Leverage Class-Specific Accuracy to Guide Data Generation for Improving Image Classification

Jay Gala [1] **Pengtao Xie** [1 2]

Abstract

In many image classification applications, the number of labeled training images is limited, which leads to model overfitting. To mitigate the lack of training data, deep generative models have been leveraged to generate synthetic training data. However, existing methods generate data for individual classes based on how much training data they have without considering their actual data needs. To address this limitation, we propose needs-aware image generation, which automatically identifies the different data needs of individual classes based on their classification performance and divides a limited data generation budget into these classes according to their needs. We propose a multi-level optimization based framework which performs four learning stages in an end-to-end manner. Experiments on both imbalanced and balanced classification datasets demonstrate the effectiveness of our proposed method.

1. Introduction

Image classification (He et al., 2016b) is a fundamental problem in computer vision and machine learning. In many applications, due to the high annotation cost (Tian et al., 2020), it is challenging to obtain large amounts of labeled training images. Trained on small-sized datasets, models tend to overfit the training data and generalize poorly on test data (Neyshabur et al., 2017). To address this problem, many approaches (Azizi et al., 2023; Trabucco et al., 2023; Zhou et al., 2023a; Zheng et al., 2023; Li et al., 2023; Antoniou et al., 2017; Mariani et al., 2018; Such et al., 2020) have been proposed to leverage deep generative models (Ho et al., 2020; Song et al., 2021; Goodfellow et al., 2014; Kingma et al., 2019; Rezende & Mohamed, 2015) to generate synthetic training images, augmenting the limited real training data, and have demonstrated promising effectiveness.

Different classes within a classification task may require varying amounts of synthetic training data, with these needs often being dictated by the classification performance of individual classes. For instance, in class-imbalanced datasets, the number of training examples can vary greatly between classes, leading to performance disparities (Cui et al., 2019b; Huang et al., 2016; Zhang et al., 2018; Kim et al., 2020). Those classes with poorer performance often need more synthetic training data. Even in class-balanced datasets, classification accuracy can differ between classes (He et al., 2016a; 2019). Some classes may possess more complex or diverse features, making them more challenging for classifiers to learn and accurately distinguish. This variation in classification accuracy results in different classes having distinct requirements for synthetic data.

Due to computation and memory constraints, there is a limit to the number of synthetic images that can be generated, referred to as the data generation budget. It is crucial to allocate more of this budget to worse-performing classes that require more synthetic training data and less to better-performing ones. However, existing data generation methods (Azizi et al., 2023; Trabucco et al., 2023; Zhou et al., 2023a; Zheng et al., 2023; Li et al., 2023; Antoniou et al., 2017; Mariani et al., 2018; Such et al., 2020) cannot properly divide a limited data generation budget between classes based on their actual data needs. They simply distribute the budget based on the number of training examples in each class, assuming that a class with fewer training examples needs more synthetic training data. However, this assumption does not always hold in many real-world applications. For example, a class with similar data examples, which are distinct from other classes, may already exhibit excellent classification performance even if the amount of training data is limited. In such a case, there would be less need for synthetic training data for this class.

To address the limitations of existing methods, we propose a multi-level optimization (MLO) (Sato et al., 2021) based framework for needs-aware data generation. Instead of relying on the number of training images in each class, our framework assesses the data needs of individual classes

[1]University of California San Diego [2]Mohamed bin Zayed University of Artificial Intelligence. Correspondence to: Pengtao Xie <p1xie@ucsd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

based on their classification performance. The framework executes an end-to-end learning process, consisting of three stages: 1) training an initial classifier to measure the performance of individual classes, thereby determining their data needs; 2) allocating a limited data generation budget among classes based on these identified needs, and subsequently generating data in accordance with this allocation; and 3) utilizing the generated data to enhance classification performance. Each stage corresponds to one level of optimization problem in the MLO formulation and different stages are performed jointly by solving different levels of interdependent optimization problems collectively, to achieve the best overall performance.

The major contributions of this work include:

- We propose a multi-level optimization based framework which conducts needs-aware data generation to mitigate the lack of labeled training data in image classification. Compared with previous works, our method has two major novelties. First, previous works are based on a simple assumption that infrequent classes need more data. Such an assumption cannot cover the cases that balanced classes may also have imbalanced performance. In contrast, our method goes beyond such a simplified assumption. It measures the performance of individual classes and generates more data for worse-performing classes. Second, some baselines perform a sequence of tasks, including measuring class-specific performance, generating data, using generated data to train classifiers, in a separate manner. Later tasks have no influence on earlier tasks, which leads to suboptimal overall performance. In contrast, our method performs the three tasks end-to-end in a multi-level optimization framework, where all tasks can mutually influence each other to achieve the overall best performance.
- We demonstrate our framework's effectiveness on both imbalanced and balanced image classification datasets.

2. Related Works

Synthetic data generation. The feasibility of utilizing generated data to train more accurate machine learning models has been demonstrated in several works (Azizi et al., 2023; Trabucco et al., 2023; Zhou et al., 2023a; Zheng et al., 2023; Li et al., 2023). Many deep generative models have been developed for data generation (or augmentation) (Azizi et al., 2023; Trabucco et al., 2023; Zhou et al., 2023a), including diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Saharia et al., 2022), generative adversarial networks (Goodfellow et al., 2014; Mirza & Osindero, 2014; Zhu et al., 2017; Lin et al., 2022), variational autoencoders (Kingma & Welling, 2014; Bowman et al., 2016), normalizing flows (Tabak & Vanden-Eijnden, 2010; Rezende & Mohamed, 2015), GPT3 (Brown et al., 2020), etc. Particularly in the field of image generation, numerous methods have been devised. For example, Antoniou et al. (Antoniou et al., 2017) leveraged conditional Generative Adversarial Networks (GANs) to perform data augmentation. Mariani et al. (Mariani et al., 2018) proposed a balancing GAN to generate images for infrequent classes in imbalanced datasets to achieve balance among classes. Frid-Adar et al. (Frid-Adar et al., 2018) leveraged GANs to generate synthetic medical images for liver lesion classification. Huang et al. (Huang et al., 2018) proposed a GAN-based data augmentation method for cross domain adaptation. Such et al. (Such et al., 2020) proposed to generate synthetic images for improving neural architecture search. Li et al. (Li et al., 2022e) proposed a GAN-based method for extremely imbalanced data augmentation. Different from these methods which determine how much data to generate for a class based on the number of real training examples that the class has, our method measures classes' actual needs of data based on their classification performance and generates data according to the needs.

Bi-level and multi-level optimization. Our method is based on multi-level optimization, which is an extension of bi-level optimization (BLO) (Dempe, 2002). BLO has been applied for formulating many machine learning methods, including model-agnostic meta learning (Finn et al., 2017), neural architecture search (Liu et al., 2019a), hyperparameter tuning (Feurer et al., 2015), data reweighting (Shu et al., 2019; Ren et al., 2020; Wang et al., 2020b), etc. In these methods, there are learnable model weights and meta parameters. Model weights are trained by minimizing a training loss while meta parameters are learned by minimizing a validation loss. The optimization problem defined on model weights is on the constraint of the optimization problem defined on meta parameters.

3. Method

Our framework (Figure 1) consists of an initial classifier W_1, a refined classifier W_2, and a class-to-image conditional generative adversarial network (CGAN) (Mirza & Osindero, 2014; Brock et al., 2018). They share learnable meta parameters A (e.g., neural architectures). The CGAN takes a class name as input and generates a set of images belonging to this class. Let $D^{(tr)}_{cls} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training set of an image classification dataset and $D^{(val)}_{cls}$ denote a validation set, where x_i is an input image and y_i is its class label. After training W_1 on $D^{(tr)}_{cls}$, we measure its performance for individual classes on $D^{(val)}_{cls}$ and divide a data generation budget based on these classes' performance. If W_1 does not perform well on a certain class c, the CGAN will generate

