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

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

Abstract. Real visual-world datasets tend to have few classes with large numbers of samples (*i.e.*, head classes) and many others with smaller numbers of samples (*i.e.*, tail classes). Unfortunately, this imbalance enables a visual recognition system to perform well on head classes but poorly on tail classes. To alleviate this imbalance, we present BLT, a novel data augmentation technique that generates extra training samples for tail classes to improve the generalization performance of a classifier. Unlike prior long-tail approaches that rely on generative models (*e.g.*, GANs or VQ-VAEs) to augment a dataset, BLT uses a gradient-ascent-based image generation algorithm that requires significantly less training time and computational resources. BLT avoids the use of dedicated generative networks, which adds significant computational overhead and require elaborate training procedures. Our experiments on natural and synthetic long-tailed datasets and across different network architectures demonstrate that BLT consistently improves the average classification performance of tail classes by 11% w.r.t. the common approach that balances the dataset by oversampling tail-class images. BLT maintains the accuracy on head classes while improving the performance on tail classes.

1 Introduction

Visual recognition systems deliver impressive performance thanks to the vast publicly available amount of data and convolutional neural networks (CNN) [1–6]. Despite these advancements, the majority of the state-of-the-art visual recognition systems learn from artificially balanced large-scale datasets. These datasets are not representative of the data distribution in most real-world applications [7–12]. The statistics of the real visual world follow a long-tailed distribution [13–17]. These distributions have a handful of classes with a large number of training instances (head classes) and many classes with only a few training samples (tail classes); Fig. 1(a) illustrates a long-tailed dataset.

The main motivation for visual recognition is to understand and learn from the real visual world [14]. While the state of the art can challenge human performance on academic datasets, it is missing an efficient mechanism for learning tail classes. As Van Horn and Perona found [14], training models using long-tailed datasets often leads to unsatisfying tail performance. This is because the

Code available at: `http://www.github.com/JKozerawski/BLT`

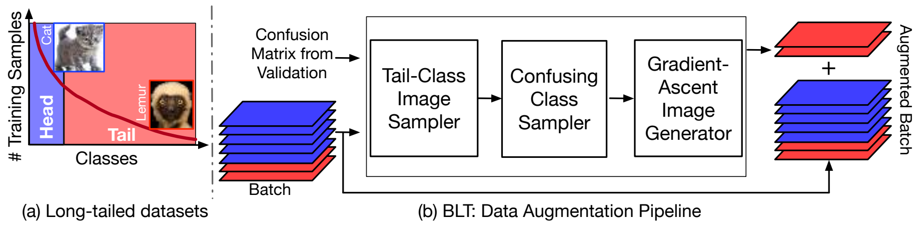

Fig. 1. **(a)** Real-world datasets are often naturally imbalanced as they present a long-tail distribution over classes. Some classes (*e.g.*, cats) have an abundant number of training instances (head classes) while others (*e.g.*, lemurs) have fewer training examples (tail classes). **(b)** BLT augments a training batch by generating images from existing tail class images to compensate for the imbalance in a long-tailed dataset. Unlike existing methods that rely on generative networks such as GANs or VAEs, BLT uses an efficient gradient ascent-based algorithm to generate hard examples that are tailored for tail classes. We show that BLT is flexible across different architectures and improves the performance of tail classes without sacrificing that of the head classes.

imbalance in real-world datasets imposes a bias that enables a visual recognition system to perform well on head classes but often poorly on tail classes.

To alleviate the bias imposed from a long-tailed dataset, learned classifiers need to generalize for tail classes while simultaneously maintaining a good performance on head classes. Recent efforts that aim to learn from long-tailed datasets modify the training loss functions [18–22], over- or under-sample a dataset to balance it [23, 24], or hallucinate or generate additional training instances (*e.g.*, images or features) [25]. Despite the progress of these efforts, the performance of visual recognition systems still falls short when trained using long-tailed datasets. There are two reasons that make these systems struggle on these long-tailed datasets. First, the information from the gradients of tail-class samples gets diminished given the prevalence of the head-class instances in the mini-batch. Second, more frequent sampling of instances from the tail classes reduces their training error but does not help the classifier to generalize.

Recent advances on generative approaches (*e.g.*, GANs [26, 27] and autoencoders [28]) enable the development of data augmentation techniques that make the generation of additional training samples for tail classes on the fly useful to address dataset imbalance. Although these generative approaches can hallucinate impressively realistic imagery, they incur adaptations that are computationally expensive. Specifically, adding these generative approaches into a per-batch data augmentation policy requires training an additional neural network and adapting its sophisticated training procedures. This adds significant overhead in terms of training time, computational complexity, and use of computational resources on top of training the CNN-based image classifier.

To circumvent the cumbersome requirements of adopting a generative approach in long-tail recognition, we propose an efficient solution for Balancing Long-Tailed datasets (BLT) which, at its core, embraces gradient ascent-based adversarial image hallucination [29–31]. This approach removes the requirement of using an additional network to generate images for tail classes (*e.g.*, GANs or autoencoders). As a result, BLT waives the need for extensive training procedures for the generator, thus keeping the computational complexity and resources low. Instead of perturbing images to purely confuse a CNN-based image classifier, as it is done for increasing robustness of a CNN [32–34], BLT perturbs tail-class images in a batch to make them hard examples, adds them to the batch, and proceeds with the regular training procedure. BLT generates hard examples by computing image perturbations that make the classifier confuse an image from a tail class with a confusing class based on the confusion matrix. Fig. 1(b) shows an overview of our proposed data augmentation technique.

Our experiments on publicly available real and synthetic long-tail image-classification datasets show that BLT consistently increases the average classification accuracy of tail classes across different network architectures while maintaining the performance on head classes. Our experiments show that BLT increases the classification performance on tail classes by 11% w.r.t. the common approach of oversampling tail-class images to balance a long-tailed dataset.

