BAYESIAN EMBEDDINGS FOR LONG-TAILED DATASETS

Anonymous authors
Paper under double-blind review

ABSTRACT

The statistics of the real visual world presents a long-tailed distribution: a few classes have significantly more training instances than the remaining classes in a dataset. This is because the real visual world has a few classes that are common while others are rare. Unfortunately, the performance of a convolutional neural network is typically unsatisfactory when trained using a long-tailed dataset. To alleviate this issue, we propose a method that discriminatively learns an embedding in which a simple Bayesian classifier can balance the class-priors to generalize well for rare classes. To this end, the proposed approach uses a Gaussian mixture model to factor out class-likelihoods and class-priors in a long-tailed dataset. The proposed method is simple and easy-to-implement in existing deep learning frameworks. Experiments on publicly available datasets show that the proposed approach improves the performance on classes with few training instances, while maintaining a comparable performance to the state-of-the-art on classes with abundant training examples.

1 INTRODUCTION

Deep convolutional neural networks (CNN) have achieved impressive results in large-scale visual recognition tasks (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015; Szegedy et al., 2015; Mnih et al., 2015; 2013; He et al., 2016; Huang et al., 2017). However, despite the significant impact in visual perception, the vast majority of these advancements learn from artificially balanced large-scale datasets that are not representative of the real visual world (Nene et al., 1996; Griffin et al., 2007; Deng et al., 2009; Quattoni & Torralba, 2009; Lin et al., 2014; Russakovsky et al., 2015). The statistics of the real visual world follow a long-tailed distribution (Zhu et al., 2014; 2016; Van Horn & Perona, 2017; Salakhutdinov et al., 2011; Wang & Hebert, 2016; Wang et al., 2017). This means that a few classes are predominant in the world while others are rare. Consequently, representative real-world datasets have a few classes with significantly more training instances than the remaining classes in the set; see Fig. 1(a) for an illustration of a long-tailed dataset. We refer to classes with abundant training instances as classes in the head, and unrepresented classes as classes in the tail.

As Van Horn & Perona (2017) note, the main motivation for visual recognition is to understand and learn from the real visual world. Thus, while the state-of-the-art can challenge humans in visual recognition tasks, it misses a mechanism that effectively learns from long-tailed datasets. As Van Horn & Perona (2017) found, training models using long-tailed datasets often leads to unsatisfying performance. This is because classifiers tend to generalize well for classes in the head, but lack generalization for classes in the tail.

To alleviate this issue, learned classifiers need to generalize for classes in the tail while maintaining a good performance for all the classes. Recent efforts that aim to learn from long-tailed datasets consider penalities in the optimization-learning problem (Huang et al., 2016), sampling-based methods (He & Garcia, 2009), and transfer-learning algorithms (Wang & Hebert, 2016; Wang et al., 2017). In contrast with these solutions, the proposed method aims to learn an embedding in which the distribution of the real visual world allows a simple Bayesian classifier to predict robustly given a long-tailed dataset.

Long-tailed datasets have class-prior statistics that heavily skew towards classes in the head. This skew can bias classifiers towards classes in the head, and consequently can reduce generalization for classes in the tail. To remove this skew, we appeal to Bayesian classifiers that can explicitly

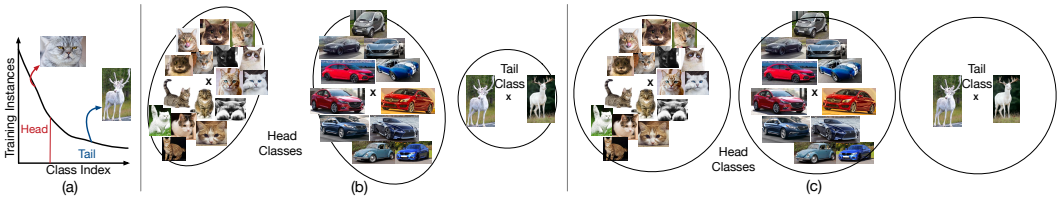

Figure 1: **(a)** The real visual world yields long-tailed datasets. Classes in the head are common (*e.g.*, cats) while classes in the tail are rare (*e.g.*, white reindeers). **(b)** The proposed approach builds a generative (Bayesian) classifier over a learned embedding to compute class-posterior probabilities. In an empirical Bayesian framework, posteriors are computed through class likelihoods and priors fit to the data (*e.g.*, sample means, variances, and counts assuming Gaussian Mixture Models). We introduce an end-to-end pipeline for jointly learning embeddings and Bayesian models built upon them. **(c)** Bayesian models are particularly well-suited for long-tailed datasets because class priors and likelihoods can be fixed to be uniform and isotropic, ensuring that the learned representation is balanced across the head and tail.

factor out the likelihood and prior when computing posteriors over class labels. Thus, the main goal of this work is to learn a feature embedding in which class prior statistics do not affect/skew class likelihoods. The proposed approach uses a simple Gaussian mixture model (GMM) to describe the statistics of a long-tailed dataset. This is because it enables a clean factorization of the class-likelihoods and class-priors. Moreover, it easily fits within an empirical Bayesian classification framework, because a GMM enables the computation of closed-form maximum likelihood estimation (MLE) of class-specific means, covariance matrices, and priors. We show that such closed-form estimates can be integrated into existing deep learning optimizers without much effort. By fixing the covariance matrices of all the classes to be the identity and the priors over each class to be uniform, we can explicitly enforce that both rare classes in the tail and dominant classes in the head have equal weight for Bayesian classification. In simple terms: *we learn a discriminative embedding of training data such that Bayesian classifiers with balanced priors produce accurate class posteriors*. As a point of clarity, the proposed approach does *not* learn an embedding in the traditional Bayesian sense, which might define a prior distribution over embeddings that is then combined with training data to produce a posterior embedding. Rather, it learns a *single* embedding that is discriminatively trained to produce accurate features for Bayesian classifiers. See Fig. 1 for an illustration about the proposed approach.