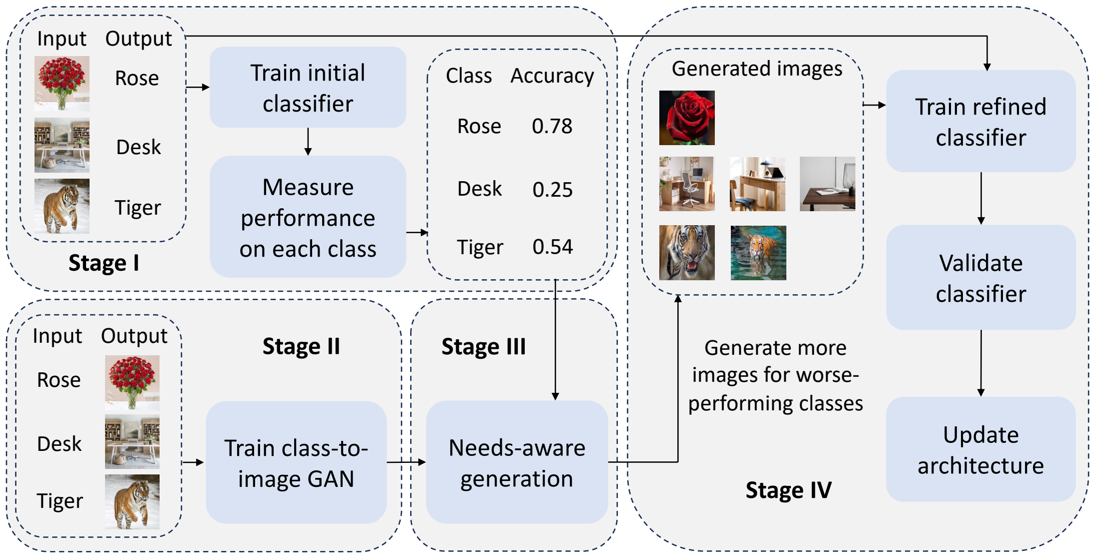


Figure 1. Overview of our framework.

more images belonging to c. Then these generated images will be used to train W_2. Table 1 shows notations.

Table 1. Notations of our method.

Notation	Meaning
W_1	Weight parameters of the initial classifier
W_2	Weight parameters of the refined classifier
A	Meta parameters
$D_{cls}^{(tr)}$	Training set of an image classification dataset
$D_{cls}^{(val)}$	Validation set of an image classification dataset
x_i	Input image
y_i	Class label
L_{cls}	Classification loss
G	Generator in the conditional GAN
H	Weights of the discriminator in the conditional GAN
D_{cgan}	Training data for the conditional GAN
L_{gan}	Loss of the conditional GAN

3.1. A Multi-level Optimization Based Framework

Our framework consists of the following four stages which are performed end-to-end.

Stage I. At the first stage, we train W_1 on $D_{cls}^{(tr)}$ by minimizing a cross-entropy based classification loss L_{cls}. The corresponding optimization problem is:

$$W_1^*(A) = \min_{W_1} L_{cls}(W_1, A, D_{cls}^{(tr)}). \quad (1)$$

The meta parameters A are needed to define the training loss. They are tentatively fixed at this stage and will be updated later on. $W_1^*(A)$ denotes that W_1^* depends on A, since W_1^* depends on L_{cls} which is a function of A.

Stage II. At the second stage, we train a conditional GAN (CGAN) consisting of a generator and a discriminator (Goodfellow et al., 2014). The generator G takes a class name as input and generates an image. The discriminator H takes an image as input and predicts whether it is synthetic or real. The training dataset for CGAN is $D_{cgan} = \{(y_i, x_i)\}_{i=1}^N$, which is obtained by switching the order of images and labels in $D_{cls}^{(tr)}$. When training a classifier with the image classification dataset $D_{cls}^{(tr)}$, images serve as the input, and their corresponding class labels are the output. Conversely, when training a conditional GAN model using D_{cgan}, the roles are reversed: class labels are the input, and images are the output. While both $D_{cls}^{(tr)}$ and D_{cgan} utilize the same collection of images and labels, their assignments of inputs and outputs are distinct. We train G and H by solving a mini-max problem where the objective L_{gan} is from GAN (Goodfellow et al., 2014):

$$G^*(A), H^* = \min_G \max_H L_{gan}(G, H, A, D_{cgan}). \quad (2)$$

Similar to Eq.(1), the conditional GAN's meta parameters A (shared with the classifiers) are tentatively fixed at this stage and will be updated later on.

Stage III. In the third stage, we measure the validation performance of the optimally trained $W_1^*(A)$ on the validation set $D_{cls}^{(val)}$. The budget of generating synthetic training data is divided into classes based on their performance (validation losses). For worse-performing classes (with larger validation losses), more synthetic images are generated. Specifically, we generate an image for class c with a probability proportional to the average validation loss $l_c(W_1^*(A), A, D_{cls}^{(val)})$ of c. We define a categorical variable s which can have values from 1 to C (C is the total number of classes). $s = c$ denotes that class c is selected. The probability of $p(s = c)$ is defined as:

$$p(s=c) = \frac{l_c(W_1^*(A), A, D_{cls}^{(val)})}{\sum_{b=1}^{C} l_b(W_1^*(A), A, D_{cls}^{(val)})}. \tag{3}$$

The following procedure is performed to generate an image. First, we sample a class c from $p(s)$ according to the probability defined in Eq.(3). Then we feed the class name n_c of class c and a random noise vector δ into the generator f parameterized by $G^*(A)$, which generates an image $\hat{x} = f(n_c, \delta, G^*(A))$. Given a total budget of generating M images, this procedure repeats M times, yielding a generated set $\{(\hat{x}_m, o_m)\}_{m=1}^M$ where $\hat{x}_m$ is a generated image and o_m is the class from which $\hat{x}_m$ is generated. These generated images are then used to train W_2, where training loss is defined as $\sum_{m=1}^M L_{cls}(W_2, A, \hat{x}_m, o_m)$. We use the Gumbel-Softmax (Jang et al., 2016) reparameterization to approximate $\sum_{m=1}^M L_{cls}(W_2, A, \hat{x}_m, o_m)$. Let b_m denote $\exp((\log p(s = o_m) + g_1^{(m)})/T)/(\exp((\log p(s = o_m) + g_1^{(m)})/T) + \exp((\log(1 - p(s = o_m)) + g_2^{(m)})/T))$, where $p(s = o_m)$ is given in Eq.(3) and $(g_1^{(m)}, g_2^{(m)})$ are samples drawn from a Gumbel distribution. $T \in (0, \infty)$ is a temperature parameter. During training, T is set via annealing. We have the approximation as:

$$\begin{aligned} &L_a(W_2, A, W_1^*(A), G^*(A)) \\ &= \sum_{m=1}^{M} b_m L_{cls}(W_2, A, \hat{x}_m, o_m). \end{aligned} \tag{4}$$

In this stage, the optimization problem is:

$$\begin{aligned} &W_2^*(A, W_1^*(A), G^*(A)) \\ &= \min_{W_2} L_{cls}(W_2, A, D_{cls}^{(tr)}) + \lambda L_a(W_2, A, W_1^*(A), G^*(A)) \end{aligned} \tag{5}$$

where λ is a tradeoff parameter.

Stage IV. In the fourth stage, we validate $W_2^*(A, W_1^*(A), G^*(A))$ and $W_1^*(A)$ on the validation dataset and learn A by minimizing validation losses:

$$\begin{aligned} \min_A \; & L_{cls}(W_2^*(A, W_1^*(A), G^*(A)), A, D_{cls}^{(val)}) \\ & + \gamma L_{cls}(W_1^*(A), A, D_{cls}^{(val)}), \end{aligned} \tag{6}$$

where γ is a tradeoff parameter.

A multi-level optimization framework. Putting these pieces together, we have the following overall formulation.

$$\begin{aligned} \min_A \; & L_{cls}(W_2^*(A, W_1^*(A), G^*(A)), A, D_{cls}^{(val)}) \\ & + \gamma L_{cls}(W_1^*(A), A, D_{cls}^{(val)}) \\ s.t. \; & W_2^*(A, W_1^*(A), G^*(A)) \\ & = \min_{W_2} L_{cls}(W_2, A, D_{cls}^{(tr)}) + \lambda L_a(W_2, A, W_1^*(A), G^*(A)) \\ & G^*(A), H^* = \min_G \max_H L_{gan}(G, H, A, D_{cgan}) \\ & W_1^*(A) = \min_{W_1} L_{cls}(W_1, A, D_{cls}^{(tr)}) \end{aligned} \tag{7}$$

The optimization algorithm is deferred to Appendix A.

3.2. Reducing Search Cost and Memory Cost

Before running our framework in Eq.(7), we pretrain the CGAN on D_{cgan}. The sum of runtime of these two steps is reported as the total computation cost of our method in the sequel. The following techniques are applied to reduce computation and memory costs.

- Parameter tying is performed. For W_1, W_2, and H, each of them consists of a data encoder and a classification head. We let W_1, W_2 and H share the same data encoder and have different classification heads. Typically, the classification heads are set to be linear vectors with a few thousand dimensions or less. Their memory consumption and computational costs are neglectable compared with those of the data encoder. With parameter tying, the total number of model parameters and the computation cost of updating these parameters can be substantially reduced.

