

*Appendix for Enhancing Representation Learning with Deep Classifiers in Presence of Shortcut*

Amirhossein Ahmadian and Fredrik Lindsten

## A Additional implementation and hyperparameter details

What follows is additional details about the *proposed method*, for both CIFAR-10 and CelebA experiments. The learning rate used for the classifier network and lens network are 0.0002 and 0.002 respectively, and batch size is 128 (in the CIFAR-10 case, the size is  $128 \times 4$  in effect due to the rotations). The quite small learning rate of the classifier is because of oscillation and convergence problems. The lens has a larger learning rate since we observed that in practice it cannot keep up with the classifier during the adversarial training (game) if it has the same learning rate and number of iterations as the classifier (possibly due to the longer gradient backpropagation path compared to the classifier). The first 2 iterations (on mini-batches) of the training are considered as warm-up iterations. During these iterations, the lens is ignored, and the classifier is trained only on the real data. It should be emphasized that the ResNet model we train on the upstream data (for our method as well as the other compared ones) is not pre-trained on any dataset, and is initialized randomly.

In the vanilla baseline (ERM) method, the learning rate and batch size is equal to the values used for the classifier in our method, to rule out any effect of learning rate. In general, when choosing the hyperparameters for the other compared methods, we try to use the values suggested in their papers or corresponding code when a similar dataset (CIFAR-10 or CelebA) has been used in those papers.

For the *Automatic Shortcut Removal* method, the learning rate is 0.0001 for both of the networks and the datasets, and the batch size is 128. In the arrow shortcut experiment, a suggested  $\lambda$  is already available since this problem is experimented in their paper as well. For the light gradient and CelebA experiments, we found the best  $\lambda$  by probing the downstream test accuracy with  $\lambda \in \{1, 10, 160, 320, 640, 1280\}$ .

For the *Spectral Decoupling* method, we followed the  $\lambda$  and learning rate values that is used in their paper/code. This means the learning rate 0.001 (with batch size 50) for CIFAR-10 and 0.0001 (with batch size 128) for CelebA experiments.

For the *Diverse Ensemble* method, the ResNet network is used as the shared feature extractor, where its pre-logits output is fed to each head. The heads are feedforward networks with 2 layers of 512 hidden units, ReLU activation functions, and softmax output, which is the same network used in their paper. The best value of  $\lambda$  was determined by probing the downstream test accuracy with  $\log_{10} \lambda \in \{-1, 0, 1, 2, 3, 3.6\}$ . The learning rate and batch size for this method are 0.0002 and 128.

The downstream classifier is always trained with ADAM optimizer, learning rate 0.001, and batch size 64.

## B Learning curves

Figure 1 shows how the loss values of the lens and classifier networks change during training epochs of the proposed method. In addition, the accuracy of the classifier on real data is shown in this figure. Each point in these plots refers to the loss/accuracy values on the training set averaged over all iterations during the epoch. It should be noted that the classification accuracy on the real data is obtained with discarding the extra (fake class) output neuron of the classifier network.

We observe that the classifier loss can reach a relatively small value in earlier epochs, but that is followed by a maximum, and then it keeps going down. This behavior can be explained by taking into account the nature of adversarial training and the presence of shortcuts. In the early iterations, when the lens is still weak, it is easy for the classifier to detect the fake lens reconstructions. Shortly after, the lens manages to take advantage of the vulnerability of the classifier to shortcuts, and thus the classifier is fooled more easily. In the next phase, the classifier starts to learn the non-shortcut features to improve its performance in both detecting the fake images and classifying real data. The other important point is that the classifier achieves an almost perfect accuracy on real images (upstream classification accuracy). This is reasonable because the shortcuts still exist in the images, and therefore it is always easy to predict the true class. This also shows that the performance of the classifier on the upstream tasks is not impaired by the modifications made by our method, that is, the upstream classifier can be used in the task as usual afterwards, albeit by removing the extra output neuron (in all of the three experiments, the classifier accuracy on the test data ends at a value around 99% as well).