A GMM not only is useful for learning an embedding using a long-tailed dataset, but also provides flexibility at the evaluation stage. This is because it enables the measurement of generalization for classes in the tail by simply setting equal class-prior probabilities. In addition, it enables the possibility of giving more importance to the most frequent classes by adjusting their respective class-prior probabilities.

In sum, the proposed approach aims to learn an embedding in which a GMM enables a Bayesian classifier to generalize well for classes in the tail by balancing out class-priors. The proposed method is simple, easy-to-train using deep learning frameworks, and increases classification performance for classes in the tail. The experiments on publicly available datasets show that this approach tends to perform better on classes in the tail than the competing methods, while performing comparable to the state-of-the-art on classes with abundant training instances.

2 Related Work

The main challenges for learning models using long-tailed datasets comprise learning parameters that generalize from a few-shots and avoiding classifier bias. While the proposed approach aims to tackle these two problems simultaneously, methods that tackle each of these problems independently are still relevant. As such, this section not only covers prior work on learning using imbalanced datasets, but also covers relevant solutions for few-shot learning. Given that the proposed approach is based on a GMM model, this section also covers recent approaches that use class-centroid representations for incremental learning and for improving discriminative properties.

2.1 LEARNING FROM LONG-TAILED DATASETS

Simple techniques that deal with imbalanced datasets use random sampling to artificially create a more balanced training set (He & Garcia, 2009). For instance, random oversampling effectively "repeats" training instances from the classes in the tail, while random undersampling "removes" instances from the classes with abundant training instances. Thus, these techniques address imbalanced datasets by means of artificially balancing the training set. An alternative approach to deal with long-tailed datasets use transfer learning techniques. Wang et al. (2017) proposed MetaModelNet, a meta-learning algorithm that learns the evolution of parameters when gradually including more training samples. MetaModelNet improves the performance of CNN models since it transfers the parameter-evolution knowledge from data-rich classes to categories in the tail. Rather than artificially modifying the training set or use transfer learning, the proposed approach aims to learn an embedding that allows classifiers to generalize when learning from a long-tailed dataset. Consequently, the proposed method can complement sampling or transfer-learning-based methods.

2.2 FEW-SHOT LEARNING AND CLASS-CENTROID-BASED REPRESENTATIONS

Recent approaches in this category aim to learn good parameters from a few training instances (Snell et al., 2017; Hariharan & Girshick, 2017). A recent approach that considers an imbalanced dataset to tackle few-shot learning is the work by Hariharan & Girshick (2017). Their proposed approach learns a feature embedding from the classes with the most samples in the dataset. Then, the approach "hallucinates" samples for classes with a few training instances in the learned embedding. While this work learns from an imabalanced dataset, it considers a different setting that that of the proposed approach. The work by Hariharan & Girshick (2017) assumes that classes with few instances are added incrementally. The proposed approach differs in this regard, since the introduced method aims to learn the embedding using the entire long-tailed dataset, generalize, and avoid any bias towards the classes with abundant training instances.

To achieve generalization given a few shots in an incremental learning context, Rebuffi et al. (2017) proposed iCaRL, a deep-learning-based incremental classifier. Similarly to the proposed approach, iCaRL represents each class using a single centroid in an embedding learned using a regular CNN model. However, instead of using the learned softmax classifier, it uses a nearest-class-mean (Mensink et al., 2013) classifier. Unlike iCaRL that uses features from a learned CNN-softmax model, the proposed approach learns an embedding using a generative model. It is worth noting that the proposed approach uses a GMM, which by default includes a nearest-class-mean classifier as part of the learning problem. The use of class-centroids in learning representations is also useful to improve discriminative properties. Wen et al. (2016) proposed a loss that aims to minimize intra-class variation in CNN-softmax models. Unlike the center-loss approach, the proposed method minimizes the intra-class variation automatically by finding the GMM parameters in the learned embedding. Different from the center-loss that requires a mechanism to estimate the class-centroids, the proposed approach uses back-propagation to learn the GMM parameters. A recent approach that aims to generalize by using class-centroid representations are the Prototypical Networks (proto-nets) by Snell et al. (2017). Proto-nets estimate the class centroids from a slice of a mini-batch-like subset of the training set. Then, they evaluate the loss from the complementary slice of the mini-batch-like subset, and update the feature encoder weights. The proposed approach has two main differences with proto-nets. First, the proposed approach is based on generative models describing the statistics of an imbalanced dataset, rather than learning an embedding tailored for a nearest-class-mean classifier that requires specific parameter-update rules. Second, the proposed approach uses regular batching mechanisms and updates parameters using back-propagation. Thus, in constrast with proto-nets, the proposed approach avoids modifying components in the deep learning frameworks.

