# Tencent ML-Images: A Large-Scale Multi-Label Image Database for Visual Representation Learning

Baoyuan Wu, Weidong Chen, Yanbo Fan, Yong Zhang, Jinlong Hou, Jie Liu, Junzhou Huang, Wei Liu, Tong Zhang

**Abstract**—In existing visual representation learning tasks, deep convolutional neural networks (CNNs) are often trained on images annotated with single tags, such as ImageNet. However, a single tag cannot describe all important contents of one image, and some useful visual information may be wasted during training. In this work, we propose to train CNNs from images annotated with multiple tags, to enhance the quality of visual representation of the trained CNN model. To this end, we build a large-scale multi-label image database with 18M images and 11K categories, dubbed *Tencent ML-Images*. We efficiently train the ResNet-101 model with multi-label outputs on Tencent ML-Images, taking 90 hours for 60 epochs, based on a large-scale distributed deep learning framework, *i.e.*, TFplus. The good quality of the visual representation of the Tencent ML-Images checkpoint is verified through three transfer learning tasks, including single-label image classification on ImageNet and Caltech-256, object detection on PASCAL VOC 2007, and semantic segmentation on PASCAL VOC 2012. The Tencent ML-Images database, the checkpoints of ResNet-101, and all the training code have been released at *https://github.com/Tencent/tencent-ml-images*. It is expected to promote other vision tasks in the research and industry community.

**Index Terms**—Visual Representation Learning, Multi-label, Image Database.

✦

## 1 INTRODUCTION

This work presents the large-scale visual representation learning on a newly built multi-label image database, dubbed *Tencent ML-Images*. We start from the discussions of the following two questions.

- Why we need large-scale image database? Deep learning had been in a long trough, until 2012 when AlexNet [1] shows surprising results on the single-label image classification task of ILSVRC2012 challenge. The excellent potential of deep neural networks is released through the large-scale image database, *i.e.*, ImageNet-ILSVRC2012 [2]. Besides, the cost of acquiring training data for many visual tasks, such as object detection and semantic segmentation, is very high. Due to the insufficient training data, they need the checkpoint with good visual presentation pre-trained on other large-scale database for other visual tasks (*e.g.*, ImageNet-ILSVRC2012 for single-label image classification) as initialization.
- Why we need multi-label image database? As there are multiple objects in most natural images, the single annotation may miss some useful information and mislead the training of CNNs. For example, two visually similar images including *cow* and *grass* may be annotated as *cow* and *grass*, respectively. The reasonable approach is "telling" the CNN model that these two images contain both *cow* and *grass*.

The above discussions explain why we need the large-scale multi-label image database for visual representation learning with deep neural networks. However, annotating one image with multiple tags are much more time-consuming than annotation with one tag, and it is difficult to control the annotation quality. To the best of our knowledge, the largest public multi-label image database is Open Images [3], which includes about 9 million images and 6 thousand categories, and with about 20% label noises. However, only CNN models with multi-label outputs have been trained on Open Images, while its generalization to other vision tasks like single-label image classification has not been studied in [4]. Recently, Sun et al. [5] fine-tuned the ResNet-101 model pre-trained on JFT-300M (a multi-label image database with 300 million images) on ImageNet, leading to 79.2% top-1 accuracy on the validation set of ImageNet. In contrast, the ResNet-101 model trained on ImageNet-ILSVRC2012 from scratch gives 77.5% top-1 accuracy. However, the training on JFT-300M takes 2 months for 4 epochs, as the training size of JFT-300M is 250 times of the size of ImageNet-ILSVRC2012. Moreover, JFT-300M and its checkpoint are not publicly available.

In this work, we build a new large-scale multi-label image database, dubbed *Tencent ML-Images*. Instead of collecting new images from Google search or Flickr as did in other databases, we collect images from existing image databases, *i.e.*, Open Images [3] and ImageNet [2]. Specifically, we merge their class vocabularies into one unified vocabulary, by remove rare and redundant categories as well as the corresponding images. We further build the semantic hierarchy of the unified vocabulary, according to WordNet [6]. We also derive the class co-occurrence among

All authors are with Tencent AI Lab, Shenzhen, China. B. Wu (email: wubaoyuan1987@gmail.com) and W. Chen (email: powerchen@tencent.com) contribute equally. Y. Fan is the corresponding author, email: fanyanbo0124@gmail.com.

