
Fast Implicit Constrained Optimization of Non-decomposable Objectives for Deep Networks

Yatong Chen^{†‡*}, Abhishek Kumar[†], Yang Liu[‡], Ehsan Amid[†]

[†] Google Research, Brain Team

[‡] Department of Computer Science, UC Santa Cruz

ychen592@ucsc.edu, abhishk@google.com, yangliu@ucsc.edu, eamid@google.com

Abstract

We consider a popular family of constrained optimization problems in machine learning that involve optimizing a non-decomposable objective while constraining another. Different from the previous approach that expresses the classifier thresholds as a function of all model parameters, we consider an alternative strategy where the thresholds are expressed as a function of only a subset of the model parameters, i.e., the last layer of the neural network. We propose new training procedures that optimize for the bottom and last layers separately, and solve them using standard gradient-based methods. Experiments on a benchmark dataset demonstrate our proposed method achieves performance comparable to the existing approach while being computationally efficient.

1 Introduction

Many machine learning applications involve optimizing for *non-decomposable metrics*, such as false-positive rate (FPR) or precision, while constraining another, such as false-negative rate (FNR) or recall. For these metrics, the loss on a set of data points cannot be expressed as the sum of losses of individual data points (e.g., FPR involves a ratio that depends on the total false-positive and true-negative classified examples). Such problems arise in several applications. For example, many fairness objectives can be expressed as rate constraints, including the popular *equal opportunity* and *equalized odds* fairness criteria [1, 4, 12, 13]. The typical baseline for solving these problems is first to train the model using regular cross-entropy (CE) loss, followed by tuning the thresholds imposed on the model predictions to classify between positive versus negative classes while the constraint is satisfied. However, the post hoc approach often yields sub-optimal results as the model parameters are trained on a different objective than the evaluation metric. To improve upon the baseline approach, *Implicit Constrained Optimization* (ICO) [6] reformulates the prediction threshold as a function of the model parameters and transforms the constrained optimization problem into an unconstrained one to jointly optimize for the model parameters and the thresholds. ICO solves the problem by decomposing the gradient into a sum of disjoint terms, which consist of the gradient of the loss function and the constraint with respect to the model parameters and the thresholds. Such a decomposition is computationally more expensive than standard training since it requires additional backward passes to compute the gradients with respect to each component. In particular, ICO becomes more expensive when there are multiple constraints, e.g., when optimizing for the area under the ROC Curve (AUC). To reduce the computational cost, ICO resorts to approximations, e.g., intermittently updating the thresholds – Section 3.3 in [6]. Thus, it is desirable to close this computation gap while retaining the simplicity of the method as well as potentially improved performance on the metrics.

*Work done as a Student Researcher at Google Brain.

Motivation of This Work: Constrained Optimization of Only the Last Layer Standard neural network architectures can be decoupled into a *feature extractor* sub-network at the bottom, followed by a single fully-connected *classifier* layer to predict the class probabilities at the top (see Figure 1). Several recent studies have shown that the feature extractor network can be trained reasonably well even when the training data is noisy [7] or imbalanced [10, 5]. These studies show the significance of retraining the last layer weights on a clean or balanced validation set in obtaining good performance based on the feature representation extracted by the bottom sub-network. While it is natural to impose the constraint on all the model parameters in our problem, the results from the earlier work raise the question of whether constraining the last layer weights is sufficient for obtaining desired performance [6].

To this end, we propose to make the prediction thresholds a function of only the last layer parameters. This simplification leaves the parameters of the earlier layers free to be optimized with any surrogate loss in an unconstrained manner. In the meantime, the constraint is imposed on the last layer parameters. The major benefit of our approach is avoiding additional backward passes to calculate the gradient of the constraint(s) with respect to the weights of the bottom sub-network.

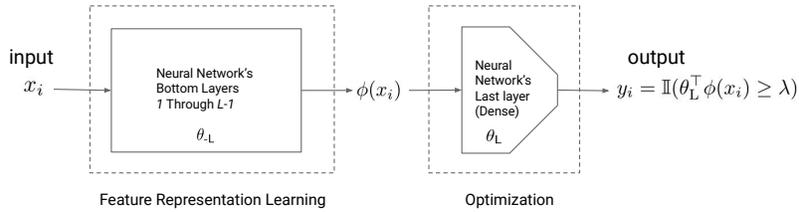


Figure 1: Demonstration of the bottom sub-network with weights θ_{-L} and the top (i.e., last) layer weights θ_L of a typical neural network.

