Mitigating Unfairness in Deep Learning

Guest Lecture – CS 594: Responsible Data Science and Algorithmic Fairness

Vishnu Dasu, Ph.D. student in Computer Sciences

PennState

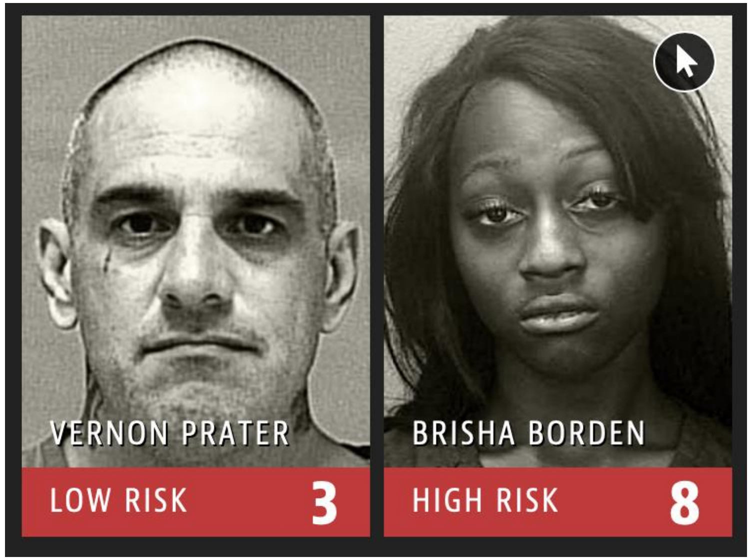
About Me

- Second year PhD student at Penn State
    - Advised by Prof. Gary Tan
- 2 years industry experience as a security researcher
- Research Interests: Trustworthy AI, Security & Privacy, Applied Cryptography
- Website: https://vdasu.github.io
- Contact: vdasu@psu.edu

Fairness in ML Applications

Tax Auditing

Amazon Hiring[1]



COMPAS[2]

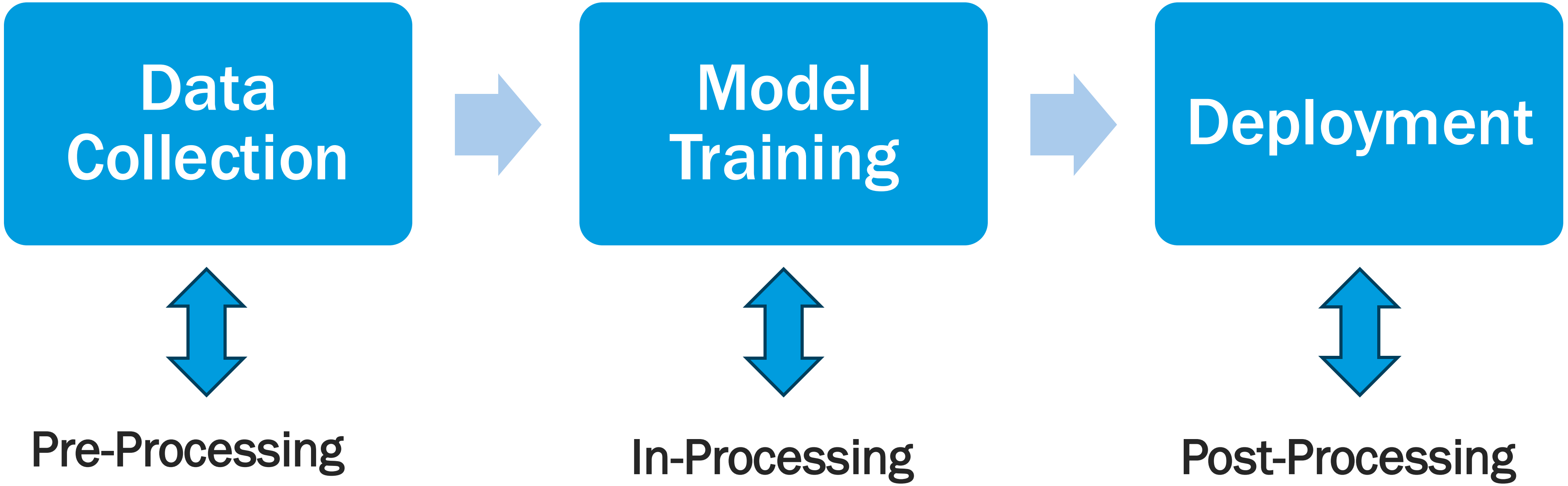
[1]https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
[2]https://www.datanami.com/2018/10/16/do-amazons-biased-algorithms-spell-the-end-of-ai-in-hiring/

ML Application Workflow

Data Collection → Model Training → Deployment

PennState

Repairing Unfairness in ML



Repairing Unfairness in ML

- Data collection and training is an expensive and difficult process
- Modifying them is often infeasible
  - Terabytes of data are scraped to train LLMs
  - Cleaning and analyzing the data is impossible
- Can we fix fairness issues after model training is complete?

Repairing Unfairness in ML

- Data collection and training is an expensive and difficult process
- Modifying them is often infeasible
  - Terabytes of data are scraped to train LLMs
  - Cleaning and analyzing the data is impossible
- Can we fix fairness issues after model training is complete?

Yes!

PennState

Overview

- Fairness in Deep Neural Networks (DNNs)
  - *NeuFair: Neural Network Fairness Repair with Dropout* [ISSTA '24]
- Fairness in Large Language Models (LLMs)
  - *Attention Pruning: Automated Fairness Repair of Language Models via Surrogate Simulated Annealing* [ICSE '26]

NeuFair: Neural Network Fairness Repair with Dropout

Vishnu Asutosh Dasu, Ashish Kumar, Saeid Tizpaz-Niari, Gang Tan

ISSTA '24

Problem Statement

Can we repair unfairness in a trained DNN without modifying dataset or re-training?

Key Observation and Idea

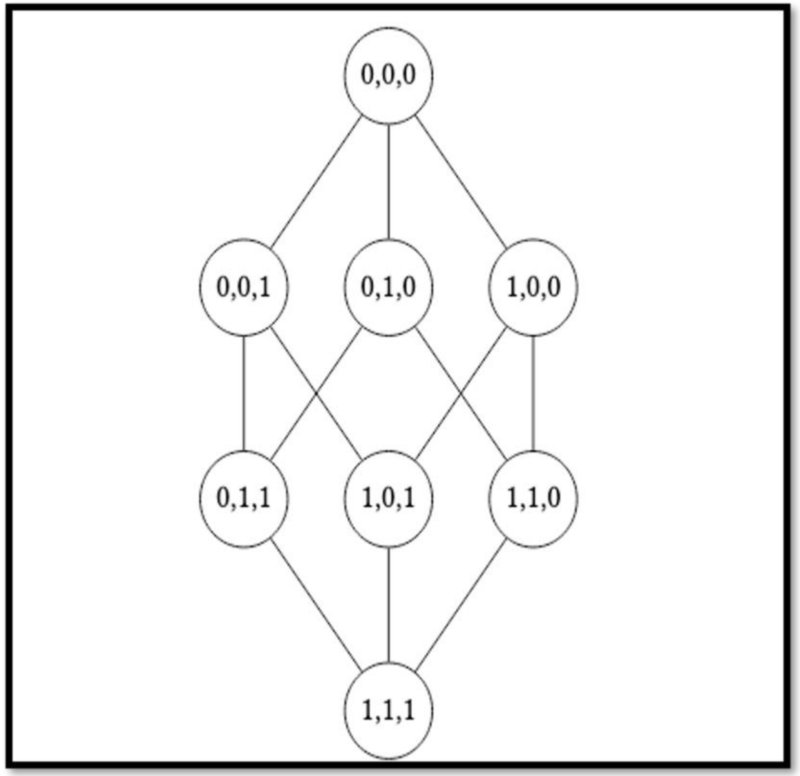
Observation: Subset of neurons disparately affects fairness

Idea: Dropping these neurons after training can improve fairness with negligible loss in utility