categories, which are then used together with the semantic hierarchy to augment the annotations, based on the original annotations from Open Images and ImageNet. To verify the quality of the built Tencent ML-Images, we conduct the large-scale visual representation learning with deep neural networks, and we choose the popular ResNet-101 V2 architecture. There are two main difficulties in the large-scale learning using multi-label image database, including the severe class imbalance and the long training process. To alleviate the side-effect of class imbalance, we design a novel loss function that simultaneously considers weighted cross entropy, the adaptive loss weight along the training process and the down-sampling of negative training images in each mini-batch. To accelerate the training process, we utilize the large-scale distributed deep learning framework, *i.e.*, TFplus, with Message Passing Interface (MPI) and NVIDIA Collective Communications Library (NCCL) [7]. Consequently, the whole training process takes 90 hours of 60 epochs, using 128 GPUs. And the efficiency of one single GPU is 5X of the efficiency reported in [5]. Furthermore, to verify the quality of visual representation of the ResNet-101 model pre-trained on Tencent ML-Images, we conduct transfer learning to three other vision tasks, including single-label image classification, object detection and semantic segmentation. We compare with the transfer learning using the checkpoints pre-trained on JFT-300M and ImageNet-ILSVRC2012, respectively. The better transfer learning results using the checkpoint pre-trained on Tencent ML-Images demonstrate the good quality of Tencent ML-Images and the trained checkpoint.

The main contributions of this work are four-fold.

- We build a multi-label image database with 18M images and 11K categories, dubbed *Tencent ML-Images*, which is the largest publicly available multi-label image database until now.
- We efficiently train the ResNet-101 model on Tencent ML-Images, utilizing a large-scale distributed deep learning framework. Besides, we design a novel loss function to alleviate the side-effect of the severe class-imbalance in large-scale multi-label database.
- We demonstrate that the good quality of Tencent ML-Images and its pre-trained checkpoint, through the transfer learning to three different vision tasks.
- We release the Tencent ML-Images database, the trained ResNet-101 checkpoints, as well as the complete codes from data preperation, pre-training, fine-tuning to image classification and feature extraction, at the GitHub address *https://github.com/Tencent/tencent-ml-images*. It is expected to promote other vision tasks for the research and industry community.

The organization of this manuscript is as follows. Related work is reviewed in Section 2. The built multi-label image database is introduced in Section 3, including the image source, class vocabulary and semantic hierarchy, as well as the tag augmentation and the statistics. The visual representation learning on Tencent ML-Images is presented in Section 4. Transfer learning to single-label image classification, object detection and semantic segmentation are presented in Section 5, followed by the conclusion in Section 6.

## 2 RELATED WORK

In this section, we review the image databases that are used for visual representation learning. They can be generally partitioned into two categories. One category is the single-label image database, where each image is annotated with only one tag. The other category is the multi-label image database, where each image is annotated with multiple tags.

Widely used single-label image databases include CIFAR-10 [8], Caltech-256 [9], MNIST [10], ImageNet [2], WebVision [11], SUN [12] and Places [13], etc. Before the deep learning era (*i.e.*, before 2012), the scales of most image databases are not very large. CIFAR-10 [8] includes 60K small-sized natural images in 10 categories. Caltech-256 [9] includes 30,607 images, with 256 object categories. MNIST [10] includes 70K images of handwritten digits, from '0' to '9'. In the deep learning era (from 2012), ImageNet [2] is the most popular database. Its first version that was used for ILSVRC 2012 includes 128K images, with 1000 object categories. And now it has been expanded to include 14M images. Many deep learning models (*e.g.*, AlexNet [14], VGG [15] and ResNet [16]) are trained and evaluated on ImageNet to demonstrate their performance, and the checkpoints pre-trained on ImageNet are widely used to help other vision tasks, such as image annotation, object detection, etc. WebVision [11] includes 2.4M images, with the same 1,000 object categories in ImageNet. The main difference between WebVision and ImageNet is that the annotations of WebVision is noisy, while the annotations of ImageNet are accurate. However, the authors of Web-Vision have experimentally demonstrated that the AlexNet trained on sufficient images with noisy labels has comparable or even better performance of visual representation, than those trained on ImageNet. However, the state-of-the-art CNN model, *i.e.*, ResNet, is not evaluated. In addition to above image databases of object categories, there are two popular databases of scene categories, including SUN and Places. SUN [12] includes 108,754 images, with 397 scene semantic categories. Places [13] includes 10M images, with 434 scene semantic categories. However, scene categories describe higher level information than object categories. The visual representation of the deep model trained on scene databases may not be suitable for other vision tasks like object recognition or detection. However, as mentioned in Section 1, the main contents of one image cannot be well described by a single label. Visual representation learning on single-label images will waste useful information of training images, and may bring in confusion to deep models, as two visually similar images could be annotated with two different categories.

