Reproducibility Challenge @ NeurIPS 2019 Learning Robust Global Representations by Penalizing Local Predictive Power (Replication Track)

Guanqun Yang
Department of Electrical and Computer Engineering
University of California, Los Angeles (UCLA)
Los Angeles, CA 90024
guanqun.yang@engineering.ucla.edu

Abstract

This work focuses on the replication of some of the results presented in [14] **from scratch**. In particular, we implement the *patch-wise adversarial regularization* (PAR) and its variants and apply them to two-layer CNN and ResNet for domain adaptation (DA) and domain generalization (DG) tasks, respectively. The comprehensive experiments confirm the claims made in the paper, showing that

- The PAR and its variants could improve out-of-domain performance than baseline models.
- The variants of the PAR do not consistently improve upon the vanilla PAR across architectures and datasets.

Besides, we make following extension of [14]

- A review of the variants and application of the adversarial training approach, which is the origin of the PAR proposed in the paper.
- A dissection of the approach described in the paper by providing additional implementation and experiment details.

Eventually, the code for this work is hosted on GitHub [1]

1 Introduction

In this work, we are trying to reproduce the paper "Learning Robust Global Representation by Penalizing Local Predictive Power" [14], which we refer to as "the paper" throughout this work.

The remaining sections are organized as follows. Section 2 restates the problem and method discussed in the paper. Section 3 shows the reproducibility setup, including experiment platform, model architectures, hyperparameters, and experiment replication results. Key points regarding replications differences of results are also discussed there. Section 4 gives advice for reproducibility. Section 5 provides an additional literature review of the adversarial training approach. This work is concluded by Section 6 that summarizes the work.

[1] https://github.com/guanqun-yang/reproducibility_challenge_neurips_2019

Preprint. Under review.

2 Problem Statement

The paper is motivated by the observations that CNNs have a tendency to learn superficial regularities like color and texture rather than underlying semantics. This propensity cripples the classifier's ability to generalize well on samples where these regularities are not available. For example, when we try to train a classifier that is to identify turtle (water-dwelling) from tortoise (land-dwelling) based on images, we anticipate the classifier to mimic what humans do by finding the differences of their appearances. However, as is shown in Figure 1, the classifier might exploit the fact that the images of turtles generally have a background of waters while that of tortoise are mostly taken on land. Then the error could occur when these superficial cues are removed. One might argue that this classifier is still useful because of the high correlation between background and subject. Indeed, Jo and Bengio show that the generalization attained in this way is only meaningful in a narrow distributional sense, where higher-level abstractions within datasets are omitted. Additionally, it is possible to utilize natural image statistics to attack extant classifiers exploiting these regularities, causing significant performance degradation [9].

Figure 1: Turtle (left) and tortoise (right) (both CC0). They have significant differences in appearance. But classifier might exploit information in the background while forgoing underlying semantics

In order to address these issues, the paper extends the training approach in [7] by paying extra attention to the *subject* of adversarial training. Specifically, the composition of the gradient-reversal layer and a one-layer CNN is applied to the individual channel (fiber) of intermediate representation, which creates the *patch-wise adversarial regularization* (PAR).

Formally, in a given CNN $f(g(\cdot;\delta);\theta)$, the first convolution layer $g(\cdot;\delta)$ is meant to extract features $\mathbf{z}$ that could capture semantics of input image $\mathbf{x}$. However, $g(\cdot;\delta)$ tends to overfit to superficial patterns like color and textures. Then we introduce another classifier $h(\cdot;\phi)$ that penalizes this tendency of overfitting. It consists of a gradient-reversal layer $R(\cdot)$ that follows incompatible forward and backward equations

$$R(\mathbf{z}) = \mathbf{z},\ \nabla_{\mathbf{z}} R = -\mathbf{I}$$

where conventional $\nabla_{\mathbf{z}} R$ should be $\mathbf{I}$ rather than $-\mathbf{I}$.