2 Problem Formulation

Basic Setup In this paper, we consider a binary classification setting with feature space X and binary label $\{0, 1\}$. Our goal is to learn a *binary threshold scoring rule* $s_{\theta|\lambda} : X \rightarrow \{0, 1\}^m$ parameterized by model parameter $\theta \in \mathbb{R}^p$ and *threshold* values $\lambda \in \mathbb{R}^m$, such that for any given input $x \in X$, $[s_{\theta|\lambda}(x)]_i = \mathbb{1}(s_{\theta}(x) \geq \lambda_i)$ where $s_{\theta}(x) : X \rightarrow \mathbb{R}$ is a scoring model parameterized by the model parameters $\theta \in \mathbb{R}^p$ and $\mathbb{1}(\cdot)$ is the indicator function.

We assume the scoring model s_{θ} is an L -layer neural network whose parameters can be divided into two disjoint parts $\theta = [\theta_{-L}, \theta_L] \in \mathbb{R}^p$: the parameters of the last dense layer $\theta_L \in \mathbb{R}^{p_L}$, and the weights from the rest of the network, which we will also refer to as the bottom layers: $\theta_{-L} \in \mathbb{R}^{p_{-L}}$, where $p = p_{-L} + p_L$. The bottom layers transform the input x into a feature representation $\phi(x)$. The last dense layer then sets $s_{\theta}(x) = \theta_L^T \phi(x)$. In this work, we explore ideas involving faster training of deep networks using non-decomposable objectives by applying different strategies for training the last layer weights θ_L and the rest of the weights θ_{-L} .

Objective Function In this paper, we consider a general form of a constrained optimization problem with m constraints:

$$\min_{\theta \in \mathbb{R}^p} f(\theta, \lambda) \quad \text{s.t.} \quad g(\theta, \lambda) = \mathbf{0}, \quad (1)$$

where $f : \mathbb{R}^p \times \mathbb{R}^m \rightarrow \mathbb{R}$ and $g : \mathbb{R}^p \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ are non-decomposable functions that cannot be expressed as the sum of losses on individual data points, $\theta \in \mathbb{R}^p$ are the model parameters, and $\lambda \in \mathbb{R}^m$ are the thresholds and chosen such that the constraint is satisfied. Similar to [6], throughout the training process, we assume that θ stay in the feasible region, namely $\forall \theta, \exists \lambda$ s.t. $g(\theta, \lambda) = \mathbf{0}$.

Here, we provide one example that satisfies this general constraint optimization form, which is also going to be the main task of our experimental demonstration in Section 4:

Example 1 (Minimizing FNR subject to a fixed FPR). Consider a setting where we want to minimize the FNR at the threshold $\lambda \in \mathbb{R}$ at which the FPR is a fixed value $\beta \in [0, 1]$:

$$f(\theta, \lambda) = \text{FNR}(s_{\theta|\lambda}) = \frac{\text{FN}(s_{\theta|\lambda})}{\text{TN}(s_{\theta|\lambda}) + \text{FP}(s_{\theta|\lambda})}, \quad g(\theta, \lambda) = \text{FPR}(s_{\theta|\lambda}) = \frac{\text{FP}(s_{\theta|\lambda})}{\text{TP}(s_{\theta|\lambda}) + \text{FN}(s_{\theta|\lambda})} - \beta,$$

where $\text{FN}(s_{\theta|\lambda})$, $\text{TP}(s_{\theta|\lambda})$, and $\text{TN}(s_{\theta|\lambda})$ denote the true-positive, false-positive, and false-negative values, respectively, for the threshold scoring rule.

Surrogate Losses It is possible that the optimization objectives and constraints described above are non-differentiable. To make the training viable using gradient descent-based methods, we replace f and g with their corresponding *smooth surrogate losses*, \tilde{f} and \tilde{g} , and relax Eq. (1) into:²

$$\min_{\theta \in \mathbb{R}^p} \tilde{f}(\theta, \lambda) \quad \text{s.t.} \quad \tilde{g}(\theta, \lambda) = \mathbf{0}. \quad (2)$$

3 Methods

We first briefly discuss the previous methods and introduce our efficient alternative.

