ActiveDC: Distribution Calibration for Active Finetuning

Supplementary Material

In this supplementary material, we augment our study with crucial experiments focused on the impact of hyperparameter α on refining category-centered information, detailed extensively in Sec. 1. Moreover, Sec. 2 furnishes additional experimental details regarding the pretraining-finetuning paradigm and the distributional calibration module. Additionally, the details of the code implementation of Tukey's Ladder power transformation are described in Sec. 3. Furthermore, detailed algorithmic descriptions concerning the filtering process are available in Sec. 4.

1. Hyperparameter Tuning

The hyperparameter α in Eq. 8 (in Submission paper), whose value determines the parameter β in Eq. 7 (in Submission paper). The parameter β represents the degree to which the true labeled data acquired through data selection contributes to the central information of the sample. The values of α were set based on experimental results as shown in Tabs. 1 to 3. We design Eq. 8 (in Submission paper) to provide a good trade-off between clustered center information and labeled category center information. This design prioritizes the utilization of cluster center information in scenarios with limited labeled data, while emphasizing the utilization of labeled category center information in situations where sufficient labeled data is available. The optimization process concerning the hyperparameters α and ξ presents a dual-stage challenge, although empirical findings indicate limited fluctuations in their influence on performance within a favorable range of values. Our primary focus revolves around assessing the efficacy of the method rather than hyperparameter optimization. Consequently, while the selected values may not represent the absolute optimal solution, they are expected to be in close proximity to it. To speed up the hyperparameter selection process, an alternative approach involving logistic regression on the feature data, instead of model finetuning on the original data, stands as a viable consideration.

Table 1. Tuning α on CIFAR10.

hyperparameter α	0.5	0.6	**0.7**	0.8	0.9
CIFAR10 (0.2%)	72.9	**73.1**	**73.1**	72.6	71.5
CIFAR10 (0.5%)	86.3	86.8	**87.3**	86.7	85.8
CIFAR10 (1%)	87.8	88.1	**88.9**	88.9	88.4

Table 2. Tuning α on CIFAR100.

hyperparameter α	0.05	0.06	**0.07**	0.08	0.09
CIFAR100 (2%)	54.3	54.5	**54.6**	53.8	53.2
CIFAR100 (5%)	71.5	71.8	**71.9**	71.2	70.9
CIFAR100 (10%)	73.1	73.8	**74.3**	74.0	73.8

Table 3. Tuning α on ImageNet.

hyperparameter α	0.10	0.12	**0.14**	0.16	0.18
ImageNet (1%)	56.3	**56.4**	56.3	55.1	54.5
ImageNet (2%)	59.8	60.0	**60.1**	59.4	58.5
ImageNet (5%)	67.3	67.8	**68.2**	68.0	67.8

2. Implementation Details

In unsupervised pretraining phase, we adopt DeiT-Small architecture[6] pretrained within the DINO framework[1], a well-established and effective choice on the ImageNet-1k dataset[5]. In the data selection phase, the parameters denoted as $\theta_{\mathcal{S}}$ are optimized employing the Adam optimizer[3] with a learning rate of $1\mathrm{e}{-3}$ until convergence. In the distribution calibration phase, the unsupervised clustering method and similarity retrieval mechanisms employed primarily rely on the FAISS (Facebook AI Similarity Search) library. Specifically, we employ the GPU-accelerated variant of K-Means for clustering and cosine similarity as the similarity metric. We run the clustering 10 times and keep the best centroids. We use a full traversal search approach in FAISS for efficient retrieval, which is also GPU-accelerated. In the supervised finetuning phase, the DeiT-Small model follows the established protocol outlined in reference [1]. For the CIFAR10 and CIFAR100 datasets, supervised finetuning involves the use of the SGD optimizer with a learning rate of $1\mathrm{e}{-3}$, weight decay of $1\mathrm{e}{-4}$, and a momentum value of 0.9. This procedure runs on a selected subset of the training data for 1000 epochs, and utilizes a learning rate schedule characterized by cosine decay. For the ImageNet dataset, finetuning is performed using the same SGD optimizer configuration as employed for CIFAR datasets. To ensure convergence, the finetuning process spans 1000 epochs when the sampling rate is below 5%, while it requires 300 epochs at a 5% sampling rate. Throughout the finetuning process, the batch size for training samples is set at 256, while the batch size for test samples is configured to 768. Similar to the pretraining phase, the input images are resized to dimensions of 224×224. The implementation of supervised finetuning is based on the official codebase of DeiT.

