Appendix

Our experiments are conducted on an Intel Xeon E5-2687W CPU and a Nvidia Tesla T4 GPU based on Pytorch 1.4.0. Detailed settings are as follows:

Models. Following the adversarial learning framework described in Sec. 3, its training involves a feature extractor (E_θ), an adversarial classifier (A_ψ) and a target classifier (C_ϕ). The model architectures of them are presented in Table 1. We train the framework for each attribute pair, respectively. Following the evaluation procedure described in Sec. 3.3, we evaluate these trained adversarial learning frameworks based on a normal baseline classification model (BC) and a validation classifier (VC) for each attribute pair, respectively. Model architectures of BC and VC are also presented in Table 1.

Hyper-parameters. The total training epoch is 50. The Adam optimizer is utilized with the learning rate 0.0001, and batch size 64. We fix the entropy term $\alpha = 0.3$, and set the tradeoff parameter $\lambda \in \{0.3, 0.5, 0.7\}$, respectively.

Implementation details. We repeat each experiment for 3 times and report the average results. All the input images are resized to 64×64. For each attribute pair, we select 3000 samples without replacement through weighted sampling to meet the corresponding SR requirement as the training set to train the adversarial learning framework, and select 1000 samples as the testing set to evaluate the framework with no statistical bias by setting $(C_{00} : C_{10} : C_{01} : C_{11}) = (1 : 1 : 1 : 1)$.

Metrics. When designing the utility and privacy evaluation metrics, we both considered accuracy and AUC metrics. Considering the accuracy metric is effective when the testing data is uniformly distributed in labels, we resample the testing data to make its label distribution balanced, where $(C_{00} : C_{10} : C_{01} : C_{11}) = (1 : 1 : 1 : 1)$ for the privacy and target attribute labels. At the same time, the accuracy metric is easier to manually calculate for a non-expert. But it is better to take AUC as the privacy and utility metrics for imbalanced testing sets considering AUC performs better when evaluating imbalanced datasets.

Table 1. Model Architecture Configurations

E_θ	C_ϕ/A_ψ/VC	BC
2×conv3-64 maxpool	3×conv3-256 maxpool	2×conv3-64 maxpool
2×conv3-128 maxpool	3×conv3-512 maxpool	2×conv3-128 maxpool
	2×fc-4096 fc-1	3×conv3-256 maxpool
		3×conv3-512 maxpool
		3×conv3-512 maxpool
		2×fc-4096 fc-1