- Instead of updating the meta parameters A and the generator G in each iteration (mini-batch), we reduce their update frequency to every 8 iterations. Empirically, this substantially reduced computation costs while achieving similar convergence quality compared to updating A and G in each iteration. The update frequency of other parameters is still set to be per-iteration.

- On data generated by the CGAN in stage III, a decorrelation regularizer (Cogswell et al., 2015) is applied. With this regularizer, our algorithm converged to a low loss value in 25 epochs. To reach a similar loss value, it took about 50 epochs without this regularizer.

- By leveraging the core-sets based approach for speeding up GAN training (Sinha et al., 2020a) and top-k training strategy of GANs (Sinha et al., 2020b) and decreasing epoch number by half, we managed to greatly decrease the pretraining time of the CGAN on various datasets.

Table 2. Test accuracy (%) on imbalanced CIFAR datasets. Results of baselines in the last panel are taken from their original papers.

Dataset	CIFAR10-LT					CIFAR100-LT				
Imbalance factor	200	100	50	20	10	200	100	50	20	10
Focal loss (Lin et al., 2017)	65.29	70.38	76.71	82.76	86.66	35.62	38.41	44.32	51.95	55.78
Class balance (Cui et al., 2019b)	68.89	74.57	79.27	84.36	87.49	36.23	39.60	45.32	52.59	57.99
L2RW (Ren et al., 2018)	66.51	74.16	78.93	82.12	85.19	33.38	40.23	44.44	51.64	53.73
Meta weight (Shu et al., 2019)	68.91	75.21	80.06	84.94	87.84	37.91	42.09	46.74	54.37	58.46
Vanilla ResNet (He et al., 2016a)	65.68	70.36	74.81	82.23	86.39	34.84	38.32	43.85	51.44	55.71
Reciprocal	68.85	75.96	79.67	83.29	87.50	37.27	42.36	47.13	54.05	57.92
Separate	67.29	76.04	80.21	82.79	87.94	37.87	41.95	46.36	54.28	57.70
MTL	68.14	76.11	79.33	83.92	87.46	37.95	42.49	47.90	54.74	58.39
Ours	**73.41**	**79.25**	**83.09**	**87.52**	**90.76**	**40.95**	**46.10**	**50.77**	**57.28**	**61.62**
MiSLAS (Zhong et al., 2021b)	77.31	82.06	85.16	-	90.00	42.33	47.50	52.62	-	63.2
GCL (Li et al., 2022b)	79.03	82.68	85.46	-	-	44.88	48.71	53.55	-	-
GCL + CR (Ma et al., 2023)	79.9	83.5	86.8	-	-	45.6	49.8	55.1	-	-
RIDE (Wang et al., 2020a)	-	-	-	-	-	-	48.6	51.4	-	59.8
RIDE + CMO (Park et al., 2022)	-	-	-	-	-	-	50.0	53.0	-	60.2
RIDE + CMO + CR (Ma et al., 2023)	-	-	-	-	-	-	50.7	54.3	-	61.4
CC-SAM (Zhou et al., 2023b)	80.94	83.92	86.22	-	-	45.66	50.83	53.91	-	-
CC-SAM + Ours	**81.89**	**84.77**	**88.15**	-	-	**47.09**	**52.36**	**55.80**	-	-

4. Experiments

We evaluated our method on four imbalanced dataset and one balanced dataset. This diverse selection was intended to test our method's ability to identify the performance-based data needs of classes, irrespective of the class balance within these datasets. We used BigGAN (Brock et al., 2018) as the CGAN. Weight parameters of BigGAN were optimized using Adam (Kingma & Ba, 2014a), with a learning rate of 2e-4. The tradeoff parameters λ and γ in Eq.(7) were set to 1. We tuned them in $\{0.1, 0.5, 1, 2, 3\}$ on 5K held-out examples. The budget M of data examples to be generated was set to be the same as the number of real training examples.

4.1. Experiments on Imbalanced Datasets

Datasets. We used three class-imbalanced datasets: CIFAR10-LT (Cui et al., 2019b), CIFAR100-LT (Cui et al., 2019b), and ImageNet-LT (Liu et al., 2019b), where LT denotes long tail. They were curated from the original balanced CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky & Hinton, 2010), and ImageNet (Deng et al., 2009), by randomly sampling skewed number of images from different classes. CIFAR10-LT and CIFAR100-LT have different versions with different imbalance factors, which are defined as the ratio between the numbers of training examples in the largest and smallest classes. ImageNet-LT has an imbalance factor of 256. It contains 115.8K images from 1,000 categories.

Experimental Settings. We set the meta parameters A in our method to be the shared weight parameters of the last two layers in the initial classifier, refined classifier, and the CGAN's discriminator. The rationale is as follows. A typical neural network-based image classifier is composed of two main components: an image encoder that extracts representations from an input image, and a classification head that predicts the class label. Generally, the classification head consists of the last two layers of the network. Our method learns the image encoder using the training dataset and the classification head using the validation set.

For experiments on CIFAR10-LT and CIFAR100-LT, we used ResNet-32 (He et al., 2016a) as the classifier, following (Shu et al., 2019). The number of epochs was set to 100. The initial learning rate was set to 0.001, which was reduced by 10 after 80 and 90 epochs. Adam (Kingma & Ba, 2014b) was used as the optimizer. Batch size was set to 64. For experiments on ImageNet-LT, we used ViT-B (Dosovitskiy et al., 2021) as the classifier, with an image patch size of 16. The image encoder in ViT-B employs a Transformer architecture with 12 layers, alternating between multiheaded self-attention and multi-layer perceptron (MLP) blocks. The classification head in ViT-B is an MLP with one hidden layer and a logits layer.

We compared our method with the following baselines.

- Reciprocal (Mariani et al., 2018; Li et al., 2022e). For a class with n training examples, the number of synthetic images generated for this class is proportional to $\frac{1}{n}$.

Table 3. Class-specific accuracy and generated data for CIFAR-10 (imbalance factor = 50).

Class	Number of real training images	Accuracy before using generated data	Number of generated images	Accuracy after using generated data
Truck	5000	97.3	237	98.7
Cat	3237	90.9	394	93.1
Frog	2096	91.4	466	92.9
Car	1357	92.6	401	94.3
Plane	878	83.3	924	87.0
Dog	568	59.5	2280	78.5
Horse	368	74.4	1468	83.4
Ship	238	78.1	1173	88.1
Bird	154	41.8	3245	58.6
Deer	100	38.6	3407	55.9
Mean	-	74.8	-	83.1
Standard deviation	-	21.3	-	14.8

- Separate (Antoniou et al., 2017; Frid-Adar et al., 2018). CGAN and classification model are trained separately. We use a pretrained CGAN to generate images instead of training it in our framework.
- Multi-task learning (MTL). CGAN and the refined classifier W_2 are trained by minimizing the weighted sum of their loss functions in an MTL framework.

Results on Imbalanced CIFAR Datasets. Table 2 shows the results. From this table, we make the following observations. **First**, our framework which generates synthetic labeled images to train ResNet achieves significantly better performance than vanilla ResNet which does not use synthetic training data. This demonstrates that labeled images generated by our framework are effective for training better classification models. **Second**, our method outperforms Reciprocal significantly, which demonstrates that our method's strategy of dividing a data generation budget into classes is superior to Reciprocal. Reciprocal makes the assumption that classes with less training data would perform worse and hence need more synthetic training data. As discussed in Section 1, this assumption is not always true. In contrast, our method does not rely on this assumption and dynamically measures individual classes' classification performance and generates more data for worse-performing classes. **Third**, our method outperforms Separate. In Separate, CGAN and classification models are trained separately, where the CGAN training is not guided by classification performance. In contrast, our method trains CGAN and classification models jointly. The CGAN is trained to generate data that is effective for improving the classification model. **Fourth**, our method performs better than MTL. In MTL, CGAN and W_2 are trained jointly by minimizing the weighted sum of their losses. It is difficult to properly balance these two loss terms: achieving more decrease of one loss renders less decrease of the other loss. Our method avoids this problem by training CGAN and W_2 in two separate optimization problems (but still in an end-to-end framework).

We compared with another six recent baselines, including Mixup Shifted Label-Aware Smoothing (MiSLAS) (Zhong et al., 2021b), Gaussian Clouded Logit (GCL) (Li et al., 2022b), Curvature Regularization (CR) (Ma et al., 2023), RoutIng Diverse Experts (RIDE) (Wang et al., 2020a), Context-rich Minority Oversampling (CMO) (Park et al., 2022), and Class-Conditional Sharpness-Aware Minimization (CC-SAM) (Zhou et al., 2023b). Following (Wang et al., 2020a; Park et al., 2022), the number of experts in RIDE is set to 3. Our method is applied on top of CC-SAM. The last panel of Table 2 shows the comparison between our method and these baselines. The performance numbers of baseline methods are taken from their original papers. As can be seen, our method outperforms these baselines, which further demonstrates the effectiveness of our proposed needs-aware image generation mechanism.

