Post-Op BERT: Improving Gradient-Surgery on Imbalanced Data

Stanford CS224N Default Project

Giancarlo Ricci
Department of Computer Science
Stanford University
gricci@stanford.edu

Abstract

Gradient surgery is a valuable technique for improving performance on multi-task learning (MTL). However, we observe that PCGrad may over-optimize on tasks with more training data. We explore existing solutions towards improving MTL with imbalanced data, and demonstrate that annealed sampling can improve the performance of PCGrad. Then, we suggest a novel approach called Alternating Round-Robin PCGrad (ARRP). We show that ARPP achieves comparable performance to annealed sampling while providing a significant speedup. We suggest AARP achieves this by balancing a task fairness objective with sufficient utilization of training data.

1 Key Information to include

- Mentor: Yuan Gao
- External Collaborators (if you have any): NA
- Sharing project: NA

2 Introduction

Fine-tuning is a powerful framework for improving deep language models (LM) on subject-specific tasks. One variant of fine-tuning is multi-task learning (MTL), where a LM is optimized for several downstream tasks simultaneously. For the CS224N default project, we are interested in fine-tuning a custom BERT implementation (Devlin et al. (2019)) on three downstream natural language processing (NLP) tasks: sentiment analysis, paraphrase detection, and sentence similarity.

One approach towards MTL is to fine-tune the LM on all tasks jointly. This approach seeks to uncover shared structures across tasks, enabling greater efficiency and improved performance compared to optimizing tasks individually. However, optimizing multiple tasks concurrently is quite challenging, sometimes resulting in lower overall performance and data efficiency than learning tasks separately (Parisotto et al. (2016)).

One reason that MTL can lead to performance degradation is because different tasks often have competing objectives, making them difficult to optimize jointly. One method to address this issue is gradient-surgery, which manually adjusts the loss gradients to resolve conflicts between tasks. Although gradient surgery is a powerful tool, it may over-optimize on tasks with more training data, leading to performance degradation and unfairness between tasks. One technique to address this problem is annealed sampling, which increases the representation of tasks with less data during training.

Motivated by the desire to increase fairness in MTL, we suggest a novel approach called Round-Robin (RR) PCGrad. We show that RR-PCGrad successfully improves fairness between objectives, at the

expense of poor data utilization. To improve upon this method, we develop an extension called Alternating Round-Robin PCGrad (ARRP). We show that ARRP is able to match the performance benefit of annealed sampling, while providing a significant speedup. Given that we do not ensemble other extensions with PCGrad, our performance validates improved gradient-surgery as a powerful tool for MTL.

3 Related Work

3.1 Gradient Surgery

Yu et al. (2020) introduce gradient surgery, an approach designed to improve MTL by altering conflicting gradients between tasks. Their method, named PCGrad, projects a task's gradient onto the normal plane of another task's conflicting gradient. To motivate this approach, observe that MTL is commonly performed on a joint loss:

$$L_{\text{total}} = \sum_k L_k$$

where L_k is the loss for individual task k. They observe that it may be difficult to optimize model parameters Θ through $\frac{\partial L}{\partial \Theta}$, since the gradient $g_i = \frac{\partial L_i}{\partial \Theta}$ and $g_j = \frac{\partial L_j}{\partial \Theta}$ may point in contrasting directions, causing the optimizer to stall when $g_{\text{total}} = \frac{\partial L_{\text{total}}}{\partial \Theta} \approx 0$. PCGrad applies the following gradient surgery to improve performance:

$$g_i \leftarrow g_i - \frac{g_i \cdot g_j}{\|g_j\|^2} g_j$$

where g_i and g_j are gradients that optimize different loss functions L_i and L_j of different tasks. By projecting the gradient of tasks onto the normal plane of conflicting gradients, the destructive interference between tasks in reduced.