3 BAYESIAN EMBEDDINGS

The goal of this work is to learn an embedding that allows a simple Bayesian classifier to robustly operate given a long-tailed training dataset. Specifically, this work aims to learn an encoder $f_w(\cdot)$, parameterized by its set of weights w, that produces a good representation for Bayesian classification given a long-tailed dataset.

In order to learn the aforementioned encoder, the proposed approach requires a model that describes the distribution of the data. Let $x = f_w(I)$ be the encoded feature for image I and y be its corresponding class label. Thus, the distribution of the training set can be described with the following joint probability:

$$\begin{aligned} p(x, y) &= p(x \mid y)p(y) \\ &= p(f_w(I) \mid y;\ \theta_y)\pi_y \\ &= p(x, y;\ w, \theta) \end{aligned} \tag{1}$$

where $p(f_w(I) \mid y;\ \theta_y)$ represents the *likelihood* of observing the feature vector x as part of class y, and π_y is the *prior* probabilities for class y. The likelihood is a function with parameters θ_y (*e.g.*, parameters of a multivariate Gaussian) that describes the distribution of the feature vectors in the embedding. Thus, the joint probability of the data $p(x, y;\ w, \theta)$ is a function with parameters composed by the the encoder w parameters, and the Bayesian parameters θ which include the likelihood parameters θ_y, and priors π_y. In practice, the likelihood parameters proves most crucial as it is not sensitive to class priors, which can be misleading in the long-tailed setting (as discussed in Section 3.1).

Given the above joint probability model, the posterior probability for class y given a feature vector x can be computed using Bayes rule as follows:

$$p(y \mid x;\ w, \theta) = \frac{p(f_w(I) \mid y;\ \theta_y)\pi_y}{\sum_k p(f_w(I) \mid k;\ \theta_k)\pi_k}, \tag{2}$$

where θ is a concatenation of the likelihood parameters and priors of all the classes. Thus, the class posterior probability is a function that depends on the encoder parameters w and the Bayesian parameters θ.

The overall objective of this work is to jointly learn the weights w of the feature encoder and the Bayesian parameters θ to guarantee a good classification performance. Given a training dataset of images and label pairs $\mathcal{D} = \{(x_i, y_i)\}$, we propose to learn parameters by maximizing the Bayesian class-posterior probability of the true class labels:

$$\underset{w}{\text{minimize}} \sum_i -\log p(y_i \mid x_i;\ w, \theta) \qquad \text{subject to} \qquad \theta = \text{MLE}(\mathcal{D}), \tag{3}$$

where MLE is a function that computes the closed form maximum likelihood estimates of the parameters of our Bayesian model, a procedure commonly known as *Empirical Bayes* (Bishop, 2006).

To use existing solvers for learning deep networks, we reformulate the problem shown in Eq. (3) as an unconstrained optimization by using a Lagrangian penalty (Boyd & Vandenberghe, 2004) that penalizes solutions which violate the constraint:

$$\underset{w,\theta}{\text{minimize}} \quad \sum_i -\log p(y_i \mid x_i;\ w, \theta) + \lambda \|\theta - \text{MLE}(\mathcal{D})\|^2 \quad , \tag{4}$$

where $\lambda \geq 0$. In this formulation, the optimization explicitly searches over the feature encoder parameters w and the Bayesian parameters θ so as to maximize class posterior probabilities. The last term penalizes deviations of the θ parameters from their MLE estimates, effectively acting as a regularizer.

3.1 GMM Embeddings for Heavily Tailed Datasets

GMMs: The likelihood models are crucial to determine the parameters that allows the proposed approach to learn the feature encoder given a long-tailed dataset. We propose to use a multivariate Gaussian probability density function as the likelihood model. Given this likelihood model, the proposed approach implicitly uses a Gaussian mixture model to represent the distribution of the training set. Using a multivariate Gaussian brings benefits to the proposed formulation. This is because its parameters (the centroid μ, covariance matrix Σ, and prior π) have an intuitive meaning and closed-form-maximum-likelihood estimators. Interestingly, as discussed by van den Oord & Schrauwen (2014) and Patel et al. (2016), a mixture of multivariate Gaussians can be used to theoretically motivate the success of deep learning.

Balancing: The use of a GMM not only brings simplicity into the formulation, but also allows the feature encoder to generalize better for classes in the tail. The generalization aspect of a GMM

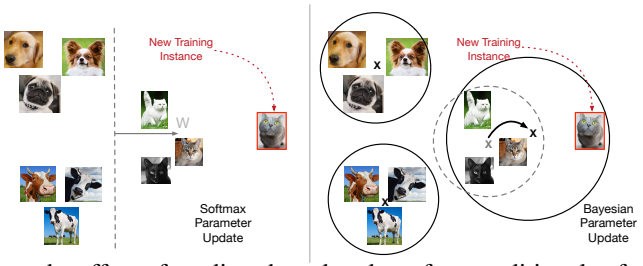

Figure 2: We compare the effect of gradient-based updates for a traditional softmax classifier versus our Bayesian embedding model. Recall that our approach learns an embedding for which Bayesian classifiers produce accurate class posteriors. During softmax training, an "easy" example of a class will tend to not generate a strong gradient update, and so is not useful for learning (**left**). This might be considered paradoxical: when children learn a new concept (for say, a never-before-seen animal), an easy or "protypical" example might be most informative for learning. On the other hand, in our framework, an easy example of a class will change its centroid, generating a strong signal for updating our learned representation (**right**).