The contributions of this work are the following:

1. BLT, a data augmentation technique that uses gradient ascent-based adversarial image generation to compensate the imbalance in a long-tailed dataset;
2. A quantitative analysis to demonstrate that BLT improves the generalization of a classifier on tail classes while maintaining its overall performance; and
3. An extensive evaluation on synthetically and organically long-tailed datasets to validate the flexibility of BLT on different network architectures.

2 Related Work

The main challenge of learning models from long-tailed datasets involves learning parameters that generalize well from few training instances while maintaining the accuracy of head classes. Many of the existing methods that address the problem of learning from a long-tailed dataset modify the training loss, balance the dataset via sampling techniques, or hallucinate data. Since BLT uses techniques designed to address classification robustness, this section also covers adversarial image perturbations, and image and feature hallucinations.

2.1 Learning from Long-Tailed Datasets

The simplest techniques that deal with long-tailed datasets use random sampling to artificially create a more balanced training set [23]. The two most common techniques are oversampling and undersampling. Oversampling picks training instances from tail classes more often. On the other hand, undersampling selects instances from head classes less frequently. In practice, oversampling tail classes tends to alleviate the bias from long-tailed datasets. Liu *et al.* [24] proposed an approach that exploits data balancing and a modular architecture to solve learning from an long-tailed dataset but also in an open-set scenario [35].

A different set of approaches adapt the training loss function to learn from long-tailed datasets. Lin *et al.* [20] proposed an object detection loss designed to penalize more the misclassified ones. Song *et al.* [21] presented a loss that forces a network to learn a feature embedding that is useful for few-shot learning.

Cui *et al.* [18] presented a loss designed to better re-weight by means of the effective number of samples. Dong *et al.* [19] presented class rectification loss which formulates a scheme for batch incremental hard sample mining of minority attribute classes. Zhang *et al.* [22] developed a loss with the goal to reduce overall intra-class variations while enlarging inter-class differences. Zhong *et al.* [36] used different loss functions for head and tail class data while simultaneously introducing a noise resistant loss. Huang *et al.* [37] presented a quintuplet loss that forces a network to have both inter-cluster and inter-class margins.

The closest group of approaches to BLT hallucinate new data for tail classes to compensate for the imbalance in the dataset. Yin *et al.* [25] presented a face-recognition approach that generates new instances in feature space for tail classes. Their approach used an encoder-decoder architecture to produced novel features. Wang *et al.* [16] introduced MetaModelNet, a network that can hallucinate parameters given some knowledge from head classes. While these approaches alleviate the imbalance in a long-tailed dataset, they require training additional networks besides the CNN-based classifier.

2.2 Generating Novel Data for Few-shot Learning

The methods that leverage image generation techniques the most are those that tackle the one- and few-shot learning problems [38–41]. Peng *et al.* [42] created a method that used a generator-discriminator network that adversarially learned to generate data augmentations. Their method aimed to generate hard examples in an on-line fashion. Hoffman *et al.* [43] presented an approach that hallucinates features obtained by a depth prediction network for improving object detection. Zhang *et al.* [44] introduced a one-shot learning approach that hallucinated foreground objects on different backgrounds by leveraging saliency maps. Hariharan and Girshick [45] presented a method that used visual analogies to generate new samples in a feature space for few-shot categories. Gidiaris and Komodakis [46] generated weights for novel classes based on attention. Pahde *et al.* [47] used StackGAN to generate images based on textual image descriptions for few-shot learning. Wang *et al.* [48] hallucinated temporal features for action recognition from few images. Wang *et al.* [49] hallucinated examples using GANs trained in an end-to-end fashion combined with a classifier for few-shot classification. Chen *et al.* [50] presented a network that learns how to deform training images for more effective one-shot learning. Although these networks did not generate realistic images, Chen and colleagues demonstrated that they were still beneficial for one-shot learning. While many of these approaches can generate realistic imagery, they unfortunately lack adoption because they require a significant amount of effort to make them work as desired. Nevertheless, inspired by Chen *et al.* [50], we argue that images do not need to look realistic in order to compensate the lack of data of tail classes. Given this argument, we focus on efficient image generation via adversarial perturbations.

2.3 Adversarially-Perturbed Images

The goal of adversarial images is to fool CNNs [30, 32, 33, 51] or increase the robustness of a CNN-based classifier [52–56]. While some techniques use GANs [51] for generating adversarial images, there exist others that construct adversarial images by means of gradient ascent [30] or by solving simple optimization problems [32, 33]. The benefit of using adversarially-perturbed images as hard examples was shown by Rozsa *et al.* [57]. Because we are interested in generating images in an efficient manner, we focus on the gradient ascent-based method of Nguyen *et al.* [30]. This method computes the gradient of the posterior probability for a specific class with respect to an input image using back propagation [58]. Then, the method uses these gradients to compute an additive perturbation yielding a new image. While these methods have been useful to show weaknesses and increase robustness of many visual recognition systems, there has not been any approach exploiting these adversarial examples to learn from a long-tailed dataset.

Unlike many methods described in Sec. 2.2, BLT does not require dedicated architectures for image generations (*e.g.*, GANs or VAEs) and complex training procedures which can take days to train [59]. Instead, BLT uses the underlying trained CNN-based model combined with a gradient ascent method [30] to generate adversarial examples from tail-class images that are added to a batch.

3 BLT: An Efficient Data Augmentation Technique for Balancing Long-Tailed Datasets

The main goal of BLT is to augment a batch by generating new images from existing ones in order to compensate for the lack of training data in tail classes. With the constraint of not increasing the computational overhead considerably, we investigate the use of adversarial image perturbations [29–31] to generate novel images. Although these techniques create noise-induced imagery, we show that they are effective in compensating the imbalance in a long-tailed dataset and efficient to generate. We first review how to generate new images by perturbing existing ones via the gradient ascent technique [29–31].