Challenges

- Exponentially large search space

- DNN with N neurons has 2^N possible subsets

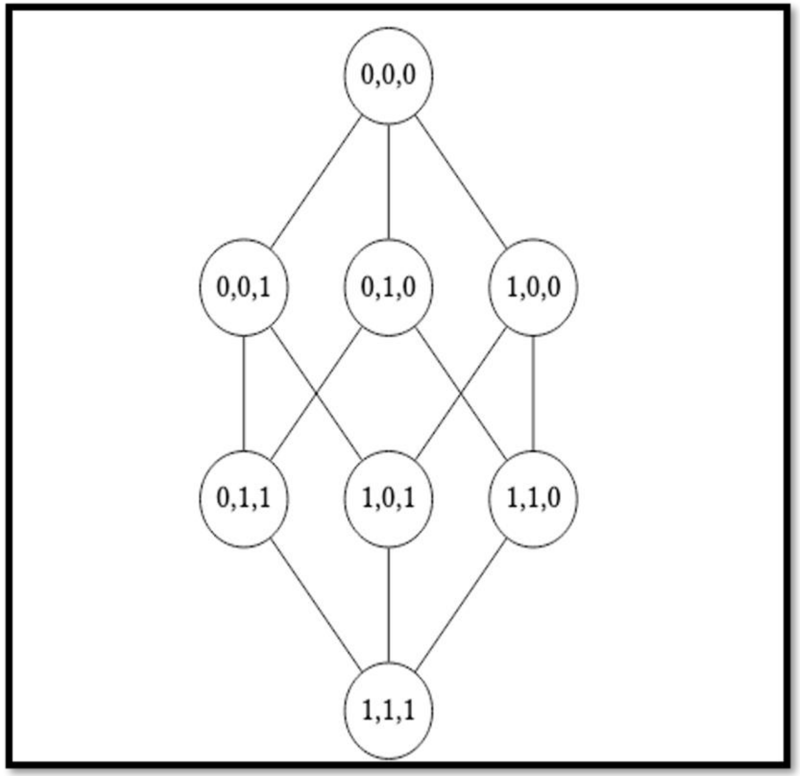


Search space with N = 3

Solution

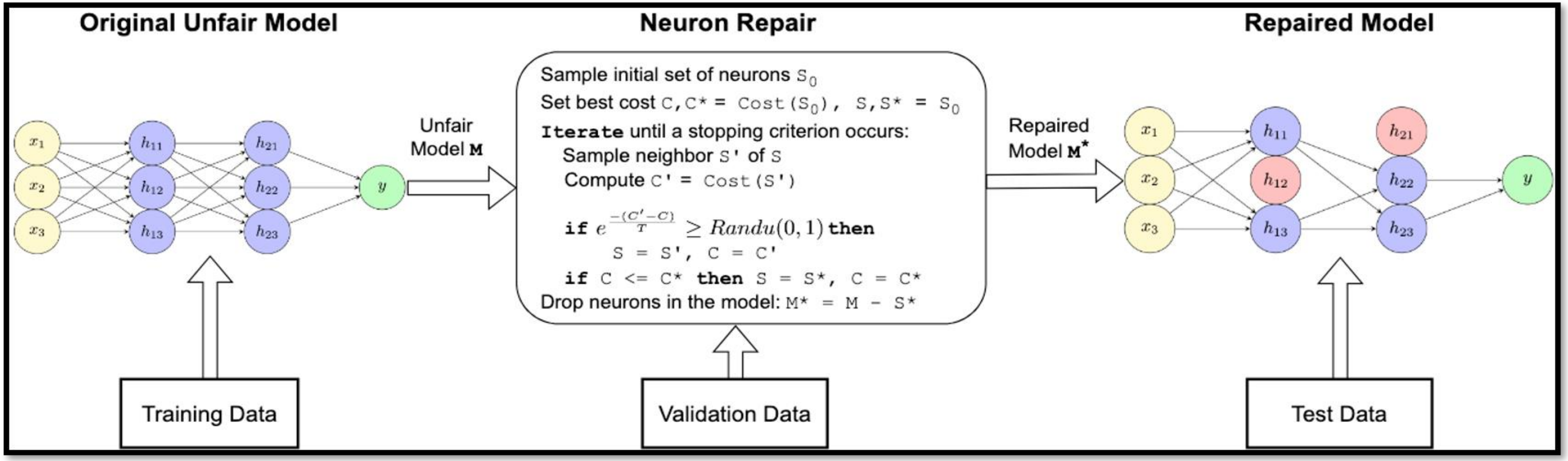
- Use randomized algorithms to explore search space

- Two strategies:
  - Simulated Annealing (SA)
  - Random Walk (RW)



Search space with N = 3

Overview of NeuFair



Fairness and Utility Definitions

- Fairness:
  - Equalized Odds Difference (EOD) is maximum of difference between true and false positive rates across protected groups

$$\boxed{EOD := \max(|TPR_A - TPR_B|, |FPR_A - FPR_B|)}$$

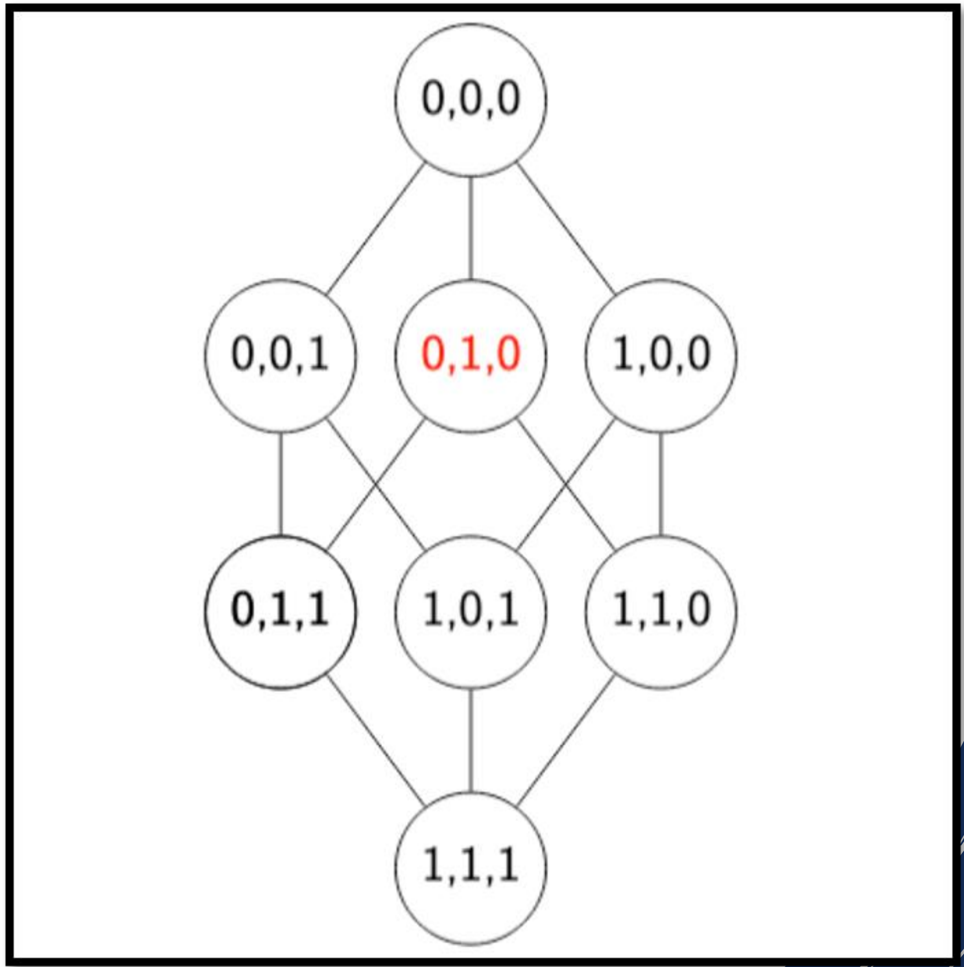
- Model Utility:
  - F1 Score and Accuracy