model comes from the fact that a class is described with a single centroid. The benefit of this class representation is that estimating the centroid with a handful of examples is simple and produces a good estimate. Perhaps more importantly, a GMM allows us to access specific parameters that control the probabilistic "footprint" of each class in the embedded space. We can set these parameters to ensure balanced footprints by fixing the covariance matrices to be the identity and the class priors to be uniform - see Fig. 1-(c). The remaining parameters to be estimated are then the class means $\mu = (\mu_1, \ldots, \mu_{n_c})$. Given this setting and considering that deep-learning frameworks use mini batches, the unconstrained problem shown in Eq. (4) becomes:

$$\underset{w,\mu}{\text{minimize}} \quad \frac{1}{m}\sum_{i=1}^{m} -\log\left(\frac{\exp\left(-\frac{1}{2}\|f_w(I_i)-\mu_{y_i}\|^2\right)}{\sum_k \exp\left(-\frac{1}{2}\|f_w(I_i)-\mu_k\|^2\right)}\right) + \lambda \sum_{j\in\mathcal{M}} \|\mu_j - \frac{1}{n_j}\sum_{i':y_{i'}=j} f_w(I_{i'})\|^2 \quad , \tag{5}$$

where y_i is the true class label/index for the i-th data point, m is the batch size, $\mathcal{M}$ is the set of class indices in the batch, n_j is the number of samples of the j-th class in the batch, and i' is the index running over instances in the batch.

3.2 Discussion

Other probabilistic models: Our analysis and experiments focus on Gaussian Mixture Models, but the general learning problem from Eq. (4) holds for other probabilistic models. For example, deep embeddings can be learned for rectified (nonnegative) or binary features (Agrawal et al., 2014; Erin Liong et al., 2015). For such embeddings, likelihood models based on rectified Gaussians or multivariate Bernoulli distributions may be more appropriate Socci et al. (1998); Teugels (1990). Such models do not appear to have closed form maximum likelihood estimates, and so may be challenging to formulate precisely as a constrained optimization problem.

Relationship to softmax: The GMM-based formulation has a direct relationship with softmax classifiers. This relationship can be obtained by expanding the squared distance terms in the class-posterior probability, yielding the following:

$$\begin{aligned} p\left(y_i \mid f_w(I_i);\ w,\mu\right) &= \frac{\exp\left(-\frac{1}{2}\|f_w(I_i)-\mu_j\|^2\right)}{\sum_k \exp\left(-\frac{1}{2}\|f_w(I_i)-\mu_k\|^2\right)} \\ &= \frac{\exp\left(\mu_j^T f_w(I_i) - \frac{1}{2}\left(\|f_w(I_i)\|^2 + \|\mu_j\|^2\right)\right)}{\sum_k \exp\left(\mu_k^T f_w(I_i) - \frac{1}{2}\left(\|f_w(I_i)\|^2 + \|\mu_k\|^2\right)\right)}, \\ &= \frac{\exp\left(v_j^T f_w(I_i) + b_j\right)}{\sum_k \exp\left(v_k^T f_w(I_i) + b_k\right)} \end{aligned} \tag{6}$$

where $v_j = \mu_j$ and $b_j = -\frac{1}{2}\|v_j\|^2$, since $-\frac{1}{2}\|f_w(I_i)\|^2$ is a common term between the numerator and denominator. This relationship thus indicates that the proposed approach fits linear classifiers

with restricted biases. This relationship is useful for an easy implementation in many deep learning frameworks. This is because this approach can be implemented using a dense layer without the bias terms. In addition, this relationship shows that the proposed approach requires fewer parameters-to-learn in comparison with classical CNN-softmax models. An more intuitive comparison between GMMs and softmax classifiers can be made with respect to to their parameter updates. Intuitively, during softmax training, an "easy" example of a class will not generate a model update. In some sense, this might be considered paradoxical. When children learn a new concept (for say, a never-before-seen animal), they tend to be presented with an easy or "protypical" example. On the other hand, an easy example of a class will change its centroid, generating a signal for learning - see Fig. 2.

4 EXPERIMENTS

This section presents a series of experiments evaluating the learned embedding computed using the proposed method and long-tailed datasets. Since the goal of the experiments is to evaluate the feature encoder, all the experiments trained all the baselines or competing methods and the proposed one from scratch. An additional goal of the experiments is to show that the proposed approach can be adapted to any CNN architecture. For this reason, the experiments also used legacy and recent CNN architectures.

Datasets: One evaluation aspect of the experiments is to measure the performance on small- and medium-scale datasets. The experiments included MNIST (LeCun et al., 1998) and CIFAR 10 (Krizhevsky & Hinton, 2009) as the small-scale datasets (each with ten classes); and CIFAR 100 (Krizhevsky & Hinton, 2009) and Tiny ImageNet [1] as the medium-scale datasets (with hundred and two hundred classes, respectively). The balanced MNIST dataset contains 60,000 and 10,000 28x28 training and testing images depicting hand-written digits, respectively. The CIFAR 10 dataset contains 50,000 and 10,000 32x32 training and testing images, respectively. The CIFAR 100 dataset contains 500 and 100 32x32 training and testing images per class, respectively. Lastly, Tiny ImageNet has 500 and 50 64x64 training and testing images for every class, respectively. However, the experiments used a 224x224 image instead. See Sec. 4.1 for details on how the experiments processed these datasets to evaluate classifiers using long-tailed datasets.