Despite the utility of PCGrad for multi-task learning (MTL), it may suffer in scenarios with vastly different dataset sizes. Observe that the vanilla approach to integrate PCGrad into an MTL training loop is as follows: for each iteration, randomly sample a batch of data points, $\mathcal{B}$, from the combined training datasets $\bigcup_i \mathcal{D}_i$ (from each task, $\mathcal{T}_i$). This sampling mechanism means that tasks with larger datasets may be over-sampled, leading to them dominating over tasks with less available data.

3.2 Improving PCGrad for Data Imbalances

One strategy to mitigate task domination involves designing a more intelligent sampling scheme for tasks with different amounts of data. Stickland and Murray (2019) propose the use of annealed sampling for this purpose. In annealed sampling, the probability p_i of selecting a batch of examples from task i is proportional to N_i^α, where N_i represents the number of training examples in task i. We employ the same definition for α as Stickland and Murray (2019):

$$\alpha = 1 - 0.8\frac{e-1}{E-1}$$

where e denotes the current epoch and E denotes the total number of epochs. Intuitively, annealed sampling biases the training towards tasks with larger datasets during the initial epochs, gradually transitioning to a more equitable distribution across tasks as training progresses.

Another approaches towards increasing fairness in MTL for imbalanced data regimes include down-sampling techniques He and Garcia (2009). While downsampling can mitigate task-specific bias, it also reduces data utilization, which may lead to performance deterioration. One final approach is gradient weighting techniques exist, which seek to manually adjust the importance of gradients for different tasks Sener and Koltun (2019). GradNorm is an extension of this method, which seeks to adjust the weights of different task gradients dynamically Chen et al. (2018).

4 Approach

4.1 Introducing RR-PCGrad

Recall that for every training iteration, Vanilla PCGrad will sample $\mathcal{B}$ batches from combined training data of each task, $\bigcup_i \mathcal{D}_i$ (where $\mathcal{D}_i$ denotes the training data of task $\mathcal{T}_i$). We observe this sampling

mechanism may lead to tasks with more data being preferred each training loop. To devise a more fair PCGrad, we introduce a novel approach: Round-Robin PCGRad (RR-PCGrad).

The fundamental intervention proposed by RR-PCGrad is to redefine the sampling mechanism during each training iteration to improve fairness. Specifically, we redefine our batch sample,$\mathcal{B}'$, as: $\bigcup_i \mathcal{B}_i$, where $\mathcal{B}_i$ refers to a batch from $\mathcal{T}_i$. Intuitively, RR-Grad forces each task to contribute to the overall MTL objective each iteration, increasing task-fairness despite data imbalances.

Besides increased fairness, RR-PCGrad also seeks to improve data efficiency. Notice that in vanilla PCGrad, the sample batch $\mathcal{B}$ may consist of examples entirely from one task, which may not have conflicting gradients. In such iterations, applying PCGrad samples may incur minimal benefit at the expense of more computation time. In contrast, RR-PCGrad ensures that each task is represented, allowing for more utility to be extracted from PCGrad each iteration. Below we introduce psuedocode to highlight the major differences in these two approaches.

Algorithm 1 Vanilla PCGrad Loop

for T iterations **do**
  Sample $\mathcal{B}$ batches from $\bigcup_i \mathcal{D}_i$
  Group data from each represented $\mathcal{T}_i$ into $\mathcal{B}_i$
  Initialize `losses` as empty
  for each $\mathcal{B}_i$ **do**
    Do forward pass to obtain logits
    Compute loss for task i
    Append loss to `losses`
  end for
  Perform PCGrad on `losses`
  Update model parameters
end for

Algorithm 2 Round-Robin PCGrad Loop

for T iterations **do**
  for each task $\mathcal{T}_i$ **do**
    Sample batch $\mathcal{B}_i$
  end for
  Initialize `losses` as empty
  for each $\mathcal{B}_i$ **do**
    Do forward pass to obtain logits
    Compute loss for task i
    Append loss to `losses`
  end for
  Perform PCGrad on `losses`
  Update model parameters
end for

Figure 1: Comparing Vanilla and RR-PCGrad algorithms

While we hypothesize that RR-PCGrad will allow for more democratic training, we also observe a significant issue with this proposed approach. Namely, we observe that RR-PCGrad is now bottlenecked by the smallest training set, $\min_i |\mathcal{D}_i|$. Unfortunately, this means that RR-PCGrad may suffer from poor data-utilization for tasks with vastly more data. For the default project, we observe the following sizes of training dataset and their relative utilization each training epoch under the proposed RR-PCGrad scheme.