More detailedly,

- **Forward Pass**
  When an image $\mathbf{x}$ is transformed by $g(\cdot;\delta)$, a representation of $\mathbf{z} \in \mathbb{R}^{c\times m'\times n'}$ is attained, where c is number of channels and m', n' are width and height of $\mathbf{z}$ respectively. On one hand, $\mathbf{z}$ is further transformed by remaining parts of $f(\cdot;\theta)$ and gives one class membership prediction at the output layer, which incurs a loss L_f. On the other hand, $h(\cdot;\phi)$ selects each individual location of $\mathbf{z}$, generating $m' \times n'$ class membership predictions by affline transformation. The $m' \times n'$ predictions incur a loss L_h with the same true class membership. Note that $h(\cdot;\phi)$ is indeed a fully connected neural network, but it is implemented through 1×1 convolution operation. See more details in Section 3.
- **Background Pass**
  Losses L_f and L_h are back-propagated to update parameter θ, δ and ϕ. The loss L_f is back-

propagated normally until the first layer $g(\cdot;\delta)$ and parameter θ is updated along the way. Another loss L_h is back-propagated normally until gradient-reversal layer $R(\cdot)$, where ϕ gets updated. When gradient stream meets $R(\cdot)$, the gradient is negated

$$\nabla_{\mathbf{z}} L_h = \nabla_{\mathbf{z}} R \cdot \nabla_R L_h = -\mathbf{I} \cdot \nabla_R L_h = -\nabla_R L_h$$

which prevents the $h(\cdot;\phi)$ from correctly predicting class memberships in the future. Then two streams join together, forming $\nabla_{\mathbf{z}} L_f + \nabla_{\mathbf{z}} L_h = \nabla_{\mathbf{z}} L_f - \nabla_R L_h$ to update δ.

Additionally, an additional parameter λ that governs the strength of regularization could be plugged in, then the gradient becomes

$$\nabla_{\mathbf{z}} L_h = -\lambda \nabla_R L_h$$

Wang et al. also introduce several variants of the PAR

- **More Powerful Pattern Predictor** (PAR_M)
  In the setting mentioned above, $h(\cdot;\phi)$ is a gradient-reversal layer plus a one-layer fully connected neural network without non-linearity. The one-layer fully connected neural network could be replaced with more powerful classifiers.
- **Broader Local Pattern** (PAR_B)
  Previously $h(\cdot;\phi)$ focuses 1×1 location of $\mathbf{z}$, yielding $m' \times n'$ predictions. The region of interest could be broadened to 3×3.
- **Higher Level Local Concept** (PAR_H)
  The $h(\cdot;\phi)$ is designed to append to the first convolution layer in the PAR. It could also be appended to the second convolution layer. Note, it is *not* advisable to append $h(\cdot;\phi)$ to very high-level convolution layer since these layers are meant to synthesize high-level abstractions and applying adversarial training to these layers undermine our objective to learn robust global representations.

The PAR and its variants are experimented under the setting of domain adaptation (DA) and domain generalization (DG), where the predictive power of a classifier is preserved while the distributions of input data $\mathbf{x}$ vary across the source and target datasets. Their main difference is whether or not they have access to target domain samples in training time [11].

- **Domain Adaptation**
  Learning from both source and (labeled or unlabeled) target domains to adapt source model to the target domain.
- **Domain Generalization**
  Learning from the source domain and extract a domain-agnostic model that could be applied to the unseen target domain.

Section 3 provides more information about experiment settings, results, and other critical implementation details.

3 Experiment Replication

This section starts with several important setups for the experiment, including the experiment platform, critical implementation details of the PAR, and many others. Then the experiment results on perturbed MNIST and perturbed CIFAR10 dataset are provided, which correspond to domain adaptation (DA) and domain generalization (DG) task, respectively.

3.1 Experiment Setup

Experiment Platform
All experiments in this work are conducted on a gaming laptop that costs $1,300 as of 2019 fall. The detailed platform specification is shown in Table 1.