Baselines: The experiments included recent approaches that deal with imbalanced datasets. These approaches include iCaRL (Rebuffi et al., 2017), center loss (Wen et al., 2016), and a plain softmax classifier. The experiments also consider a variation of iCaRL. This variation does not use normalized feature vectors as originally proposed by Rebuffi et al. (2017). While prototypical networks (Snell et al., 2017) are similar to the proposed approach, they require a balanced dataset with a few training instances for every class. Since prototypical networks do not assume a long-tailed dataset, these experiments did not include it as a competing method. The experiment also considered a method that uses a full GMM model (*i.e.*, full covariance, means, and priors) of a softmax representation of the training set. As discussed in Sec. 2, MetaModelNet (Wang et al., 2017) deals with long-tailed datasets by operating at the classifier-parameter level, since it is a meta-learning algorithm. Thus, MetaModelNet does not learn an embedding, and consequently complements the proposed method.

Implementation Details: All the experiments were implemented on top of TensorFlow Models (TFM)[2]. This open-source project implements various legacy architectures, and several pre-processing imaging techniques (*e.g.*, random translations and shifts). The experiments used the following CNN architectures: LeNet (LeCun et al., 1998) for MNIST; CifarNet (Krizhevsky & Hinton, 2009) for CIFAR 10; AllCNN (Springenberg et al., 2014) for CIFAR 100; and VGG 16 (Simonyan & Zisserman, 2015) for Tiny ImageNet. We implemented center loss (Wen et al., 2016) and verified correctness using a balanced setting. We implemented the proposed approach in TFM using a fully connected layer with a restrictive bias. This is possible thanks to the relationship with linear classifiers discussed in Sec. 3.2. The regularizer was implemented using plain Tensorflow operations and was added as a regularizer function for the fully connected layer with restrictive bias. We will release the code upon publication. The hyperparameters for center-loss and the proposed approach were estimated using a validation set for every dataset. See Sec. A in the Appendix for the specific parameters.

[1] https://tiny-imagenet.herokuapp.com/

[2] https://github.com/tensorflow/models

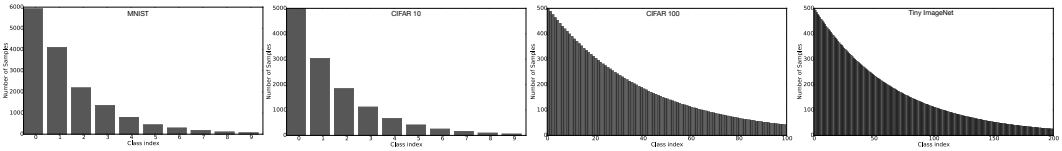

Figure 3: Histogram of the number of training instances per class in the long-tailed datasets. From left to right, the datasets are MNIST, CIFAR 10, CIFAR 100, and Tiny ImageNet.

4.1 Evaluation for Long-Tailed Datasets

The main motivation of this work is to learn from a realistic dataset representing the statistics of the real visual world. Recall that realistic datasets are long-tailed since the visual world has a few predominant classes while others are rare. As such, the performance evaluation of the visual recognition system in this setting needs to be discussed, since common evaluation methods may not be adequate given this context.

Intuitively, since the visual world yields long-tailed datasets, then the test set should ideally be long-tailed as well. While this rationale is logical given the statistics of the world, it has a main drawback: a simple classifier that is biased towards classes in the head is likely to perform well using a long-tailed testing set. While this setting may reflect a good performance for the common classes in practice, achieving a good performance for classes in the tail is still desirable in real practical applications. For instance, consider a self-driving car: the vehicle may easily detect common objects or events, *e.g.*, pedestrians walking on the sidewalk. However, children playing soccer on the street is a rare event that can occur in the real world, and it is important to evaluate autonomous systems on such rare but crucial events. Thus, although rare events are infrequent, classifiers still need to account for them. Consequently, average accuracy on a long-tailed dataset is not an adequate measure of performance across rare classes.

An alternative approach using long-tailed testing sets is to evaluate *per-class* accuracy. This explicitly weights all classes – both in the head and tail – equally. However, this has the drawback that performance estimates of rare classes in the tail have high variability and can be unreliable. In the autonomous vehicle scenario above, we might encounter very few (or even no) examples of children playing street soccer in any finite testset. This means that performance estimates fort tail classes can be unreliable.

We propose an evaluation approach that addresses the bias towards the head and intra-class variation of classes in the tail. The proposed evaluation protocol requires a training and an evaluation procedure. The training setting includes several training trials that used different versions of long-tailed training sets. The evaluation procedure uses a balanced dataset. The use of a balanced testing set addresses the issue of classifiers that are biased towards the head since the class-priors are uniform and both classes in the head and tail contribute to the performance measure. Training a classifier using different long-tailed sets accounts for intra-class variation for classes in the tail. Consequently, aggregates of performance from these different trials account for the intra-class variation noise from classes in the tail. The experiments report a per-class accuracy average, the average class-accuracy, and their standard deviations over three different trails.