3.1 Previous Approach: Implicit Constraint Optimization (ICO) on All Weights

Different from traditional Lagrangian based methods [3], ICO [6] avoids explicitly solving the constrained problem in Eq. (2) by formulating an equivalent unconstrained problem in which the thresholds λ are expressed as an implicit function of the model parameters θ (within a neighborhood around θ). Specifically, the implicit function theorem [11] implies: suppose (θ_0, λ_0) satisfies the constraints, namely $\tilde{g}(\theta_0, \lambda_0) = \mathbf{0}$. Then, we can express the thresholds as $\lambda_0 = \tilde{h}(\theta_0)$ in the neighborhood of θ_0 where \tilde{h} is some implicit function that depends on problem setup and the local structure of (θ_0, λ_0) . See the detailed description of the theorem (Theorem 1) in the appendix.

The benefit of using Theorem 1 is that we can change the constrained optimization problem to be an unconstrained optimization problem with only one free variable θ :

$$\min_{\theta} \tilde{f}(\theta, \lambda) \quad \text{s.t.} \quad \tilde{g}(\theta, \lambda) = \mathbf{0} \xrightarrow{\text{Theorem 1}} \min_{\theta} \tilde{f}(\theta, \tilde{h}(\theta)). \quad (3)$$

A differentiable function $\tilde{h}(\theta)$ that provides us $\tilde{g}(\theta, \tilde{h}(\theta)) = \mathbf{0}$ for a particular threshold that we care about might not always exist. However, [6] identifies some conditions under which the resulting Eq. (3) is convex in θ .³ We also provide detailed derivations in the appendix.

3.2 Proposed Approach: Constraining the Last Layer Weights θ_L only

In this work, we propose making the thresholds λ a function of only the last layer’s weights θ_L , and applying different training procedures for the bottom layer weights θ_{-L} and last layer weights θ_L separately.

Implicit function theorem on θ_L First, we propose to make the thresholds λ in the constraint $\tilde{g}(\theta, \lambda) = \mathbf{0}$ a function of only θ_L , which gives us the following modified form of the implicit function theorem (see the appendix). The modified theorem implies that given $\phi(x)$, the classifier weights θ_L control the value of the thresholds λ in a feasible region.

New objective form: decoupled training for θ_{-L} and θ_L We propose a new optimization procedure for training a neural network that optimizes its bottom layer weights θ_{-L} and last layer weights θ_L separately using different training objectives:

$$\min_{\theta_{-L}} \tilde{r}(\theta_{-L} | \theta_L, \lambda) \quad \text{and} \quad \min_{\theta_L} \tilde{\ell}(\theta_L, \lambda | \theta_{-L}) \quad \text{s.t.} \quad \tilde{g}(\theta_L, \lambda | \theta_{-L}) = \mathbf{0}. \quad (4)$$

where $\tilde{r} : \mathbb{R}^{p-L} \rightarrow \mathbb{R}$ and $\tilde{\ell} : \mathbb{R}^{p_L} \times \mathbb{R}^m \rightarrow \mathbb{R}$ are differential objective function for θ_{-L} and θ_L , respectively. For the objective function \tilde{r} , the notation $(\theta_{-L} | \theta_L, \lambda)$ means we optimize over θ_{-L} while fixing θ_L and λ . The notation holds similarly for $\tilde{\ell}$ and \tilde{g} . Using this convention, we can write $\tilde{f}(\theta, \lambda) = \tilde{r}(\theta_{-L} | \theta_L, \lambda) + \tilde{\ell}(\theta_L, \lambda | \theta_{-L})$.

We assume the conditions in Theorem 2 hold, which means that we can re-write the thresholds λ as a function of the last layer weights θ_L (i.e., $\lambda = \tilde{h}_L(\theta_L)$). The specification of the function form $\tilde{h}_L(\cdot)$ holds with respect to the modified setting.

²We provide an example of a surrogate loss for our particular setting involving $f = \text{FNR}$ in the appendix.

³See Proposition 1 in Section 3 in [6].