Implementation Detail of the PAR and its Variants
When implementing $h(\cdot;\phi)$, one critical detail is that all $m' \times n'$ channel ls for an individual image share the same parameter ϕ. However, if it is implemented directly through a fully-connected

Table 1: Platform specification for experiment replication

Item	Specification
CPU	Intel Core i7-9750H
GPU	NVIDIA GeForce RTX 2060 6GB
Operating System	Ubuntu 18.04.3 LTS
Framework	PyTorch 1.3,1

layer (`torch.nn.Linear` in PyTorch), nested loops are inevitable and could consume unnecessarily more runtime. Equivalently, it could be implemented through a 1×1 convolution operation (`torch.nn.Conv2d` in PyTorch) with C output channels, where C is number of classes.

Precisely, they could be described with Protocol 1 and Protocol 2. In these protocols, $\mathbf{z} \in \mathbb{R}^{c \times m' \times n'}$ is the feature extracted by lower level convolution layer $g(\cdot; \delta)$, where c is number of channels and m', n' are height and width of $\mathbf{z}$. The ultimate goal for representation is to be identified as $\hat{y} \in \{1, 2, \cdots, C\}$ by $h(\cdot; \phi)$ given true membership y. Additionally, $\ell(\cdot, \cdot)$ is some loss function like cross entropy loss, α is learning rate, and λ is the strength of regularization.

Protocol 1 PAR through FC layer

procedure PAR($\mathbf{z}$)
  $L_h \leftarrow 0$
  Initialize FC layer with number of input features c' and output features C.
  ▷ Initialization
  for $i \leftarrow 1 : m'$ **do**
    for $j \leftarrow 1 : n'$ **do**
      $\mathbf{z}_{i,j} \in \mathbb{R}^{c'}$ is extracted
      $\hat{y}_{i,j} \leftarrow \text{SoftMax}(\text{FC}(\mathbf{z}_{i,j}))$
      $L_h \leftarrow L_h + \ell(y, \hat{y}_{i,j})$
    end for
  end for
  ▷ Forward pass
  $\nabla_{\mathbf{z}} L_h \leftarrow \nabla_{\mathbf{z}} R \cdot \nabla_R L_h = -\nabla_R L_h$
  $\nabla_{\mathbf{z}} L_h \leftarrow \lambda \nabla_{\mathbf{z}} L_h$
  ▷ Adjust regularization strength λ
  $\phi \leftarrow \phi - \alpha \nabla_{\mathbf{z}} L_h = \phi + \alpha \lambda \nabla_R L_h$
  ▷ Backward pass
end procedure

Protocol 2 PAR through 1×1 CONV layer

procedure PAR($\mathbf{z}$)
  $L_h \leftarrow 0$
  Initialize 2D CONV layer with kernel size 1×1, stride 1 , padding 0, input channels c', and output channels C.
  ▷ Initialization
  $\hat{\mathbf{y}} \leftarrow \text{SoftMax}(\text{CONV}(\mathbf{z}))$
  ▷ $\hat{\mathbf{y}} \in \mathbb{R}^{C \times m' \times n'}$
  Construct $\mathbf{y}$ by repeating y $m' \times n'$ times.
  $L_h \leftarrow \ell(\mathbf{y}, \hat{\mathbf{y}})$
  ▷ Forward Pass
  $\nabla_{\mathbf{z}} L_h \leftarrow \nabla_{\mathbf{z}} R \cdot \nabla_R L_h = -\nabla_R L_h$
  $\nabla_{\mathbf{z}} L_h \leftarrow \lambda \nabla_{\mathbf{z}} L_h$
  ▷ Adjust regularization strength λ
  $\phi \leftarrow \phi - \alpha \nabla_{\mathbf{z}} L_h = \phi + \alpha \lambda \nabla_R L_h$
  ▷ Backward pass
end procedure

PAR Variants

The specifications of PAR variants in the experiments are summarized in the Table 2.

