Delving into Semantic Scale Imbalance

Motivation

Model bias triggered by long-tailed data has been widely studied. However, measure based on the number of samples cannot explicate **three phenomena simultaneously**:

Given enough data, the classification performance gain is marginal with additional samples.

Classification performance decays precipitously as the number of training samples decreases when there is insufficient data.

Model trained on sample-balanced datasets still has different biases for different classes.

Our key contributions

We propose the novel idea of leveraging the volume of manifold to measure the semantic scale. It is also innovative to find that the semantic scale has the marginal effect, and that the semantic scale of the dataset is highly consistent with model performance in terms of trends.

We introduce and define semantic scale imbalance, aiming to measure the degree of class imbalance by semantic scale rather than sample number, which reveals a new perspective for the study of class imbalance problem. Experiments show that semantic scale imbalance is prevalent in the dataset and can more accurately reflect model bias that affects model performance.

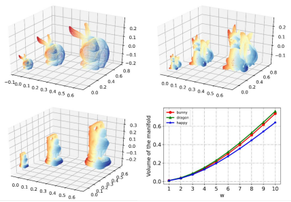
Semantic-scale-balanced learning is proposed to mitigate model bias, which includes a general loss improvement scheme and a dynamic re-weighting training framework that overcomes the challenge of calculating semantic scales in real-time during iterations. Comprehensive experiments demonstrate that semantic-scale-balanced learning is applicable to a variety of datasets and achieves significant performance gains on multiple vision tasks.

Quantification of Semantic Scale

The volume of the space spanned by $Z = [Z_1, Z_2, \ldots, Z_m]$ can be can be written as:

$$Vol(Z) = \frac{1}{2}\log_2 \det(I + \frac{d}{m}(Z - Z_{mean})(Z - Z_{mean})^T)$$

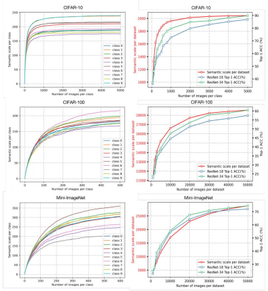
The semantic scales of multiple Stanford point cloud manifolds with different sizes are calculated and plotted in figure. Let the center point of bunny be C(bunny).



As the object manifold is scaled up, the calculated volume then increases slowly and monotonically, indicating that our method can accurately measure the relative size of the manifold volume and is numerically stable, an advantage that will help mitigate the effects of noisy samples.

Marginal Effect of Semantic Scale

The marginal effect describes that the feature richness will gradually saturate as the number of samples increases. Figure illustrates that as the number of samples increases, the semantic scale S' measured by sample volume gradually saturates, which indicates that the quantitative measurement of semantic scale is reasonable.



We train ResNet-18 and ResNet-34 on each of the training sets in Table, and the sum of the semantic scales for all classes and the corresponding top-1 accuracy are shown in Figure. **We are pleasantly surprised to find that when the semantic scale increases rapidly, the model performance improves swiftly with it, and when the semantic scale becomes saturated, the improvement is small.**

Semantic Scale Imbalance on Long-Tailed Data

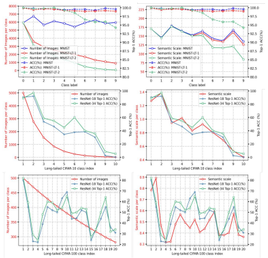
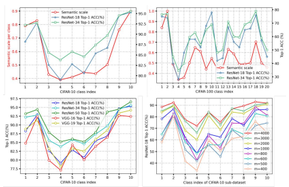


Figure indicate that models on certain classes with fewer samples outperform those on classes with more samples, and that the semantic scale S reflects model bias more accurately.

Previous studies have roughly attributed model bias to the imbalance in the number of samples. The experimental results in the first row of Figure show that even though the number of MNIST-LT-1 is similar to that of MNIST-LT-2, the class-wise accuracy on MNIST-LT-1 is closer to that on MNIST, just as their semantic scales are also more similar.

Semantic Scale Imbalance on Non-Long-Tailed Data



In summary, semantic scale imbalance can represent model bias more generally and appropriately.

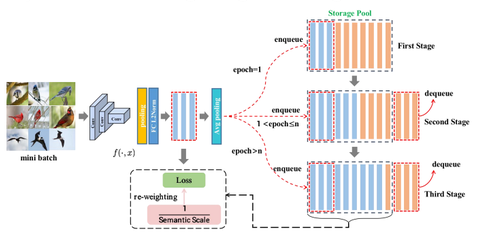
Figure demonstrates the model bias not only on long-tailed data but also on sample-balanced data. Usually, the classes with smaller semantic scales have lower accuracies. Depending on the size of the semantic scale, it can make it possible for the weaker and dominant classes to be well differentiated. It should be noted that the weaker classes are not random, and experiments in Figure show that the models always perform worse on the same classes.

