[Re]CUDA: Curriculum of Data Augmentation for Long-Tailed Recognition

Anonymous authors
Paper under double-blind review

Abstract

In this reproducibility study, we present our results and experience during replicating the paper, titled CUDA: Curriculum of Data Augmentation for Long-Tailed Recognition(Ahn et al., 2023).Traditional datasets used in image recognition, such as Imagenet, are often synthetically balanced, meaning each class has an equal number of samples. In practical scenarios, datasets frequently exhibit significant class imbalances, with certain classes having a disproportionately larger number of samples compared to others. This discrepancy poses a challenge for traditional image recognition models, as they tend to favor classes with larger sample sizes, leading to poor performance on minority classes. CUDA proposes a class-wise data augmentation technique which can be used over any existing model to improve the accuracy for LTR: Long Tailed Recognition. We were able to reproduce a significant part of the results for the long tailed dataset Cifar100-LT.

1 Introduction

Long-tailed recognition presents one of the most formidable challenges in visual recognition. This problem revolves around training highly effective models from datasets characterized by a large number of images distributed along a long-tailed class distribution. In such datasets, a notable imbalance exists among the number of samples belonging to different classes, with some classes comprising a vast majority of training instances (Head classes), while others contain significantly fewer instances (Tail classes).Consider a practical example of a disease screening test dataset: the Head classes would predominantly consist of non-patients, whereas the Tail classes would represent the minority of patients. In such scenarios, the performance of deep learning models tends to be disproportionately influenced by the Head classes, while the learning of Tail classes is often inadequately developed.

Solutions to Long Tailed recognition primary involves three methods: (1) Resampling (Buda et al., 2018) ,Up-sampling minority classes and Down-sampling majority classes. (2) Reweighting (Cao et al., 2019): Rebalancing the loss to give more weights to minority classes. (3) Transfer learning(Kim et al., 2020).: enriching the information of minority classes by transferring information gathered from majority classes to the minority classes.

The original authors propose that applying an algorithm to determine class-wise augmentation strength can potentially address the imbalance problem in long-tailed visual recognition tasks. The other key finding, as highlighted in the paper, is that after training when we examine the class-wise strength of augmentation the majority classes have a stronger degree of augmentation and the minority classes have a weaker degree of augmentation.

In this reproducibility study, we examine the performance of CUDA over the dataset Cifar100-LT across methods like CE(Cross Entropy), CE-DRW (Cross entropy Dynamic reweighting)(Cao et al., 2019), LDAM-DRW (label-distribution-aware margin loss)(Cao et al., 2019), BS (balanced soft-max)(Ren et al., 2020),BCL (balanced contrastive learning)(Zhu et al., 2022).We analyze the standard deviation of the weight L1 norm to demonstrate how CUDA contributes to achieving more balanced feature representation across different classes.. We investigate the LOL score, representing the augmentation strength of each class after training with CUDA. We extend our experiments to observe how changes in CUDA hyperparameters affect model

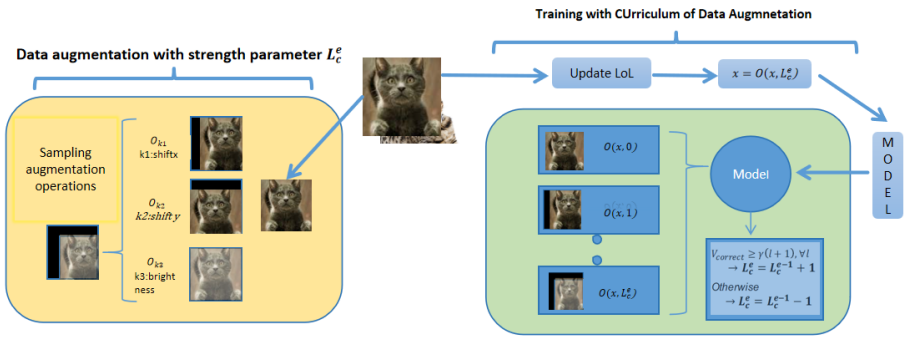

Figure 1: Schematic of CUDA.The left part shows the strength based augmentation and the right part shows how the Level-of-Learning(LOL) score is updated in a given epoch

performance . We conduct a component analysis to dissect the contribution of curriculum learning and class-wise augmentation strengths to overall model performance

2 Scope of Reproducibility

The main idea of the paper is to introduce a data augmentation technique called CUDA, which is designed to complement existing Long-Tailed Recognition (LTR) models. When integrated with LTR models, the paper claims that CUDA offers the following benefits

- CUDA enhances the top-1 validation accuracy across all LTR models and effectively addresses class imbalance.
- Implementing stronger augmentation on majority classes and milder augmentation on minority classes leads to superior model performance compared to the reverse strategy.