Table 2: Specifications of the PAR variants

PAR variant	Specification
PAR_{M}	Replace one-layer FC network with thee-layer ReLU network
PAR_{H}	Apply $h(\cdot; \phi)$ at the second layer
PAR_{B}	Apply $h(\cdot; \phi)$ to 3×3 regions rather than 1×1 regions

Hyperparameter Choice

The following two sections share the same hyperparameter setting shown in Table 3. These choices are consistent with the one in the paper except the number of epochs because of limited computation resources. The hyperparameter search is not conducted for this work as a result of extensive runtime.

Table 3: Hyperparameter choice for perturbed MNIST and perturbed CIFAR experiment

Hyperparameter	Value
Learning rate α	$1e-4$
Batch size	64
Dropout rate	0.5
Adversarial strength λ	1.0
Number of epochs (MNIST)	100
Epoch that starts adversarial training (MNIST)	50
Number of epochs (CIFAR10)	80
Epoch that starts adversarial training (CIFAR10)	50

Ablation Study
We argue that ablation study is *not* applicable in this work since the architecture of main contribution of the paper (i.e. PAR) has already been its simplest possible case (one-layer fully connected network without non-linearity). Taking into account of the variants of the PAR, including PAR_H, PAR_M, and PAR_B, have validated the soundness of this approach.

Perturbation to Dataset
The perturbation to the dataset is either used to mimic different superficial patterns present in the natural images or to intentionally shift the distribution of dataset. For example, the paper proposes to apply frequency domain filtering using radial and random kernel in MNIST and CIFAR10 dataset and apply negative color and grayscale transformation to CIFAR10 dataset. All of these preprocessing operations are based on the paper's code repository [2] for this work.

3.2 MNIST with Perturbation

The architecture used for MNIST classification is shown in Table 4. Most of the operations are routine except L_2 normalization, which normalizes each entry of tensor $\mathbf{v}$ with the maximum $\|\mathbf{v}\|_2$ across one axis.

Table 4: Two-layer CNN for MNIST classification

Operation	Output dimension	Activation
Input $\mathbf{x}$	$(N, 1, 28, 28)$	-
Layer1		
CONV	$(N, 32, 28, 28)$	ReLU
POOL	$(N, 32, 14, 14)$	-
Layer2		
CONV	$(N, 64, 14, 14)$	ReLU
POOL	$(N, 64, 7, 7)$	-
Output		
Flatten	$(N, 64 \times 7 \times 7)$	-
Linear	$(N, 1024)$	ReLU
L_2 Normalization	$(N, 1024)$	-
Dropout	$(N, 1024)$	-
Linear	$(N, 10)$	-

Table 5 corresponds to Figure 1 in the original paper. Here the figure is replaced with a table for clarity. It could be seen that

- **PAR and its variants consistently improve out-of-domain accuracy**
  Across six different cases, the test accuracies are *consistently* improved due to the application of the PAR and its variants. The most significant improvement occurs at "Original independent" for PAR_H with a 15.81% improvement in test accuracy. Half of the cases have more than 8% improvement. Other cases embody at least a 1% increase in test accuracy.

[2] https://github.com/HaohanWang/PAR_experiments

Table 5: Test accuracy of vanilla CNN, PAR and its variants on perturbed MNIST dataset

	Vanilla	PAR	PAR_B	PAR_M	PAR_H
Original independent	0.7744	0.8814	0.85070	0.82020	**0.9325**
Random independent	0.7100	0.6953	0.75540	**0.79380**	0.7692
Radial independent	0.7435	**0.7721**	0.68830	0.71260	0.7111
Original dependent	0.7222	0.7714	**0.80590**	0.71840	0.7757
Random dependent	0.4253	0.3875	0.43530	**0.44090**	0.4239
Radial dependent	0.5336	0.3956	0.47640	0.47690	**0.6158**
Average	0.6515	0.65055	0.668667	0.660467	**0.7047**