Table 1: Datasets and their respective sizes

Dataset	Training Examples	Dev Examples	Test Examples	% Utilized during each Epoch
STS	6,040	863	1,725	100.0%
SST	8,544	1,101	2,210	70.7%
QQP	283,010	40,429	80,859	2.1%

While shuffling each epoch helps increase the utilization of training data for such tasks, this table suggest that our largest training dataset (QQP) will experience poor data utilization, and as a consequence, may experience performance degradation. This observation motivates the following extension to RR-PCGrad to improve data utilization.

4.2 AAPR: Overcoming the Data Bottleneck of RR-PCGrad

To overcome the data bottleneck introduced in RR-PCGrad, we introduce the following novel approach: Alternating Round-Robin PCGrad (ARRP). ARRP aims to overcome this data bottleneck introduced by alternating between RR-PGrad and individually fine-tuning tasks with large amounts of training data every epoch. This algorithm is motivated by the following observations.

(i) Additional task-specific fine-tuning will increase data utilization of the task with the largest amount of training data. This helps alleviate the bottleneck introduced by RR-PGrad.

(ii) RR-PCGrad should correct for task-specific overfitting introduced by fine-tuning on the data dominant task, providing an overall normalizing effect.

To illustrate this approach, we provide the following psuedocode.

Algorithm 3 ARRP

```
Selected task with most training data, t
for E epochs do
    Do RR-PCGrad on all tasks
    Compute remaining batches in t as B'
    Sample B batches from B'
    Freeze select model params, Θ'
    Initialize loss penalty as λ
    for B batches do
        Do forward pass for logits
        Compute loss on t
        loss ← λ× loss
        Perform backward pass
        Update unfrozen params, Θ \ Θ'
    end for
```

Intuitively, AARP seeks to balance the fairness introduced by RR-PCGrad, while still maintaining sufficient data utilization for the tasks with most data. AARP introduces three hyperparameters.

- λ corresponds to the penalty applied to the task-specific weight. This is inspired by Gradient-weighting approaches Sener and Koltun (2019). Intuitively, we want to penalize the task-specific loss so it does not dominate the overall MTL objective.
- B corresponds to the number of batches used for fine-tuning.
- Θ' corresponds to which set of parameters to freeze during the task-specific fine-tuning.

5 Experiments

Given these approaches, we now proceed with fine-tuning BERT for our three downstream tasks. All the code used to run experiments is available at the following repository [1] (which is private to maintain the Stanford Honor Code, but permission is available upon request).

5.1 Data

We use the following datasets for each task:

(i) For sentiment classification, we use the Stanford Sentiment Treebank (SST) dataset [2] SST consists of sentences extracted from movie reviews, where each sentence has a label of negative, somewhat negative, neutral, somewhat positive, or positive.

(ii) For paraphrase detection, we use the Quora Question Pairs (QQP) dataset [3] , which consists of questions pairs with labels indicating whether they are paraphrases of one another.

(iii) For semantic textual similarity, we use the SemEval STS Benchmark dataset [4], which consists of sentence pairs of varying similarity on a scale of 0 (not at all related) to 5 (equivalent meaning).

[1] `https://github.com/giancarloricci/CS224N)`

[2] `https://nlp.stanford.edu/sentiment/treebank.html`

[3] `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

[4] `https://aclanthology.org/S13-1004.pdf`

5.2 Evaluation method

For SST and QQP, we measure the model's performance using accuracy between the true and predicted labels. For STS, we use the Pearson correlation coefficient, which measures the linear correlation between the true and predicted similarity values.