To further substantiate the aforementioned claims, we conducted the following studies:

- Conduct the performance analysis across three different imbalance ratios (100, 50, 10) to examine how CUDA performance varies with dataset imbalance ratios.
- Examine how accuracy changes with the three hyper-parameters augmentation probability, number of test samples and acceptance rate. This analysis aims to validate the assertion that both excessive and insufficient augmentation adversely affect performance.
- To evaluate the contribution of curriculum learning and class-wise score to performance of CUDA ,we do a component analysis.

3 Methodology

3.1 Model descriptions

We use Resnet-32 as our backbone model for all 5 models CE, CE-DRW, BS, LDAM-DRW, and RIDE.However RIDE uses modified version of Resnet-32 implemented according to the original paper.The core philosophy of CUDA is to "generate an augmented sample that becomes the most difficult sample without losing its original information."CUDA uses two main parts to achieve this:(1)Strength-based augmentation(2)Using Level-of-Learning (LOL) score.

3.1.1 Strength-based augmentation

We need to define a metric to quantify how complex augmentations we are applying on an image. Let us assume that there exist pre-defined K augmentation operations. We utilize visual augmentation operations indexed as $k \in \{1, \ldots, K\}$, e.g., Gaussian blur, Rotation, Horizontal flip. Each augmentation operation $O_k^{m_k(s)}$ has its own predefined augmentation magnitude function $m_k(s)$ with the strength parameter $s \in \{0, \ldots, S\}$.Given an augmentation strength parameter s and an input image x, we model a sequence of augmentation operations O(x; s) as follows:

$$\mathcal{O}(x; s) = \mathcal{O}_{k_s}^{m_{k_s}(s)} \circ \mathcal{O}_{k_{s-1}}^{m_{k_{s-1}}(s)} \circ \cdots \circ \mathcal{O}_{k_1}^{m_{k_1}(s)}(x), \quad k_i \sim \mathrm{Cat}(K, \mathcal{U}(K)) \quad \forall i = \{1, \ldots, s\}$$

The sequential augmentation operation O(x; s) involves sampling s operations from a categorical distribution, where each operation is chosen from a uniform distribution among K possible operations. In essence, out of the K possible operations, only s operations are selected, each with a magnitude m(3).For example, let's consider s=3. Suppose the selected augmentation operations k1,k2,k3 correspond to brightness adjustment, X-shift, and Y-shift, respectively. In this case,O(x;3) outputs an image where the brightness is increased by m-bright(3) and shifted by m-x-shift(3) on the x-axis, and shifted by m-y-shift(3) on the y-axis.As s increases, both the number of augmentation operations applied to each image and the magnitude of each of these operations increase. Consequently, the complexity of the augmentation process increases.

3.1.2 Updating Level of Learning Score

To control the strength of augmentation properly, we check whether the model can correctly predict augmented versions without losing the original information. To enable this, we define the LOL for each class c at epoch e, i.e., L_c^e which is adaptively updated as the training continues as follows:

- Initialize L_c^e to zero for all values of c before training.Suppose, at the beginning of the third epoch L_c^2 (LOL value for class c after the second iteration) for a specific c is 2.
- First, update the LOL value for each class using the function V_{lol} which takes inputs: D_c (all images belonging to class c) , L_c^2 (previous value of LOL), f_θ (the model we are gonna use for prediction), $\gamma \in [0, 1]$ (threshold hyperparameter) , T (coefficient of the number of samples used to updating LOL).
$$L_c^e = V_{\mathrm{LoL}}\left(\mathcal{D}_c, L_c^{e-1}, f_\theta, \gamma, T\right)$$
- Inside the function V_{lol} we loop over values of l lesser than L_c^2 =2 (i.e) for l=0,1,2 We randomly sample T(l+1) samples out of D_c which is $D_c^{'}$, $D_c^{'}$ ∈D_c s.t. $|D_c^{'}| = T(l + 1)$
$$V_{\mathrm{LoL}}\left(\mathcal{D}_c, L_c^{e-1}, f_\theta, \gamma, T\right) = \begin{cases} L_c^{e-1} + 1 & \text{if } V_{\text{Correct}}\ (\mathcal{D}_c, l, f_\theta, T) \geq \gamma T(l+1) \quad \forall l \in \left\{0, \ldots, L_c^{e-1}\right\} \\ L_c^{e-1} - 1 & \text{otherwise} \end{cases}$$
- Compute $V_{correct}$ where the model f_θ predict the class for all samples in $D_c^{'}$ counting the total correct predictions among the T(l+1) samples.
$$V_{\text{Correct}}\ (\mathcal{D}_c, l, f_\theta, T) = \sum_{x \in \mathcal{D}_c'} \mathbb{1}_{\{f_\theta(\mathcal{O}(x;l))=c\}} \quad \text{where } \mathcal{D}_c' \subset \mathcal{D}_c$$
- If the number of correct predictions is above the threshold T(l+1) (i.e if the ratio of correct samples/total samples is above) for all values of l (0,1,2 in this case)increase the L_c^e by 1 (3 in this case) otherwise decrease it by 1 (1 in this case) for the next epoch.
- By running V_{lol} for all classes we could then define a sequence of augmentation operations $O(x_i, L_{y_i}^e)$ each with their strengths defined by the respective $L_{y_i}^e$ where each y_i is a class .We create a new training dataset according to equation 2.