3.1 Generating Images with Gradient Ascent-based Techniques

Generating an image via gradient ascent [29–31] requires evolving an image by applying a sequence of additive image perturbations. We review this technique assuming that we aim to confuse a classifier. Confusing a classifier requires maximizing the posterior probability or logit of a non-true class given an input image I. Mathematically, this confusion can be posed as follows: $I^\star = \arg\max_I S_c(I)$, where $S_c(I)$ is the score (*e.g.*, logit) of class c given I.

To confuse a classifier, the goal is to maximize the score $S_c(I)$ for a non-true class c. To generate image $I^\star$, the technique first computes the gradient of the scoring function $\nabla_I S_c(I)$ corresponding to a non-true class c w.r.t. to an input image I using backpropagation. Then, the technique adds a scaled gradient to the input image I, *i.e.*, $I \leftarrow I + \delta \nabla_I S_c(I)$, to produce a new image I. This technique repeats this process until the score $S_c(I)$ for a non-true class is large enough to confuse a classifier. Unlike generative approaches (*e.g.*, GANs or VQ-VAEs) that require an additional architecture to generate images (*e.g.*, encoder-decoder networks), specialized losses, and sophisticated training procedures, this

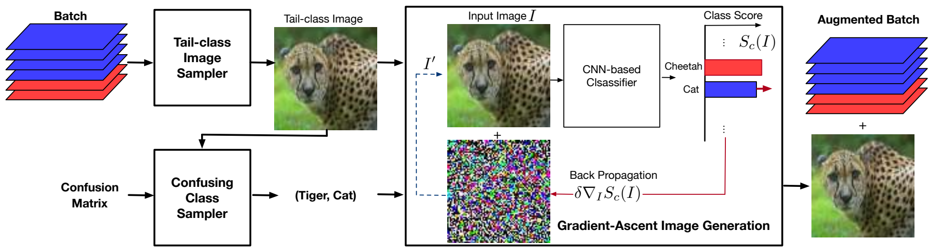

Fig. 2. BLT samples a tail-class image I from the batch and its confusion matrix from the latest validation epoch. Then, our algorithm passes I through the CNN and evaluates its class scores $S_c(I)$. Via back-propagation, our method computes the image perturbation that increases the class score of a selected confusing class (*e.g.*, cat) and adds the perturbation to the original image to produce I'. The perturbed image becomes the new input, *i.e.*, $I \leftarrow I'$. The technique iterates until the class score of a target non-true class reaches certain threshold or an iteration limit. Finally, BLT augments the input batch with the generated image to resume the regular training procedure.

technique evolves the image I using the underlying neural network and keeps its parameters frozen. Thus, BLT saves memory because it avoids the parameters of a generative model and uses efficient implementations of backpropagation from deep learning libraries to compute the image perturbations. Further, BLT is about 7 times more efficient than GANs as generating images for ImageNet-LT adds 3 hours and 53 minutes to the regular 3 hours 10 minutes training time for a vanilla CNN (compared to additional 48 hours to just train a GAN [59]).

3.2 Augmenting a Batch with Generated Tail-class Hard Examples

The goal of BLT is to generate images from tail classes using gradient ascent techniques to compensate for the imbalance in a long-tailed dataset. As a data augmentation technique, BLT generates new images from existing tail-class images in a batch. These additional images are generated in such a way that they become hard examples (*i.e.*, confusing examples for tail classes). To this end, BLT uses the results of a validation process to detect the most confusing classes for tail classes. Then, it perturbs the images in the batch belonging to tail classes in such a way that the the resultant images get a higher confusing class score. Subsequently, BLT appends the hard examples to the batch preserving their original tail-class labels and resumes the normal training procedure.

Algorithm 1 summarizes BLT. Given a batch $\mathcal{B}$, a list of tail classes $\mathcal{T}$, the fraction p of tail-class samples to process, and the confusion matrix from the latest validation epoch $\mathcal{C}$, BLT first initializes the augmented batch $\mathcal{B}'$ by copying the original input batch $\mathcal{B}$. Then, it iterates the training samples in the batch $\mathcal{B}$ and creates a list l which contains the identified tail-class samples (step 3). Next, BLT computes the number $n_{\mathcal{T}}$ of tail samples to process using the fraction p where $0 \leq p \leq 1$ in step 5. Then in steps 6-17, for each tail-class sample $(I, c) \in l$, BLT selects a confusing class c' for the tail class c from the confusion matrix $\mathcal{C}$ (step 10). Then, in step 12 BLT computes a minimum class score $s_{c'}$. Next, in step 14, BLT triggers the generation of a new image via the

Algorithm 1: BLT

```
Input  : Batch $\mathcal{B}$, list of tail classes $\mathcal{T}$, fraction $p$ of tail classes to process, and confusion
         matrix $\mathcal{C}$ from the latest validation epoch
Output: Augmented Batch $\mathcal{B}'$
1  $\mathcal{B}' \leftarrow \mathcal{B}$    // Initialize the output batch.
2  // Identify the tail classes present in the original batch.
3  $l \leftarrow$ IdentifyTailClasses $(\mathcal{B}, \mathcal{T})$
4  // Calculate the number of the tail classes to process.
5  $n_{\mathcal{T}} \leftarrow \lceil p \times \texttt{Length}(l) \rceil$
6  for $i \leftarrow 0$ to $n_{\mathcal{T}}$ do
7      // For the i-th tail class c, sample an image I of class c in the training set.
8      $(I, c) \leftarrow l\,[i]$
9      // Select a confusing class $c'$ for the i-th tail class c.
10     $c' \leftarrow$ SelectConfusingClass $(\mathcal{C}, c)$
11     // Sample a class score for $S_{c'}(\cdot)$.
12     $s_{c'} \leftarrow$ SampleClassScore ()
13     // Generate an adversarial image via iterative gradient ascent; see Sec. 3.1.
14     $I' \leftarrow$ HallucinateImage $(I, c', s_{c'})$
15     // Augment batch with the generated hard example.
16     $\mathcal{B}' += (I', c)$
17 end
18 return $\mathcal{B}'$
```

gradient ascent technique with a starting image I, target class c', and class score threshold $s_{c'} \geq S_{c'}(I')$. Lastly, BLT appends the new hard example (I', c) to the augmented batch $\mathcal{B}'$ (step 16) and returns it in step 18. When the input batch $\mathcal{B}$ does not contain any tail classes, then we return the input batch, *i.e.*, $\mathcal{B}' = \mathcal{B}$.