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}$$

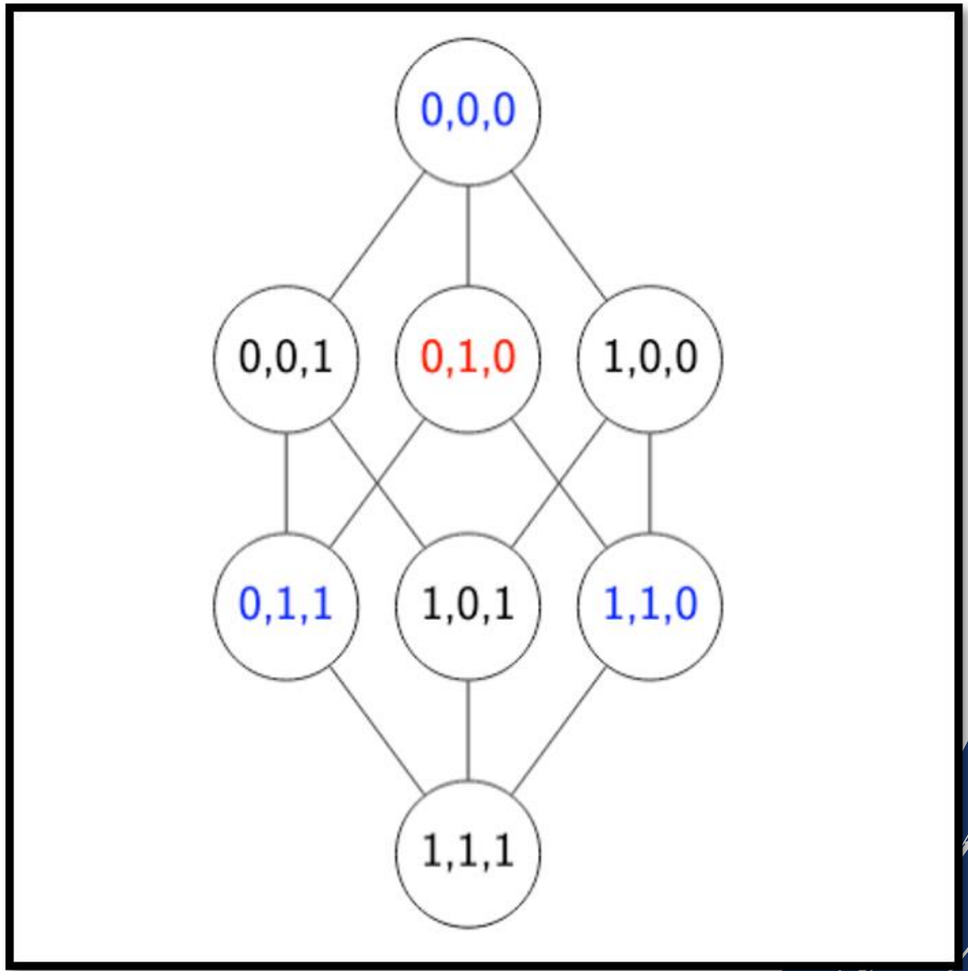
Methodology

1. Compute cost of current state



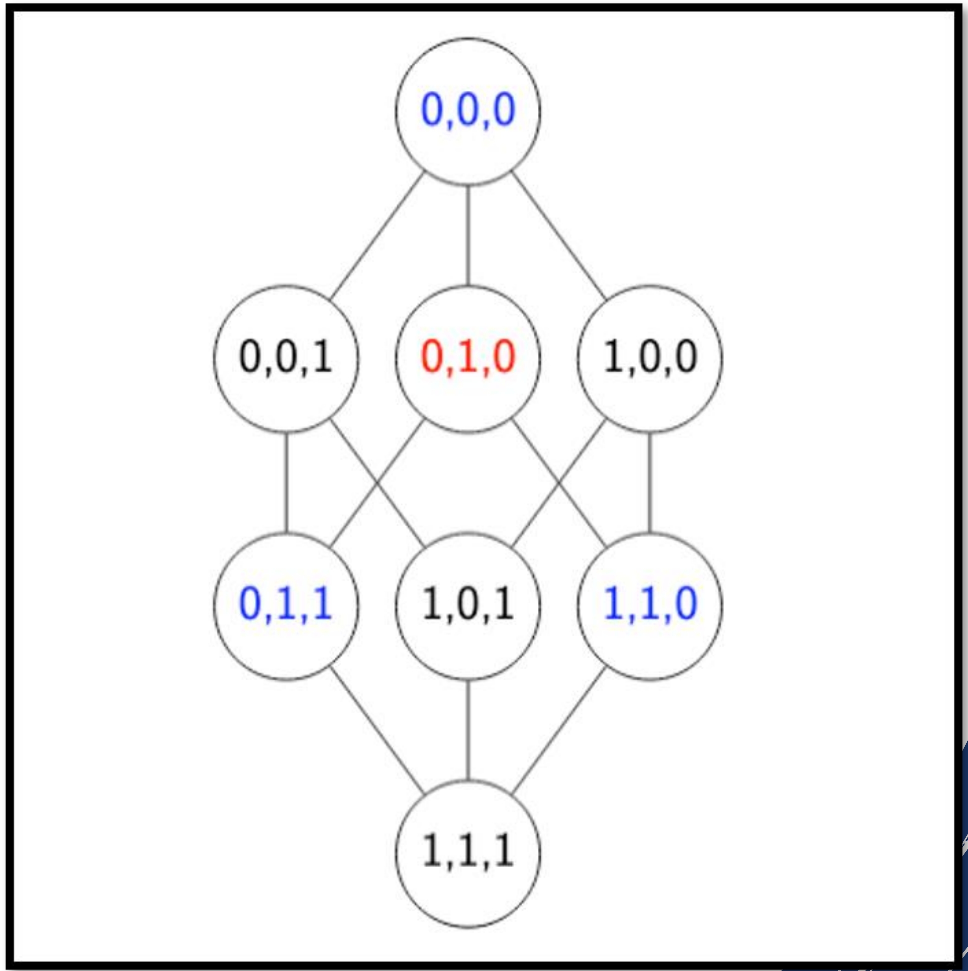
Methodology

1. Compute cost of current state
2. Sample neighbor and compute cost



Methodology

1. Compute cost of current state
2. Sample neighbor and compute cost
3. If new cost is better, accept state.
   Else, accept with some probability



Cost Function

- Minimize EOD and maintain F1 score

1. Compute cost of current state
2. Sample a neighbor state and compute cost
3. If new cost is better, accept state. Else, accept with some probability

PennState

Cost Function

- Minimize EOD and maintain F1 score
- Linear combination of EOD and penalty for poor F1

$$\text{cost}(s) := EOD_s + p \cdot EOD_{s_0} \cdot \mathbb{1}(F1_s < tF1_{s_0})$$

- Fair states are encouraged
- States with low F1 are penalized

1. Compute cost of current state
2. Sample a neighbor state and compute cost
3. If new cost is better, accept state. Else, accept with some probability

PennState

Neighbors of a State

- State stores a subset of neurons that need to dropped
- Neighbors of a state are all states that add or remove a neuron to its subset
- Hamming Difference is 1
- Pick random neighbor

1. Compute cost of current state
2. Sample a neighbor state and compute cost
3. If new cost is better, accept state. Else, accept with some probability

PennState

Acceptance Criteria

- Always accept good states
- For bad states, two strategies:
  - Simulated Annealing (SA): Accept with probability $p = e^{-(\Delta cost/temperature)}$
  - Random Walk (RW): Always accept state i.e. probability $p = 1$
- SA balances exploration/exploitation
- RW always explores

1. Compute cost of current state
2. Sample a neighbor state and compute cost
3. If new cost is better, accept state. Else, accept with some probability

NeuFair

Algorithm 2: NeuFair to mitigate unfairness in trained neural networks

Input: Unfair neural network $\mathcal{M}$, Penalty multiplier p, Threshold multiplier t, Minimum and maximum number of neurons to drop $[n_l, n_u]$, Algorithm Type *alg_type*, Time Limit *time_limit*

Output: Repaired neural network $\mathcal{M}_\star$, Desirable state $s_\star$, Best cost $cost_\star$