3. Power Transformation

To make the feature distribution more Gaussian-like, we first transform the features in $\mathcal{F}^u$ using Tukey's Ladder of Powers transformation [2]. This step is a prerequisite for the subsequent generation of features aligned with calibrated statistics conforming to a Gaussian distribution. Based on the experimental results, we set the value of λ to 0.5, as shown in Fig. 5 (in Submission paper). Since the value of λ in Tukey's Ladder Power Transformation cannot handle negative values in some cases, such as λ equals to 0.5, we need to make minor changes in the code implementation. For instance, if the value in features is between -1 and 1, we can start with an offset (*e.g.* $+1$) and then apply the transformation. Also, in order not to change the expected value of the variable, we need to re-scale and offset. The transformation is formulated as:

$$\hat{x} = \sqrt{2(x+1)} - 1 \tag{1}$$

The purpose of applying the transformation has been indicated at the beginning of this section, it is to make the distribution closer to Gaussian distribution, so Box-Cox Transformation, Log-Tukey transformation are also transformations that can be considered.

4. Feature Filtering

The labeled feature pool is denoted as $\mathcal{F}^l_{\mathcal{S}}$. The feature pool consisting of all the features in $\mathbb{R}_y$ is denoted as $\mathcal{F}^u_{\mathcal{G}} = \{\hat{f}^g_y\}_{(\hat{f}^g_y, y)\in\mathbb{R}_y}$. The extended feature pool $\mathcal{F}^u_E$ $(\mathcal{F}^u_{\mathcal{S}} \bigcup \mathcal{F}^u_{\mathcal{G}})$ is also associated with the extended data subset in $\mathcal{P}_{EL}$, with the corresponding distribution over $\mathcal{F}^u_E$ in the feature space denoted as p_{f_E}. The filtering process for generated features is conducted through a tripartite approach: Firstly, features already present within the pool of previously annotated features are excluded. Secondly, features significantly deviating from the established category center are also filtered out. Lastly, any features deemed detrimental to the overall distribution of the annotated feature pool are excluded from consideration.

The Earth Mover's Distance (EMD) metric [4] is employed as a quantitative measure for assessing the dissimilarity between a subset distribution p_{f_E} and the overall distribution p_{f_u}. Its application enables the identification and elimination of generative features that pose detriment to the overall distribution. The EMD between p_{f_u}, p_{f_E} is written as:

$$EMD(p_{f_u}, p_{f_E}) = \inf_{\gamma\in\Pi(p_{f_u}, p_{f_E})} \underset{(f_i, f_{E_j})\sim\gamma}{E} [\left\| f_i - f_{E_j} \right\|_2] \tag{2}$$

where $\Pi(p_{f_u}, p_{f_E})$ is the set of all possible joint distributions whose marginals are p_{f_u} and p_{f_E} . The pseudo-code detailing the feature filtering procedure is accessible in Algorithm 1.

Algorithm 1: Feature Filtering in ActiveDC

input : the labeled feature pool $\mathcal{F}^l_{\mathcal{S}}$
output: the generated feature pool $\mathbb{R}_y$

1 **foreach** $(f_{\mathcal{S}_j}, y_{\mathcal{S}_j})$ **in** $\mathcal{F}^l_{\mathcal{S}}$ **do**
  // a new feature As Per Eq. 9 (in Submission paper)
2 $f^g_y \leftarrow \text{FeatureGene}\left((f_{\mathcal{S}_j}, y_{\mathcal{S}_j})\right)$;
  // search feature per Eq. 10 (in Submission paper)
3 $\hat{f}^g_y \leftarrow \text{SearchFeature}\left(f^g_y\right)$;
  // filter redundant features or features too far offset
4 **if** $\hat{f}^g_y$ **in** $\mathcal{F}^u_E$ **or** $cos(\hat{f}^g_y, \mu_y) < cos(\hat{f}^g_y, \mu_{!y})$ **then**
5 **goto** line 2
  // filter the features harmful to the overall distribution
6 **else if** $\text{EMD}\left(p_{f_E \bigcup \hat{f}^g_y}, p_{f_u}\right) > \text{EMD}\left(p_{f_E}, p_{f_u}\right)$ **then**
7 **goto** line 2
8 **else**
9 $\mathbb{R}_y \leftarrow \mathbb{R}_y \bigcup (\hat{f}^g_y, y)$
10 **end**

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1

[2] Vaibhav Ganatra. Logarithm-transform aided gaussian sampling for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 247–252, 2023. 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[4] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998. 2

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 1

[6] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 1