BLT: Balancing Long-Tailed Datasets with Adversarially-Perturbed Images Supplemental Material

Jedrzej Kozerawski[1], Victor Fragoso[2], Nikolaos Karianakis[2], Gaurav Mittal[2], Matthew Turk[1,3], and Mei Chen[2]

UC Santa Barbara[1] Microsoft[2] Toyota Technological Institute at Chicago[3]
jkozerawski@ucsb.edu {victor.fragoso, nikolaos.karianakis, gaurav.mittal, mei.chen}@microsoft.com mturk@ttic.edu

In this document, we present a series of additional experiments showcasing the effect of some parameters, visualize perturbed images via the gradient-ascent technique, and discuss architecture and implementation details.

1 Extended Ablation Studies

1.1 Adversarially Perturbed Images

In Fig. 1 we show a few examples of images perturbed using our gradient-ascent-based technique. The figure shows at the top row the original images, the middle row shows visualization of the perturbations, and bottom row shows the perturbed images. We can see that the perturbed images do not show significant visual artifacts or modifications compared to the original images. However, as our experiments demonstrated, the perturbations are beneficial in improving performance on tail classes.

1.2 BLT Reduces Confusion as Epochs Progress

We visualize in Fig. 3 how the most confusing categories for **chow** and **stone wall** tail classes from ImageNet-LT evolve throughout different epochs when using our proposed BLT. The y-axis of the visualizations show the epochs and progress from top to bottom. On the x-axis we show the confusing category indices. The visualization shows a white pixel when there is confusion and black pixel otherwise. The reader should see the progression of the confusion in the visualizations from top to bottom. The Figure shows on the left column the confusion progression when BLT is not used, and on the right column when BLTis used. We can observe that from epoch 0 to about epoch 20, the confusion is present across different categories. After epoch 20 BLT (see right column) the confusion overall decreases for both cases but BLT shows that only a few categories remain as confusing (see how there are more black pixels on the right column compared to the left one). This experiment shows that generating images at every batch effectively reduces the confusion for few shot classes.

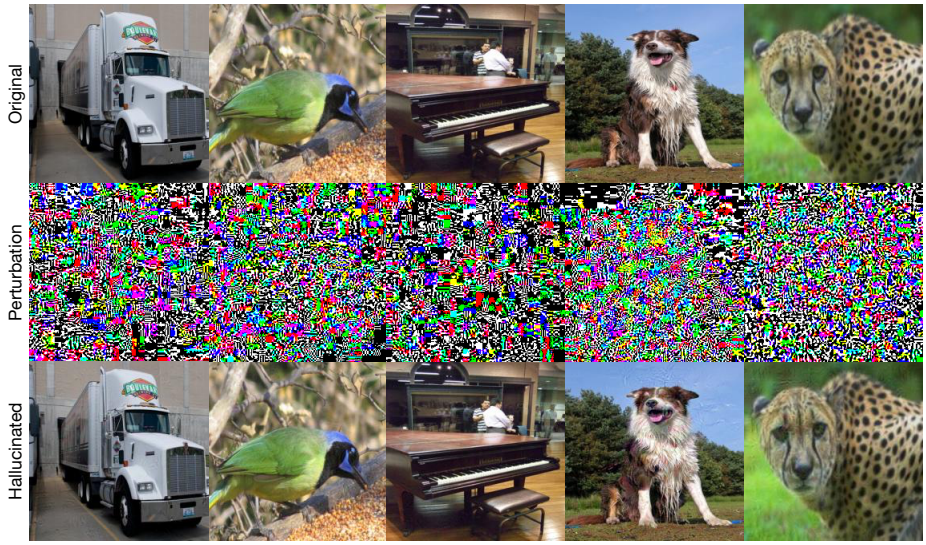

Fig. 1: Visualization of gradient-ascent-based images. Top-row shows original images, middle-row shows visualization of the perturbations calculated via the gradient-ascent technique, and bottom row shows the hallucinated or perturbed images.