Our implementation of BLT selects a confusing class in step 4 by using information from the confusion matrix $\mathcal{C}$ for a given tail class c. Specifically, BLT computes a probability distribution over all classes using the confusion matrix scores for a tail class c. Then, it uses the computed distribution to sample for a confusing class c'. This strategy will select the most confusing classes more often. Subsequently, BLT computes the minimum class score $s_{c'}$ by randomly choosing a confidence value from within 0.15 and 0.25. Our implementation runs the gradient ascent image generation procedure with a learning rate $\delta = 0.7$. It stops running when $S_{c'}(I') \geq s_{c'}$ or when it reaches 15 iterations. BLT freezes the weights of the underlying network, since the goal is to generate new images. Fig. 2 illustrates how BLT operates.

BLT is independent of model architecture. However, there is an important aspect of using BLT and a class balancer (*e.g.*, oversampling [23]). Since BLT operates on a batch $\mathcal{B}$, it is possible that the batch contains many tail-class samples triggering BLT more often. When this happens, our experiments show that the performance of the head classes decreases. To mitigate this issue, the balancer needs to reduce the sampling frequency for tail classes. We introduce a procedure to achieve this for the widely adopted balancer: oversampling via class weights.

The simplest balancer uses class weights $w_i \geq 0$ to define its sampling policy using the inverse frequency, *i.e.*, $w_i = n_i^{-1} \cdot \sum_i^N n_i$, where n_i is the number of training samples for the i-th class. This balancer then normalizes the weights to compute a probability distribution over the N classes, and uses this distribution as a sampling policy. This balancer samples tail classes more frequently because their corresponding weights w_i tend to be higher. To reduce these weights of tail-classes, we introduce the following adaptation,

$$w_i = \frac{\sum_i^N n_i}{n_i^\gamma}, \tag{1}$$

where γ is the exponent that inflates or deflates the weights w_i. When $0 < \gamma < 1$, the proposed balancer samples head-class instances more frequently than the inverse-frequency balancer. On the other hand, when $\gamma > 1$, the balancer favors tail classes more frequently than the inverse-frequency balancer. This simple adaptation is effective in maintaining the performance of head-classes while significantly increasing the performance of tail classes (see Sec. 4.1).

3.3 Squashing-Cosine Classifier

We use an adapted cosine classifier combined with the Large-Margin Softmax Loss [60]. This is because it is a strict loss and forces a classifier to find a decision boundary with a desired margin. We generalize the squashing cosine classifier implemented by Liu *et al.* [24] by adding two parameters that allow us to balance the accuracy drop of head classes and the accuracy gain of tail classes. The adapted squashing-cosine classifier computes the following class scores or logits for class c as follows:

$$\text{logit}_c(\mathbf{x}) = \left(\frac{\alpha \cdot \|\mathbf{x}\|}{\beta + \|\mathbf{x}\|}\right) \frac{\mathbf{w}_c^\intercal \mathbf{x}}{\|\mathbf{w}_c\|\|\mathbf{x}\|}, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the feature vector of an image I, $\mathbf{w}_c \in \mathbb{R}^d$ is the weight vector for class c, α is a scale parameter, and β controls the squashing factor. We obtain the cosine classifier used by Liu *et al.* [24] when $\alpha = 16$ and $\beta = 1$.

3.4 BLT as a Bi-level Optimization and Regularization Per Batch

BLT can be seen as a learning process that uses bi-level optimization and regularization terms for tail classes at every batch. This is because the added images to the batch come from a gradient ascent procedure. Since the images in a batch go through the training loss and procedure, they consequently contribute gradients for the learning process. BLT can be seen as the following per-batch problem:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \frac{1}{|\mathcal{B}|} \sum_{(I_i, c_i) \in \mathcal{B}} \mathcal{H}\left(f_\theta\left(I_i\right), c_i\right) + \lambda [\![c_i \in \mathcal{T}]\!] \mathcal{H}\left(f_\theta\left(I'_{c_i}\right), c_i\right) \\ \text{subject to} \quad & I'_{c_i} = \arg\max_I f_\theta\left(I_i\right), s_{c'_i} \geq f_\theta\left(I_i\right); \forall c_i \in \mathcal{T}, \end{aligned} \tag{3}$$

where $f_\theta(\cdot)$ is the CNN-based classifier with parameters θ; $\mathcal{H}(\cdot)$ is a classification loss (*e.g.*, the Large-Margin Softmax loss or binary cross entropy loss); $[\![\cdot]\!]$ is the Iverson bracket; c_i is the class of I_i; c'_i is the class to confuse the classifier using gradient ascent techniques; and λ is the penalizing factor for mistakes on the generated images. Our implementation uses $\lambda = 1$.

BLT adapts its learning process at every batch. This is because in a stochastic gradient descent learning process, the parameters θ of the CNN-based classifier change at every batch. Thanks to this bi-level optimization and regularization, BLT generates images for tail classes that compensate the long-tailed dataset and forces the CNN-based classifier to generalize well on few-shot classes.