We can train any LTR algorithm (model + preprocessing) on the new dataset and obtain the overall accuracy for that epoch.

Algorithm 1: CUrriculum of Data Augmentation

Input: LTR algorithm $\mathcal{A}_{\text{LTR}}(f, \mathcal{D})$, training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, train epochs E, aug. probability p_{aug}, threshold γ, number of sample coefficient T.
Output: trained model f_θ
Initialize: $L_c^0 = 0 \; \forall c \in \{1, ..., C\}$
for $e \leq E$ **do**
  Update $L_c^e = V_{\text{LoL}}(\mathcal{D}_c, L_c^{e-1}, f_\theta, \gamma, T) \quad \forall c$ // Alg. 2
  Generate $\mathcal{D}_{\text{CUDA}} = \{(\bar{x}_i, y_i) | (x_i, y_i) \in \mathcal{D}\}$ where

$$\bar{x}_i = \begin{cases} \mathcal{O}(x_i, L_{y_i}^e) & \text{with prob. } p_{\text{aug}} \\ x_i & \text{otherwise.} \end{cases}$$

  Run LTR algorithm using $\mathcal{D}_{\text{CUDA}}$, *i.e.*, $\mathcal{A}_{\text{LTR}}(f_\theta, \mathcal{D}_{\text{CUDA}})$.
end

Algorithm 2: V_{LoL}: Update LoL score

Input: $\mathcal{D}_c$, L, f_θ, γ, T
Output: updated L
Initialize: check $= 1$
for $l \leq L$ **do**
  /* $V_{\text{correct}}(\mathcal{D}_c, l, f_\theta, T)$ */
  Sample $\mathcal{D}'_c \subset \mathcal{D}_c$ s.t. $|\mathcal{D}'_c| = T(l+1)$
  Compute $v = \sum_{x \in \mathcal{D}'_c} \mathbb{1}_{\{f(\mathcal{O}(x;l)=c\}}$
  if $v \leq \gamma T(l+1)$ **then**
    check $\leftarrow 0$; **break**
  end
end
if check $= 1$ **then** $L \leftarrow L + 1$
else $L \leftarrow L - 1$

Figure 2: Algorithm of CUDA (Ahn et al., 2023)

3.2 Dataset and Hyper Parameters

The reproduction study is done on the dataset Cifar100-LT as mentioned in the previous works(Cao et al., 2019) ,Three different datasets are derived from Cifar100 dataset with imbalance ratio 100,50 and 10,where an imbalance ratio is defined as $N_{\max}/N_{\min}$. The CIFAR-100 dataset (Canadian Institute for Advanced Research, 100 classes) is a subset of the Tiny Images dataset and consists of 60000 32x32 color images. $|D_k|$ between $|D_1| = N_{\max}$ and $|D_{100}| = N_{\min}$ follows an linear decay.The Hyper parameters augmentation probability ,Number of test samples and Acceptance Rate as mentioned in the original paper.The values used are 0.6 for augmentation probability,10 for number of test samples and 0.5 for Acceptance rate.The hyper parameter sensitivity analysis done on the three hyper parameters further substantiated the values used.

3.3 Experimental setup and code

We have conducted all experiments for the dataset Cifar100-LT by using the official repository, which is implemented in PyTorch .The code from the repository was reorganized into a Jupyter notebook to enhance portability and offer better control over the environment.The dependencies were not explicitly provided, and the versions of different libraries were determined through trial and error. Deprecated elements within the code were replaced with suitable, up-to-date alternatives.For training the model, parameters were set based on the specifications outlined in the paper. Any parameters not explicitly mentioned in the paper were assumed to use default values. The code for conducting component analysis on curriculum learning, class-wise score measurement, and measuring the standard deviation in weight L1 norm were re-implemented based on the specifications provided in the paper.The code and the readings are available here.

3.4 Computational requirements

We trained the model in kaggle with 1 NVIDIA Tesla P100 as the GPU accelerator.The average training time of the model was approximately 40 minutes with a batch size of 128 for 200 epochs.The overall budget of the study was 250 GPU hours.For each model, include a measure of the average runtime (e.g. average time to predict labels for a given validation set with a particular batch size).For each experiment, include the total computational requirements (e.g. the total GPU hours spent).