There are also many multi-label image databases. Before the deep learning era, most multi-label image databases are used to evaluate multi-label models or image annotation methods. Some widely used databases include Corel 5k [17] (including 4,999 images with 260 object categories), ESP Game [18] (including 20,770 images with 268 categories), IAPRTC-12 [19] (including 19627 images with 291 categories), NUSWIDE [20] (including 270K images with 81 categories), MS COCO [21] (including 330K images with 80 categories), and PASCAL VOC 2007 [22] (including 9,963 images with 2.47 averaged annotated tags per image). How-

ever, they are rarely used to train deep models for visual representation learning. Their scales are not enough to train good parameters of popular deep models, such as VGG or ResNet, as their numbers of images are even smaller than the number of parameters of these deep models. Besides, their small-scale category vocabularies are not diverse enough to train models with good generalization to other vision tasks. In contrast, there are also large-scale multi-label image databases. For example, Open Images [3] includes 9M image, with 6K categories. However, the authors of Open Images have only provided a ResNet-101 model with 5K independent outputs (corresponding to 5K categories) trained on Open Images. Its generalization to single-label image classification has not been studied. JFT-300M is an "internal dataset" in Google, including 300M images, with 18,291 categories, as well as 1.25 averaged annotated tags per image. Sun et al. [5] trained the ResNet-101 model on JFT-300M, and transferred the trained checkpoint to other vision tasks, including single-label image classification on ImageNet, object detection on MS-COCO and PASCAL VOC 2007, semantic segmentation on PASCAL VOC 2012, and human pose estimation on MS-COCO. Specifically, the checkpoint of ResNet-101 pre-trained on JFT-300M is fine-tuned on ImageNet, leading to 79.2% top-1 accuracy on the validation set of ImageNet. In contrast, the ResNet-101 model trained on ImageNet from scratch gives 77.5% top-1 accuracy. This improvement demonstrates that JFT-300M is helpful for learning more generalized visual representation, compared with ImageNet. However, it is notable that the scale of JFT-300M is 250X of ImageNet. Training ResNet-101 on 300M images with 18,291 categories is very costly. As reported in [5], their training process takes 2 months for 4 epochs, using "asynchronous gradient descent training on 50 NVIDIA K80 GPUs and 17 parameter servers"). Moreover, JFT-300M and its checkpoints have not been published. They cannot be utilized by the research community to help other vision tasks. In contrast, the built Tencent ML-Images will be the largest public multi-label image database, and our training based on distributed training framework is much more efficient.

# 3 THE TENCENT ML-IMAGES DATABASE

## 3.1 Image Source and Class Vocabulary

The images and class vocabulary of Tencent ML-Images are collected from ImageNet [2] and Open Images [3]. In the following we introduce the construction of training set, validation set and class vocabulary, respectively.

**URLs and Vocabulary**. Firstly, we extract image URLs from ImageNet-11k[1]. It is a subset of the whole database of ImageNet, collected by MXNet. It originally includes 11,797,630 training images, covering 11,221 categories. However, 1,989 categories out of 11,221 categories are very abstract in visual domain, like *event, summer*. We think the images annotated with such abstract categories will not help the learning of visual representation. Thus, we remove these abstract categories, as well as their corresponding images, with 10,322,935 images of 9,232 categories. Moreover, according to the semantic relationship among categories, we add

1. Downloaded from http://data.mxnet.io/models/imagenet-11k/

another 800 finer-grained categories from the vocabulary of the whole database of ImageNet. For example, if *dog* is included in the above 9,232 categories, we also add *Husky* into the vocabulary of Tencent ML-Images, as well as the corresponding images from ImageNet. Consequently, we obtain 10,756,941 URLs, covering 10,032 categories from ImageNet. We randomly choose 50,000 URLs as validation URLs, while ensuring that the number of selected URLs of each category is not larger than 5. On the other hand, we filter all images of Open Images using a per-category criteria. If one category occurs in less than 650 images, then we remove this category from all these images. And we also remove some abstract categories in visual domain as did above. Besides, as some categories from Open Images are similar to or synonyms of above 10,032 categories, we merge these redundant categories into unique categories. If all tags of one image are removed, then this image is also abandoned. Consequently, 6,902,811 training images and 38,739 validation images are remained, covering 1,134 unique categories. Finally, we merge the selected URLs and categories from ImageNet and Open Images to construct the URLs and vocabulary of Tencent ML-Images, which includes 17,609,752 training and 88,739 validation image URLs, covering 11,166 categories. Then, we download the original images from these URLs.