5.3 Experiment 1: Establishing Baselines

We first establish a performance baseline of the existing methods. We train for 10 epochs with learning rate `1e-5`. We denote Vanilla PCRad as V-PCGrad, PCGread with Annealed Sampling as A-PCGrad. B corresponds to the number of batches sampled each iteration (note that RR-PCGrad is contrained to be the minimum training size, 6040). We report the wall-clock time on a T4 GPU deployed on Google Colab. Additionally, we use Tseng (2020) as a reference implementation for PCGRad. We observe the following results.

Table 2: Establishing Baselines for PCGrad

Method	**Sentiment**	**Paraphrase**	**Similarity**	**Dev Acc**	**Time / Epoch (Minutes)**
RR-PCGrad, $B = 6040$	**0.528**	0.829	**0.869**	0.763	**7.93**
A-PCGrad, $B = 16000$	0.512	0.874	0.858	0.774	18.1
A-PCGrad, $B = 56000$	**0.535**	**0.890**	0.865	**0.785**	65.8
A-PCGrad, $B = 80000$	0.516	0.895	0.862	0.781	93.1
V-PCGrad, $B = 16000$	0.496	0.887	0.851	0.770	18.8
V-PCGrad, $B = 56000$	0.511	**0.891**	0.869	**0.779**	65.1
Full Model Finetuning	0.378	0.778	0.363	0.613	91.1
Random Baseline	0.144	0.547	-0.023	0.393	0

From these results, we extract the following takeaways:

(i) PCGrad significantly outperforms the baseline of individually finetuning each task, in much less time, validating it as an approach for MTL.

(ii) As expected, A-PCGrad outperforms V-PCGrad by increasing performance on the under-represented tasks.

(iii) As B increases, the performance of both V-PCGrad and A-PCGrad tends to increase, as well as the runtime. This makes sense, as we are utilizing more training data. We also observe that there is an upper-bound for B with the largest performance gain, after which performance tends to degrade. Given that the smaller datasets are likely exhausted, this performance degradation likely indicates the over-represented task has begun to dominate.

(iv) RR-PCGrad is more fair than V-PCGrad. In particular, it is the second best-performing approach on the under-represented tasks. Furthermore, RR-PCGrad is much more efficient than the other tasks, finishing in significantly less time However, due to its poor utilization of QQP data, it vastly under-performs on the Paraphrase Task.

5.4 Experiment 2: Exploring AARP

Next, we investigate approaches towards overcoming the data bottleneck introduced by RR-PCGrad. Before proceeding with the main experiment, we also notice the following result after running AARP for 1 epoch.

Table 3: AARP performance vs. Number of Task-Specific Batches

Method	**Paraphrase Accuracy**	**Additional Minutes / Epoch**
RR-PCGrad	0.752	0
ARRP, $B = 18000$	0.759	**1.6**
ARRP, $B = 270466$	**0.762**	18.1

This indicates that we can achieve large performance benefits by deploying AARP on only a fraction of the QQP dataset, with a huge speedup. As such, we also investigate different choices of B and their affect on performance and run-time.

To proceed with our next experiment, we also train for 10 epochs with learning rate `1e-5`. We denote the penalty applied to the task-specific loss as λ. Let F denote which parameters were frozen: $F = 0$ corresponds to not freezing any model parameters during task-specific finetuning, and $F = 1$ corresponding to freezing all parameters except the linear layer for the paraphrase task. We observe the following results