Table 6: ResNet for CIFAR10 classification

(a) ResNet

Operation	Output dimension	Activation
Input $\mathbf{x}$	$(N, 3, 32, 32)$	-
Layer1		
CONV	$(N, 16, 32, 32)$	-
BN	$(N, 16, 32, 32)$	ReLU
Layer2		
ResidualBlock$\times 5$	$(N, 16, 32, 32)$	-
Layer3		
ResidualBlock$\times 5$	$(N, 32, 16, 16)$	-
Layer4		
ResidualBlock$\times 5$	$(N, 64, 8, 8)$	-
Output		
BN	$(N, 64, 8, 8)$	ReLU
AdaptiveAvgPool	$(N, 64, 1, 1)$	-
Flatten	$(N, 64)$	-
FC	$(N, 10)$	-

(b) Residual block

Input $\mathbf{z}$	
Shortcut	CONV
	BN
	ReLU
	CONV
	BN
Output ReLU$(\mathbf{z} + f(\mathbf{z}))$	

- **No consistent winner for one particular PAR variant**
  Despite the consistent increase in testing accuracy than the vanilla baseline, there is no consistently better regularization method among the PAR and its variants. The responsibility of choosing a particular type of the PAR is shifted to their users through understandings of the problem in hand.

3.3 CIFAR with Perturbation

The architecture of ResNet used in the paper is outlined in Table 6. Note that

- Some revisions are made to the original architecture proposed in [8] because the size of the image is reduced from $(3, 227, 227)$ (ImageNet) to $(3, 32, 32)$ (CIFAR10). Specifically, the first layer is no more made from residual block. At the same time, the first few layers are removed since reducing image size is not necessary in this application.
- The paper proposes a different way to construct shortcut in residual block. When number of channels increases because of convolution operation in f : CONV $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ CONV $\rightarrow$ BN, the shorcut first reduces the size of $\mathbf{z}$ using average pooling, then increased channels of in $\mathbf{z}$ are padded with 0 to match the dimension of $f(\mathbf{z})$. For example, when a representation $\mathbf{z}$ of size $(16, 32, 32)$ becomes $(32, 16, 16)$ because of convolution operation in $f(\cdot)$. The shortcut in residual block will pad 0 to increased 16 channels to $\text{AvgPOOL}(\mathbf{z}) \in \mathbb{R}^{16\times 16\times 16}$.

The replication results in Table 7 correspond to the ones Table 1 in the original paper. One might frown upon this performance at first glance due to its large deviation from the original results. However, we argue this comes from significantly fewer number of training epochs in this work. The number of epochs is just 20% (80 versus 400) of the original paper, where Wang et al. achieve

Table 7: Test accuracy of vanilla ResNet, PAR and its variants on perturbed CIFAR10 dataset

	Vanilla	PAR	PAR_B	PAR_M	PAR_H
Original	0.7069	**0.73550**	0.7265	0.7210	0.7241
GrayScale	0.1000	0.10000	0.1000	0.1000	0.1000
NegColor	0.2617	**0.29790**	0.2676	0.2849	0.2645
RandKernel	0.2409	0.16680	0.2104	**0.2547**	0.1850
RadialKernel	0.1420	0.13200	**0.1560**	0.1256	0.1351
Average	0.1862	0.17418	0.1835	**0.1913**	0.1712

92% accuracy in original test image. We believe the original results are reproducible with additional program runtime.

Besides the similar conclusions we could draw from the MNIST experiment, we could find that

- **CIFAR10 dataset is more difficult than the MNIST dataset**
  Despite consistent improvement except "GrayScale", the improvements become significantly smaller than that of the MNIST experiment, with maximum test accuracy boost merely 3.62%. However, it is still exciting to know that all cases except "GrayScale" still have at least a 1% improvement.
- **The "GrayScale" case requires extra attention**
  In the original paper, the best performance is attained in the case of "GrayScale". However, the replication of experiments shows the otherwise, where it has the worst performance. Future work is required to take a closer look at why this happens.

4 Advice on Reproducibility

Use existing code as much as possible
Due to the page limits of the publications and overwhelmingly many fine details of experiment, it is not likely that one could recover all the details needed to reproduce the experiment by just reading the paper. Thanks to the fact that machine learning community are open to share research codes, we could refrain from rebuilding the wheels and spare our time and energy to most interesting parts.