Because the considered datasets are balanced, the experiments "long-tailed" these datasets following the procedure proposed by Wang et al. (2017). For every class, the procedure computed the number of samples to draw from the balanced set using an exponential distribution. Thus, as the class index grows, the number of training instances decreases according to the exponential distribution. Given the computed number of samples to draw, the procedure randomly selects these instances from the balanced set to generate a training long-tailed dataset version. Fig. 3 shows a visualization of the training-instance distribution of the resultant long-tailed datasets. The experiments used the balanced testing sets because the goal is to measure generalization and overall performance.

4.2 Performance on Long-Tailed Datasets

The goal of this experiment is to evaluate the learned embedding using a long-tailed dataset. To do so, the experiments used the long-tailed datasets described above. In particular, the target is to measure any classification improvement for classes in the tail with respect to the regular softmax classifier. Since most of the baselines rely on class-centroids to classify, the experiments use a

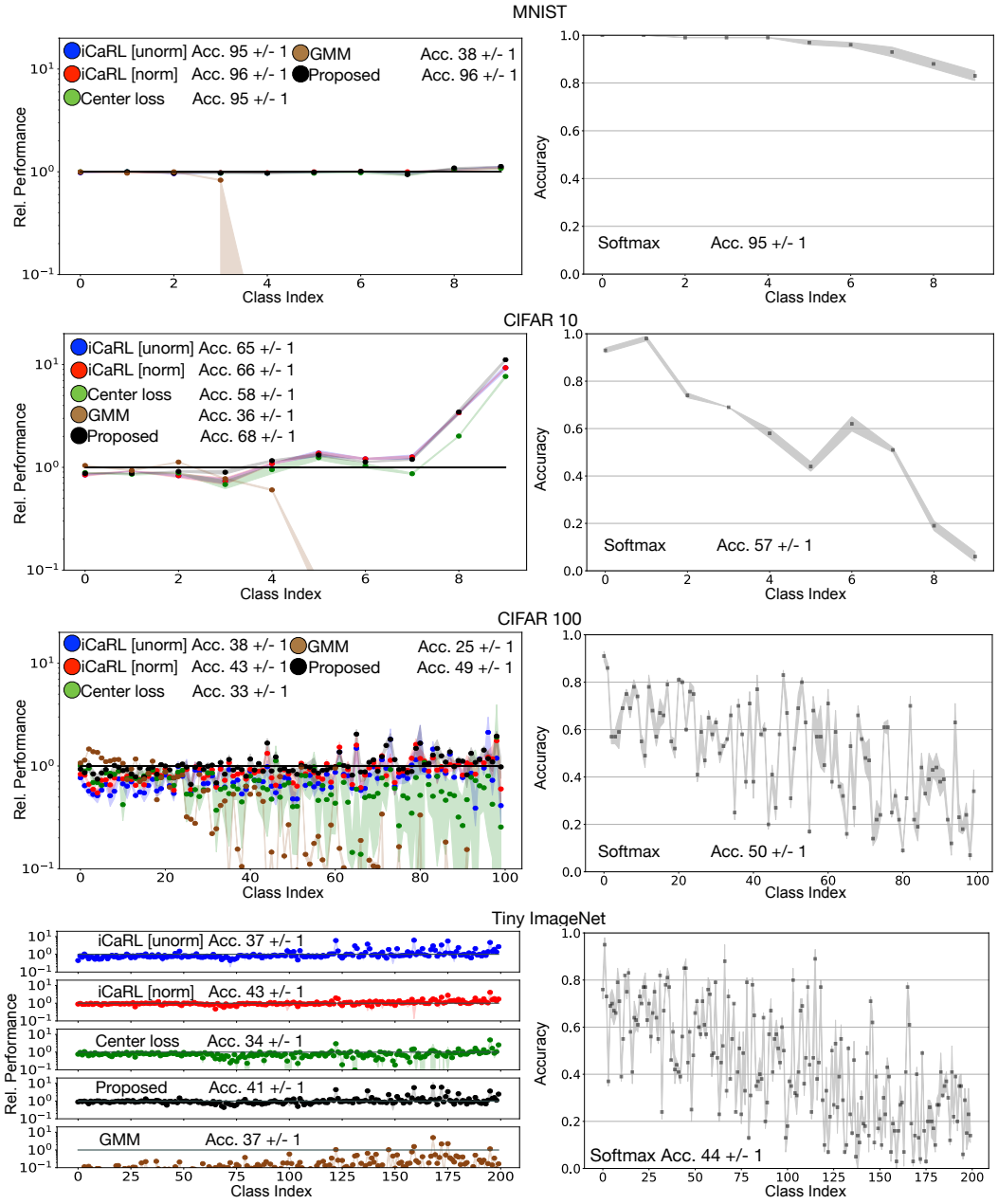

Figure 4: **Left column:** Relative classification accuracy gain of the competing and proposed methods with respect to a softmax classifier using long-tailed datasets. Overall, the proposed method tends to achieve a comparable accuracy to that of a softmax classifier while delivering an increase for tail classes. **Right column:** The performance of a softmax classifier. The performance for classes in the head is higher than that of the classes in the tail.

nearest-class-mean (Mensink et al., 2013) classifier. Thus, the experiments computed the deep-features for the training and testing sets after learning the feature encoder $f_w(\cdot)$; a deep feature is the output of $f_w(\cdot)$ which is the input tensor to the classifier or softmax layer. Then, the experiments computed a class centroid using the long-tailed training set for every method.