1.3 Minimum confidence value

Our gradient-ascent-based image generation requires a minimum confidence value score as one termination criterion for the image generation step. Recall that an image is being perturbed to be classified as a category c' until the class score for that category reaches a predefined threshold, *e.g.*, $s_{c'} \geq S_{c'}(I')$ or when the process reaches 15 iterations. In Fig. 4 we show top-1 accuracy on the ImageNet-LT dataset for our BLT method with different minimum confidence thresholds $s_{c'}$. As we can see in Fig. 4, the higher the minimum confidence threshold is, the lower the few-shot accuracy is (by up to 5.9% lower). Simultaneously, both the many-shot and medium-shot performance slightly increase (by around 1%). Choosing a single constant value (0.2 here) as a minimum confidence threshold vs one drawn from a range of possible values (0.15-0.25) shows that the former method yields higher few-shot accuracy (by 1.2%), but lower many- and medium-shot performance (1% and 0.4% respectively) and lower overall accuracy. For that reason we chose in our implementation the threshold to be between 0.15-0.25 as it yields the highest overall accuracy, while simultaneously allowing for a high few-shot accuracy.

1.4 Impact of batch size and learning rate

We are interested to see the impact of varying batch size and learning rate combined when used in BLT. Table 1 shows the ablation study for different batch sizes with two different learning policies. Based on the results, we can see

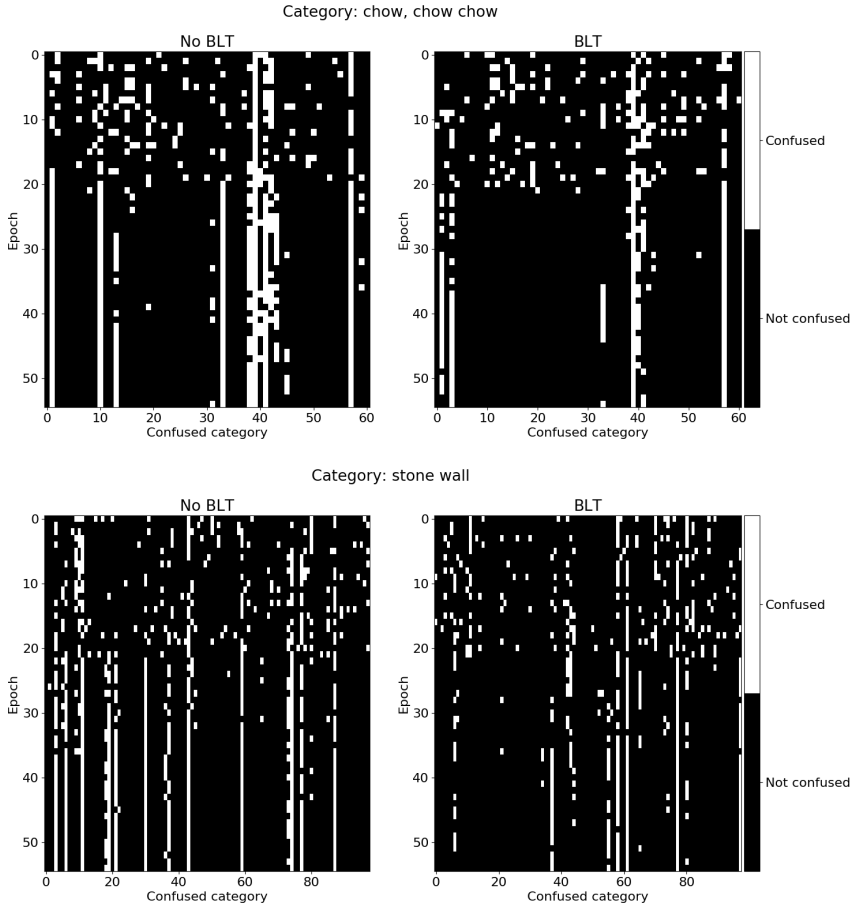

Fig. 3: Comparison of confused categories (x-axis) throughout different epochs (y-axis) when using BLT (right column) and without long-tail augmentation (left column). A white pixel indicates confusion while a black pixel indicates no confusion. Top row shows the results for chow category while the bottom row shows the results for stone wall class; both classes are tail classes. We observe that before 20 epochs the confusion is present across different categories. However, this confusion decreases after 20 epochs. In particular, we observe that BLT reduces the confusion (there are more black pixels compared to the left column).