4 Results

The performance analysis of accuracy with CUDA consistently demonstrates an increase across all models. The examination of LOL scores, representing augmentation strength, broadly supports the claim that

ALGORITHM	IR=100	IR=50	IR=10	IR=100			IR 50			IR 10	
				Many	Med	few	Many	Med	few	Many	few
CE	38.49	42.92	56.61	65.8	36.27	8.43	63.8	34.73	13.83	63	42.53
CE+CMO	42.08	46.93	60.08	68.87	41.23	11.7	67.47	38.73	18.53	66.4	45.83
CE + CUDA	42.27	46.91	59.79	70.47	41.93	9.6	69.17	38.59	14.83	66.7	44.27
CE + CMO + CUDA	43.01	48.14	58.62	70.2	42.47	11.77	68.6	40.03	19.6	65.5	43.17
CE-DRW (Cao et al., 2019)	40.86	46.22	57.83	62.33	41.2	15.23	61.63	40.17	24.57	62.1	48.13
CE-DRW + Remix (Chou et al., 2020)	45.8	49.5	59.2	~	~	~	~	~	~	~	~
CE-DRW + CUDA	47.23	51.9	61.76	62.5	49.2	27.03	62.3	48.3	36.17	64.2	56.1
LDAM-DRW (Cao et al., 2019)	42.48	47.15	57.96	62.17	42.43	19.4	62.43	41.07	25.8	63.3	45.9
LDAM + M2m (Kim et al., 2020)‡	43.5	~	57.6	~	~	~	~	~	~	~	~
LDAM-DRW + CUDA	47.52	50.57	58.18	66.43	50.27	22.03	66.07	45.63	26.3	63.33	46.6
BS (Ren et al., 2020)	42.24	45.99	58.29	59.63	41.33	22.8	58	41	29.87	61.5	51
BS + CUDA	47.47	52	61.52	63.03	48.8	27.83	62.53	47.03	39	64.13	55.53
RIDE (3 experts) (Wang et al., 2021)	48.6	51.4	59.8	~	~	~	~	~	~	~	~
RIDE (3 experts)	48.87	51.95	59.13	67.7	50.17	25.27	67.2	46.5	29.3	64.9	46.1
RIDE + CMO (Park et al., 2022)†	0	53	60.2	~	~	~	~	~	~	~	~
RIDE + CMO	49.61	52.78	59.91	68.03	50.9	26.47	67.23	47.17	32.37	65.3	47.73
RIDE (3 experts) + CUDA	49.53	53.58	61.21	68.23	50.7	24.63	67.7	49	31.6	66.24	50

Figure 3: Validation accuracy on CIFAR-100-LT dataset. † are from Park et al. (2022) and ‡, are from the original papers (Kim et al. (2020); Zhu et al. (2022)). Other results are from our implementation.We report the average results of three random trials.

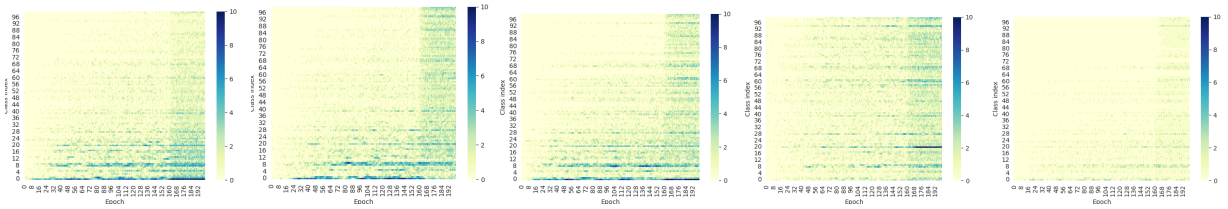

Figure 4: Evolution of LOL score in the order CE,CE-DRW,LDAM-DRW,BS,RIDE

"Stronger augmentation on majority classes and milder augmentation on minority classes improves performance," although a definitive trend is not readily apparent. The gain in accuracy attributed to CUDA diminishes as the imbalance ratio decreases. We observe a concavity in performance when hyper-parameters augmentation probability and accept rate are varied. Additionally, minimal fluctuations in accuracy are observed with variations in the hyper-parameter number of test samples. A drop in performance is observed when classwise score or curriculum learning is omitted from the model training process.

4.1 Results reproducing original paper

4.1.1 Comparison of validation accuracy

We measure the validation accuracy of CUDA when used with CE(Cross Entropy), CE-DRW (Cross entropy Dynamic reweighting)(Cao et al., 2019), LDAM-DRW (label-distribution-aware margin loss)(Cao et al., 2019), BS (balanced soft-max)(Ren et al., 2020) and RIDE(Wang et al., 2021) for the CIFAR-100-LT dataset following the general settings outlined in Cao et al. (2019). Specifically, we use ResNet-32 (He & Garcia, 2009)as the backbone network. The network is trained using stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 0.0002. The initial learning rate is set to 0.1, and a linear learning rate warm-up is applied during the first 5 epochs to reach the initial learning rate. The training process spans over 200 epochs, during which the learning rate is decayed at the 160th and 180th epochs by a factor of 0.01.The hyperparmeters: acceptance rate, augmentation probability and Number of test samples were 0.6,0.5 and 10 respectively. From figure3 we can observe a consistent increase in accuracy across all 5 models for the imbalance ratios 100 and 50 when paired with CUDA compared to the Vanilla edition. However for the imbalance ratio 10 we see a minor improvement in accuracy for all models except when CUDA is paired with CE+CMO.