**Semantic Hierarchy**. We firstly map the categories from both ImageNet and Open Images to the WordIDs in Word-Net. According to the WordIDs, we construct the semantic hierarchy among these 11,166 categories. It include 4 independent trees, of which the root nodes are *thing, matter, physical object* and *atmospheric phenomenon*, respectively. The length of the longest semantic path from root to leaf nodes is 16, and the average length is 7.47.

## 3.2 Tag Augmentation of Images

Note that each image from ImageNet-11K is annotated by a single tag. We propose to augment the tags of these images by utilizing the semantic hierarchy and the co-occurrence among categories. Firstly, according to the semantic hierarchy, all ancestor categories of the original single tag are also annotated as tags of the same image. Secondly, we compute the co-occurrence matrix $CO$ between categories from ImageNet-11k and categories from Open Images. Specifically, we train a ResNet-101 model with 1,134 outputs, based on Open Images. Using this trained model, we predict the labels among these 1,134 categories for the images from ImageNet-11k. If the posterior probability with respect to one category of one image is larger than 95%, then we set this category as the positive label of this image. Then, we compute the co-occurrence matrix as follows: for category $i$ from ImageNet-11k and category $j$ from Open Images, we denote the number of positive images of category $i$ in ImageNet-11k as $n_i$, among which the number of images also annotated as category $j$ is denoted as $n_{i,j}$, then $CO(i,j) = n_{i,j}/n_i \in [0,1]$. If $CO(i,j) > 0.5$ and there is no semantic relationship between category $i$ and $j$ (*i.e.*, there is no path from $i$ to $j$ or reverse in the semantic hierarchy), then we determine that category $i$ and $j$ is a strongly co-occurrent pair of categories. Then, we augment the tags
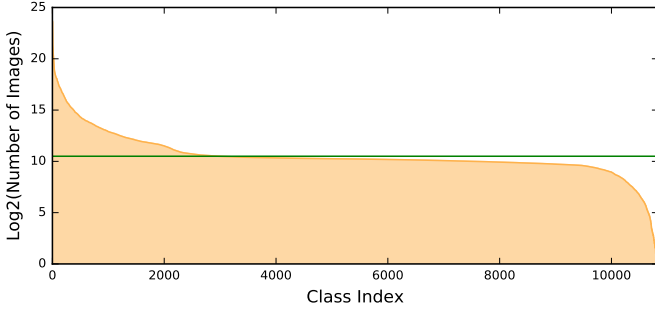
Fig. 1. Number of images ($\log_2$) per category in Tencent ML-Images. The red line indicates the average number of images of all categories.
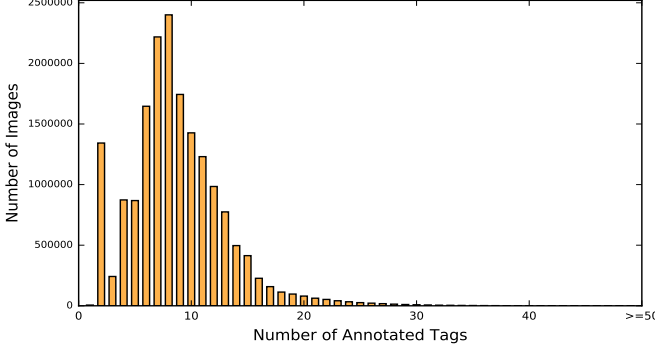


Fig. 2. The statistics of the numbers of annotated tags of all training images in Tencent ML-Images.

of images from ImageNet-11k as follows: if one image is originally annotated as $i$, then we also label it as category $j$.

### 3.3 Data Statistics

**Distribution of annotations**. The statistics of the number of images per category is shown in Fig. 1. Specifically, the maximum number of images of one category is 13,180,586, corresponding to the category 'object, physical object'; the minimum number is 0; the average number is 1,447.2. The distributions of different categories are extremely unbalanced. Some categories are frequent, while many others are very rare. There are 10,505 trainable categories, of which the number of images are larger than 100. It is referred to as the imbalance among categories [23]. The statistics of the numbers of annotated tags of all training images is shown in Fig. 2. Specifically, the numbers of annotated tags of all training images range from 1 to 91, and the average number is 8. Considering the size of the class vocabulary (*i.e.*, 11K), the numbers of annotated tags of all images are very small. In other words, the number of positive tags of each image is much smaller than the number of negative tags. It is referred to as the imbalance between positive and negative tags per image [23]. Above two types of imbalance bring in difficulty to model training. They will be considered during the training process of our model, as shown in Section 4.