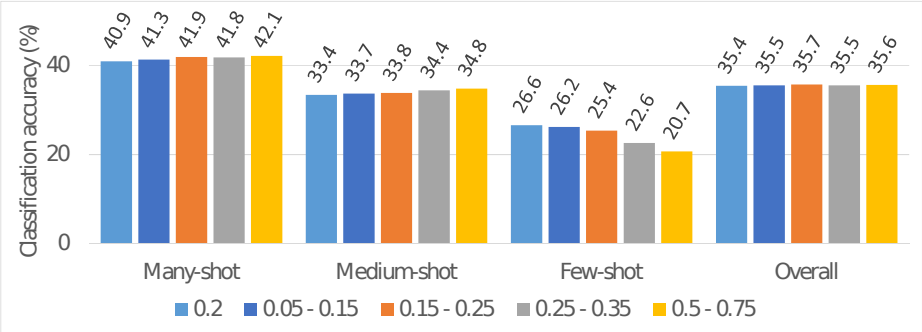

Fig. 4: Top-1 classification accuracy on ImageNet-LT for our BLT approach with different minimum confidence values used.

Table 1: Performance effect of BLT as a function of batch size and two different learning policies. **Left:** results with the learning policy described in Section 2 (with starting learning rates of 0.1, 0.01, and 0.1). **Right:** results when dividing the learning rates by a factor of 10 (starting learning rates of 0.01, 0.001, and 0.01). Smaller learning rate enables promotes gains in the many-shot accuracy over the few-shot accuracy while maintaining a comparable few-shot accuracy.

	Base learning rate				Smaller learning rate			
Batch size	Many	Medium	Few	Overall	Many	Medium	Few	Overall
256	41.6	33.4	26.3	35.5	36.2	25.3	20.5	28.8
128	40.8	33.0	25.4	34.9	42.0	31.0	22.8	34.1
64	39.9	32.9	25.9	34.6	45.5	33.6	22.8	36.6
32	33.4	27.8	**27.0**	31.0	**46.8**	**35.5**	22.6	**38.0**
16	22.7	17.2	21.4	20.0	44.1	34.3	21.3	36.2

that the smaller learning rate is more effective for smaller batch sizes with gains of 7% and 16.2% in overall accuracy for batch sizes of 32 and 16, respectively. In addition to that, we see that a smaller learning rate is responsible for an increase in many- and medium-shot accuracy (higher for smaller batch sizes) and a decrease in the few-shot accuracy. To mitigate this effect, we decided to increase the fraction of tail classes to adversarially perturb images from $p = 0.25$ to $p = 0.5$. Table 2 presents the results of this experiment. Increasing p improved the few-shot accuracy, while keeping the accuracy gains high for many- and medium-shot classes due to the lower learning rate. This works best for smaller batch sizes (e.g. batch size of 32 registers increase of 13% for many-shot classes, 6.9% for medium-shot and 7.1% overall compared to the base learning policy results presented on the left side of Table 1).

1.5 Different network architectures

To support our claim that our proposed approach does not require a specific architecture as a backbone - we have trained BLT on ImageNet-LT using six

Table 2: Results for a smaller learning rate with a higher fraction of tail classes to adversarially perturb ($p = 0.5$).

Batch size	Many	Medium	Few	Overall
256	35.6	24.6	22.7	28.6
128	41.3	30.0	25.5	33.7
64	44.6	33.0	26.3	36.5
32	**46.4**	**34.7**	**27.0**	**38.1**
16	41.2	32.4	24.6	34.6

Table 3: Top-1 accuracy comparison between a plain model + sampling baseline and BLT for different architectures.

	Plain + sampling				BLT			
Methods	Many	Medium	Few	Overall	Many	Medium	Few	Overall
EfficientNet-b0	29.1	25.2	16.5	25.5	33.1	27.3	24.3	29.1
ResNet-10	41.1	31.6	14.5	32.7	41.9	33.9	25.4	35.7
ResNet-18	47.3	34.2	14.4	36.3	47.9	36.1	26.1	39.2
ResNet-34	51.4	38.0	16.2	40.0	51.0	37.3	25.4	40.8
ResNet-152	41.1	34.8	16.8	35.0	48.0	39.5	29.8	41.4
DenseNet-121	53.4	39.2	17.9	41.5	54.3	41.7	28.9	44.7

different architectures. The results can be seen in the Table 3 comparing accuracy for different architectures between the plain model + sampling baseline and BLT. We can see that simply increasing the depth of the network does not help with the few-shot accuracy for the baseline approach, whereas for BLT both the ResNet-152 and the DenseNet-121 observe a significant increase in the few-shot accuracy (13% and 11% respectively) compared to the baseline method. Moreover we can observe, that the BLT keeps the many-shot and medium-shot accuracies the comparable or higher w.r.t. to the baseline method.