Table 1: Minimizing false-negative rate (FNR) at a given false-positive rate (FPR) for **celebA**. The mean FNR (in %) are reported over five random trails for different methods including (CE / TFCO [2, 9] / ICO[6]) / CE + Constrained FNR (Method 1) / FPR Regularizer on θ_{-L} + Constrained FNR on θ_L (Method 2) / FPR Regularizer on θ (Ablation), respectively. Among the three proposed methods, Method 2 outperforms the other two. **Bold** indicates the best performance among the three newly proposed methods. Lower values are better.

FPR	High-cheekbones		Smiling		Wearing-lipstick	
1%	(53.5/49.0/46.9)	51.3/ 46.4 /48.2	(37.4/35.9/33.7)	37.5/ 32.7 /37.1	(44.0/42.6/37.5)	42.40/ 38.8 /40.5
2%	(44.8/40.9/39.8)	45.1/ 40.0 /41.8	(29.4/27.8/26.1)	30.2/ 26.5 /27.5	(32.7/30.4/26.7)	31.74/29.7/ 28.3
5%	(32.9/30.1/28.5)	31.9/ 28.9 /29.4	(18.7/17.0/16.9)	18.9/ 16.4 /16.6	(16.3/14.9/13.1)	15.4/15.4/ 15.1
10%	(22.9/20.4/19.7)	22.8/ 20.2 / 20.2	(11.7/10.7/10.2)	12.1/ 10.7 /11.0	(6.6/5.9/4.7)	7.0/ 5.3 /5.5

The main focus of this work is to explore different training objectives and training strategies that take the form of Eq. (4) to speed up the training procedure and get better prediction outcomes.

• **Method 1: minimizing CE for θ_{-L} and constrained FNR for θ_L** Our first proposed method is to use cross-entropy (CE) loss as the objective function for the bottom network θ_{-L} , and use constrained false-negative rate (FNR) only for updating the gradient for θ_L . This corresponds to specifying \tilde{r} , $\tilde{\ell}$, and \tilde{g} in Eq. (4) as follows:⁴

$$\tilde{r} = \text{CE}, \quad \tilde{\ell} = \widetilde{\text{FNR}}, \quad \tilde{g} = \widetilde{\text{FPR}} - \beta,$$

where $\beta \in [0, 1]$ is the target FPR rate for the constraint.

• **Method 2: minimizing FNR with regularization** Our second proposed method involves minimizing FNR for the whole network while imposing FPR constraints on the last layer weight. In addition, we impose a constraint regularizer on θ_{-L} . This corresponds to specify \tilde{r} , $\tilde{\ell}$, and \tilde{g} in Eq. (4) as follows:

$$\tilde{r} = \widetilde{\text{FNR}} + \eta |\widetilde{\text{FPR}} - \beta|, \quad \tilde{\ell} = \widetilde{\text{FNR}}, \quad \tilde{g} = \widetilde{\text{FPR}} - \beta, \quad (5)$$

where $\beta \in [0, 1]$ is the target FPR rate for the constraint and $\eta > 0$ balances between minimizing the FNR loss and the regularization term. The intuition behind adding this regularization term is to avoid trivially minimizing FNR by always predicting one at the output for any given input. Also, the regularized objective form in Eq. (5) resembles the Lagrangian method proposed in [3, 2, 9] for solving constrained optimization problems.

• **Ablation: regularization on all layers’ weights** For comparison, we also try imposing the constraint regularizer on all weights. This is equivalent of modifying $\tilde{\ell}$, the objective function for θ_L in Eq. (5) by adding a similar regularization term as in \tilde{r} :

$$\tilde{\ell} = \widetilde{\text{FNR}} + \eta |\widetilde{\text{FPR}} - \beta| \quad (6)$$

Interestingly, we observe that applying ICO on the last layer weights improves the performance over using the same regularized loss for training all weights. We provide a detailed comparison in Eq. (4) (see Table 1).

4 Experiments

We evaluate our approach on an image classification task. In particular, we use the **celebA** dataset [8]. We consider the task of minimizing the false-negative rate (FNR) at a given false-positive rate (FPR) of β . Please refer to the appendix for more details.