4 Experiments

This section presents a series of experiments designed to validate the benefits of BLT on long-tailed datasets. The experiments comprise an ablation study that reveals the performance effect of the BLT parameters and image classification experiments on synthetic and naturally long-tailed datasets that measure the accuracy of BLT applied on different architectures. We implemented BLT on PyTorch, and trained and ran CNN-based classifiers; see the supplementary material for all implementation details (*e.g.*, learning rate policies, optimizer, etc.). Our code is available at: `http://www.github.com/JKozerawski/BLT`

Datasets. We use two synthetic long-tailed datasets, ImageNet-LT [24] (1k classes, 5-1280 images/class) and Places-LT [24] (365 classes, 5-4980 images/class), and a naturally long-tailed dataset, iNaturalist 2018 [61]. We create a validation set from the training set for iNaturalist because BLT selects confusing classes at each validation epoch; the iNaturalist dataset does not contain a test set. To do so, we selected 5 training samples for every class and discarded the classes with less than 5 samples in the training set. We used the iNaturalist validation set modulo the removed classes. The modified iNaturalist dataset contains $8,122$ classes and preserves its natural imbalance with minimum of 2 and maximum of 995 imgs/class. Unless otherwise specified, we assume that the many-shot classes have more than 100 training images, the medium-shot classes have more than 20 and less or equal to 100 images, and the few-shot classes have less or equal to 20 training images. Every experiment reports the overall accuracy which is calculated as the average of per-class accuracies.

4.1 Ablation Study

We study the performance effect of the parameters in the adapted cosine classifier (see Sec. 3.3), the adapted balancer detailed in Sec. 3.2, the fraction p of tail-class images in a batch to process (see Sec. 3.2), the compensation of imbalance with common image augmentations versus those of BLT, and the effect of batch size on the accuracy achieved by BLT. We use ImageNet-LT dataset and ResNet-10 backbone for this study, and use a batch size of 256 for most experiments.

Squashing-Cosine Classifier. We study the effect on performance of the parameters of the adapted cosine classifier. For this experiment, we set $p = 0.5$ and $\gamma = 1.0$ and keep them fixed while varying the parameters α (scaling factor) and β (squashing factor) of the classifier. In Fig. 3(a) we see that the performance of few-shot classes decreases by about 3% and the accuracy of many-shot or head

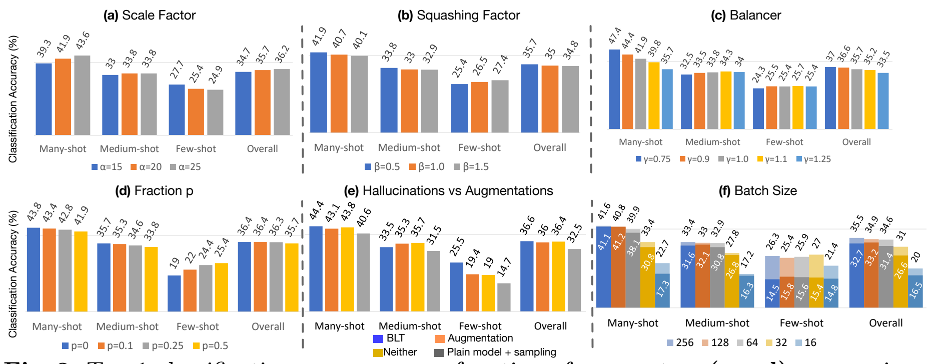

Fig. 3. Top-1 classification accuracy as a function of parameters **(a - d)**; comparison between BLT, common image augmentation, and sample balancing baselines **(e)**; and the effect of batch size **(f)**. **(a)** The performance of few-shot or tail classes deteriorates and the accuracy of many-shot or head classes improves when α increases. **(b)** The accuracy of tail classes improves and the accuracy of head classes decreases when β increases. **(c)** The many-shot accuracy decreases while the medium-shot and few-shot accuracy improves when γ increases. **(d)** The few-shot accuracy improves while the medium-shot and many-shot accuracy decreases as p increases. **(e)** Adding tail-class images in the batch (via sample balancing or image augmentations) improves the accuracy of few-shot classes. However, BLT further improves the accuracy of tail classes compared to common augmentations and BLT without appending images to the batch (Neither) while preserving the medium-shot and many-shot accuracy. **(f)** BLT (lightly colored bars) maintains the accuracy improvement on few-shot classes over plain balanced models (solidly colored bars) as the batch size decreases.

classes improves by about 4% when α increases. We see in Fig. 3(b) that the accuracy of few-shot or tail classes improves by about 2% and the performance of many-shot or head classes drops on average by 1% when β increases. Thus, setting these parameters properly can help a recognition system control gains or losses in the performance on head and tail classes.

Balancer. We analyze the effect of the parameter γ in the adapted weight-based balancer described in Sec. 3.2. For this experiment, we set $p = 0.5$, $\alpha = 20$, and $\beta = 0.5$ and keep them fixed while varying the γ parameter. In Fig. 3(c), we observe that the accuracy of many-shot or head classes decreases by about 11% while the performance of medium-shot and few-shot or tail classes improves by about 2% when γ increases. Thus, this parameter helps BLT control the decrease in the performance on head classes.

Fraction of Tail-class Images to Adversarially Perturb. We examine the classification accuracy as a function of the fraction of tail-class images in a batch to process (*i.e.*, p) by BLT. For this experiment we set $\alpha = 20$, $\beta = 0.5$, $\gamma = 1.0$ and vary p between 0 and 0.5. We observe in Fig. 3(d) that the accuracy of few-shot improves by about 6% while the performance of many- and medium-shot classes fall by about 2% when p increases.

Hallucinations vs Augmentations. BLT changes the statistics of the batch by supplementing it with hallucinated tail-class images. While this technique is effective in improving the accuracy of tail classes (see Sec. 4.2), it prompts the

question whether one can improve the accuracy of tail classes by augmenting the batch with images computed with an alternative approach, such as common image-augmentation techniques. To answer this question, we augment a batch with the same number of tail-class images using common augmentation techniques (*i.e.*, rotations, crops, mirror flips and color jitter) instead of hallucinated samples from BLT. For this experiment, we set $\alpha = 20$, $\beta = 0.5$, $\gamma = 1.0$, $p = 0.5$ and let the gradient ascent technique iterate in BLT for no more than 15 iterations; and included BLT without appending images to the batch and dubbed it "Neither". Fig. 3(e) shows that the performance of tail classes increases by augmenting the batch with tail-class images regardless of the image generation techniques (*i.e.*, image augmentations or gradient ascent techniques). However, the hard examples generated by BLT increase the accuracy of few-shot or tail classes compared to common image augmentation techniques by about 6% at a cost of an increase in confusion between medium- and few-shot classes.
Batch Size. Given that BLT operates on a batch, its size can affect the performance of BLT. We train a Resnet-10 model combined with BLT and a balancer with batch sizes varying from 16 to 256 and measure their accuracies. Fig. 3(f) shows the accuracies of the model combined with BLT (lightly colored bars) and sampling or balancer (solidly colored bars). We can observe that the accuracies of many- and medium-shot from BLT remain similar to those of the balancer and decrease when the batch size decreases. On the other hand, accuracies of few-shot classes remain stable when the batch size decreases and the accuracies of BLT are higher than those of the balancer.