**Noisy and missing tags**. Noisy tag indicates the incorrectly annotated tag, while missing tag [24], [25], [26], [27], [28] means that one class occurs in the image, but it is not annotated. As demonstrated in [3], the annotated tags for most images in Open Images are generated by machine, while the annotations of only a few fraction of images are verified by humans. Thus, many of these noisy annotations

are also included in Tencent ML-Images. Missing tag means that the positive tag of one image is not annotated. Most missing tags occur in images that are from ImageNet-11K, as these images were originally annotated by one single tag. As demonstrated in Section 3.2, we augment the tags of these single-label images by utilizing the category co-occurrence. Compared to automatically generating tags by machine as did in Open Images, our augmentation is rather conservative. Our concern is that it is difficult to control the noise proportion of the machine-generated annotations, and we believe that the negative influence of one noisy tag is larger than that of one missing tag. Some works (*e.g.*, [4], [5]) have demonstrated that learning from massive noisy labeled images is still able to show good visual representation. But they have not studied the trade-off between noisy and missing annotations, as the accurate proportions of both types of annotations are costly to calculate on large-scale databases. In this work, we choose the setting of more missing but less noisy annotations.

## 4 VISUAL REPRESENTATION LEARNING ON TENCENT ML-IMAGES

### 4.1 Training ResNet-101 with Multi-label Outputs on Tencent ML-Images

#### 4.1.1 Model and Loss Function

We implement the ResNet-101 model using Tensorflow-1.6.0, following the network architecture demonstrated in [29]. As our task is multi-label classification, the outputs of ResNet-101 are the activations of $m$ independent Sigmoid functions, with $m$ being the size of the class vocabulary. To alleviate the category imbalance, we propose a novel weighted cross entropy loss function. For clarity, we only present the loss function over one training image $\mathbf{x}_i$, as follows:

$$\mathcal{L}_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i) = \frac{1}{m} \sum_{j}^{m} r_t^j \left[ -\eta y_{ij} \log(p_{ij}) - (1 - y_{ij}) \log(1 - p_{ij}) \right],$$
(1)

where $p_{ij} = f_{\mathbf{W}}(\mathbf{x}_i, j) \in [0, 1]$ denotes the posterior probability with respect to category $j$, with $\mathbf{W}$ being the trainable parameters. $\mathbf{y}_i = [y_{i1}, \ldots, y_{ij}, \ldots, y_{im}] \in \{0, 1\}^m$ indicates the ground-truth label vector of image $\mathbf{x}_i$.

- The cost parameter $\eta > 1$ is introduced to set a larger cost on positive labels than negative labels, to alleviate the imbalance between positive and negative labels in each category. In our experiments, we specify $\eta$ as 12.
- $r_t^j = 0.9^{t-1}$ denotes an adaptive weight during the training process. For category $j$, if all training images in one mini-batch are negative, then we record the status as 0; if at least one training image in this mini-batch is positive, then we record the status as 1. Consequently, we record a status vector like $(\ldots, 0, 1, 1, 1, 0, 0, 1, 0, \ldots)$. Then, $t$ is defined as follows: if the status of the current mini-batch is different with that of the last mini-batch, *i.e.*, 01 or 10, then $t = 1$; if the current status is same with the last status, then $t = t + 1$. With $r_t^j$, if the parameters corresponding to category $j$ are positively or negatively updated
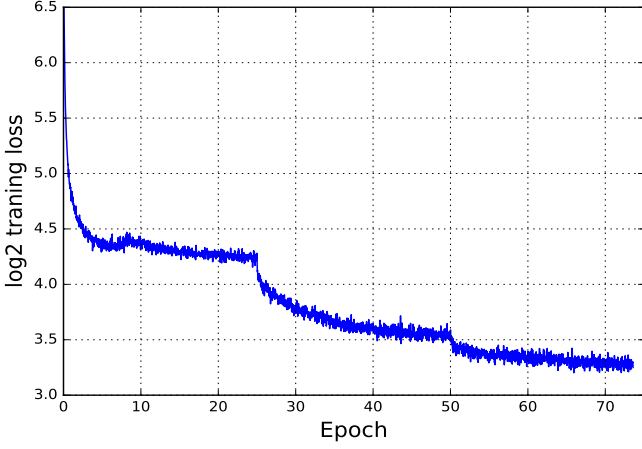
Fig. 3. The curve of the $\log_2$ training loss of pre-training the ResNet-101 model on Tencent ML-Images.

in sequential mini-batches, the weight of the corresponding loss function is decayed. It will alleviate the imbalance among frequent and rare categories.