Table 3 presents the number of images generated by our method for each class in CIFAR-10 (imbalance factor = 10), alongside the accuracy before and after using the generated data for training. Notably, there is an approximate inverse relationship between the number of generated images and the accuracy of each class before using generated data, indicating that our method preferentially generates more images for worse-performing classes. After using the generated images for training, there is a significant increase of accuracy for each class. The magnitude of increase is more prominent on worse-performing classes. This demonstrates that the images generated by our method are effective for improv-

Table 4. Top-1 accuracy (%) on ImageNet-LT. ResNet-50 is used as the classifier. Results marked with † use ResNeXt-50 (Xie et al., 2017) as the classifier.

Method	Many	Med.	Few	Acc
CE (Cui et al., 2019a)	64.0	33.8	5.8	41.6
LDAM (Cao et al., 2019)	60.4	46.9	30.7	49.8
c-RT (Kang et al., 2020)	61.8	46.2	27.3	49.6
τ-Norm (Kang et al., 2020)	59.1	46.9	30.7	49.4
Causal (Tang et al., 2020)	62.7	48.8	31.6	51.8
Logit Adj. (Menon et al., 2021)	61.1	47.5	27.6	50.1
RIDE(4E)† (Wang et al., 2021)	68.3	53.5	35.9	56.8
MiSLAS (Zhong et al., 2021a)	62.9	50.7	34.3	52.7
DisAlign (Zhang et al., 2021a)	61.3	52.2	31.4	52.9
ACE† (Cai et al., 2021)	71.7	54.6	23.5	56.6
PaCo† (Cui et al., 2021)	68.0	56.4	37.2	58.2
TADE† (Zhang et al., 2021b)	66.5	57.0	43.5	58.8
TSC (Li et al., 2022d)	63.5	49.7	30.4	52.4
GCL (Li et al., 2022c)	63.0	52.7	37.1	54.5
TLC (Diego, 2022)	68.9	55.7	40.8	55.1
BCL† (Zhu et al., 2022)	67.6	54.6	36.6	57.2
NCL (Li et al., 2022a)	67.3	55.4	39.0	57.7
SAFA (Hong et al., 2022)	63.8	49.9	33.4	53.1
DOC (Wang et al., 2022a)	65.1	52.8	34.2	55.0
DLSA (Xu et al., 2022)	67.8	54.5	38.8	57.5
LiVT (Xu et al., 2023)	73.6	56.4	41.0	60.9
LiVT + Ours	**74.3**	**57.8**	**43.9**	**61.8**

ing classification performance, particularly for the worse-performing ones. Furthermore, after training with generated images, there is a noticeable decrease in the performance variance among the classes, with the standard deviation of the accuracy dropping from 21.3% to 14.8%. This reduction underscores the effectiveness of our method in mitigating the performance disparities across classes by generating more training data for the worse-performing ones.

Another noteworthy finding is that classes with fewer real training examples do not necessarily have worse performance. For instance, although there are fewer real training samples in the horse category compared to the dog category, the accuracy for the horse category before using generated data surpasses that of the dog category. This underscores the paper's premise that data generation budget should be allocated based on the performance of each class rather than their frequency.

Results on Imbalanced ImageNet. Table 4 shows results on the imbalanced ImageNet-LT dataset. Following (Liu et al., 2019b), besides reporting top-1 accuracy on the entire test set, we also report accuracy on three categories of classes: Many (classes with more than 100 images), Medium (classes with 20 to 100 images), and Few (classes with less than 20 images). After applying our method to LiVT (Xu et al., 2023), the classification accuracy is improved notably. In particular, the improvement is more prominent on classes marked as Few. This further demonstrates that the data synthesized by our method is effective for improving the performance of imbalanced classification.

4.2. Experiments on Balanced Dataset

To demonstrate the broad applicability of our method to various scenarios, we used a setting which automatically searches for neural architectures (Zoph & Le, 2017; Liu et al., 2019a) to perform image classification. Neural architecture search (NAS) requires more training data than training fixed-architecture neural networks since NAS involves optimizing both the architecture and the weights of a neural network. This requires more data to sufficiently explore and evaluate the search space of possible architectures. Following the standard practice of the NAS literature which treats the architecture as hyperparameters, our method sets the meta parameters A to be the shared learnable architecture of the two classifiers and the CGAN's discriminator.

Dataset. We used the original ImageNet (Deng et al., 2009) dataset, which is class balanced. It contains 1.3M training images and 50K test images, belonging to 1000 classes.

Experimental Settings. Similar to (Xu et al., 2020), from the original 1.3M training images, we randomly sample 10% as training data and another 2.5% as validation data, to perform architecture search. Following (Liu et al., 2019a), each experiment consists of two phrases - architecture search and architecture evaluation. In the search phrase, an optimal cell is searched. In the evaluation phase, the cell is used to compose a larger network, which is trained from scratch on the combination of training and validation data. For the search algorithm and space, we followed PC-DARTS (Xu et al., 2020), which is computationally efficient. We compared our method with the Separate and MTL baselines mentioned in Section 4.1. In addition, we compared with a baseline (Such et al., 2020) denoted as Equal where an equal number of synthetic images are generated for each class. We conducted all experiments on Nvidia 1080Ti GPUs. Each experiment ran for 10 times with different random initializations. We report the mean and standard deviation of results.

Results. In Table 5, we show the top-1 and top-5 errors on the test set of ImageNet. From this table, we make the following observations. First, when applied to PCDARTS, our method achieves significant improvement over this baseline. This shows that the data generated by our method is highly useful for improving image classification. Second, our method performs better than Equal, which further demonstrates the effectiveness of our proposed needs-awarere data generation mechanism. In Equal, the same number of images are generated for different classes. For

Table 5. Top-1 and top-5 classification errors (%) on the test set of ImageNet. Results marked with * are taken from DARTS$^-$ (Chu et al., 2020), DrNAS (Chen et al., 2020), and β-DARTS (Ye et al., 2022).

Method	Top-1	Top-5
*Inception-v1 (Szegedy et al., 2015)	30.2	10.1
*MobileNet (Howard et al., 2017)	29.4	10.5
*ShuffleNet 2× (v2) (Ma et al., 2018)	25.1	7.6
*NASNet-A (Zoph et al., 2018)	26.0	8.4
*AmoebaNet-C (Real et al., 2019)	24.3	7.6
*SNAS-CIFAR10 (Xie et al., 2019)	27.3	9.2
*DFNAS (Liu et al., 2022)	26.4	-
*DSNAS-ImageNet (Hu et al., 2020)	25.7	8.1
*PCDARTS-CIFAR10 (Xu et al., 2020)	25.1	7.8
*ProxylessNAS-ImageNet (Cai et al., 2019)	24.9	7.5
*FairDARTS-ImageNet (Chu et al., 2019)	24.4	7.4
*DOTS (Gu et al., 2021)	24.3	7.4
*ZARTS (Wang et al., 2022b)	24.3	-
*DrNAS-ImageNet (Chen et al., 2020)	24.2	7.3
*PR-DARTS (Zhou et al., 2020)	24.1	7.3
*DARTS$^-$-ImageNet (Chu et al., 2020)	23.8	7.0
$*\beta$-DARTS (Ye et al., 2022)	23.9	7.0
*RF-PCDARTS (Zhang et al., 2023)	23.9	7.1
*Pcdarts (Xu et al., 2020)	24.2	7.3
Equal-pcdarts	23.9	7.2
Separate-pcdarts	24.1	7.3
MTL-pcdarts	24.4	7.4
Ours-pcdarts	**23.2**	**6.6**

Table 6. Test accuracy (%) achieved by performing different stages sequentially and end-to-end, on imbalanced CIFAR datasets, with an imbalance factor of 200.

Method	CIFAR10-LT	CIFA100-LT
Sequential	69.24	37.79
End-to-end (ours)	**73.41**	**40.95**

certain classes which already have good performance, it is a waste of effort to generate more images for them. In contrast, our method focuses on generating more images for worse-performing classes, which can help to improve the performance of worse-performing classes. Third, our method outperforms Separate and MTL. The analysis of reasons is similar to that for Table 2. Fourth, our method outperforms all other baselines.