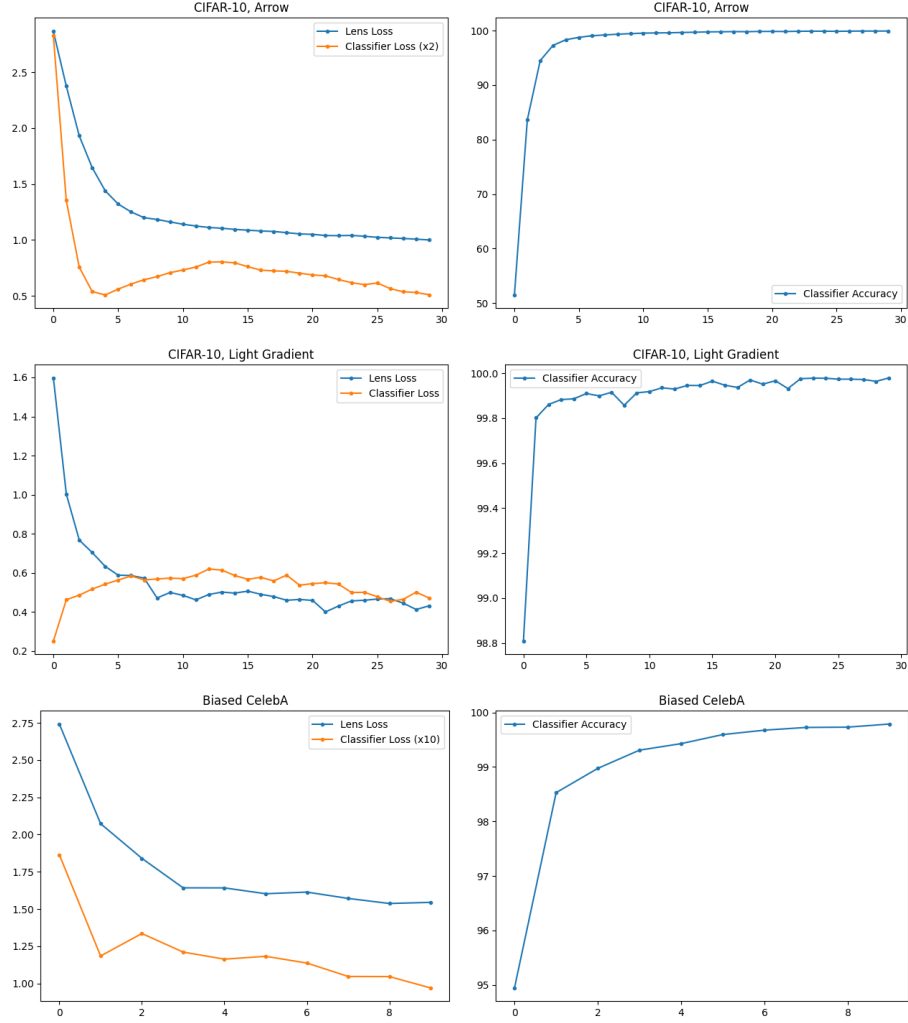


Figure 1: Loss curves of the classifier and lens, and accuracy in real images classification, in our proposed method. The values are per epoch and on training data.

## C Lens reconstruction plots

In the following, some more examples of the images from the three datasets studied in the paper, and their corresponding lens output obtained by our method, have been shown.









Input	Lens Reconstructions		
			
			

Table 1: CIFAR-10 Light Gradient Examples. Reconstructions (from left to right) obtained in epochs 2, 10, 20 .





















Input	Lens Reconstructions			
				
				
				
				

Table 2: CIFAR-10 Arrow Examples. Reconstructions (from left to right) obtained in epochs 2, 3, 10, 20 .

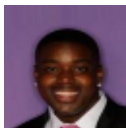
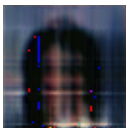


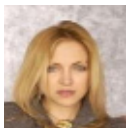


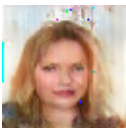

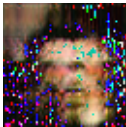



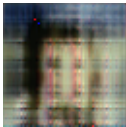


Input	Lens Reconstructions		
			
			
			
			

Table 3: Biased CelebA Examples. Reconstructions (from left to right) obtained in epochs 2, 5, 10 .