4.2 Image Classification on Long-Tailed Datasets

The goal of this experiment is to measure the accuracy gain on tail classes that BLT brings. Similar to the experiments presented by Liu *et al.* [24], we used ResNet-10 and two-stage training approach. The first stage trains the underlying model without special long-tail techniques. On the other hand, the second stage starts from the weights learned in the first stage and applies all the techniques that reduce the bias from long-tailed datasets.
BLT maintains the accuracy of head classes while increasing the accuracy of tail classes on ImageNet-LT and Places-LT. Table 1 and Table 2 show the results of image classification on ImageNet-LT and Places-LT datasets [24], respectively. These Tables report results of methods that were only trained from scratch. Every row in both Tables present the results of different state-of-the-art approaches or baselines that deal with long-tailed datasets. The results in Table 1 of Lifted Loss [21], Range Loss [22], FSLwF [46], and OLTR [24] come from those reported by Liu *et al.* [24]. We reproduce the results with publicly available code for the remaining baselines. The columns in both Tables show the top-1 accuracy for many-shot, medium-shot and few-shot classes. The right-most column shows the overall top-1 accuracy. We can observe that the results of the baseline model trained without any technique to address the bias in long-tailed datasets shows that the head-classes (Many column) achieve higher accuracy than classes with fewer training examples; compare

Table 1. Top-1 classification accuracy on ImageNet-LT. BLT maintains high many-shot accuracy, improves the accuracy of few-shot classes, and keeps the overall accuracy high. We show the highest accuracy in **bold** and the second highest in **blue**.

Methods	Many	Medium	Few	Overall
Plain model	**52.4**	23.1	4.5	31.6
Plain model + sampling	40.6	31.5	14.7	32.5
Lifted Loss [21]	35.8	30.4	17.9	30.8
Range Loss [22]	35.8	30.3	17.6	30.7
FSLwF [46]	40.9	22.1	15.0	28.4
OLTR [24]	43.2	**35.1**	**18.5**	**35.6**
OLTR [24] (Repr.)	39.5	32.5	18.4	33.1
Focal Loss [20]	37.8	31.2	15.0	31.4
CB [18]	29.7	24.7	17.4	25.6
CB Focal Loss [18]	28.0	23.6	17.4	24.4
BLT (Ours)	**44.4**	**33.5**	**25.5**	**36.6**

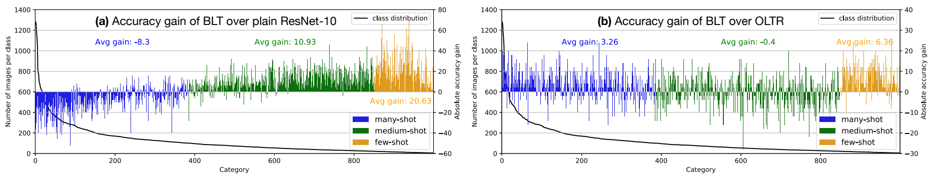

Fig. 4. Accuracy gains for every class on the ImageNet-LT dataset of BLT w.r.t. the plain ResNet-10 model and OLTR [24]. We see in **(a)** that BLT has average gains on medium- and few-shot classes of 10.93% and 20.63%, respectively. We can observe in **(b)** that BLT achieved 3.26% and 6.36% average classification gains on many- and few-shot classes, respectively.

with Medium and Few columns. When adding a sampling balancer method in order to select few-shot examples more often, the performance of tail classes (see Few column) improves. We can observe that our proposed solution increases the accuracy of the few-shot categories while maintaining a competitive accuracy compared to the baselines on the many-shot and medium-shot classes. Please see supplemental material for additional results that include variants of BLT.

BLT maintains the accuracy of head classes high while lifting the accuracy of tail classes on iNaturalist 2018. Table 2 shows the classification results on the naturally long-tailed dataset iNaturalist [61]. All methods use ResNet-34 as the backbone. Although many few-shot classes only have two images, our solution increased the accuracy of tail classes (see Few column). In particular, BLT increases the overall accuracy and keeps the performance of many- and medium-shot classes high. The difference in behavior of all methods between ImageNet-LT, Places-LT, and iNaturalist can be attributed to the "longer tail" of iNaturalist. The number of few-shot classes in iNaturalist is about 63% of all classes, compared to 21% for Places-LT, and 15% for ImageNet-LT. Moreover, many few-shot classes only have two images for training in iNaturalist dataset

Table 2. Top-1 classification accuracy on Places-LT and iNaturalist 2018. BLT maintains high many-shot accuracy, while it improves the few-shot and overall accuracy. We show in **bold** and **blue** the highest and the second highest accuracy, respectively.