4.3. Ablation Studies

Ablation on End-to-End Learning. To investigate the effectiveness of performing all stages in our framework in an end-to-end manner, we compare with an ablation setting called Sequential, where different stages are performed sequentially. We first train a classifier and CGAN. Then we measure the validation performance of the classifier on individual classes. For classes with worse performance, more data is generated for them using the CGAN. On generated data, we train another classifier. The study was performed on imbalanced CIFAR datasets, both with an imbalance factor of 200. Table 6 shows the results. Our end-to-end method achieves higher test accuracy than Sequential. The reason is: in our method, W_1 and W_2 mutually influence each other during training. In the Sequential ablation setting, W_2 cannot influence W_1.

Table 7. Test accuracy (%) in the ablation study of different CGANs, on imbalanced CIFAR datasets, with an imbalance factor of 200.

Method	Imb. CIFAR-10	Imb. CIFAR-100
VCGAN	72.33	39.88
ACGAN	72.06	40.07
BigGAN	**73.41**	**40.95**

Table 8. Human evaluation of generated images.

Method	Natural	Correct
Ours	**4.2**±0.3	**4.3**±0.3
Separate	3.6±0.4	3.7±0.3
Equal	3.7±0.3	3.9±0.4

Ablation on CGANs. We experimented with different CGANs, including the vanilla conditional GAN (Mirza & Osindero, 2014) (VCGAN), ACGAN (Odena et al., 2017), BigGAN (Brock et al., 2018), and investigated how they affect classification performance on test data. We conducted the experiments on the imbalanced CIFAR datasets, with an imbalance factor of 200. Table 7 shows the results. As can be seen, BigGAN performs better than VCGAN and ACGAN. This is because BigGAN has better capability in generating high-fidelity data.

4.4. Evaluation of Generated Images

The evaluation was conducted on images generated by BigGAN in our method trained on the balanced ImageNet.

Human Evaluation. Given 500 randomly sampled images, we asked 5 undergraduate students to annotate whether these images are 1) visually natural (i.e., look like real images); 2) semantically correct (i.e., content is consistent with class label). The ratings are 1-5. Higher is better. Table 8 shows the results. As can be seen, our method outperforms baselines. In our method, the quality of generated data is evaluated directly by applying them for training classification models in the same framework. Classification performance can inform BigGAN whether the generated

Table 9. Automatic evaluation of generated images.

Method	Inception↑	FID↓
Separate	92.4	9.1
Equal	96.2	8.8
Ours	**108.5**	**7.9**

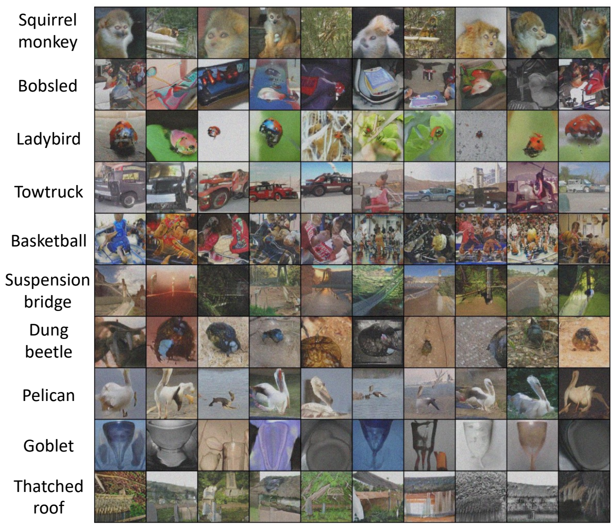


Figure 2. Exemplar images generated by BigGAN in our method trained on the balanced ImageNet.

data is useful. In Separate, such a feedback loop is missing. In Equal, BigGAN generates the same number of images for different classes. For a class where the classifier is already performing well, continuing to generate data for this class is a waste of effort.

Automatic Evaluation. In addition to human evaluation, we also performed an automatic evaluation of generated images, using metrics including inception score (Salimans et al., 2016) and Frechet inception distance (FID) (Heusel et al., 2017). Table 9 shows that our method outperforms baselines. The reason is: the data generated by our method is explicitly encouraged to be useful for training the classifier. If a generated image has poor quality, the classifier trained by it will perform worse. Our framework minimizes classification losses to ensure generated images are of high quality.

Qualitative Evaluation. Figure 2 shows some randomly sampled generated images. As can be seen, the generated images are natural and semantically meaningful.

Table 10. Computation costs (GPU days on Nvidia 1080Ti GPUs).

Method	Top-1 Error	Top-5 Error	Runtime
Pcdarts (Xu et al., 2020)	24.2	7.3	3.8
Original (ours)	23.2	6.6	5.1
Reduce (ours)	23.4	6.7	3.9

4.5. Computation Costs

Let Reduce denote our method after applying cost reduction methods outlined in Section 3.2. Let Original denote our original method without using cost reduction methods. Table 10 compares the computation costs (in GPU days) of Reduce and Original in the experiments on balanced ImageNet. From this table, we make two observations. First, the computation cost of our Reduce method is similar to that of vanilla PCDARTS, while the classification errors of Reduce are much smaller than those of PCDARTS. Second, compared with Original, the computation cost of Reduce is much smaller without substantially increasing classification errors. This demonstrates that the cost reduction methods can greatly decrease computation costs without significantly compromising classification performance. We applied these cost reduction methods to PCDARTS as well, including reducing architecture's update frequency from every iteration to every 8 iterations and decreasing the number of epochs by half, which resulted in significantly poorer performance. Therefore, we maintained PCDARTS' default hyperparameter settings. The rest of cost reduction methods, including parameter tying and decreasing CGAN pretraining time, are not applicable to PCDARTS.

5. Conclusions and Future Works

We propose a multi-level optimization based framework to conduct needs-aware image generation. Our framework trains an initial classification model and measures its class-specific performance on a validation set. A conditional GAN is trained to generate training data for less well-performing classes. Generated data is used to improve classification performance. All these steps are performed end-to-end. Experiments on both imbalanced and balanced datasets demonstrate our method's effectiveness.

In future work, we plan to extend the framework to machine learning applications beyond classification, such as semantic segmentation, object detection, named entity recognition.

Impact Statement

One potential negative societal impact of our work is: if the images generated by our method are not meaningful, models trained using these images may make incorrect predictions, which are not acceptable in application areas such as healthcare and finance. One major limitation of this work is that it cannot be applied to reinforcement learning and evolutionary algorithm based neural architecture search methods since they are not differentiable.

References

Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Azizi, S., Kornblith, S., Saharia, C., Norouzi, M., and Fleet, D. J. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. Generating sentences from a continuous space. In *20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pp. 10–21. Association for Computational Linguistics (ACL), 2016.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

Cai, J., Wang, Y., Hwang, J.-N., et al. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *ICCV*, pp. 112–121, 2021.

Cao, K., Wei, C., Gaidon, A., Arechiga, N., and Ma, T. Learning imbalanced datasets with label-distribution-aware margin loss. *NeurIPS*, 32, 2019.

Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C. Drnas: Dirichlet neural architecture search. *CoRR*, abs/2006.10355, 2020.

Chu, X., Zhou, T., Zhang, B., and Li, J. Fair DARTS: eliminating unfair advantages in differentiable architecture search. *CoRR*, abs/1911.12126, 2019.

Chu, X., Wang, X., Zhang, B., Lu, S., Wei, X., and Yan, J. DARTS-: robustly stepping out of performance collapse without indicators. *CoRR*, abs/2009.01027, 2020.

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., and Batra, D. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

Cui, J., Zhong, Z., Liu, S., Yu, B., and Jia, J. Parametric contrastive learning. In *ICCV*, pp. 715–724, 2021.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *CVPR*, pp. 9268–9277, 2019a.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019b.

Dempe, S. *Foundations of bilevel programming*. Springer Science & Business Media, 2002.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Diego, U. S. Teaching + learning commons. `https://commons.ucsd.edu/`, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Feurer, M., Springenberg, J., and Hutter, F. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pp. 289–293. IEEE, 2018.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pp. 3748–3758. PMLR, 2020.

Gu, Y.-C., Wang, L.-J., Liu, Y., Yang, Y., Wu, Y.-H., Lu, S.-P., and Cheng, M.-M. Dots: Decoupling operation and topology in differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12311–12320, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016b.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hong, Y., Zhang, J., Sun, Z., and Yan, K. Safa: Sample-adaptive feature augmentation for long-tailed image classification. In *ECCV*, 2022.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

Hu, S., Xie, S., Zheng, H., Liu, C., Shi, J., Liu, X., and Lin, D. DSNAS: direct neural architecture search without parameter retraining. In *CVPR*, 2020.

Huang, C., Li, Y., Loy, C. C., and Tang, X. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5375–5384, 2016.