$s \leftarrow \texttt{random\_state}(\mathcal{M}, n_l, n_u)$
$s_\star, start\_time \leftarrow \phi, \text{curr\_time}()$
$cost \leftarrow \texttt{compute\_cost}(\mathcal{M}, s, p, t)$
$cost_\star \leftarrow \texttt{compute\_cost}(\mathcal{M}, s_\star, p, t)$
$T_0 \leftarrow \texttt{estimate\_temperature}(\mathcal{M}, s)$
while $curr\_time() - start\_time \leq time\_limit$ **do**
  $T \leftarrow \texttt{update\_temperature}(T_0, curr\_time())$
  $s_i \leftarrow \texttt{generate\_state}(s, n_l, n_u)$
  $cost_i \leftarrow \texttt{compute\_cost}(\mathcal{M}, s_i, p, t)$
  $\Delta E \leftarrow cost_i - cost$
  if $\Delta E \leq 0$ **then**
    $cost \leftarrow cost_i$
    $s \leftarrow s_i$
  else if $(alg\_type == \text{RW}) \lor (alg\_type == \text{SA} \land e^{-\Delta E/T} \geq Uniform(0,1))$ **then**
    $cost \leftarrow cost_i$
    $s \leftarrow s_i$
  if $cost \leq cost_\star$ **then**
    $cost_\star \leftarrow cost_i$
    $s_\star \leftarrow s_i$
$\mathcal{M}_\star \leftarrow \mathcal{M} \setminus s_\star$
return $\mathcal{M}_\star, s_\star, cost_\star$

Repeat Until Time Limit

Initialization

Compute cost of current and new state

Transition to new state

Experiments and Results

- RQ1: How effective are randomized algorithms in repairing unfairness?
- RQ2: Can dropout improve fairness and utility together?
- RQ3: What are the design considerations of search algorithms?
- RQ4: How does NeuFair perform against SOTA post-processing algorithms?

PennState

Experiments and Results

- Seven settings with five different datasets
- Repeat with 10 random seeds each, 1 hr timeout
- Penalty multiplier = 3.0
- F1 threshold = 0.98

$$\boxed{C(s) = EOD_s + 3.0 \cdot EOD_{s_0} \cdot \mathbb{1}(F1_s < (0.98 \cdot F1_{s_0}))}$$

- Restricted search space:
  - Min. 2 neurons
  - Max. 20-40% of DNN
- 60%-20%-20% Train-Validation-Test split

RQ1: Effective of randomized algorithms

Datasets	Original Model			Simulated Annealing		
	EOD	*F1*	*Accuracy*	*EOD*	*F1*	*Accuracy*
Adult (Sex)	$11.639\% \pm 2.326$	0.667 ± 0.008	0.851 ± 0.003	$7.259\% \pm 1.697$	0.652 ± 0.01	0.849 ± 0.004
Adult (Race)	$8.251\% \pm 3.195$			$4.976\% \pm 1.816$	0.656 ± 0.008	0.849 ± 0.004
COMPAS (Sex)	$2.522\% \pm 0.817$	0.967 ± 0.004	0.969 ± 0.004	$2.921\% \pm 1.446$	0.954 ± 0.08	0.957 ± 0.008
COMPAS (Race)	$2.96\% \pm 1.088$			$2.239\% \pm 1.003$	0.954 ± 0.005	0.957 ± 0.004
Bank	$14.665\% \pm 2.114$	0.553 ± 0.004	0.84 ± 0.003	$7.257\% \pm 3.533$	0.537 ± 0.014	0.888 ± 0.01
Default	$8.962\% \pm 1.772$	0.53 ± 0.006	0.769 ± 0.007	$2.749\% \pm 0.827$	0.519 ± 0.006	0.79 ± 0.015
MEPS16	$20.641\% \pm 2.527$	0.533 ± 0.01	0.788 ± 0.009	$8.426\% \pm 2.311$	0.507 ± 0.02	0.853 ± 0.005

Simulated Annealing on Test split

RQ1: Effective of randomized algorithms

Datasets	Original Model			Random Walk		
	EOD	*F1*	*Accuracy*	*EOD*	*F1*	*Accuracy*
Adult (Sex)	$11.639\% \pm 2.326$	0.667 ± 0.008	0.851 ± 0.003	$7.358\% \pm 1.063$	0.652 ± 0.01	0.849 ± 0.004
Adult (Race)	$8.251\% \pm 3.195$			$4.785\% \pm 2.085$	0.658 ± 0.009	0.849 ± 0.003
COMPAS (Sex)	$2.522\% \pm 0.817$	0.967 ± 0.004	0.969 ± 0.004	$2.233\% \pm 1.022$	0.954 ± 0.007	0.957 ± 0.006
COMPAS (Race)	2.96%1.088			$2.159\% \pm 1.07$	0.955 ± 0.004	0.958 ± 0.004
Bank	$14.665\% \pm 2.114$	0.553 ± 0.004	0.84 ± 0.003	$7.595\% \pm 2.733$	0.548 ± 0.008	0.881 ± 0.008
Default	$8.962\% \pm 1.772$	0.53 ± 0.006	0.769 ± 0.007	$3.124\% \pm 0.937$	0.523 ± 0.005	0.79 ± 0.015
MEPS16	$20.641\% \pm 2.527$	0.533 ± 0.01	0.788 ± 0.009	$9.86\% \pm 2.623$	0.513 ± 0.018	0.851 ± 0.007

Random Walk on Test split

RQ2: Improve Fairness and Utility together

Datasets	Original Model			Simulated Annealing		
	EOD	*F1*	*Accuracy*	*EOD*	*F1*	*Accuracy*
Adult (Sex)	$11.639\% \pm 2.326$	0.667 ± 0.008	0.851 ± 0.003	$7.259\% \pm 1.697$	0.652 ± 0.01	0.849 ± 0.004
Adult (Race)	$8.251\% \pm 3.195$			$4.976\% \pm 1.816$	0.656 ± 0.008	0.849 ± 0.004
COMPAS (Sex)	$2.522\% \pm 0.817$	0.967 ± 0.004	0.969 ± 0.004	$2.921\% \pm 1.446$	0.954 ± 0.08	0.957 ± 0.008
COMPAS (Race)	$2.96\% \pm 1.088$			$2.239\% \pm 1.003$	0.954 ± 0.005	0.957 ± 0.004
Bank	$14.665\% \pm 2.114$	0.553 ± 0.004	0.84 ± 0.003	$7.257\% \pm 3.533$	0.537 ± 0.014	0.888 ± 0.01
Default	$8.962\% \pm 1.772$	0.53 ± 0.006	0.769 ± 0.007	$2.749\% \pm 0.827$	0.519 ± 0.006	0.79 ± 0.015
MEPS16	$20.641\% \pm 2.527$	0.533 ± 0.01	0.788 ± 0.009	$8.426\% \pm 2.311$	0.507 ± 0.02	0.853 ± 0.005