To measure the classification improvements, the experiments trained all the methods with three different long-tailed datasets and used the balanced testing sets. Then, the experiment computed an average class-accuracy from the three trials for every baseline. To measure the *relative performance* with respect to a softmax classifier, the experiment computed the ratio between the average class-accuracy of a competing method (*i.e.*, iCaRL (Rebuffi et al., 2017), center loss (Wen et al., 2016), and

Table 1: Average *relative performance* with respect to a softmax classifier for classes in the head (H column) and tail (T column); the relative performance is the ratio between the class accuracies of a competing method and a softmax classifier. The proposed method increases the performance on classes in the tail while maintaining a comparable performance to that of a softmax classifier for classes in the head. Bold numbers indicate the highest performance per dataset in each row.

Datasets	iCarl [Unorm]		iCarl [Norm]		Center loss		Proposed	
	H	T	H	T	H	T	H	T
MNIST	0.98	1.00	**0.99**	**1.01**	**0.99**	1.00	**0.99**	**1.01**
CIFAR 10	0.95	2.22	0.95	2.22	0.92	1.8	**0.98**	**2.44**
CIFAR 100	0.72	0.80	0.83	0.90	0.71	0.66	**0.96**	**1.04**
Tiny ImageNet	0.86	1.07	0.95	1.06	0.76	0.8	**0.97**	**1.12**

the proposed method) and the average class-accuracy of a softmax classifier; the softmax classifier is the reference because it tends to bias towards classes in the head (Wang et al., 2017).

Fig. 4 shows the results of this experiment on small- and medium-scale datasets in the first two rows and last two rows, respectively. The Figure shows the relative performance for all the classes in a dataset in the left column, the class accuracy of a softmax classifier on the right column, and the average class-accuracy of the compared methods in the labels. All the plots in the left column show a black solid line indicating the performance of a softmax classifier. Thus, a decrement falls below the line while an increment raises above the line.

The results on the MNIST dataset (first row) show that most of the competing methods perform comparable to that of a softmax classifier. However, the GMM method underperforms for classes in the tail. For this dataset, a softmax classifier does not present a significant bias towards classes in the head. Thus, these results indicate that the competing and proposed methods, with the exception of the GMM, operate well given a dataset with minor visual variations (*e.g.*, illumination variations, pose, occlusion, among others). Consequently, these results effectively work as a sanity check of the proposed and competing methods (*i.e.*, iCaRl and Centerloss). The results on CIFAR 10 (second row) show that the proposed approach and competing methods tend to perform comparable to a softmax classifier for classes in the head (*i.e.*, the first three classes). In addition, the results show that the GMM also underperforms for classes in the tail. However, the proposed approach and competing methods tend to increase relative performance for classes in the tail. In this dataset, the proposed approach achieved an average class-accuracy of 68%, which is the highest compared to all the methods.

The plots in the third row show the results on CIFAR 100. The plot in the left shows that the proposed method achieves a comparable performance with respect to a softmax classifier for classes in the head (*i.e.*, the first twenty classes). On the other hand, the competing methods have a larger decrease in accuracy for classes in the head. The GMM in this dataset again underperforms for classes in the tail. The plot in the left shows that the proposed approach tends to increase the relative performance for classes in the tail. Overall, they tend to be larger than those of the competing methods and a softmax classifier. Lastly, the plot at the bottom shows the results on Tiny ImageNet. The plot in the left shows similar observations. The proposed approach maintains a comparable performance with respect to a softmax classifier for classes in the head. However, it delivers an increase in relative performance for classes in the tail. The GMM approach suffers for classes in the tail because the covariance estimates are poor due to the lack of data.

To highlight the previous observations, Table 1 shows a break down of the average relative performance for classes in the head (H column) and in the tail (T column); this experiment excludes the GMM approach. To measure an average relative performance for classes in the head, the experiments used a weighted average of the relative performance considering all the classes. The average used the fraction of instances for a given class in the training set as its corresponding weight. Specifically, the weight for the i-th class is $w_i = \frac{n_i}{n}$, where n_i is the number of training instances for the i-th class and n is the total number of training instances in the long-tailed training set. Thus, this average emphasizes the relative performance of classes with abundant training instances while decreasing the contribution of the classes with scarce training data. To compute a weighted average of the relative performance for classes in the tail, the experiment calculated the weight w'_i for the i-th class as follows: $w'_i = \frac{1-w_i}{\sum_i 1-w_i}$. These weights emphasize the relative performance of the classes in the tail while diminishing the relative performance of classes in the head. The results in Table 1

Table 2: Classification performance improvement by using the regularizer in the proposed approach on CIFAR 10. The proposed approach with regularizer achieves a higher classification accuracy than the approach without the regularizer.

Configuration	Class Index										Avg.
	1	2	3	4	5	6	7	8	9	10	Acc.
w/ regularizer	81	85	68	62	67	58	70	61	65	68	**68**
w/o regularizer	72	72	57	63	65	55	67	63	68	58	64

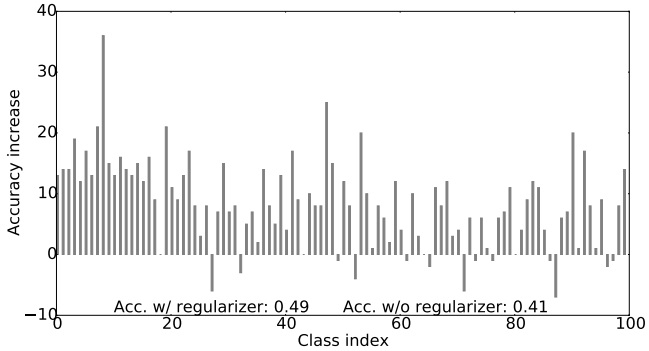

Figure 5: Accuracy increase achieved by using the proposed regularizer on CIFAR 100. Overall, the proposed approach with regularizer tends to increase the accuracy across all classes compared to the proposed approach without one.

show that the proposed method maintains a comparable peformance for classes in the head with respect to a softmax classifier. At the same time, the proposed method consistently improves the performance for classes in the tail.