Huang, S.-W., Lin, C.-T., Chen, S.-P., Wu, Y.-Y., Hsu, P.-H., and Lai, S.-H. Auggan: Cross domain adaptation with gan-based data augmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 718–731, 2018.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Ji, K., Yang, J., and Liang, Y. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pp. 4882–4892. PMLR, 2021.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020.

Kim, J., Jeong, J., and Shin, J. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13896–13905, 2020.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014a.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014b.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Kingma, D. P., Welling, M., et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

Krizhevsky, A. and Hinton, G. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7): 1–9, 2010.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Li, J., Tan, Z., Wan, J., Lei, Z., and Guo, G. Nested collaborative learning for long-tailed visual recognition. In *CVPR*, pp. 6949–6958, 2022a.

Li, M., Cheung, Y.-m., and Lu, Y. Long-tailed visual recognition via gaussian clouded logit adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6929–6938, 2022b.

Li, M., Cheung, Y.-m., Lu, Y., et al. Long-tailed visual recognition via gaussian clouded logit adjustment. In *CVPR*, pp. 6929–6938, 2022c.

Li, T., Cao, P., Yuan, Y., Fan, L., Yang, Y., Feris, R. S., Indyk, P., and Katabi, D. Targeted supervised contrastive learning for long-tailed recognition. In *CVPR*, pp. 6918–6928, 2022d.

Li, W., Chen, J., Cao, J., Ma, C., Wang, J., Cui, X., and Chen, P. Eid-gan: Generative adversarial nets for extremely imbalanced data augmentation. *IEEE Transactions on Industrial Informatics*, 2022e.

Li, W., Lotz, J. F., Qiu, C., and Elliott, D. Data curation for image captioning with text-to-image generative models. *arXiv preprint arXiv:2305.03610*, 2023.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Lin, Z., Liang, H., Fanti, G., and Sekar, V. Raregan: Generating samples for rare classes. *AAAI Conference on Artificial Intelligence*, 2022.

Liu, H., Simonyan, K., and Yang, Y. DARTS: differentiable architecture search. In *ICLR*, 2019a.

Liu, R., Liu, Y., Zeng, S., and Zhang, J. Towards gradient-based bilevel optimization with non-convex followers and beyond. *Advances in Neural Information Processing Systems*, 34, 2021.

Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., and Yu, S. X. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019b.

Liu, Z., Shen, Z., Long, Y., Xing, E., Cheng, K.-T., and Leichner, C. Data-free neural architecture search via recursive label calibration. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pp. 391–406. Springer, 2022.

Ma, N., Zhang, X., Zheng, H., and Sun, J. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

Ma, Y., Jiao, L., Liu, F., Yang, S., Liu, X., and Li, L. Curvature-balanced feature manifold learning for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15824–15835, 2023.

Mariani, G., Scheidegger, F., Istrate, R., Bekas, C., and Malossi, C. Bagan: Data augmentation with balancing gan. *arXiv preprint arXiv:1803.09655*, 2018.

Menon, A. K., Jayasumana, S., Rawat, A. S., Jain, H., Veit, A., and Kumar, S. Long-tail learning via logit adjustment. In *ICLR*, 2021.

Mirza, M. and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Odena, A., Olah, C., and Shlens, J. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pp. 2642–2651. PMLR, 2017.

Park, S., Hong, Y., Heo, B., Yun, S., and Choi, J. Y. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6887–6896, 2022.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Ren, M., Zeng, W., Yang, B., and Urtasun, R. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

Ren, Z., Yeh, R., and Schwing, A. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21786–21797. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f7ac67a9aa8d255282de7d11391e1b69-Paper.pdf.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *NeurIPS*, 2016.

Sato, R., Tanaka, M., and Takeda, A. A gradient method for multilevel optimization. *Advances in Neural Information Processing Systems*, 34:7522–7533, 2021.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., and Meng, D. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pp. 1919–1930, 2019.

Sinha, S., Zhang, H., Goyal, A., Bengio, Y., Larochelle, H., and Odena, A. Small-gan: Speeding up gan training using core-sets. In *International Conference on Machine Learning*, pp. 9005–9015. PMLR, 2020a.

Sinha, S., Zhao, Z., ALIAS PARTH GOYAL, A. G., Raffel, C. A., and Odena, A. Top-k training of gans: Improving gan performance by throwing away bad samples. *Advances in Neural Information Processing Systems*, 33: 14638–14649, 2020b.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations*, 2021.

Such, F. P., Rawal, A., Lehman, J., Stanley, K., and Clune, J. Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In *International Conference on Machine Learning*, pp. 9206–9216. PMLR, 2020.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, 2015.

Tabak, E. G. and Vanden-Eijnden, E. Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233, 2010.

Tang, K., Huang, J., and Zhang, H. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *NeurIPS*, 33:1513–1524, 2020.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 266–282. Springer, 2020.

Trabucco, B., Doherty, K., Gurinas, M., and Salakhutdinov, R. Effective data augmentation with diffusion models. *arXiv preprint arXiv:2302.07944*, 2023.

Wang, H., Fu, S., He, X., Fang, H., Liu, Z., and Hu, H. Towards calibrated hyper-sphere representation via distribution overlap coefficient for long-tailed learning. In *ECCV*, 2022a.

Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. X. Long-tailed recognition by routing diverse distribution-aware experts. *arXiv preprint arXiv:2010.01809*, 2020a.

Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. X. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*. OpenReview.net, 2021.

Wang, X., Guo, W., Su, J., Yang, X., and Yan, J. Zarts: On zero-order optimization for neural architecture search. *Advances in Neural Information Processing Systems*, 35: 12868–12880, 2022b.

Wang, Y., Guo, J., Song, S., and Huang, G. Meta-semi: A meta-learning approach for semi-supervised learning. *CoRR*, abs/2007.02394, 2020b. URL `https://arxiv.org/abs/2007.02394`.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. Aggregated residual transformations for deep neural networks. In *CVPR*, pp. 1492–1500, 2017.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *ICLR*, 2019.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G., Tian, Q., and Xiong, H. PC-DARTS: partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

Xu, Y., Li, Y.-L., Li, J., and Lu, C. Constructing balance from imbalance for long-tailed image recognition. In *ECCV*, pp. 38–56. Springer, 2022.

Xu, Z., Liu, R., Yang, S., Chai, Z., and Yuan, C. Learning imbalanced data with vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15793–15803, 2023.

Yang, J., Ji, K., and Liang, Y. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

Ye, P., Li, B., Li, Y., Chen, T., Fan, J., and Ouyang, W. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10874–10883, 2022.

Zhang, C., Tan, K. C., Li, H., and Hong, G. S. A cost-sensitive deep belief network for imbalanced classification. *IEEE transactions on neural networks and learning systems*, 30(1):109–122, 2018.

Zhang, S., Li, Z., Yan, S., He, X., and Sun, J. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, pp. 2361–2370, 2021a.

Zhang, X., Li, Y., Zhang, X., Wang, Y., and Sun, J. Differentiable architecture search with random features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16060–16069, 2023.

Zhang, Y., Hooi, B., Hong, L., and Feng, J. Test-agnostic long-tailed recognition by test-time aggregating diverse experts with self-supervision. *arXiv preprint arXiv:2107.09249*, 2021b.

Zheng, C., Wu, G., and Li, C. Toward understanding generative data augmentation. *arXiv preprint arXiv:2305.17476*, 2023.

Zhong, Z., Cui, J., Liu, S., and Jia, J. Improving calibration for long-tailed recognition. In *CVPR*, pp. 16489–16498. Computer Vision Foundation / IEEE, 2021a.

Zhong, Z., Cui, J., Liu, S., and Jia, J. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16489–16498, 2021b.

Zhou, P., Xiong, C., Socher, R., and Hoi, S. C. H. Theory-inspired path-regularized differential network architecture search. *CoRR*, abs/2006.16537, 2020. URL https://arxiv.org/abs/2006.16537.

Zhou, Y., Sahak, H., and Ba, J. Training on thin air: Improve image classification with generated data. *arXiv preprint arXiv:2305.15316*, 2023a.

Zhou, Z., Li, L., Zhao, P., Heng, P.-A., and Gong, W. Class-conditional sharpness-aware minimization for deep long-tailed recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3499–3509, 2023b.

Zhu, J., Wang, Z., Chen, J., Chen, Y.-P. P., and Jiang, Y.-G. Balanced contrastive learning for long-tailed visual recognition. In *CVPR*, pp. 6908–6917, 2022.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.