Simulated Annealing on Test split

PennState

RQ2: Improve Fairness and Utility together

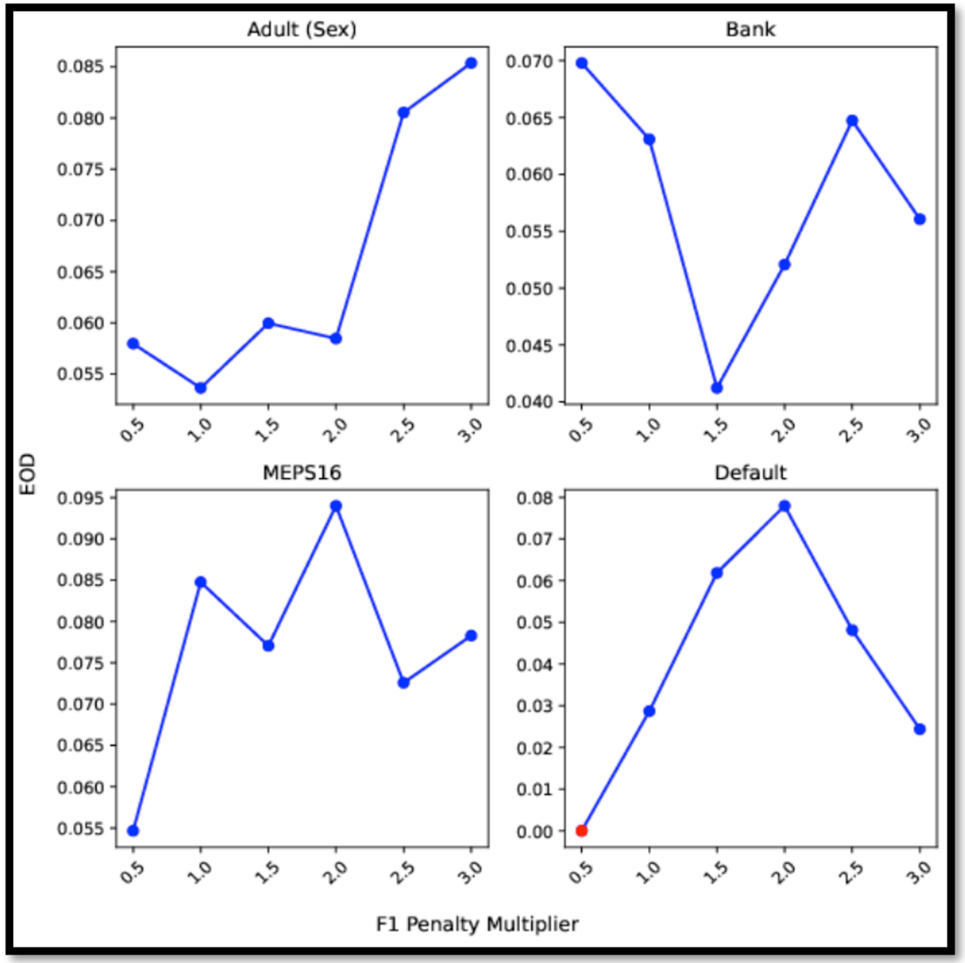
Datasets	Original Model			Random Walk		
	EOD	*F1*	*Accuracy*	*EOD*	*F1*	*Accuracy*
Adult (Sex)	$11.639\% \pm 2.326$	0.667 ± 0.008	0.851 ± 0.003	$7.358\% \pm 1.063$	0.652 ± 0.01	0.849 ± 0.004
Adult (Race)	$8.251\% \pm 3.195$			$4.785\% \pm 2.085$	0.658 ± 0.009	0.849 ± 0.003
COMPAS (Sex)	$2.522\% \pm 0.817$	0.967 ± 0.004	0.969 ± 0.004	$2.233\% \pm 1.022$	0.954 ± 0.007	0.957 ± 0.006
COMPAS (Race)	2.96%1.088			$2.159\% \pm 1.07$	0.955 ± 0.004	0.958 ± 0.004
Bank	$14.665\% \pm 2.114$	0.553 ± 0.004	0.84 ± 0.003	$7.595\% \pm 2.733$	0.548 ± 0.008	0.881 ± 0.008
Default	$8.962\% \pm 1.772$	0.53 ± 0.006	0.769 ± 0.007	$3.124\% \pm 0.937$	0.523 ± 0.005	0.79 ± 0.015
MEPS16	$20.641\% \pm 2.527$	0.533 ± 0.01	0.788 ± 0.009	$9.86\% \pm 2.623$	0.513 ± 0.018	0.851 ± 0.007

Random Walk on Test split

RQ3: Design Considerations of Search Algorithms

- Increasing F1 threshold multiplier may decrease fairness
- Increasing search space (min/max no: of neurons) and may increase fairness
- Increasing time out may increase fairness
- F1 penalty multiplier has non-trivial effect

RQ3: Design Considerations of Search Algorithms



Effect of Penalty Multiplier on EOD

$$\text{cost}(s) := EOD_s + p \cdot EOD_{s_0} \cdot \mathbb{1}(F1_s < tF1_{s_0})$$

SA acceptance probability $p = e^{-(\Delta cost / temperature)}$

RQ4: Comparison to SOTA post-processing algorithms

Datasets	EOD		
	Original	DICE	NeuFair (SA)
Adult (Sex)	$11.639\% \pm 2.326$	$10.453\% \pm 2.266$	$7.259\% \pm 1.697$
Adult (Race)	$8.251\% \pm 2.236$	$8.092\% \pm 3.253$	$4.976\% \pm 1.816$
COMPAS (Sex)	$2.522\% \pm 0.817$	$3.229\% \pm 0.774$	$2.921\% \pm 1.446$
COMPAS (Race)	$2.96\% \pm 1.088$	$2.964\% \pm 1.088$	$2.239\% \pm 1.003$
Bank	$14.665\% \pm 2.114$	$12.205\% \pm 2.731$	$7.257\% \pm 3.533$
Default	$8.962\% \pm 1.772$	$5.845\% \pm 1.816$	$2.749\% \pm 0.827$
MEPS16	$20.641\% \pm 2.527$	$19.204\% \pm 2.592$	$8.426\% \pm 2.311$

NeuFair compared to DICE (ICSE 23')

Conclusion

- A subset of neurons disparately affects fairness
- Deterministic dropout after training can improve fairness without loss in utility
- Randomized algorithms are effective for neuron dropout

Attention Pruning: Automated Fairness Repair of Language Models via Surrogate Simulated Annealing

Vishnu Asutosh Dasu, Md Rafi Rashid, Vipul Gupta, Saeid Tizpaz-Niari, Gang Tan

ICSE '26

Problem Statement

Can we extend the idea of *NeuFair* to repair fairness in pre-trained LLMs?

Challenges

1. <u>Size</u>: LLMs contain billions of parameters
    - *NeuFair* experiments with small DNNs with 100s of neurons
2. <u>Time</u>: LLMs have high-inference time
    - Exploring each state takes several minutes
    - SA needs to explore 1000s of states in the exponential search space

Fairness Definition

- Bias *HolisticBias* dataset:
  - A collection of prompts belonging to different groups *G* (e.g. race, gender, sex, etc.)
  - Each group *G* contains sub-groups *g* (gender: man, woman, transgender, etc.)
  - Bias is measured as differences in model behavior (e.g., toxicity) across sub-groups within the same group

$$\text{bias}_\Theta(G) = \sum_{g \in G} \left| T_G - T_g \right|$$

$$\text{where} \quad T_g = \frac{1}{|D_g|} \sum_{x \in D_g} \text{tox}_\Theta(x)$$

$$\text{and} \quad T_G = \frac{1}{|G|} \sum_{g \in G} T_g$$

PennState

Utility Definition

- *Perplexity (PPL)* measures how well a model Θ predicts a sequence **y**
- It is defined as the exponential of the average negative log-likelihood the model assigns to each token
- We use the WikiText-2 dataset to measure perplexity

$$PPL_{\Theta}(\mathbf{y}) = \exp\left(-\frac{1}{n}\sum_{i=1}^{n}\log(\Theta(y_i|y_1,\ldots,y_{i-1}))\right)$$

PennState

Dealing with the Size of LLMs

PennState

Attention in LLMs

- Scaled dot-product attention (SDPA) operates on Query, Key, and Value vectors

$$\boxed{\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V}$$

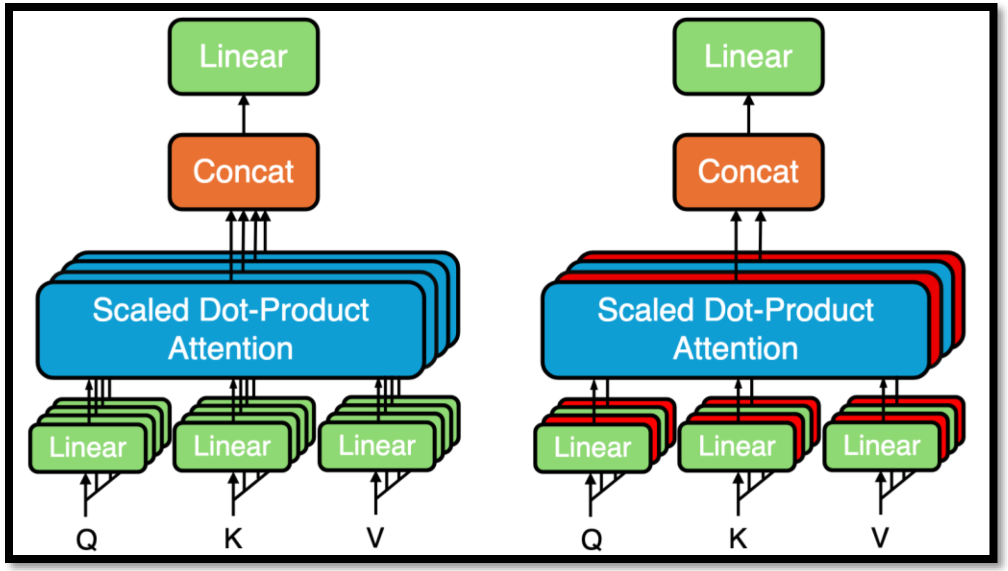
- Transformer-based LLMs consist of stacked blocks of Multi Head Attention (MHA) in each block

$$\boxed{MultiHead(Q, K, V) = Concat(head_1, \ldots, head_n)W^O}$$

$$\boxed{head_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)}$$

Pruning Attention Heads

- Several works have shown that LLMs are over-parameterized
- Pruning a subset of head of attention heads can improve efficiency with minimal loss in performance