Table 4: Performance After 10 Epoch

Method	**Sentiment**	**Paraphrase**	**Similarity**	**Dev**	**Minutes / Epoch**
RR-PCGrad, $B = 6040$	0.528	0.829	0.866	0.763	7.13
A-PCGrad: $B = 56000$	0.535	**0.892**	0.869	**0.785**	**45.18**
ARRP, $F = 1, \lambda = 1, B = 24000$	0.508	0.838	0.8	0.761	8.10
ARRP, $F = 1, \lambda = 0.5, B = 24000$	0.527	0.830	0.868	0.764	8.11
ARRP, $F = 0, \lambda = 0.3, B = 24000$	0.521	0.867	0.872	0.775	14.47
ARRP, $F = 0, \lambda = 0.5, B = 24000$	0.520	0.878	0.866	**0.777**	**14.47**
ARRP, $F = 0, \lambda = 0.7, B = 24000$	0.511	0.875	0.867	0.773	14.47
ARRP, $F = 0, \lambda = 0.1, B = 100000$	0.511	**0.899**	0.879	**0.783**	**30.76**

From this experiment, we observe the following punchlines:

(i) The most significant finding is that AARP is able to achieve compare performance to A-PCGrad. This is an exciting result which validates this approach for MTL.

(ii) AARP provides significant performance speed-up to A-PCGrad, even when using more overall data. This speed-up makes sense, since A-PCGrad is computing a more computationally expensive joint loss every iteration, whereas AARP is mostly computing single-task loss.

(iii) As B increases, the performance on the paraphrase task increases. However, this comes at the expense of the performance on other tasks. Achieving good performance across all tasks requires carefully balancing increases in B with decreases in λ. This also comes at the expense of additional run-time. We also notice that can tradeoff between performance and run-time by modulating B.

(iv) $F = 1$ does not offer any significant benefit for any value of λ. One explanation for this behavior is that too few model parameters are updated to have a substantial affect on overall performance.

5.5 Discussion

Our experiments validated AARP as an approach to MTL. We suggest that AARP is able to use RR-PCGrad to increase fairness across joint optimization, while still exploiting extra training data for tasks with large amounts of available data. We suggest this behavior largely arises due to careful selection of λ to avoid this task-specific fine-tuning from dominating the overall training objective.

6 Analysis

6.1 Sentiment Analysis

Given the deployment of AARP, we seek to understand the overall behavior of our model. We begin by looking at the sentiment analysis task. First, we construct a normalized confusion matrix for the dev set predictions, as well as a plot of the overall distribution of categories in the original training data.

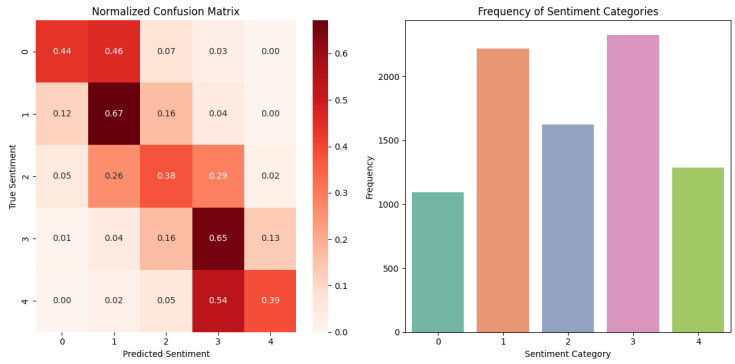


Figure 2: Confusion Matrix and Category Distribution for Sentiment

This confusion matrix reveals the following takeways:

(i) There is a strong trend along the diagonal, which indicates that even inaccurate classifications are generally close to the original label. This behavior is reassuring, and makes sense given that certain categories (e.g. negative vs somehwat negative) are quite subjective.

(ii) We notice that categories with more training instances have the highest classification accuracy. One future experiment could explore whether performance on this task improves by balancing the distribution of categories.

Finally, we also observe a number of rollouts which may indicate problems in the training set itself. In these examples, we personally believe the model's prediction to be more accurate than the annotated label.