1.6 Additional ablation studies

In Table 4 we present results for variants of BLT on ImageNet-LT dataset. We substitute the Squashing-Cosine Classifier for a softmax classifier, add hallucinated images for all categories (instead of just for tail classes) and we modify the sampling strategy to be uniform instead of oversampling the tail categories. Both the uniform sampling and the BLT for entire batch result in higher many-shot accuracy, but much lower few-shot accuracy. The change in the classifier to softmax adds 0.5% accuracy to few-shot categories, but lowers all the other more significantly thus showing the necessity of the squashing-cosine classifer.

Additional experiments performed on Places-LT dataset are present in Table 5. They reflect the same experiments done on ImageNet-LT and shown in Fig.3a of our main paper and described in the section Hallucinations vs Augmentations. These experiments show consistently that substituting hallucinations for augmentations (Augmentations) or removing them (Neither) in BLT result in a deteriorated performance among few-shot categories.

Table 4: Additional ablation studies for BLTon ImageNet-LT. All presented variants of BLT show how delicate the balance between many- and few-shot accuracy is. BLT with Squashing-Cosine Classifier, with the proposed balancer and augmenting only images from tail classes (row 1 below) maintains the optimal balance. We show in **bold** and **blue** the highest and the second highest accuracy, respectively.

Batch size	Many	Medium	Few	Overall
BLT	44.4	**33.5**	**25.5**	**36.6**
BLT with softmax	41.2	32.1	**25.9**	34.5
BLT for entire batch	**46.1**	**35.1**	19.0	**37.0**
BLT with uniform sampling	**54.6**	23.5	8.1	33.2

Table 5: Comparison of hallucinations vs augmentations on Places-LT. Results below show the impact of hallucinations on the few-shot accuracy that cannot be substituted by an oversampling with additional just augmented images.

Batch size	Many	Medium	Few	Overall
Plain model + sampling	37.8	13.0	0.8	19.3
BLT	31.0	27.4	14.1	25.9
Augmentations	32.4	27.1	8.2	25.0
Neither	32.3	27.5	8.6	25.2

2 Implementation Details

In our experiments we have used two different architectures: ResNet-10 for ImageNet-LT and Places-LT datasets and ResNet-34 for iNaturalist 2018. Both implementations follow a simple technique, where we remove the last fully-connected layer and substitute it with a different fully connected layer - in this case of dimensions 512×512 (same as Liu *et al.* [1]). On top of that we have added the squashing-cosine classifier (with $\alpha = 20$ and $\beta = 0.5$) as the final layer. We train the network in two stages (both with the same architecture): first for 35 epochs without data balancing (or with $\gamma = 0$) and without the gradient-ascent image generation; and at the second stage for 55 epochs with data balancing ($\gamma = 0.9$) and gradient-ascent image generation. We used SGD optimizer with a momentum of 0.9, weight decay of 0.0005 and with an initial learning rate of 0.1 for stage 1 and 0.01 for stage 2 for the backbone and of 0.1 for the squashing-cosine classifier. For stage 1 the learning rate drops by a factor of 10 at epochs 25 and 32, and for stage 2 at epochs 23, 38 and 52.

3 Time analysis

To provide more in-depth information about the time efficiency of using per-batch gradient ascent method for image generation, we measure the average time it takes to run BLT per image, per batch, and per epoch. The results of this experiments are shown in Table 6. We ran the experiments on a PC with 20 CPUs, 128 GB of RAM, and 1 Nvidia RTX 2080 Ti GPU. BLT is about 7 times

Table 6: Time analysis for the training procedure on ImageNet-LT dataset between the plain model with sampling baseline and BLT. Per image, batch, and epoch times show the normalized training time in seconds, whereas the final row shows the total training time. As the proposed method can be flexible and the user can decide to augment less batches, or add BLT augmentation to only few epochs - we can see that the additional time can be even smaller.

Time	Plain + sampling	BLT
Per image [s]	0.00109	0.00309
Per batch (256) [s]	0.28	0.852
Per epoch [s]	126	259
Total training time	3h 10min	7h 3min

more efficient than GANs as generating images for ImageNet-LT adds 3hrs and 53 mins to the regular 3hrs 10 mins training time for a vanilla CNN (compared to additional 48 hrs to just train a GAN [2]). To show the impact of the BLT on the training time in more detail, we also provide normalized times on per epoch, per image, and per batch of size 256. As BLT augments the batch once - the bigger the batch size, the more time efficient the training procedure will be, and the BLT does not change the inference time at all.

References

1. Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., Yu, S.X.: Large-scale long-tailed recognition in an open world. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2019)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)