Dynamic Semantic-Scale-Balanced Learning

Dynamic Semantic-Scale-Balanced Loss

$$DSB-NSM(z, y_i) = -\frac{1}{S_i}\log(\frac{\exp(w_{y_i}^T z/\sigma)}{\sum_{j=1}^{C}\exp(w_j^T z/\sigma)}) \quad DSB-ST(z, y_i) = -\frac{1}{S_i}\log(\frac{\exp(\lambda(D'_{z,y_i} - \delta))}{\exp(\lambda(D'_{z,y_i} - \delta)) + \sum_{j\neq y}\exp(\lambda D'_{i,j})})$$

Dynamic Re-Weighting Training Framewor



Algorithm 2 Dynamic Re-Weighting Training Framework

Input: Training set $D = \{(x_i, y_i)\}_{i=1}^{M}$, total training epochs N, defined encoder is model()
Initialize the queue Q
for $epoch = 1$ to N **do**
  for $iteration = 0$ to $\frac{M}{batchsize}$ **do**
    Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^{batchsize}$ from D
    Calculate features $F = [f_1, \ldots, f_i, \ldots, f_{batchsize}]$, $f_i = \text{model}(x_i), i = 1, \ldots, batchsize$
    Store F and label y into Q: $enqueue(Q, [F, y])$
    if $epoch < n$ **then**
      if $epoch > 1$ **then**
        Dequeue the oldest mini-batch features from Q
      end if
      Calculate loss $L = SoftTripleloss(F, y)$
    else
      Dequeue the oldest mini-batch features from Q
      Calculate loss $L = DSB.SoftTripleloss(F, y)$
    end if
    Perform back propagation: L.backward()
    optimizer.step()
  end for
end for

In the first stage, all the features and labels generated by the 1st epoch are stored in Q, but they cannot be used directly to calculate semantic scales due to the large drift of historical features from current features in the early stage.

The second stage corresponds to epoch 2 to epoch n. At each iteration, the oldest mini-batch features and labels in Q are removed and those generated by the current iteration are stored. The goal is to continuously update the features in Q until the feature drift is small enough. We set n to 5 in our experiments, and the original loss function is used in the first two stages.

The third stage corresponds to epoch>n. At each iteration, the semantic scales are calculated using the features in Q after updating Q, and the original loss is re-weighted.

Experiments

Table 2: Top-1 Acc(%) on ImageNet-LT and iNaturalist2018. We use ResNext-50 (61) on ImageNet-LT and ResNet-50 (14) on iNaturalist2018 as the network backbone for all methods. And we conduct model training with the SGD optimizer based on batch size 256 (for ImageNet-LT) / 512 (for iNaturalist), momentum 0.9, weight decay factor 0.0005, and learning rate 0.1 (linear LR decay).

Methods	ImageNet-LT(ResNeXt50)				iNaturalist 2018(ResNet50)			
	Head	Middle	Tail	Overall	Head	Middle	Tail	Overall
BBN (70)	43.3	45.9	43.7	44.7	49.4	70.8	65.3	66.3
PaCo (9)	63.2	51.6	39.2	54.4	69.5	72.3	73.1	72.3
DIVE (15)	64.1	50.4	31.5	53.1	70.6	70	67.6	69.1
CE	65.9	37.5	7.70	44.4	67.2	63.0	56.2	61.7
CB-CE (10)	39.6	32.7	16.8	33.2	53.4	54.8	53.2	54.0
DSB-CE	**67.3**	**42.5**	**21.4**(+13.7)	**49.2**(+4.8)	**68.5**	**63.4**	**62.7**(+6.5)	**64.3**(+2.6)
Focal (10)	67.0	41.0	13.1	47.2	-	-	-	61.1
CB-Focal (10)	-	-	-	-	-	-	-	61.2
DSB-Focal	**68.1**	**44.2**	**23.7**(+10.6)	**50.6**(+3.4)	**70.6**	**62.8**	**58.4**	**63.5**(+2.4)
LDAM (3)	60.0	49.2	31.9	51.1	-	-	-	64.6
DSB-LDAM	**60.7**	**50.5**	**33.4**(+1.5)	**52.3**(+1.2)	**69.4**	**66.5**	**61.9**	**65.7**(+1.1)
BS (38)	47.8	37.7	24.9	40.4	60.1	51.4	46.7	53.2
DSB-BS	**48.5**	**39.3**	**28.7**(+3.8)	**42.9**(+2.5)	**61.4**	**52.8**	**49.4**(+2.7)	**55.1**(+1.9)
LADE (16)	62.3	49.3	31.2	51.9	-	-	-	69.7
DSB-LADE	**62.6**	**50.4**	**33.6**(+2.4)	**53.2**(+1.3)	**72.3**	**70.7**	**65.8**	**70.5**(+0.8)
MBJ (30)	61.6	48.4	39.0	52.1	-	-	-	70.0
DSB+MBJ	**63.2**	**49.6**	**40.7**(+1.7)	**53.3**(+1.2)	**73.6**	**70.2**	**66.2**	**70.9**(+0.9)
RIDE (56)	67.9	52.3	36.0	56.1	70.9	72.4	73.1	72.6
MBJ+RIDE (30)	68.4	54.1	37.7	57.7	-	-	-	73.0
DSB+RIDE	**68.6**	**54.5**	**38.5**(+2.5)	**58.2**(+2.1)	**70.7**	**74.0**	**74.2**(+1.1)	**73.4**(+0.8)