Text	**True Label**	**Predicted Label**
It 's a lovely film with lovely performances by Buy and Accorsi.	Somewhat Positive	Positive
Creepy, authentic and dark.	Neutral	Somewhat Positive

6.2 Paraphrase Accuracy

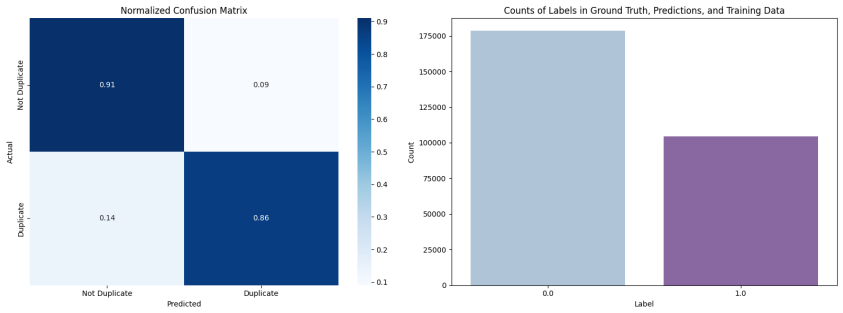


Figure 3: Confusion Matrix and Category Distribution for Paraphrase

Similarly, we plot a normalized confusion matrix and the distribution of categories in training data for the paraphrase detection task. This graph reveals that there is also a slight bias towards predicting the label with over-represented training data, however, this is much less pronounced than for the sentiment classification case. One explanation for this behavior is that the greater quantity of training data reduces this biasing affect.

We also notice the same trend in sentiment classification that certain output from the model may be more accurate than the annotation. This highlights that the model's accuracy may also be artificially deflated by annotation errors in the QQP dataset.

Sentences	True Label	Predicted Label
Is jailbreaking iphone 6s worth it? Is it worth jailbreaking the iPhone 6s?	0.0	1.0
How does it feels to be in love? How does it feel to be doing what you love?	1.0	0.0

6.3 Textual Similarity

Finally, we plot a scatterplot of the true and predicted labels on the STS dev set. We also compare the L^1 distance between labels, and group predictions into categories of based on this distance.

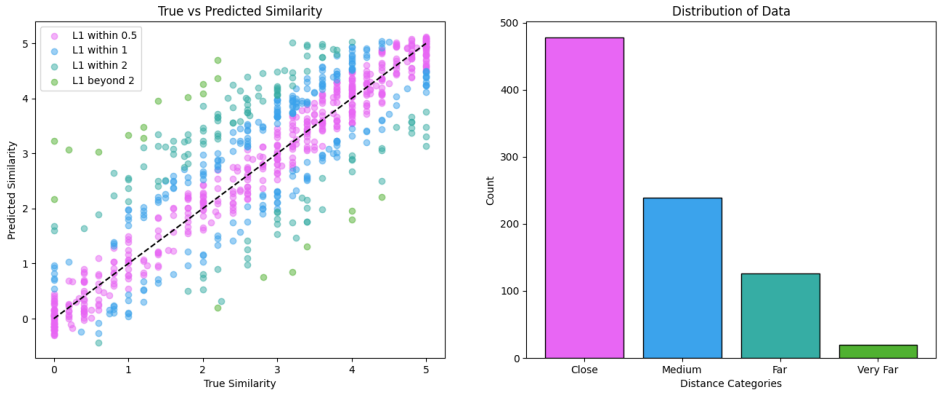


Figure 4: Similarity Comparison

This plot reveals the most predictions fall into the "close" category, which relatively few extreme outliers. However, the outliers which do exist reveal quite concerning behavior. For instance, we notice the following failures, which indicate the model has limited semantic understanding in some contexts. These failures could be the result of the limited training data for this task, or over-optimization on other tasks (e.g. Paraphrase Detection).