4.3 Effect of the Centroids Regularizer

The goal of this experiment is to measure the benefits of the regularizer in the proposed method. To do so, this experiment compared the proposed method with a hyperparemeter $\lambda = 0$, leaving only the Bayesian classifier, and the configuration tested in the previous Section. Note that this setting is equivalent to only using a linear classifier with restricted bias, according to the discussion in Sec. 3.2. This experiment only considered CIFAR 10 and 100, and tested performance also considering three different long-tailed training sets for each dataset.

The results of this experiment on CIFAR 10 are shown in Table 2. The table shows the average accuracies per class for the proposed method using a regularizer with $\lambda = 0.001$ (top row), and without a regularizer (bottom row). The last column of the table shows the average classification performance. This table shows that the regularizer overall improves classification performance. This is expected since the regularizer aims to retain the centroid-parameters that are as close as possible to the batch-sample-mean centroids.

Fig. 5 presents the results of this experiment on CIFAR 100. The plot shows the accuracy gains obtained by comparing the accuracies of the proposed method using a regularizer with $\lambda = 0.0001$ across all classes. Also, the plot shows the average accuracies for both methods. The plot indicates that the regularizer consistently provides an accuracy increase across classes. Thus, the regularizer is an important component that overall improves the classification performance.

5 Conclusion

This work introduced a method that improves the classification performance for classes in the tail. The proposed approach is based on a Gaussian mixture model that allows a Bayesian classifier to represent the distribution of a long-tailed dataset and to compute the class-prediction probabilities. The experiments on publicly available dataset show that the proposed approach tends to increase the classification accuracy for classes in the tail while maintaining a comparable accuracy to that of a softmax classifier for classes in the head. In addition, this work introduced an evaluation method for methods that tackle the learning of concepts from a long-tailed dataset. Finally, this work demon-

strated that class-centroid approaches overall tend to generalize well for classes in the tail while maintaining a comparable performance to that of a softmax classifiers for classes in the head.

REFERENCES

Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European conference on computer vision*, pp. 329–344. Springer, 2014.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2475–2483, 2015.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset, 2007.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, October 2017.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Trans. on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proc. of the IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755. Springer, 2014.

Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Sameer A. Nene, Shree K. Nayar, and Hiroshi Murase. Columbia object image library (coil-20. Technical report, 1996.

Ankit B Patel, Minh Tan Nguyen, and Richard Baraniuk. A probabilistic framework for deep learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2558–2566. 2016.

Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *Proc of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision (IJCV)*, 115(3): 211–252, December 2015.

Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2015.

J. Snell, K. Swersky, and R. S. Zemel. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems (NIPS)*. 2017.

Nicholas D Socci, Daniel D Lee, and H Sebastian Seung. The rectified gaussian distribution. In *Advances in neural information processing systems*, pp. 350–356, 1998.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Jozef L Teugels. Some representations of the multivariate bernoulli and binomial distributions. *Journal of multivariate analysis*, 32(2):256–268, 1990.

Aaron van den Oord and Benjamin Schrauwen. Factoring variations in natural images with deep gaussian mixture models. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3518–3526. 2014.

Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

Yu-Xiong Wang and Martial Hebert. Learning from Small Sample Sets by Combining Unsupervised Meta-Training with CNNs. In *Advances in Neural Information Processing Systems (NIPS)*. 2016.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning To Model the Tail. In *Advances in Neural Information Processing Systems (NIPS)*. 2017.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 499–515. Springer, 2016.

Xiangxin Zhu, Dragomir Anguelov, and Deva Ramanan. Capturing long-tail distributions of object subcategories. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Xiangxin Zhu, Carl Vondrick, Charless C. Fowlkes, and Deva Ramanan. Do we need more training data? *International Journal of Computer Vision (IJCV)*, 119(1):76–92, Aug 2016.

A HYPERPARAMETERS AND IMPLEMENTATION DETAILS

In order to guarantee similar training conditions for all the methods, the experiments used the same framework parameters (*e.g.*, number of steps, learning rate, decay factors, among others) for all the considered methods.

All the tested methods used an Adam optimizer (Kingma & Ba, 2014) and a batch size of 32. The experiments used a learning rate of 0.01 and 0.1 for the small- and medium-scale datasets, respectively. The experiments used the default exponential-learning-rate decay, weight decay, and drop-out parameters provided in TensorFlow Models. The hyperparameters used for center-loss are 0.5 for the centroids learning rate and a scale value of 0.001 for MNIST and CIFAR 10, 0.0001 for CIFAR 100 and Tiny ImageNet the proposed method. The hyperparameter for the proposed approach was set to 0.001 for MNIST and CIFAR 10, and 0.0001 for CIFAR 100 and Tiny ImageNet.