### 4.1.2 Image Pre-processing

The image pre-processing used in our experiments consists of the following six sequential steps.

1) Crop a bounding box within the image randomly, to ensure the box area to be the range $[0.05, 1.0]$ of the whole image, and the aspect between the width and height to be the range $[\frac{3}{4}, \frac{4}{3}]$.
2) Re-size the cropped box to be $224 \times 224$.
3) Make the horizontal flip with the probability 0.5.
4) Rotate the image within the degree range $[-45, 45]$, with the probability 0.25.
5) Shift the color with the probability 0.5.
6) Re-scale the pixel value from $[0, 1]$ to $[-1, 1]$.

### 4.1.3 Training Algorithm and Hyper-parameters

We adopt the stochastic gradient descent (SGD) with momentum and back-propagation [30] to train the ResNet-101 model. The hyper-parameters used in our training are specified as follows. There are 17,609,752 training images. The batch-size is 4,096, and one epoch includes 4,300 steps. The learning rate is adjusted with a warm-up strategy [31]. Specifically, in the first 8 epochs, the learning rate starts at 0.01 and is increased with the factor 1.297 after each epoch. The learning rate will be 0.08 at the $9^{th}$ epoch. Then, the learning rate is decayed with the factor 0.1 in every 25 epochs. The momentum is 0.9. The maximal epoch is 60. When updating the parameters of BatchNorm, the decay factor of moving average is 0.9, and the constant $\epsilon$ is set to 0.001 to avoid the 0 value of the variance. The parameter decay is 0.0001.

In each batch, for each category, most or even all training images are negative. For category $i$, if there are no positive images in the current batch, the parameters of the fully connected layer corresponding to category $i$ will be updated with the probability of 0.1; if there are positive images for category $i$, we will down-sample negative images with the number of 5 times of positive images, then the corresponding parameters are updated according to positive and down-sampled negative images. With this setting, the classifier's
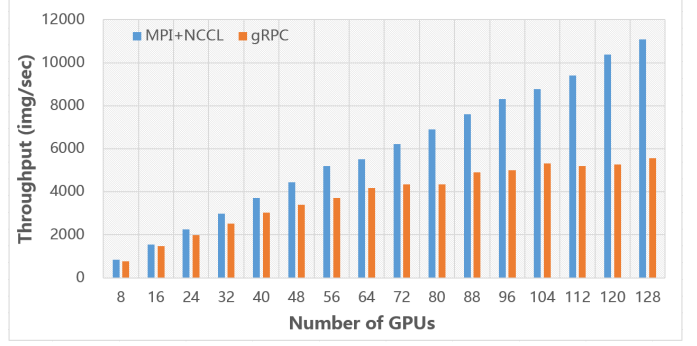


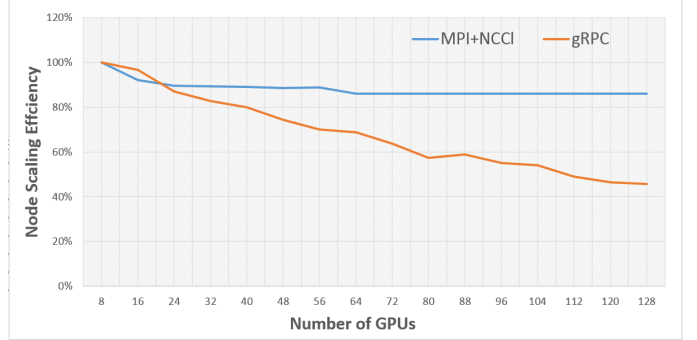Fig. 4. Throughout of distributed training based on MPI+NCCL and gRPC.



Fig. 5. Scaling efficiency of each GPU in distributed training based on MPI+NCCL and gRPC.

parameters corresponding to every category will be learned in a similar distribution, *i.e.*, the negative images are 5 times of the positive images, though the absolute numbers of training images among categories are still different. It can somewhat alleviates the negative influence of the distribution imbalance among categories. Besides, we set the cost parameter $\eta$ as 12, *i.e.*, a higher cost of positive labels than negative labels. It will alleviate the negative influence of the distribution imbalance between positive and negative labels. The curve of the $\log_2$ training loss is shown in Fig. 3.