Experimental Results We present our experimental results in Table 1. Overall our proposed method 2 (which we refer to as “FPR Regularizer on θ_L + Constrained FNR on θ_L ”) achieves comparable performance compared to ICO, and we observe a consistent improvement of both our proposed Method 2 and Ablation (which we refer to as “FPR Regularizer on θ ”) compared to the other

⁴When it is clear from the content, we use $\tilde{\ell}$ to denote $\tilde{\ell}(\theta_{-L}|\theta_L, \lambda)$ to ease the notation. Same applies for \tilde{r} , \tilde{g} , $\widetilde{\text{FPR}}$, and $\widetilde{\text{FNR}}$.

two baseline methods (CE and TFCO [2, 9]). Our method involves training θ_L and θ_{-L} separately while fixing the other parameters and making the thresholds as a function of only θ_L . Thus, our approach is significantly cheaper compared to the baseline method ICO, which trains all parameters together, making the thresholds a function of all the parameters. This observation demonstrates that our proposed method can successfully preserve performance while reducing the cost of training compared to previous methods. Interestingly, among the two proposed methods, imposing ICO on θ_L while using FPR-regularizer on θ_{-L} leads to better performance compared to imposing FPR-regularizer on all weights θ . This supports our hypothesis that θ_{-L} and θ_L take up different roles during training, and the last layer weights θ_L play a more significant role in overall performance.

5 Conclusion and Future Work

We propose new training techniques for optimizing a popular constrained optimization problem that involves non-decomposable rate metrics. Compared to the previous approach, where the classifier thresholds are expressed as a function of all model parameters, we consider an alternative technique where the thresholds are expressed as a function of only the last layer weights of a neural network. Our empirical results show that among all proposed methods, adding a regularizer on the bottom layers while solving the constrained optimization problem for the top layer achieves a comparable performance to the previous approach, supporting our hypothesis on the important role of last layer training in deep neural networks.

References

- [1] Yatong Chen, Reilly Raab, Jialu Wang, and Yang Liu. Fairness transferability subject to bounded distribution shift. *Advances in Neural Information Processing Systems*, 2022.
- [2] Andrew Cotter, Heinrich Jiang, and Karthik Sridharan. Two-player games for efficient non-convex constrained optimization. In *Algorithmic Learning Theory*. PMLR, 2019.
- [3] Elad Eban, Mariano Schain, Alan Mackey, Ariel Gordon, Ryan Rifkin, and Gal Elidan. Scalable learning of non-decomposable objectives. In *Artificial intelligence and statistics*. PMLR, 2017.
- [4] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 2016.
- [5] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.
- [6] Abhishek Kumar, Harikrishna Narasimhan, and Andrew Cotter. Implicit rate-constrained optimization of non-decomposable objectives. *International Conference on Machine Learning*, 2021.
- [7] Jingling Li, Mozhi Zhang, Keyulu Xu, John P Dickerson, and Jimmy Ba. Noisy labels can induce good representations. *arXiv preprint arXiv:2012.12896*, 2020.
- [8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- [9] Harikrishna Narasimhan, Andrew Cotter, and Maya Gupta. Optimizing generalized rate metrics with three players. *Advances in Neural Information Processing Systems*, 32, 2019.
- [10] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: ERM may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.
- [11] Loring W Tu. Manifolds. In *An Introduction to Manifolds*. Springer, 2011.
- [12] Jimmy Wu, Yatong Chen, and Yang Liu. Metric-fair classifier derandomization. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022.
- [13] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*. PMLR, 2017.

Appendix

An Example of Surrogate Loss for FNR For our particular example involving minimizing $f = \text{FNR} = \frac{\text{FN}}{\text{total positives}}$, recall that $\text{FN}(s_{\theta|\lambda}) = \sum_{i:y_i=1} \mathbb{1}(p_i(\theta) \leq \lambda)$ where p_i is the predicted probability for the i -th positive example. We replace the indicator function with its smooth surrogate and get $\tilde{\text{FN}}(s_{\theta|\lambda}) = \sum_{i:y_i=1} \sigma_{\tau}(-(p_i - \lambda))$, where $\sigma_{\tau}(u) = 1/(1 + \exp(-\tau u))$ denotes a temperature scaled sigmoid function. This gives us $\tilde{f} = \frac{\tilde{\text{FN}}}{\text{total positives}}$ as the surrogate loss.