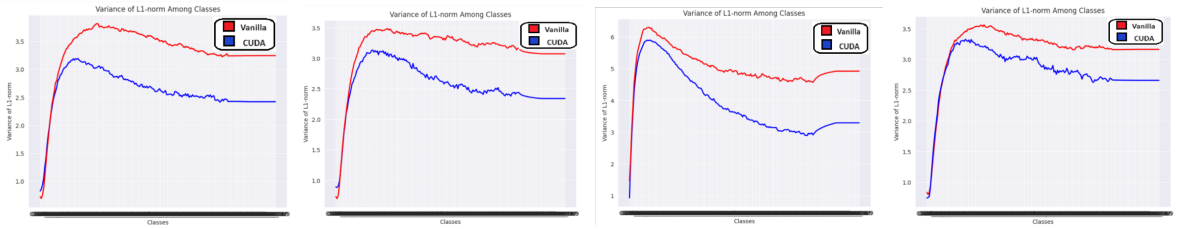

Figure 5: Variation Of weight-L1 norm in the order CE,CE-DRW,LDAM-DRW,BS

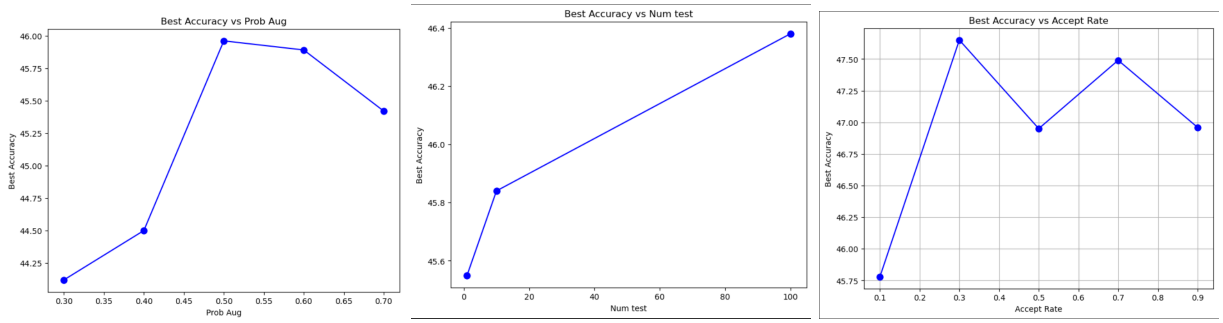

Figure 6: Hyper parameter Analysis on the LDAM-DRW

4.1.2 Dynamics of LOL Score

We plot the progression of LOL scores for various classes across five models: CE, CE-DRW, BS, LDAM-DRW, and RIDE. Notably, while there isn't a linear drop in augmentation strength as we move along the y-axis, there's a clear trend of higher average augmentation strength for the majority classes (0-49) compared to minority classes (50-99). The heatmaps presented in Figure 4 validate the assertion that "Stronger augmentation on majority classes and weaker augmentation on minority classes yields better performance." A notable surge in LOL scores across most classes occurs after the 160th epoch, likely attributable to the decay in learning rate beyond this epoch.

4.1.3 Variance of Weight L1-norm

Image recognition can be considered to be a coupling of two tasks:feature learning where we extract features from the images and embed it into a feature space and classifier learning where we train a classifier over the learned features(Kang et al., 2020).The long-tailed data distribution can corrupt the representation space, where the distance between head and tail categories is much larger than the distance between two tail categories (Fu et al., 2023).A naively trained model on long-tailed class distributed data tends to have "artificially" large weights for the head classes.This yields a wider classification boundary in feature space for the head classes, allowing the classifier to have much higher accuracy on head classes, but hurting the performance of the tail classes (Kang et al., 2020).The variance of classifier weight norm is usually used to measure how balanced the classifier is considering the input from a class-wise perspective(Kang et al., 2021). A lower variance in weight L1 norm indicates that the classifier assigns similar importance to all classes.

We analyze how CUDA affects standard deviation of weight L1 norm across the 5 models over the epochs. We observe that there is a significant decrease in the standard deviation of weight L1 norm when we use CUDA compared to the vanilla version from Figure 5.Any enhancements in performance can be attributed to either improved feature representation or a more effective classifier. In this case, since the classifier remains constant for each case, the improvements are likely due to CUDA optimizing the feature representation space.The variance in weight norm decreases implying a reduction in distinction between different classification boundaries(between two head classes or between two tail classes or between a head and a tail class).Specifically we can infer that CUDA is able to make the feature representation more balanced in terms of inter class distances (Song et al., 2015).