In this work, even though we are trying to rewrite the code from scratch with the different framework (PyTorch vs. TensorFlow in the original paper), we use some of the codes released by authors for data preprocessing, which greatly accelerate our experiment reproduction process.

Make good use of OpenReview platform
The ambiguities might occur even for a well-written paper. This problem is made even more so when trying to approach the fine details of experiments. OpenReview provides a platform for authors and readers to discuss details of the research paper.

This work is not possible without clarifications made by authors on the OpenReview platform.

Writing everything in one script might *not* something to avoid
We used to believe that structured project management could always help streamline working flow, where the main script with various parameter settings, models, utility functions, and many other components are organized in different folders and scripts. Even though this might be true for research projects, the goal of the reproducibility challenge is to *reproduce* existing work. Since the degree of uncertainty is largely reduced and quick reproduction is the concern, it might be preferable to try alternative project management strategy.

In this work, we organize codes according experiment in the original paper rather than carefully designed project structure.

5 Extended Literature Review

Motivated by the domain adaptation problem, the adversarial training approach is first introduced in [7]. Ganin et al. propose a generic gradient-reversal layer appended to multi-task branch in helping neural networks learn representations that are invariant across domains but discriminate within the

domain, thus encouraging generalization to new domains. After this trailblazing work, there have been significant efforts in its improvement and application.

- **Improvement**
  The improvement focus on *location* and *subject* of adversarial training branch. In terms of *location*, most works follow the setup in [7] and place the adversarial branch at the output layer. However, this choice is *not* backed by strong theoretical support. Yang proposes to inspire the location of the adversarial branch using mutual information and reduce the discriminative power of intermediate representation with maximum mutual information to labels [16]. On the other hand, the study of *subject* is still in its nascent stage. Most works do *not* focus on a particular form of representation when try to apply adversarial training approach for the regularization. Wang et al. divert their attention to adversarially train against local image fiber (individual image channel) and achieve state-of-the-art performance on multiple domain generalization (DG) and domain adaptation (DA) tasks.
- **Application**
  The works employing adversarial training approach mostly come from the study of domain adaptation (DA) and domain generalization (DG), including image classification [13], low-resource machine translation [10], and human-to-human speech recognition [12]. When algorithmic fairness becomes concerns because of discrimination of automated decisions against under-represented groups [5, 4], learning representations that could yield predictive results under some fairness constraints becomes a new direction of study [17]. The adversarial training approach is among all methods that learn fair representation and show empirical success [6, 3, 15, 2, 1, 16].

6 Summary

This work reproduces two of the experiments presented in [14]. We confirm the main findings made by authors through validating the usefulness of the patch-wise adversarial regularization (PAR) and its variants. Additionally, we extend this previous work by providing more details about the implementation and an extended literature review.

Acknowledgments

We are grateful for the productive discussion with Haohan Wang and Songwei Ge, the authors of the original paper. This work is not possible without your timely and detailed clarifications. We also thank the organizers of the NeurIPS Reproducibility Challenge for their efforts to provide this opportunity for us to reproduce state-of-the-art works in machine learning.

References

[1] Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[2] Ehsan Adeli, Qingyu Zhao, Adolf Pfefferbaum, Edith V Sullivan, Li Fei-Fei, Juan Carlos Niebles, and Kilian M Pohl. Bias-resilient neural network. *arXiv preprint arXiv:1910.03676*, 2019.

[3] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*, 2017.

[4] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[5] Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. *San Fransico, CA: Reuters. Retrieved on October*, 9:2018, 2018.

[6] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015.

[7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

[10] Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. 2018.

[11] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5542–5550, 2017.

[12] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. *arXiv preprint arXiv:1703.02136*, 2017.

[13] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[14] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

[15] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. *arXiv preprint arXiv:1811.08489*, 2018.

[16] Guanqun Yang. Fairness-preserving empirical risk minimization. Master's thesis, UCLA, 2019.

[17] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013.