	Places-LT				iNaturalist 2018			
Methods	Many	Medium	Few	Overall	Many	Medium	Few	Overall
Plain model	**37.8**	13.0	0.8	19.3	**70.6**	53.0	40.4	46.8
Plain model + sampling	27.8	25.3	7.0	22.4	48.8	**53.4**	47.1	49.0
OLTR [24] (Repr.)	29.1	**26.0**	8.3	**23.4**	44.8	**53.7**	**52.1**	**51.8**
Focal Loss [20]	27.6	25.5	7.0	22.3	28.6	39.0	36.9	36.6
CB [18]	20.5	19.0	12.6	18.2	16.6	25.4	29.1	26.8
CB Focal Loss [18]	18.6	17.7	**12.8**	17.0	14.0	22.1	27.2	24.5
BLT (Ours)	**31.0**	**27.4**	**14.1**	**25.9**	**53.7**	52.5	**49.9**	**51.0**

while ImageNet-LT and Places-LT have at least five. Thus, iNaturalist presents a more challenging scenario for the baselines because few-shot classes dominate.

Accuracy gains on ImageNet-LT. Figs. 4(a,b) show the accuracy boost for many-shot (head classes), medium-shot and few-shot (tail) classes w.r.t. to the plain ResNet-10 model and OLTR [24]. We can see that BLT achieved average gains in accuracy for medium- and few-shot classes by 10.93% and 20.63%, respectively. The performance drop of head (many-shot) classes occurs because the baseline model has a strong bias due to the imbalance in the dataset. In Fig. 4(b) we observe that BLT achieves 3.26% and 6.36% average gains respectively on many- and few-shot classes w.r.t. OLTR. The accuracy gains on tail classes of BLT over OLTR are consistent; only a few tail classes declined (see yellow bars).

Performance as a function of network depth on ImageNet-LT. Fig. 5(a-b) demonstrates that BLT increases the overall top-1 accuracy compared to the plain model with a balancer oversampling tail-classes for all tested backbones (see Fig. 5(a)). It also improves the accuracy on few-shot classes by a significant margin (see Fig. 5(b)). We used architectures with different depths and complexity (in FLOPS) such as EfficientNet-b0 [62], ResNet-28, and ResNet-152 [1].

Influence of dynamic image generation. Because a network changes the topology of its feature space every batch, we study the effect of generating new tail-class images at different training stages (*e.g.*, at every batch or one time) and using them for training. To do so, we trained BLT excluding augmentations from scratch on ImageNet-LT and computed its confusion matrix $\mathcal{C}'$. We tested two augmentation strategies. The first is BLT static targeted: we generated images using BLT strategy using $\mathcal{C}'$. The second is BLT static random: we generated images using gradient ascent techniques and randomly selected confusing classes for tail categories. In both cases, we used the generated images to train BLT replacing its per-batch image generation. Fig. 5(c) shows that BLT with per-batch operation increases accuracy by 5% w.r.t. the described methods earlier.

Classification error reduction on tail classes. Since BLT generates hard examples by forcing the classifier to learn a more robust decision function between each tail class and its most confusing categories, we computed the average classification error (confusion) as a function of the most mistaken classes for tail

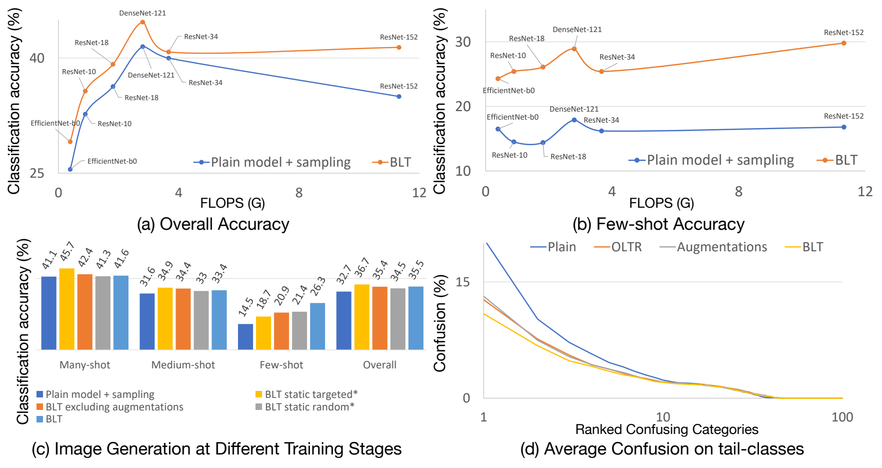

Fig. 5. Top-1 classification accuracy vs FLOPS for BLT and plain model with sample balancing across different architectures **(a-b)**. BLT preserves high overall accuracy for all backbones **(a)** while significantly increasing the performance for few-shot classes **(b)**. Influence of generating images at different training stages **(c)**. Two approaches of generating images statically (*) cannot increase the few-shot accuracy above the level of BLT excluding augmentations, while dynamic image generation (BLT) increases the performance by 5.4%. Overall, we see a favorable trade-off as a 7.6% increase in few-shot accuracy leads to a modest 1.2% drop in overall accuracy. Average classification error (confusion) on tail classes as a function of the ranked misclassified categories **(d)**. Because BLT uses hard examples to force the CNN-based classifier learn a more robust decision function for tail classes, it reduces the errors on the most confusing classes.

categories. Fig. 5(d) shows that BLT reduces the confusion against the most frequently mistaken categories without increasing the error for less confusing classes. Although augmentations and OLTR also decrease the error of tail classes on their most confusing categories, Fig. 5(d) demonstrates that BLT is the most effective approach, thereby increasing the performance on tail classes.