Table 3: Comparison on ImageNet and CIFAR-100. On ImageNet, we use random clipping, mixup (67), and cutmix (66) to augment the training data, and all models are optimized by Adam with batch size of 512, learning rate of 0.05, momentum of 0.9, and weight decay factor of 0.0005. On CIFAR-100, we set the batch size to 64 and augment the training data using random clipping, mixup, and cutmix. An Adam optimizer with learning rate of 0.1 (linear decay), momentum of 0.9, and weight decay factor of 0.005 is used to train all networks.

Methods	ImageNet Top-1 Acc(%)			CIFAR-100 Top-1 Acc(%)		
	CE	DSB-CE	Δ	CE	DSB-CE	Δ
VGG16 (42)	71.6	72.9	+1.3	71.9	73.4	+1.5
BN-Inception (47)	73.5	74.4	+0.9	74.1	75.2	+1.1
ResNet-18	70.1	71.2	+1.1	75.6	76.9	+1.3
ResNet-34	73.5	74.3	+0.8	76.8	77.9	+1.1
ResNet-50	76.0	76.8	+0.8	77.4	78.3	+0.9
DenseNet-201 (19)	77.2	78.1	+0.9	78.5	79.7	+1.2
SE-ResNet-50 (18)	77.6	78.4	+0.8	78.6	79.3	+0.7
ResNeXt-101 (61)	78.8	79.7	+0.9	77.8	78.8	+1.0

Table 4: Results on CUB-2011 and Cars196. We evaluate the model performance with Recall@K (36) and Normalized Mutual Information (NMI) (41).

	Dataset	CUB-2011			Cars196		
	Metric	R@1	R@2	NMI	R@1	R@2	NMI
dim 64	NormSoftmax	57.8	70.0	65.3	76.8	85.6	66.7
	DSB-NSM	**59.2**(+1.4)	**70.7**(+0.7)	**66.5** (+1.2)	**77.9**(+1.1)	**86.4**(+0.8)	**67.8**(+1.1)
	SoftTriple	60.1	71.9	66.2	78.6	86.6	67.0
	DSB-ST	**61.3** (+1.2)	**72.7**(+0.8)	**67.3**(+1.1)	**79.8**(+1.2)	**87.5**(+0.9)	**68.3** (+1.3)
dim 512	Circle	66.7	77.4	-	83.4	89.8	-
	NormSoftmax	63.9	75.5	68.3	83.2	89.5	69.7
	DSB-NSM	**65.1**(+1.2)	**76.3**(+0.8)	**69.2**(+0.9)	**84.0**(+0.8)	**90.2**(+0.7)	**70.9**(+1.2)
	SoftTriple	65.4	76.4	69.3	84.5	90.7	70.1
	DSB-ST	**66.4**(+1.0)	**77.0**(+0.6)	**70.6**(+1.3)	**85.6**(+1.1)	**91.3**(+0.6)	**71.1**(+1.0)

Table 5: Results on CIFAR-100-LT. The imbalance factor of a dataset is defined as the value of the number of training samples in the largest class divided by that in the smallest class.

	Dataset	CIFAR-100-LT								
	Imbalance factor	10			50			200		
	Metric	R@1	R@2	NMI	R@1	R@2	NMI	R@1	R@2	NMI
dim 64	NormSoftmax	54.6	65.2	62.4	49.6	60.5	58.0	43.4	54.5	52.9
	CB-NSM	55.7	66.1	63.3	50.5	61.1	58.7	45.5	55.3	53.8
	DSB-NSM	**56.3**	**66.7**	**63.5**	**51.3**	**61.4**	**59.1**	**46.0**	**56.1**	**54.4**
	SoftTriple	56.6	67.6	63.9	49.5	61.0	58.3	46.6	57.8	55.4
	CB-ST	58.1	68.4	65.1	51.1	62.8	59.4	48.2	59.5	56.6
	DSB-ST	**58.8**	**69.0**	**65.8**	**51.5**	**62.5**	**59.7**	**49.3**	**60.6**	**57.3**

Table 10: Comparison of DSB-ST and SoftTriple in terms of memory consumption and training speed. The speed is measured by the average number of iterations per second. The additional video memory consumption due to our method is almost negligible.

Dataset	GPU Memory		Training speed	
	SoftTriple	DSB-ST	SoftTriple	DSB-ST
ImageNet-LT	24.29 GB	24.75 GB	6.01 it/s	5.56 it/s
iNaturalist2018	45.13 GB	46.88 GB	3.32 it/s	3.03 it/s
Cars196	3491 MB	4097 MB	20.21 it/s	18.95 it/s
CUB-2011	3225 MB	3647 MB	21.95 it/s	18.58 it/s
Mini-ImageNet	3491 MB	3713 MB	23.34 it/s	20.36 it/s

See more in the paper!