Sentences	True Label	Predicted Label
A man is riding a bicycle. A man is riding a bike.	5.0	2.93
A swimmer is doing the backstroke in the swimming pool. A person doing the back stroke in a swimming pool.	4.8	2.34

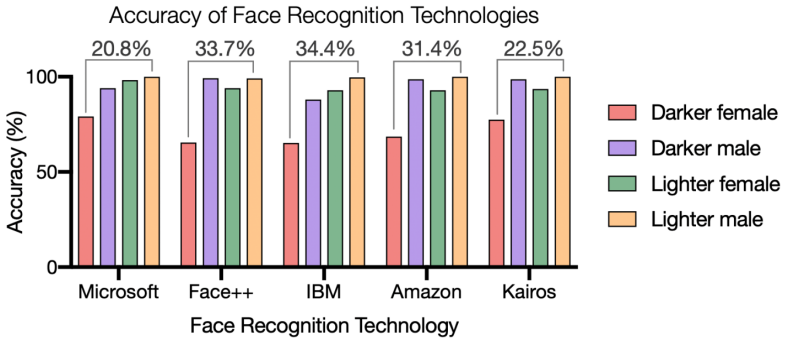
7 Conclusion

This paper introduced a novel extension to PCGrad, called AARP, which alternates between computing a more fair Round-Robin PCGrad and fine-tuning on tasks with underutilized training data. We found that AARP provided a significant performance speed-up compared to other approaches, such as annealed sampling, with a comparable performance. In particular, we achieve a score of 0.783 on the dev data set using AARP (compared to 0.785 for PCGrad with annealed sampling), while providing a roughly 1.5x speedup. Moving forward, it may be interesting to extend AARP to regimes in which more than one tasks have large amounts of training data, or when tasks have several different tiers of training data magnitude. Overall, this project suggests there are many future research directions for factoring in fairness objectives into MTL.

8 Ethics Statement

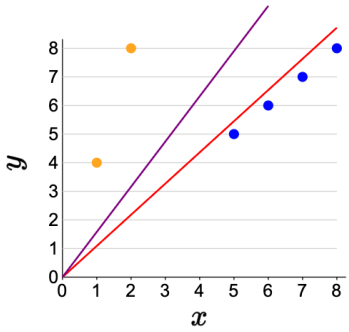
One driving force behind this project is the observation that PCGrad may over-optimize for tasks with more representative data, leading to task-unfairness. We suggest that that AARP tries to balance some notion of task-fairness with sufficient data utilization. However, we acknowledge that this tradeoff is largely dependent on a hyperpameter, λ. For certain choices of λ, AARP will also over-optimize for the task with more data.

This issue of over-optimizing for specific tasks with more data raises serious justice concerns. We observe that in many settings, imbalances in training data often originate from social inequality and biases. For example, consider the task of automated facial recognition. Buolamwini and Gebru (2018) report that many datasets used to train facial recognition models models are overwhelmingly composed of lighter-skinned subjects (79.6% for IJB-A and 86.2% for Adience). As a result, these models underperform on people of color, and especially women of color. Consider the following illustration [5] showcasing disparities in model performance for women with darker skin tones.



Given this example, we are weary of over-optimizing for certain tasks with more available data, as this may perpetuate social biases. Beyond the results highlighted by the Gender Shades project, dataset imbalances has serious impact across different domains, including ecological conversation and autonomous driving (e.g. considering how to design robust policies in states with little data).

One technical intervention to solve this issue is to reformulate the loss objective to more carefully weight different data points. For example, consider this illustration [6] of computing an average loss, compared to a more fair scheme: group DRO (Sagawa et al. (2020)).



Group DRO indicates that technical interventions can be designed to increase fairness in ML algorithms, even in cases with imbalanced training data. Practically, for AARP, the main technical intervention to maximize fairness is to rigorously search for the correct balance between utilizing available data, B, while decreasing its importance through the gradient-weight λ. This careful hyperparameter tuning, in addition rigorous oversight when deploying this method, are critical to fairly operationalize AARP.

[5] https://sitn.hms.harvard.edu/flash/2020/racial-discrimination-in-face-recognition-technology/

[6] https://stanford-cs221.github.io/autumn2023-extra/modules/machine-learning/group-dro.pdf

References

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. PMLR.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization.

Ozan Sener and Vladlen Koltun. 2019. Multi-task learning as multi-objective optimization.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.