5 Conclusion

We presented BLT, a data augmentation technique that compensates the imbalance of long-tailed classes by generating hard examples via gradient ascent techniques [29–31] from existing training tail-class examples. It generates hard examples for tail classes via gradient ascent at every batch using information from the latest confusion matrix. BLT circumvents the use of dedicated generative models (*e.g.*, GANs [26, 27] or VAEs [28]), which increase computational overhead and require sophisticated training procedures. These hard examples force the CNN-based classifier to produce a more robust decision function yielding an accuracy increase for tail classes while maintaining the performance on head classes. BLT is a novel, efficient, and effective approach. The experiments on synthetically and organic long-tailed datasets as well as across different architectures show that BLT improves learning from long-tailed datasets.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2016)
2. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2017)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012)
4. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518** (2015) 529
5. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations. (2015)
6. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2015)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2009)
8. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. (2007)
9. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proc. of the European Conference on Computer Vision, Springer (2014)
10. Nene, S.A., Nayar, S.K., Murase, H., et al.: Columbia object image library. (1996)
11. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2009)
12. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115** (2015) 211–252
13. Salakhutdinov, R., Torralba, A., Tenenbaum, J.: Learning to share visual appearance for multiclass object detection. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2011)
14. Van Horn, G., Perona, P.: The devil is in the tails: Fine-grained classification in the wild. arXiv preprint arXiv:1709.01450 (2017)
15. Wang, Y.X., Hebert, M.: Learning from small sample sets by combining unsupervised meta-training with cnns. In: Proc. of the Advances in Neural Information Processing Systems. (2016)
16. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In: Proc. of the Advances in Neural Information Processing Systems. (2017)
17. Zhu, X., Anguelov, D., Ramanan, D.: Capturing long-tail distributions of object subcategories. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2014)
18. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2019)
19. Dong, Q., Gong, S., Zhu, X.: Class rectification hard mining for imbalanced deep learning. In: Proc. of the IEEE Intl. Conference on Computer Vision. (2017)

20. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proc. of the IEEE Intl. Conference on Computer Vision. (2017)
21. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2016)
22. Zhang, X., Fang, Z., Wen, Y., Li, Z., Qiao, Y.: Range loss for deep face recognition with long-tailed training data. In: Proc. of the IEEE Intl. Conference on Computer Vision. (2017)
23. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on knowledge and data engineering **21** (2009) 1263–1284
24. 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)
25. Yin, X., Yu, X., Sohn, K., Liu, X., Chandraker, M.: Feature transfer learning for face recognition with under-represented data. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2019)
26. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: Proc. of the International Conference on Machine Learning. (2017)
27. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proc. of the Advances in Neural Information Processing Systems. (2014)
28. van den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: Proc. of the Advances in Neural Information Processing Systems. (2017)
29. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. University of Montreal **1341** (2009) 1
30. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2015)
31. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: Workshop at International Conference on Learning Representations. (2014)
32. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2016)
33. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2017)
34. Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., Yang, G.: Provably robust deep learning via adversarially trained smoothed classifiers. In: Proc. of the Advances in Neural Information Processing Systems. (2019)
35. Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E.: Toward open set recognition. IEEE transactions on pattern analysis and machine intelligence **35** (2012) 1757–1772
36. Zhong, Y., Deng, W., Wang, M., Hu, J., Peng, J., Tao, X., Huang, Y.: Unequal-training for deep face recognition with long-tailed noisy data. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2019)
37. Huang, C., Li, Y., Change Loy, C., Tang, X.: Learning deep representation for imbalanced classification. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2016)

38. Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., Song, Y.: Metagan: An adversarial approach to few-shot learning. In: Proc. of the Advances in Neural Information Processing Systems. (2018)
39. Jang, Y., Zhao, T., Hong, S., Lee, H.: Adversarial defense via learning to generate diverse attacks. In: Proc. of the IEEE Intl. Conference on Computer Vision. (2019)
40. Zhang, J., Zhao, C., Ni, B., Xu, M., Yang, X.: Variational few-shot learning. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 1685–1694
41. Mullick, S.S., Datta, S., Das, S.: Generative adversarial minority oversampling. In: The IEEE International Conference on Computer Vision (ICCV). (2019)
42. Peng, X., Tang, Z., Yang, F., Feris, R.S., Metaxas, D.: Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2018)
43. Hoffman, J., Gupta, S., Darrell, T.: Learning with side information through modality hallucination. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2016)
44. Zhang, H., Zhang, J., Koniusz, P.: Few-shot learning via saliency-guided hallucination of samples. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2019)
45. Hariharan, B., Girshick, R.: Low-shot visual recognition by shrinking and hallucinating features. In: Proc. of the IEEE Intl. Conference on Computer Vision. (2017)
46. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2018)
47. Pahde, F., Nabi, M., Klein, T., Jahnichen, P.: Discriminative hallucination for multi-modal few-shot learning. In: Proc. of the IEEE International Conference on Image Processing. (2018)
48. Wang, Y., Zhou, L., Qiao, Y.: Temporal hallucinating for action recognition with few still images. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition. (2018)
49. Wang, Y.X., Girshick, R., Hebert, M., Hariharan, B.: Low-shot learning from imaginary data. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2018)
50. Chen, Z., Fu, Y., Wang, Y.X., Ma, L., Liu, W., Hebert, M.: Image deformation meta-networks for one-shot learning. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2019)
51. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Proc. of the Intl. Conference on Learning Representations. (2015)
52. Chen, H., Zhang, H., Boning, D., Hsieh, C.J.: Robust decision trees against adversarial examples. arXiv preprint arXiv:1902.10660 (2019)
53. Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., Madry, A.: Adversarial examples are not bugs, they are features. In: Proc. of the Advances in Neural Information Processing Systems. (2019)
54. Liu, A., Liu, X., Zhang, C., Yu, H., Liu, Q., He, J.: Training robust deep neural networks via adversarial noise propagation. arXiv preprint arXiv:1909.09034 (2019)
55. Lopes, R.G., Yin, D., Poole, B., Gilmer, J., Cubuk, E.D.: Improving robustness without sacrificing accuracy with patch gaussian augmentation. arXiv preprint arXiv:1906.02611 (2019)
56. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)

57. Rozsa, A., Rudd, E.M., Boult, T.E.: Adversarial diversity and hard positive generation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops. (2016)
58. Rumelhart, D.E., Hinton, G.E., Williams, R.J., et al.: Learning representations by back-propagating errors. Cognitive modeling **5** (1988) 1
59. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
60. Liu, W., Wen, Y., Yu, Z., Yang, M.: Large-margin softmax loss for convolutional neural networks. In: Proc. of the Intl. Conference on Machine Learning. (2016)
61. Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. (2018)
62. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)