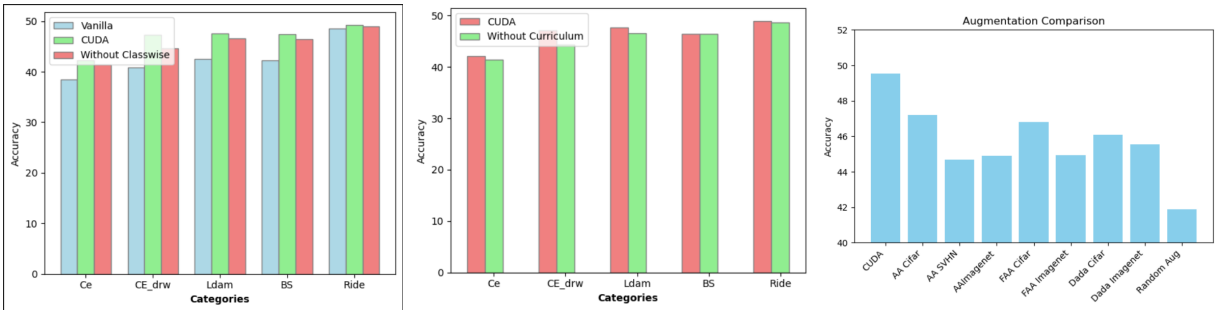

Figure 7: CA on Classwise Score and Curriculum Learning,Comparison of accuracy on different augmentation techniques

4.1.4 Hyper-Parameter Analysis

The paper proposed training on RIDE but we went with LDAM-DRW as LDAM-DRW showed a higher increase in performance when paired with CUDA which means that the magnitude of change in performance with change in hyper parameter will be higher.For Acceptance-Rate we train the model LDAM-DRW with CUDA for 5 equally spaced values from 0.1 to 0.9. Accept rate of 0.1 means that threshold for accepting the augmented samples is low leading to higher augmentation strength. Accept rate of 0.9 means that threshold for accepting the augmented samples is high leading to lower augmentation strength. For Probability-augmentation we train the model LDAM-DRW with CUDA for 5 equally spaced values from 0.3 to 0.7 .Probability-augmentation of 0.3 means that most of the original images is retained when forming the data-loader. Probability-augmentation of 0.7 means that most of the original images is replaced by the augmented images when forming the data-loader . From Figure 6 we observe that Acceptance-Rate and Probability-augmentation shows a concavity in the performance. For the hyper parameter number of test samples we train the model LDAM-DRW with CUDA for three different values of 1,10,100.We can see that the performance increases slightly with increase in Number of Test samples from Figure 6.

4.1.5 Curriculum Learning

Curriculum Learning (CL) is a training strategy designed to enhance machine learning models by progressively exposing them to increasingly complex or challenging data during training.Previous works (Zhou et al., 2020) have shown that curriculum learning can improve accuracy of LTR models.In the context of the CUDA algorithm, the LOL (Learning Objective Level) scores for each class initially start at zero and are iteratively updated at the end of each epoch based on the model's performance with augmented images. After 200 epochs, an optimal combination of LOL scores is achieved, leading to the final model performance.To assess the impact of curriculum learning on accuracy, a two-step approach is employed. In the first step, a model is trained using the standard CUDA procedure. Subsequently, the LOL scores obtained from this initial training run are extracted and utilized as fixed scores in a subsequent run. In this second run, the model is trained without updating the LOL scores. Figure7 shows that there is an decrease in performance across all 5 models when CUDA is trained without curriculum learning.

4.1.6 Classwise-Score

To examine the validity of class-wise augmentation of CUDA, we apply CUDA with the same strength of DA for all classes. Instead of computing LOL score class-wisely, we computed only one LOL score for the entire dataset by uniformly random sampling instances in the training dataset regardless of class. Figure7 shows a significant performance degradation of CUDA across all 5 models without class-wise augmentation compared to CUDA.

ALGORITHM	IR=100	IR=50	IR=10
CE	38.49	42.92	56.61
CE+CUDA(without cutout)	40.02	45.36	58.21
CE + CUDA(with cutout)	42.27	46.91	59.79

ALGORITHM	IR=100	IR=50	IR=10
CE + CMO + CUDA	0.93	1.23	-1.46
CE-DRW + CUDA	6.37	5.68	3.93
LDAM-DRW + CUDA	5.04	3.42	0.22
BS + CUDA	5.23	6.01	3.23
RIDE (3 experts) + CUDA	0.66	1.63	2.08

Figure 8: Analysis of accuracy on Cutout and Analysis of gain in accuracy with imbalance ratio

4.1.7 Comparison with other Augmentation techniques

We compare the performance of CUDA with other augmentation techniques:Auto-Augmentation(Cifar,Imagenet and SVHN policy) (Cubuk et al., 2019),Fast Auto-Augmentation(Cifar,Imagenet and SVHN policy)(Lim et al., 2019),DADA (Li et al., 2020) and Rand-Augmentation(m=1,n=2) (Cubuk et al., 2020)for the model LDAM-drw.Our analysis on figure 7 reveals that CUDA consistently outperforms all other augmentation techniques.

4.2 Results beyond original paper

4.2.1 Cutout

The usage of the augmentation operation Cutout was not explicitly mentioned in the paper. Cutout has proved to improve generalization performance of CNNs (DeVries & Taylor, 2017). Our analysis focused on investigating how Cutout affects the performance of CUDA for the CE (cross-entropy) model. Our findings in Figure8 suggest that incorporating cutout during training is crucial for achieving the reported accuracy levels stated in the original paper.