A. Full Description of Optimization Algorithm

We use a well-established algorithm developed in (Liu et al., 2019a) to solve the proposed multi-level optimization problem. Theoretic convergence of this algorithm has been broadly analyzed in (Ghadimi & Wang, 2018; Grazzi et al., 2020; Ji et al., 2021; Liu et al., 2021; Yang et al., 2021). At each level of optimization problem, the optimal solution (on the left-hand side of the equal sign, marked with *), its exact value is computationally expensive to compute. To address this problem, following (Liu et al., 2019a), we approximate the optimal solution using a one-step gradient descent update and plug the approximation into the next level of optimization problem. In the sequel, $\frac{\partial \cdot}{\partial \cdot}$ denotes partial derivative. $\frac{d \cdot}{d \cdot}$ denotes an ordinary derivative.

Following (Liu et al., 2019a), we approximate $W_1^*(A)$ using one-step gradient descent update of W_1 w.r.t $L_{cls}(W_1, A, D_{cls}^{(tr)})$:

$$W_1^*(A) \approx W_1' = W_1 - \eta_{w_1} \nabla_{W_1} L_{cls}(W_1, A, D_{cls}^{(tr)}). \tag{8}$$

We approximate $G^*(A)$ using one-step gradient ascent update of G w.r.t $L_{gan}(G, H, A, D_{cgan})$:

$$G^*(A) \approx G' = G + \eta_g \nabla_G L_{gan}(G, H, A, D_{cgan}). \tag{9}$$

For H, it can be updated using gradient descent:

$$H = H - \eta_h \nabla_H L_{gan}(G, H, A, D_{cgan}). \tag{10}$$

Plugging $W_1^*(A) \approx W_1'$ and $G^*(A) \approx G'$ into $L_{cls}(W_2, A, D_{cls}^{(tr)}) + \lambda L_{approx}(W_2, A, W_1^*(A), G^*(A))$, we get an approximated objective. Then we approximate $W_2^*(A, W_1^*(A), G^*(A))$ using one-step gradient descent update of W_2 w.r.t the approximated objective:

$$W_2^*(A, W_1^*(A), G^*(A)) \approx W_2' = W_2 - \eta_{w_2} \nabla_{W_2}(L_{cls}(W_2, A, D_{cls}^{(tr)}) + \lambda L_{approx}(W_2, A, W_1', G')). \tag{11}$$

Finally, we plug $W_2^*(A, W_1^*(A), G^*(A)) \approx W_2'$ and $W_1^*(A) \approx W_1'$ into $L_{cls}(W_2', A, D_{cls}^{(val)}) + L_{cls}(W_1', A, D_{cls}^{(val)})$ and get an approximated objective. We update A using gradient descent w.r.t the approximated objective:

$$A \leftarrow A - \eta_a \nabla_A (L_{cls}(W_2', A, D_{cls}^{(val)}) + L_{cls}(W_1', A, D_{cls}^{(val)})). \tag{12}$$

$\nabla_A L_{cls}(W_2', A, D_{cls}^{(val)})$ can be computed as:

$$\nabla_A L_{cls}(W_2', A, D_{cls}^{(val)}) \quad = (\tfrac{dW_1'}{dA} \tfrac{\partial W_2'}{\partial W_1'} + \tfrac{dG'}{dA} \tfrac{\partial W_2'}{\partial G'}) \tfrac{\partial L_{cls}(W_2', A, D_{cls}^{(val)})}{\partial W_2'} + \tfrac{\partial L_{cls}(W_2', A, D_{cls}^{(val)})}{\partial A}, \tag{13}$$

where $\frac{\partial \cdot}{\partial \cdot}$ denotes partial derivative and $\frac{d}{d}$ denotes total derivative, and further

$$\tfrac{\partial W_2'}{\partial W_1'} = -\eta_{w_2} \lambda \nabla^2_{W_1', W_2} L_{approx}(W_2, A, W_1', G') \tag{14}$$

$$\tfrac{\partial W_2'}{\partial G'} = -\eta_{w_2} \lambda \nabla^2_{G', W_2} L_{approx}(W_2, A, W_1', G') \tag{15}$$

$$\tfrac{dW_1'}{dA} = -\eta_{w_1} \nabla^2_{A, W_1} L_{cls}(W_1, A, D_{cls}^{(tr)}) \tag{16}$$

$$\tfrac{dG'}{dA} = \eta_g \nabla^2_{A, G} L_{gan}(G, H, A, D_{cgan}) \tag{17}$$

For $\nabla_A L_{cls}(W_1', A, D_{cls}^{(val)})$, it can be calculated as:

$$\nabla_A L_{cls}(W_1', A, D_{cls}^{(val)}) = \frac{dW_1'}{dA} \frac{\partial L_{cls}(W_1', A, D_{cls}^{(val)})}{\partial W_1'} + \frac{\partial L_{cls}(W_1', A, D_{cls}^{(val)})}{\partial A} \tag{18}$$

where $\frac{dW_1'}{dA}$ is given in Eq.(16). In Eq.(18) and Eq.(13), matrix-vector multiplication is approximated using finite-difference approximation similar to (Liu et al., 2019a).

Table 11. Model calibration results.

Method	ECE	Accuracy
GCL (Li et al., 2022b)	8.47	54.5
LiVT (Xu et al., 2023)	9.33	60.9
Our method	6.15	61.8

The gradient descent update of A in equation 5 can run one or more steps. After A is updated, the one-step gradient-descent approximations (in equation 1-4), which are functions of A, change with A and need to be re-updated. Then, the gradient of A, which is a function of one-step gradient-descent approximations, needs to be re-calculated and is used to refresh A. In sum, the update of A and the updates of one-step gradient-descent approximations mutually depend on each other. These updates are performed iteratively until convergence. Algorithm 1 shows the algorithm.

Algorithm 1 Optimization algorithm

```
While not converged
1. Update the approximation $W_1'$ of $W_1^*$ using Eq.(8)
2. Update the approximation $G'$ of $G^*$ using Eq.(9)
3. Update $H$ using Eq.(10)
4. Update the approximation $W_2'$ of $W_2^*$ using Eq.(11)
5. Update $A$ using Eq.(12)
```

B. Additional Settings of Experiments on Balanced Dataset

During architecture search, each network was a stack of 8 cells and each cell had 7 nodes. Initial channel number was set to 16. Weight parameters were optimized using SGD, with a learning rate of 0.025, a weight decay of 3e-4, and a momentum of 0.9. Learning rate was reduced using a cosine scheduler. Architecture variables were optimized using Adam (Kingma & Ba, 2014a), with a learning rate of 3e-4 and a weight decay of 1e-3. The algorithm ran for 50 epochs. Batch size was set to 64.

During architecture evaluation, 14 copies of the optimally searched cell were stacked to form a large network. Initial channel number was set to 40. Network weights were optimized using SGD, with an initial learning rate of 0.5, a weight decay of 3e-5, a batch size of 1024, and a momentum of 0.9. The algorithm ran for 250 epochs.

C. Analysis of Model Calibration

We evaluated our method's Expected Calibration Error (ECE) (Zhong et al., 2021b) on the ImageNet-LT dataset, as presented in Table 11. Our method demonstrates a lower ECE compared to baseline approaches, signifying enhanced calibration and reduced over-confidence. This improvement is primarily due to our method's capability to generate more training data for underperforming classes, thereby equalizing accuracy across various classes. This aligns with the results reported in (Zhong et al., 2021b) which show that balancing the amount of training data and achieving equitable accuracy across classes through data augmentation can alleviate model miscalibration and reduce overconfidence.

D. Analysis of the connection between image quality per class and incremental accuracy improvement

We evaluated the Inception Score of images generated for each class within the ImageNet-LT dataset and assessed the incremental accuracy improvements for each class, before and after using generated data for model training. Higher Inception Scores are indicative of superior image quality. Subsequently, we calculated the Pearson correlation coefficient between the Inception Scores and the accuracy enhancements for all classes. A Pearson correlation of 0.52 was observed, signifying a positive correlation. This indicates that higher quality of generated images contributes to greater enhancements in accuracy.

E. Apply Cost Reducing Strategies to NAS Baselines

We applied cost-reducing strategies to DARTS as well, including reducing architecture's update frequency from every iteration to every 8 iterations and decreasing the number of epochs from 50 to 25), but this resulted in significantly poorer performance, as shown in Table 12. Therefore, we maintained DARTS' default settings.

The rest of cost reduction strategies, including parameter tying and decreasing the pretraining time of the CGAN, are not applicable to DARTS.

Table 12. Apply cost reduction to DARTS.