### 4.1.4 Acceleration by Distributed Training

Training Resnet-101 with 11K outputs on 18M images requires lots of computation. The training will take a few dozen days if using one single machine. In this work, all training experiments are conducted on a large-scale distributed deep learning framework, *i.e.*, TFplus, which is built upon Tensorflow with several communication optimized techniques. We replace the original gRPC implementation with Message Passing Interface (MPI) and NVIDIA Collective Communications Library (NCCL) [7]. NCCL provides a highly optimized version of routines, such as all-gather, all-reduce, broadcast, reduce, reduce-scatter, especially the integrated bandwidth-optimal ring all-reduce algorithm [32], to achieve high bandwidth over PCIe on NVIDIA GPU. In order to scale from one GPU to multiple nodes and multiple GPUs, we implement several APIs for communication: 1) a broadcast operation to synchronize parameters among all GPUs at the initialization stage or the recovery from the checkpoint; 2) a distributed optimizer wrapper for synchronization update of parameters; 3) some operations for data partition and barrier, etc. Since both MPI and NCCL support the remote direct memory access

| Evaluation metric → Top-k prediction ↓ | Precision | Recall | F1 |
|---|---|---|---|
| Top-5 prediction | 35.7 | 18.2 | 23.3 |
| Top-10 prediction | 29.0 | 29.5 | 28.1 |

TABLE 1
Results (%) of the ResNet-101 with multi-label outputs, evaluated on the validation set of Tencent ML-Images.

(RDMA), we run all distributed training jobs over a 40-GbE RDMA-capable networking. We achieve about 2X speed up compared with the original gRPC based distributed implementation on a cluster of 16 nodes and each node with 8 NVIDIA P40 GPUs, as shown in Fig. 4. Specifically, when training with 128 GPUs, the throughout (*i.e.*, the number of processed images per second) of MPI+NCCL is up to 11077, while the throughout of gRPC is 5551. Besides, the distributed training jobs based on MPI+NCCL achieve 86% scaling efficiency from 8 to 128 GPUs, while those based on gRPC are only 46%, as shown in Fig. 5.

The whole training process with 60 epochs takes 90 hours, *i.e.*, 1.5 hours per epoch. In contrast, as reported in [5], the training of ResNet-101 on JFT-300M takes 2 months for 4 epochs, based on 50 NVIDIA K80 GPUs. Specifically, our training takes 218 GPU-hours (*i.e.*, one GPU runs 218 hour) to process 18M images; the training on JFT-300M [5] takes 18,000 GPU-hours to process 300M images. The speed of each GPU in our implementation is 5X of that of the implementation in [5].

## 4.2 Evaluations and Results

To evaluate the performance of the trained ResNet-101 with multi-label outputs, we adopt the widely used instance-level metrics in multi-label learning, including instance-level precision, recall and F1 score. As the output for each category is the posterior probability, we need to transform the continuous predictions to binary predictions, to calculate above metrics. Specifically, for image $i$, we determine the categories corresponding to top-k largest posterior probabilities as positive labels (*i.e.*, 1), while all other categories are negative labels (*i.e.*, 0). We obtain a binary prediction vector $\hat{\mathbf{y}}_i^k \in \{0, 1\}^m$. Then, the evaluation metrics are calculated as follows:

$$P_k = \frac{1}{n} \sum_i^n P_{i,k} = \frac{1}{n} \sum_i^n \frac{\mathbf{y}_i * \hat{\mathbf{y}}_i^k}{k}, \quad (2)$$

$$R_k = \frac{1}{n} \sum_i^n R_{i,k} = \frac{1}{n} \sum_i^n \frac{\mathbf{y}_i * \hat{\mathbf{y}}_i^k}{\mathbf{1}^\top * \mathbf{y}_i}, \quad (3)$$

$$F1_k = \frac{1}{n} \sum_i^n \frac{2P_{i,k} * R_{i,k}}{P_{i,k} + R_{i,k}}. \quad (4)$$

We present results of top-5 and top-10 predictions, as shown in Table 1. The evaluation values are not very high. As demonstrated in Section 3.1, the size of the validation set is only about $\frac{1}{200}$ of the size of the training set. And, there should be many missing labels for validation images. Thus, the evaluation scores on this small validation set are not reliable enough to measure the visual representation performance of the model trained on Tencent ML-Images. Instead, its performance could be evaluated through transfer learning to some other visual tasks, as follows.