Implicit Function Theorem on θ Here, we provide the statement for the implicit function theorem and its modification for the last layer weights:

Theorem 1 (Implicit Function Theorem on θ [11], *informal*). *For any $(\theta_0, \lambda_0) \in U \subseteq \mathbb{R}^p \times \mathbb{R}^m$ pair that satisfies $\tilde{g}(\theta_0, \lambda_0) = \mathbf{0}$, if the determinant of the Jacobian matrix is nonzero, i.e., $\det[\frac{\partial \tilde{g}^i}{\partial \theta^j}(\theta_0, \lambda_0)] \neq 0$, then there exists a neighborhood $\Theta \times \Lambda$ of (θ_0, λ_0) in U and a unique function $\tilde{h} : \Theta \Rightarrow \Lambda$:*

$$\tilde{g}(\theta, \lambda) = \mathbf{0} \Leftrightarrow \lambda = \tilde{h}(\theta). \quad (7)$$

Theorem 2 (Implicit Function Theorem on θ_L). *For any $(\theta_0, \lambda_0) \in U \subseteq \mathbb{R}^p \times \mathbb{R}^m$ pair that satisfies $\tilde{g}(\theta_0, \lambda_0) = \mathbf{0}$, if the determinant of the Jacobian matrix w.r.t θ_L is nonzero, i.e. $\det[\frac{\partial \tilde{g}^i}{\partial \theta_L^j}(\theta_0, \lambda_0)] \neq 0$, then there exists a neighborhood $\Theta \times \Lambda$ of (θ_0, λ_0) in U and a unique function $\tilde{h}_L : \Theta \Rightarrow \Lambda$ such that*

$$\tilde{g}(\theta, \lambda) = \mathbf{0} \Leftrightarrow \lambda = \tilde{h}_L(\theta_L).$$

Gradient Update Rule for Eq. (3) To compute a local derivative for $\tilde{f}(\theta, \tilde{h}(\theta))$ within the neighborhood of θ_0 using Theorem 1, we have the following update rule:

$$\nabla_{\theta} \tilde{f}(\theta, \tilde{h}(\theta)) = \nabla_{\theta} \tilde{f}(\theta, \lambda) + \frac{\partial \tilde{f}(\theta, \lambda)}{\partial \lambda} \nabla_{\theta} \tilde{h}(\theta). \quad (8)$$

We will further need the derivative of the implicit function \tilde{h} w.r.t. θ , i.e., $\nabla_{\theta} \tilde{h}(\theta)$. Since $\tilde{g}(\theta, \tilde{h}(\theta)) = \mathbf{0}$ in the neighborhood of θ_0 , we have:

$$\nabla_{\theta} \tilde{g}(\theta, \lambda) + \frac{\partial \tilde{g}(\theta, \lambda)}{\partial \lambda} \nabla_{\theta} \tilde{h}(\theta) = \mathbf{0} \Rightarrow \nabla_{\theta} \tilde{h}(\theta) = -\frac{\nabla_{\theta} \tilde{g}(\theta, \lambda)}{\frac{\partial \tilde{g}(\theta, \lambda)}{\partial \lambda}}. \quad (9)$$

Plugging Eq. (9) back to Eq. (8), we can get the final gradient for the model parameter θ . See section 3.3 of [6] for a more detailed derivation for the update rule of θ and λ .

Gradient Update Rule for Eq. (4) Furthermore, we can break the gradient of Eq. (4) into two parts: the gradient w.r.t θ_{-L} and the gradient w.r.t θ_L . We thus obtain the following gradients:

$$\partial_{\theta_{-L}} \tilde{f}(\theta, \lambda) = \nabla_{\theta_{-L}} \tilde{r}(\theta_{-L} | \theta_L, \lambda) \quad \text{and} \quad \partial_{\theta_L} \tilde{f}(\theta, \lambda) = \nabla_{\theta_L} \tilde{\ell}(\theta_L, \lambda | \theta_{-L}) + \frac{\partial \tilde{\ell}}{\partial \lambda} \nabla_{\theta_L} \tilde{h}_L(\theta_L). \quad (10)$$

Experimental Details Following [6], we choose three binary attributes (i.e., High-cheekbones, Smiling, and Wearing-lipsticks) as target attributes for our experiments and train a binary classifier for each attribute. Similar to [6], we use a 6-layer neural network with 5 convolutional layers with 128, 256, 512, 512 filters, respectively, and we use ReLU as our activation functions and apply batch normalization layers in the networks. We conduct 5 random trials for each experiment and report the average values of the metric.