4.2.2 Gain in accuracy for imbalance ratio

We conducted a comparative analysis of the gain in accuracy among five models when paired with CUDA across three different imbalance ratios: 100, 50, and 10.Imbalance ratios reflect the disparity in class distribution within the dataset, with higher ratios indicating more pronounced class imbalances. Our findings in Figure 8 reveal a general trend where the gain in accuracy diminishes as the imbalance ratio decreases.We can infer that CUDA performs better when the imbalance ration of the dataset is higher.

5 Discussion

The experimental results presented in the paper effectively support its claims, demonstrating improvements in validation accuracy. However, the performance of CUDA with model BCL couldn't be fully evaluated due to inadequate time. Nevertheless, we can still validate the first claim based on the available data. There is a noticeable difference in the LOL scores between the majority class (0-49) and minority class (50-100) before the 160th epoch. However, after training for 200 epochs, this difference becomes insignificant. Despite this, a holistic analysis of the evolution of the LOL scores across epochs still supports the second claim. By analyzing the variance of weight L1-norm, the study showcases how CUDA effectively addresses the imbalance problem, leading the model to assign equal importance to each class. We were unable to conduct a study on classwise feature cosine similarity, which could have further validated the efficiency of CUDA on solving the imbalance problem.The impact of curriculum and classwise score on the performance of CUDA is evident from our findings. Notably, Cutout plays a significant role in the accuracy gain observed when using CUDA, even more so than curriculum learning or classwise score.The Hyper-parameter sensitivity analysis for augmentation probability and Accept rate on validation accuracy reveals a concavity as claimed by the paper.Additionally, our analysis on the imbalance ratio indicates that CUDA's efficiency diminishes as we move towards datasets with lesser imbalance. From this, we can also infer that CUDA may not provide significant improvements with balanced datasets.

5.1 What was easy

The paper was really easy to follow. The section on the repository for Cifar100-LT was clearly written. The description of the arguments that we pass during training was properly stated.The algorithm, which makes up CUDA, was completely logical in its implementation.The lack of significant barriers in setting up the code enhances its portability.

5.2 What was difficult

The dependencies were not clearly mentioned by the author requiring additional time to find the versions by trial and error. The original paper had also included performance comparison on datasets Imagenet-LT and Inaturalist-18, the section of the original repository for imagenet-LT and inat-18 contains redundant code and uncleaned up code.The paper claims to deviate from the model recipes of BCL and NCL to ensure a fair comparison. However, it fails to clearly state these deviations, making it difficult to assess their performance in this study.

5.3 Communication with original authors

The initial attempts to contact the authors through the email IDs given in the paper was not successful.We were able to contact the authors through linked-in in latter half of the study.The authors were able to clarify our doubts on implementing the component analysis for curriculum learning and class-wise score.Regrettably, the authors were unable to provide a clear recipe for BCL.

References

Sumyeong Ahn, Jongwoo Ko, and Se-Young Yun. Cuda: Curriculum of data augmentation for long-tailed recognition, 2023.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106:249–259, 2018.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Siming Fu, Xiaoxuan He, Xinpeng Ding, Yuchen Cao, and Hualiang Wang. Uniformly distributed category prototype-guided vision-language framework for long-tail recognition, 2023.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1gRTCVFvB`.

Bingyi Kang, Yu Li, Sa Xie, Zehuan Yuan, and Jiashi Feng. Exploring balanced feature spaces for representation learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=OqtLIabPTit`.

Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. 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.

Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. Differentiable automatic data augmentation. In *European Conference on Computer Vision*, pp. 580–595. Springer, 2020.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *Advances in Neural Information Processing Systems*, 32, 2019.

Seulki Park, Youngkyu Hong, Byeongho Heo, Sangdoo Yun, and Jin Young Choi. 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.

Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding, 2015.

Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=D9I3drBz4UC`.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9719–9728, 2020.

Jianggang Zhu, Zheng Wang, Jingjing Chen, Yi-Ping Phoebe Chen, and Yu-Gang Jiang. Balanced contrastive learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6908–6917, 2022.

A Augmentation preset

We use 22 different augmentation operations for CUDA each having their own parameter.The details of each of these operation has been described in table 1.The magnitude parameter divides the augmentation parameter into 30 values linearly.For example for Rotate max value is 30 and min value is 0,the magnitude of parameter for rotate is defined by

$$m_{\text{rotate}}(s) = (30-0)/30 * s, \text{ thus } m_{\text{rotate}}(1) = 1 = (30-0)/30 * 1$$