Before Pruning After Pruning

Pruning Attention Heads

- Can we prune attention heads to improve fairness?
  - Less granular than infeasible neuron-level pruning and faster

Pruning Attention Heads

- Can we prune attention heads to improve fairness?
  - Less granular than infeasible neuron-level pruning and faster

- *FASP Algorithm:* Fairness-Aware Structured Pruning in Transformers [Zayed et. al, AAAI 2024][1]

[1]https://ojs.aaai.org/index.php/AAAI/article/view/30256/32240

The FASP algorithm

1. Prune each attention head one at a time while keeping others intact

The FASP algorithm

1. Prune each attention head one at a time while keeping others intact
2. Calculate the change in fairness and utility after pruning

The FASP algorithm

1. Prune each attention head one at a time while keeping others intact
2. Calculate the change in fairness and utility after pruning
3. Rank attention heads by magnitude of change

PennState

The FASP algorithm

1. Prune each attention head one at a time while keeping others intact
2. Calculate the change in fairness and utility after pruning
3. Rank attention heads by magnitude of change
4. Define *critical* heads: Heads that are important for utility

The FASP algorithm

1. Prune each attention head one at a time while keeping others intact
2. Calculate the change in fairness and utility after pruning
3. Rank attention heads by magnitude of change
4. Define *critical* heads: Heads that are important for utility
5. Prune a subset of *non-critical* heads that improve fairness the most

The FASP algorithm: Issues

- Effect of pruning attention heads is non-linear!
  - Heads interact through residual connections and layer norms
- E.g. If pruning the 1st and 3rd individually improves fairness, pruning them together may not

The FASP algorithm: Issues

- Effect of pruning attention heads is non-linear!
  - Heads interact through residual connections and layer norms
- E.g. If pruning the 1^{st} and 3^{rd} individually improves fairness, pruning them together may not
- **Solution:** Use Simulated Annealing to consider *all* possible subsets of attention heads!

Dealing with the Inference Time of LLMs

PennState

Efficiency of Simulated Annealing (SA) with LLMs

- SA for pruning attention heads solves the non-linearity problem
- However, SA needs to explore thousands of subsets of attention heads and inferring the fairness/utility of each state takes several minutes
- E.g. One round of inference on LLama-2-7B takes 13 minutes on RTX A6000
- How do we scale?

Approximating Fairness/Utility

- Instead of computing the real fairness/utility of every state, can we approximate it?

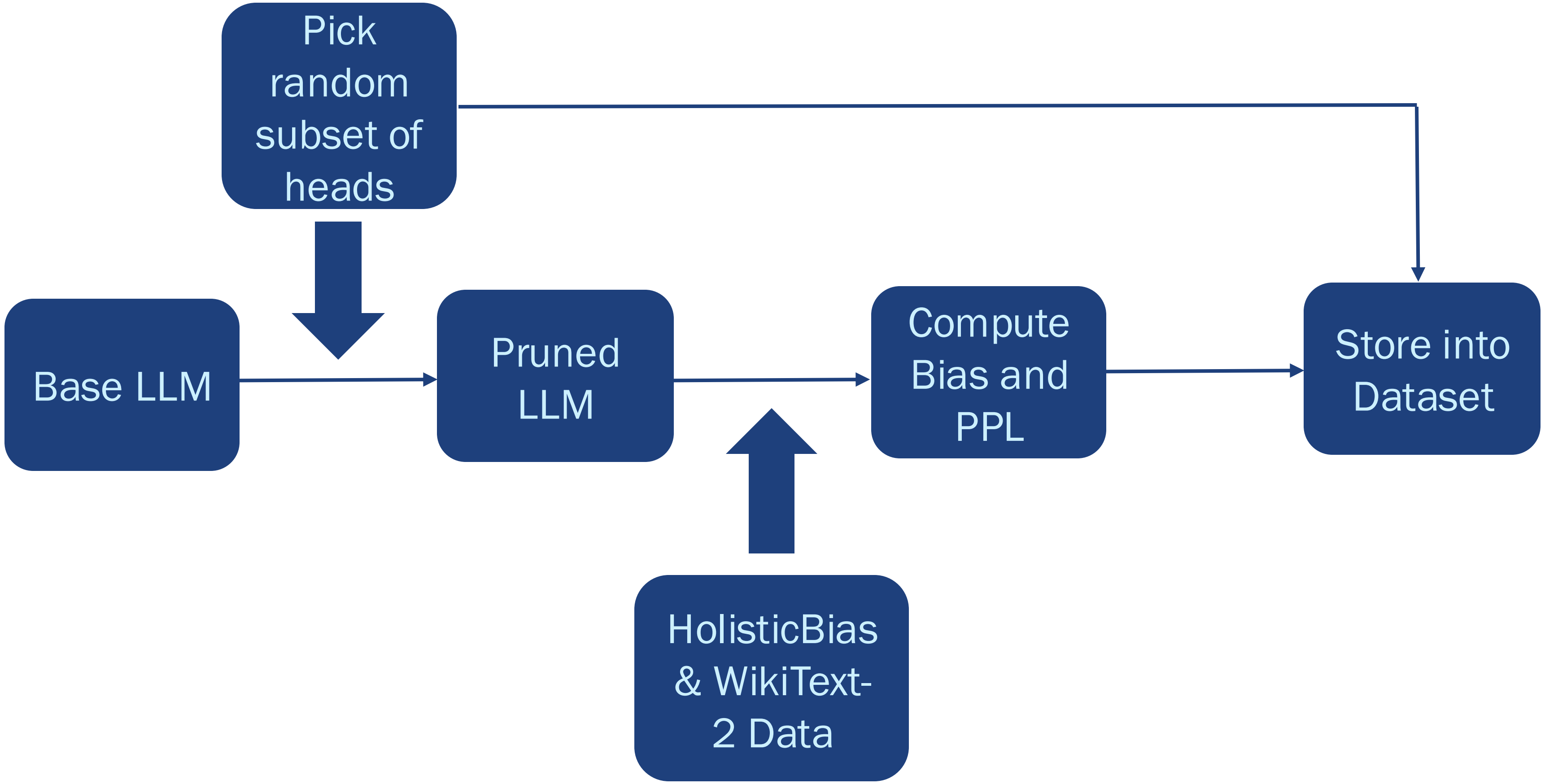
PennState

Approximating Fairness/Utility

- Instead of computing the real fairness/utility of every state, can we approximate it?
- **Solution:** Train DNNs to predict what the fairness/utility after pruning a subset of attention heads would be

PennState

Surrogate DNN Training: Dataset Creation



Examples from Dataset

Model	Attention Head Configuration	Bias	Perplexity
Llama-2-7B	0, 1, 0, 0, 0, . . . , 0, 0, 0, 0, 0	0.295	20.75
	1, 0, 0, 0, 0, . . . , 0, 0, 1, 0, 0	0.323	8.0
GPT-J-6B	1, 1, 0, 0, 0, . . . , 1, 0, 1, 0, 1	0.487	15.461
	0, 0, 1, 0, 0, . . . , 0, 1, 0, 0, 1	0.428	13.383

PennState

Surrogate DNN training

- Train two DNNs, θ_{bias} and θ_{PPL} , to predict the bias and perplexity
- The input to the DNN is a bit-vector representing a subset of attention heads
- DNNs are trained for regression using MSE loss

Algorithm 1: Training surrogate DNNs to capture the effect of pruning attention heads on the bias and perplexity.

Input: Language Model Θ, Prompts for bias $\mathcal{D}_{\text{bias}}$, Text data for perplexity $\mathcal{D}_{\text{ppl}}$, Min. and max. number of attention heads to drop $[n_l, n_u]$, Fraction of dataset to use for bias and perplexity $[\eta_{bias}, \eta_{ppl}]$, and Time Limit *time_limit*.