Method	Error on Cifar-100	Error on Cifar-10	Search cost (GPU days)
Darts	20.58 ± 0.44	2.76 ± 0.09	4.0
Darts + Cost Reduction	26.31 ± 0.27	4.59 ± 0.17	1.5

F. Limitations and potential solutions

One major limitation of this work is that it is difficult to be applied to reinforcement learning (RL) and evolutionary algorithm based neural architecture search (NAS) methods since they are not differentiable. To address this limitation and apply our method to RL based NAS methods, we can calculate the policy gradient of the validation losses in the fourth stage w.r.t to A and update A using policy gradient descent.

G. Additional Discussion

Does the success of our method highly depend on the quality of CGANs? Our framework is orthogonal to CGANs. Any CGAN can be plugged into our framework to perform needs-aware data generation. The primary source of our framework's performance improvement is the needs-aware data generation mechanism, rather than the specific CGAN employed.

H. Instructions Given to Participants in Human Studies

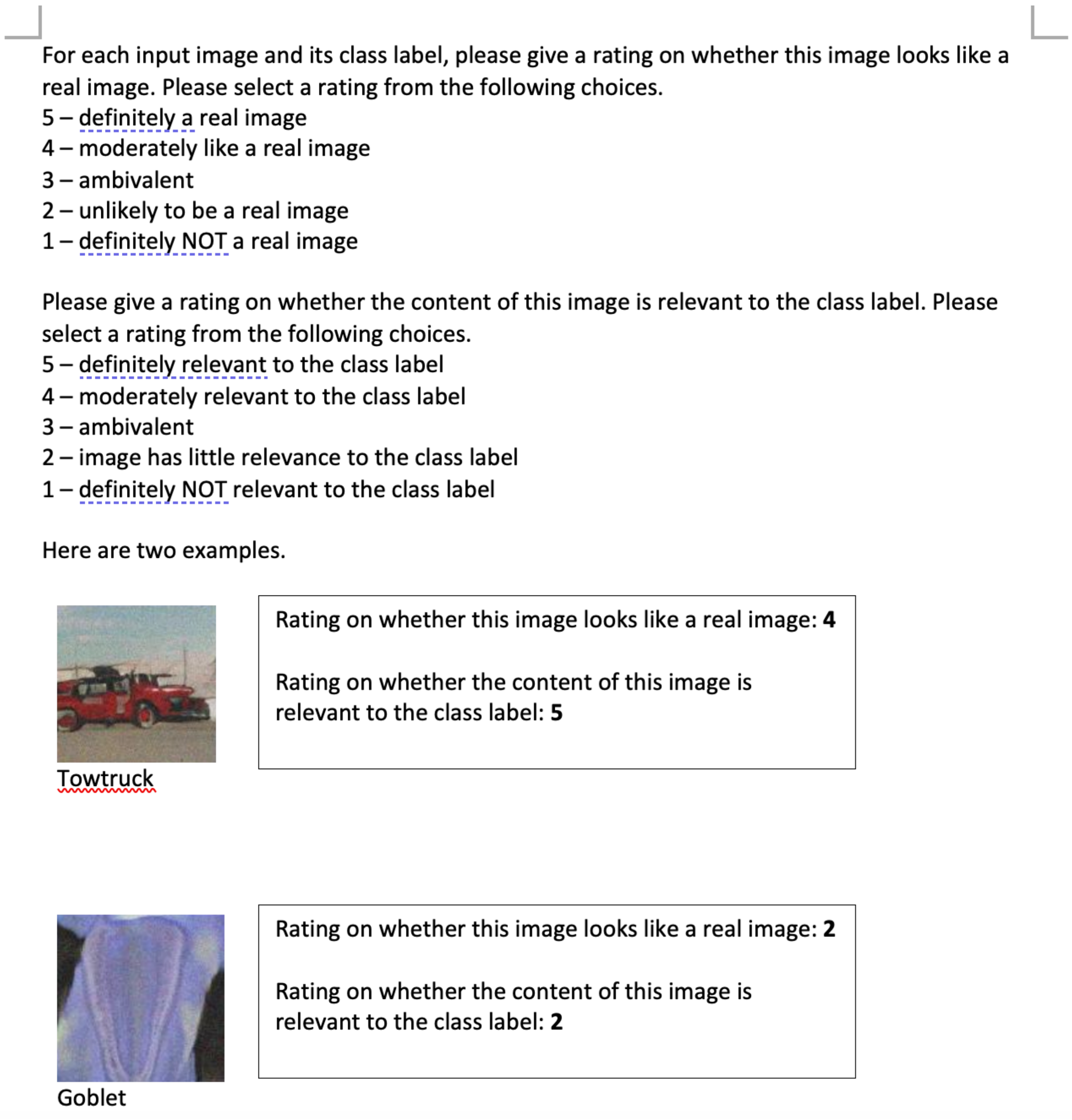
Figure 3 shows the screenshot of instructions given to participants in human studies.

I. Full Lists of Hyperparameter Settings

Table 14 shows the hyperparameter settings used in the search phase on balaced ImageNet. Table 15 shows the hyperparameter settings used in the evaluation phase. Notations used in these tables are given in Table 13.

Notation	Meaning
W_1	The first set of weight parameters of the classification model
W_2	The second set of weight parameters of the classification model
G	Generator of the GAN model
H	Discriminator of the GAN model
A	Architecture of the classification model
$D^{(\mathrm{tr})}$	Training data
$D^{(\mathrm{val})}$	Validation data

Table 13. Notations in our method



For each input image and its class label, please give a rating on whether this image looks like a real image. Please select a rating from the following choices.
5 – definitely a real image
4 – moderately like a real image
3 – ambivalent
2 – unlikely to be a real image
1 – definitely NOT a real image

Please give a rating on whether the content of this image is relevant to the class label. Please select a rating from the following choices.
5 – definitely relevant to the class label
4 – moderately relevant to the class label
3 – ambivalent
2 – image has little relevance to the class label
1 – definitely NOT relevant to the class label

Here are two examples.

Towtruck

Rating on whether this image looks like a real image: **4**

Rating on whether the content of this image is relevant to the class label: **5**

Goblet

Rating on whether this image looks like a real image: **2**

Rating on whether the content of this image is relevant to the class label: **2**

Figure 3. Screenshot of instructions given to participants in human studies.

Name	Value
Optimizer for W_1, W_2, G, H	SGD
Initial learning rate for W_1, W_2, G, H	0.5
Learning rate scheduler for W_1, W_2, G, H	Cosine decay
Minimum learning rate for W_1, W_2, G, H	0.0
Momentum for W_1, W_2, G, H	0.9
Weight decay for W_1, W_2, G, H	0.0003
Optimizer for A	Adam
Learning rate for A	0.006
Weight decay for A	0.001
Initial channels for W_1, W_2	16
Layers for W_1, W_2	8
Gradient Clip for W_1, W_2, G, H	5
Batch size	768
Epochs	50
λ	1

Table 14. Hyperparameter settings in Ours-pcdarts on ImageNet during architecture search

Name	Value
Optimizer	SGD
Initial learning rate	0.5
Learning rate scheduler	Cosine decay
Momentum	0.9
Weight decay	0.00003
Initial channels	48
Layers	14
Auxiliary weight	0.4
Label smooth	0.1
Drop path prob	0.0
Gradient Clip	5
Batch size	1024
Epochs	250

Table 15. Hyperparameter settings on ImageNet during architecture evaluation

J. Visualization of Searched Architectures

We visualize the architectures searched by our methods in Figure 4.

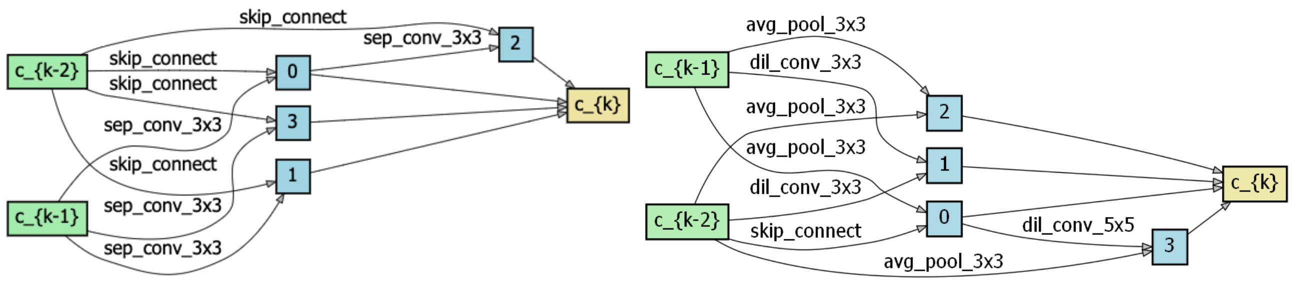


Figure 4. Architectures searched by Ours-PCDARTS on ImageNet.