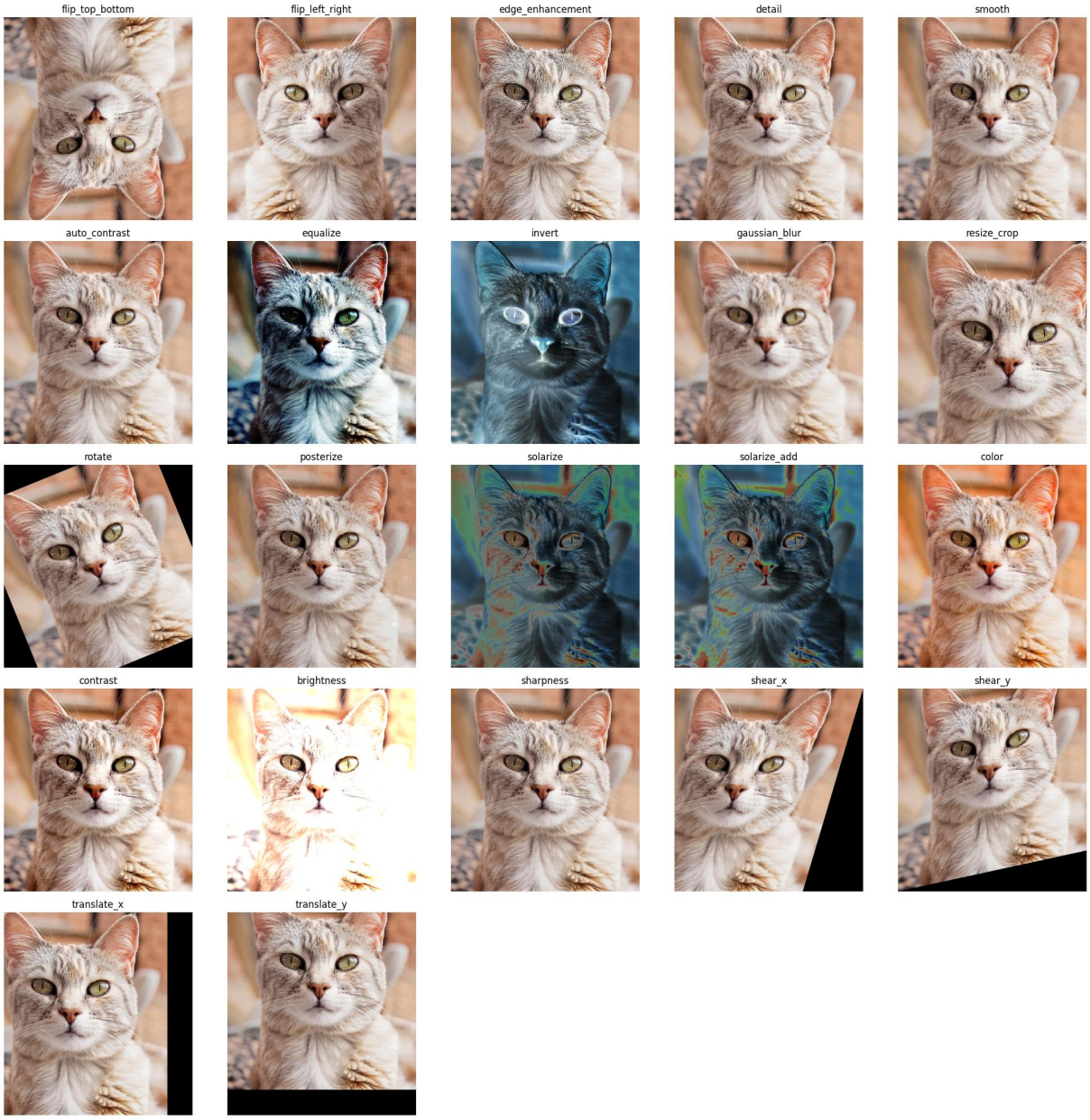

Figure 9: The 22 augmentation operations we use for CUDA

Table 1: Operation Parameter Description

Operation	**Description**
Flip On/Off	Flip top and bottom
Mirror On/Off	Flip left and right
Edge Enhancement On/Off	Increasing the contrast of the pixels around the targeted edges
Detail On/Off	Utilize convolutional kernel $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 10 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
Smooth On/Off	Utilize convolutional kernel $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
AutoContrast On/Off	Remove a specific percent of the lightest and darkest pixels
Equalize On/Off	Apply non-linear mapping to make uniform distribution
Invert On/Off	Negate the image
Gaussian Blur	Blurring an image using Gaussian function with radius [0,2]
Resize Crop	Resizing and center random cropping with scale [1,1.3]
Rotate	Rotate the image with angle [0,30]
Posterize	Reduce the number of bits for each channel in the range [0,4]
Solarize	Invert all pixel values above a threshold in the range [0,256]
SolarizeAdd	Adding value and run solarize in the range [0,110]
Color	Colorize gray scale values in the range [0.1, 1.9]
Contrast	Adjust the distance between colors in the range [0.1,1.9]
Brightness	Adjust image brightness in the range [0.1,1.9]
Sharpness	Adjust image sharpness in the range [0.1,1.9]
Shear X	Shearing X-axis in the range [0,0.3]
Shear Y	Shearing Y-axis in the range [0,0.3]
Translate X	Shift X-axis in the range [0,100]
Translate Y	Shift Y-axis in the range [0,100]