Output: Trained DNNs θ_{bias} and θ_{ppl} that predict the bias and perplexity of Θ after dropping attention heads.

$\mathcal{B}, \mathcal{P} \leftarrow \emptyset, \emptyset$

while $curr\_time() - start\_time \leq time\_limit$ **do**

- $s \leftarrow \texttt{random\_heads}(\Theta, n_l, n_u)$
- $\Theta' \leftarrow \texttt{prune\_heads}(\Theta, s)$
- $S_{bias} \overset{R}{\subseteq} \mathcal{D}_{bias}$ such that $|S_{bias}| = \eta_{bias}|\mathcal{D}_{bias}|$
- $S_{ppl} \overset{R}{\subseteq} \mathcal{D}_{ppl}$ such that $|S_{ppl}| = \eta_{ppl}|\mathcal{D}_{ppl}|$
- $bias \leftarrow \texttt{compute\_bias}(\Theta'(S_{bias}))$
- $ppl \leftarrow \texttt{compute\_ppl}(\Theta'(S_{ppl}))$
- $\mathcal{B} = \mathcal{B} \cup \{(s, bias)\}$
- $\mathcal{P} = \mathcal{P} \cup \{(s, ppl)\}$

$\theta_{bias} = \arg\min_{\theta} \frac{1}{|\mathcal{B}|} \sum_{(s,y)\in\mathcal{B}} \|\theta(s) - y\|^2$

$\theta_{ppl} = \arg\min_{\theta} \frac{1}{|\mathcal{P}|} \sum_{(s,y)\in\mathcal{P}} \|\theta(s) - y\|^2$

Surrogate Simulated Annealing

- Problem is now reduced to finding best input to both DNNs
- **Significantly** scales up SA with LLMs
  - E.g. 2,260,000x speed-up for LLama-2-7b

Find subset of attention heads to prune

Equivalent

Find best input to both DNNs

PennState

Cost Function

- Cost function of SA is weighted combination of bias and PPL DNN
- Ɛ controls the bias vs. perplexity trade off
  - Higher epsilon trades of better bias for worse perplexity

$$\boxed{\text{cost}(s) := \epsilon \cdot \theta_{bias}(s) + (1 - \epsilon) \cdot \theta_{PPL}(s)}$$

PennState

Attention Pruning Algorithm

Algorithm 2: AP: Surrogate Simulated Annealing for Fairness-Aware Attention Head Pruning.

Input: Unfair LLM Θ, DNNs to predict bias and perplexity $[\theta_{bias}, \theta_{ppl}]$, min. and max. number of attention heads to drop $[n_l, n_u]$, and Timeout *time_limit*

Output: Repaired LLM $\Theta_\star$, Ideal state $s_\star$, Best cost $cost_\star$

$s \leftarrow \texttt{random\_state}(\Theta, n_l, n_u)$

$s_\star, start\_time \leftarrow [0, 0, \ldots, 0, 0], \text{curr\_time}()$

$cost \leftarrow \theta_{bias}(s) + \theta_{ppl}(s)$

$cost_\star \leftarrow \theta_{bias}(s_\star) + \theta_{ppl}(s_\star)$

$T_0 \leftarrow \texttt{estimate\_temperature}(\theta_{bias}, \theta_{ppl}, s)$

while $curr\_time() - start\_time \leq time\_limit$ **do**

- $T \leftarrow \texttt{update\_temperature}(T_0, curr\_time())$
- $s_i \leftarrow \texttt{generate\_state}(s, n_l, n_u)$
- $cost_i \leftarrow \theta_{bias}(s_i) + \theta_{ppl}(s_i)$
- $\Delta E \leftarrow cost_i - cost$
- **if** $\Delta E \leq 0$ **then**
  - $cost \leftarrow cost_i$
  - $s \leftarrow s_i$
- **else if** $e^{-\Delta E/T} \geq Uniform(0,1)$ **then**
  - $cost \leftarrow cost_i$
  - $s \leftarrow s_i$
- **if** $cost \leq cost_\star$ **then**
  - $cost_\star \leftarrow cost_i$
  - $s_\star \leftarrow s_i$

$\Theta_\star \leftarrow \texttt{prune\_heads}(\Theta, s_\star)$

return $\Theta_\star, s_\star, cost_\star$

Repeat Until Time Limit

Initialization after training DNNs

Compute cost using DNNs

Transition to new state

Finally prune LLM

Experiments and Results

- RQ1: How effective are surrogate DNNs at predicting bias/perplexity?
- RQ2: How does Attention Pruning compare to SOTA?
- RQ3: What are the design considerations of Attention Pruning?
- RQ4: Can Attention Pruning generalize beyond Gender bias?

Experiments and Results

- Six different LLMs evaluated against Gender bias from HolisticBias
- Repeat with 3 different seeds
- Standard pre-processing to scale down bias/PPL in [0,1]
- Time limit of 3hrs

RQ1: Effectiveness of Surrogate DNNs

Model	θ_{bias} MSE	θ_{ppl} MSE
Distilgpt-2	0.0038	0.0005
GPT-2	0.004	0.004
GPT-Neo-125M	0.007	0.007
GPT-Neo-1.3B	0.0049	0.026
GPT-J-6B	0.0073	0.0048
Llama-2-7B	0.0046	0.010

MSE of Trained Surrogate DNNs

RQ1: Effectiveness of Surrogate DNNs

Model	cost = $\theta_{bias}(s)$		cost = $\theta_{ppl}(s)$		cost = $\epsilon \cdot \theta_{bias}(s) + (1-\epsilon) \cdot \theta_{ppl}(s)$	
	Bias	PPL	Bias	PPL	Bias	PPL
Distilgpt-2	0.275	211.61	0.415	65.28	0.285	74.981
GPT-2	0.245	80.694	0.415	43.533	0.233	52.071
GPT-Neo 125M	0.196	20226574.0	0.35	39.377	0.236	41.912
GPT-Neo 1.3B	0.249	21.7	0.42	18.323	0.282	18.5
GPT-J 6B	0.276	14.492	0.38	12.258	0.275	13.17
Llama-2 7B	0.37	7.5	0.405	6.688	0.317	7.219

Effect of using only one surrogate DNN with SA

RQ2: Comparison against state-of-the-art

Model	Baseline		AP ($\epsilon = 0.5$)		FASP [72] ($\gamma = 0.3$)	
	Bias	PPL	Bias	PPL	Bias	PPL
Distilgpt-2	0.428 ± 0.028	65.325	$\mathbf{0.288 \pm 0.002}$ ($\epsilon = 0.88, \eta = 0.2$)	**74.891**	0.31 ± 0.013 ($\alpha = 0.2$)	78.212
GPT-2	0.402 ± 0.01	43.588	$\mathbf{0.255 \pm 0.02}$ ($\epsilon = 0.8, \eta = 0.2$)	**52.071**	0.27 ± 0.018 ($\alpha = 0.2$)	58.125
GPT-Neo-125M	0.399 ± 0.008	35.628	0.241 ± 0.004 ($\eta = 0.1$)	**41.912**	$\mathbf{0.221 \pm 0.004}$ ($\alpha = 0.1$)	47.237
GPT-Neo-1.3B	0.435 ± 0.001	17.422	$\mathbf{0.285 \pm 0.003}$ ($\eta = 0.05$)	**18.548**	0.339 ± 0.012 ($\alpha = 0.16, \gamma = 0.6$)	19.391
GPT-J-6B	0.446 ± 0.013	11.695	$\mathbf{0.264 \pm 0.01}$ ($\eta = 0.1$)	**13.17**	0.288 ± 0.005 ($\alpha = 0.18$)	13.227
Llama-2-7B	0.4 ± 0.006	6.781	$\mathbf{0.316 \pm 0.003}$ ($\eta = 0.05$)	7.219	0.342 ± 0.004 ($\alpha = 0.06$)	**6.781**

