 在部分优化 `fairness` 指标的同时，`accuracy` 实现了回升，没有显著的准确率损失

参考文章

[1707.00075] Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations

[2106.12674] Fairness via Representation Neutralization