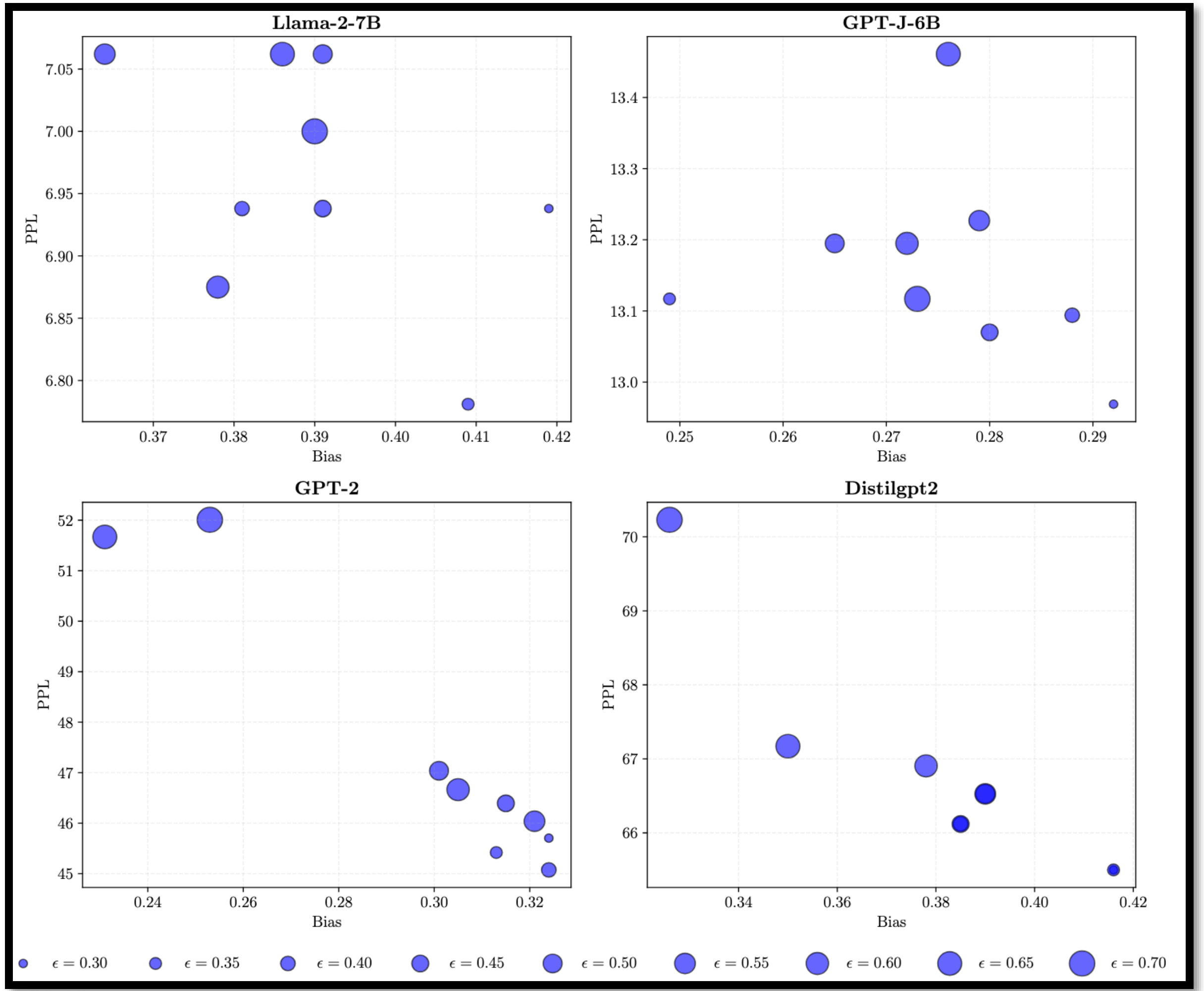
Overall, AP finds states with better bias and PPL

RQ3: Design Considerations of Attention Pruning

- AP has several hyperparameters that need to tuned
- Two important ones:
  - Ɛ in cost function
  - Running Time

PennState

RQ3: Design Considerations of Attention Pruning

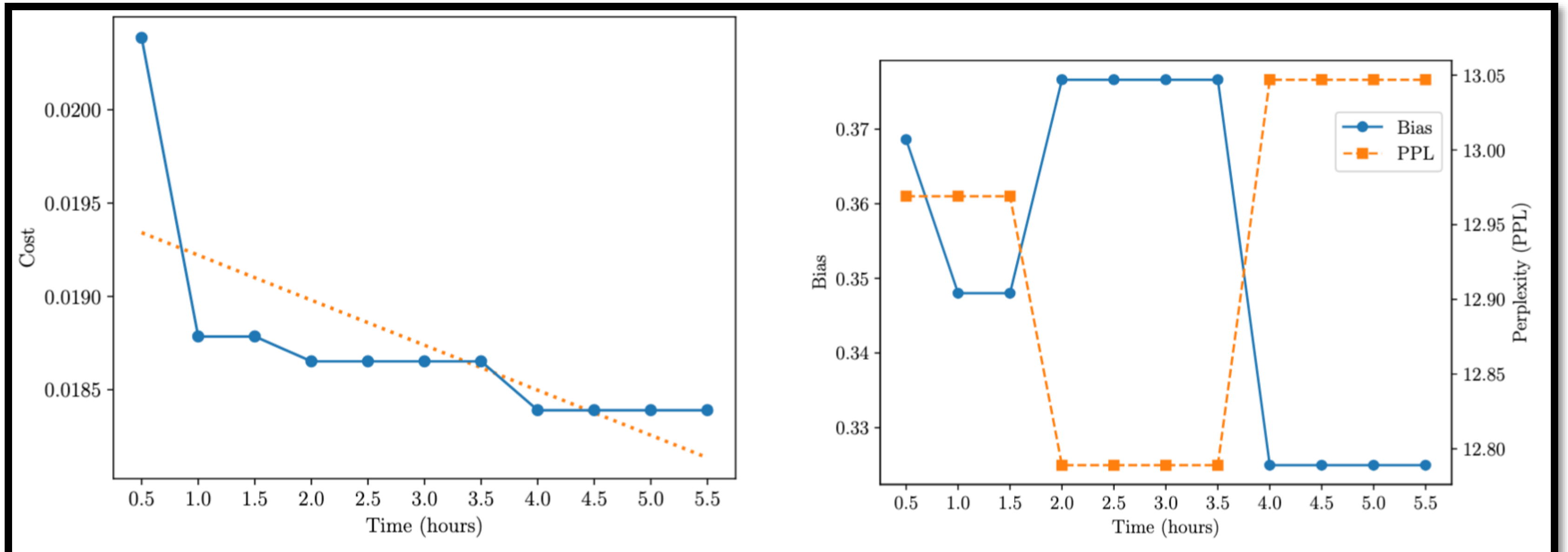


$$\boxed{\mathrm{cost}(s) := \epsilon \cdot \theta_{bias}(s) + (1 - \epsilon) \cdot \theta_{PPL}(s)}$$

Higher values of ε yield better bias at the expense of PPL

PennState

RQ3: Design Considerations of Attention Pruning



(a) Cost of best state (b) Bias/Perplexity of best state

Effect of running time on cost and bias/PPL

RQ4: Effect of reducing gender bias on other biases

Model	Race		Nationality		Sexual Orientation		Age	
	Baseline	AP	Baseline	AP	Baseline	AP	Baseline	AP
Distilgpt-2	0.529	0.395	0.288	0.225	0.737	0.57	0.054	0.036
GPT-2	0.518	0.357	0.301	0.217	0.632	0.512	0.058	0.025
GPT-Neo-125M	0.448	0.33	0.23	0.163	0.792	0.494	0.044	0.027
GPT-Neo-1.3B	0.463	0.374	0.226	0.201	0.672	0.445	0.078	0.048
GPT-J-6B	0.496	0.369	0.258	0.201	0.614	0.471	0.069	0.024
Llama-2-7B	0.458	0.385	0.252	0.217	0.612	0.49	0.057	0.033

Reducing Gender bias also reduces other biases

Conclusion

- Surrogate DNNs are effective at estimating the effect of pruning attention heads on bias/PPL
- Surrogate Simulated Annealing is effective at considering non-linear relationships between attention heads

Future Works

- Surrogate DNN idea is very interesting and powerful
  - Can we extend to other areas like robustness, privacy, etc.?
- Data collection process to train DNNs is expensive
  - Several hours to collect 1000s of samples
  - Can we find more efficient ways to train surrogate models?
- Using genetic algorithms instead of SA

PennState

Thank You!

PennState