## 5 TRANSFER LEARNING

### 5.1 Transfer Learning to Single-Label Image Classification on ImageNet-ILSVRC2012

To verify the quality of the visual representation of the ResNet-101 model pre-trained on Tencent ML-Images, we conduct transfer learning to image classification on the benchmark single-label image database, *i.e.*, ImageNet-ILSVRC2012. Specifically, we utilize the ResNet-101 model pre-trained on Tencent ML-Images as the initial checkpoint, and replace the output layer with 1,000 outputs nodes, as well as the loss with softmax loss. Then, we fine-tune this checkpoint on ImageNet-ILSVRC2012. Hereafter, we denote ImageNet-ILSVRC2012 as ImageNet for clarity.

#### 5.1.1 Fine-Tuning Approaches

**Learning rate**. Note that there are significant differences between Tencent ML-Images and ImageNet. First, the distributions of visual features and class vocabulary are different. Second, images in Tencent ML-Images are annotated with multiple tags, while images in ImageNet are annotated with one single label. Last, the annotations of Tencent ML-Images are noisy, while the annotations of ImageNet are clean. Considering these significant differences, one cannot expect that the ResNet-101 model pre-trained on Tencent ML-Images show good classification performance on ImageNet, if without fine-tuning of the parameters. The standard fine-tuning approach adopts one consistent learning rate of all layers, that is often smaller than the learning rate used in pre-training. It is referred to as fine-tuning with layer-wise consistent learning rates. However, as verified in later experiments, the ResNet-101 model with the above fine-tuning approach even shows worse performance than the ImageNet baseline. It reveals that the useful information contained in the Tencent ML-Images checkpoint has not been well utilized, due to aforementioned three differences. To tackle this issue, we adopt the fine-tuning approach with layer-wise adaptive learning rates. Specifically, we set larger learning rates on top layers, while smaller learning rates on bottom layers. The rationale behind this setting is that the parameters of top-layers are more dependent on training images and labels, while the parameters in bottom layers represent low-level visual features. To alleviate the negative influence of the significant differences between Tencent ML-Images and ImageNet, the top-layers' parameters should be changed to be further from the initialized parameters from the checkpoint, compared to the bottom layers' parameters.

**Image size**. As demonstrated in [29], the image size in training and testing has a significant influence to the results. Further, YOLO9000 [33] proposes to adjust the image size during training, which is proved to be helpful for the detection performance. And, we note that the image size of pre-training, fine-tuning and evaluation used in [5] is 299, while the image size in our pre-training is 224. To conduct a fair comparison with the JFT-300M, and also to explore the influence of image size to image classification, we design three fine-tuning settings with different image sizes, including: 1) the image size is kept at $224 \times 224$; 2) the image size in early epochs is set as $224 \times 224$, while $299 \times 299$ in late epochs; 3) the image size is kept at $299 \times 299$.

| Hyper-parameters | ckpt-1 | ckpt-2 | ckpt-3/4/5 |
|---|---|---|---|
| Batch size | | 2048 | |
| Maximum epoch | | 120 | |
| LR-top2-stages | 0.8 | 0.008 | 0.8 |
| LR-bottom2-stages | 0.8 | 0.008 | 0 |
| LR-decay-factor | | 0.1 | |
| LR-decay-step | | 18750 (30 epochs) | |
| Weight-decay | | 0.0001 | |
| warmup-steps | | 2500 (the first 4 epochs) | |
| LR-warmup | 0.1 | 0.001 | 0.1 |
| LR-warmup-decay-factor | | 1.681 | |
| LR-warmup-decay-step | | 625 (one epoch) | |
| BatchNorm-decay | | 0.9 | |
| BatchNorm-eps | 0.00001 | 0.001 | 0.001 |

TABLE 2
Hyper-parameters of different checkpoints in transfer learning to ImageNet. 'ckpt' denotes our ResNet-101 checkpoint. ckpt-1 means the checkpoint trained on ImageNet from scratch; ckpt-2 indicates the checkpoint pre-trained on Tencent ML-Images and fine-tuned on ImageNet with layer-wise consistent learning rates; ckpt-3/4/5 represent three checkpoints pre-trained on Tencent ML-Images and fine-tuned on ImageNet with layer-wise adaptive learning rates, and they are different at the image size in fine-tuning. "LR" denotes learning rate. ResNet-101 consists of 4 stages of residual blocks, and top2-stages indicates the two stages close to the output layer, while bottom2-stages represents those close to the input layer. Note that if there is only one value in a row, then it means that the checkpoints of different columns are trained with the same value of the corresponding hyper-parameter.

### 5.1.2 Comparisons and Hyper-parameters

To verify the quality of visual representation of the Tencent ML-Images checkpoint, we compare five ResNet-101 checkpoints with different fine-tuning approaches. The hyper-parameters of training these checkpoints are summarized in Table 2. Besides, we also present the reported results of others' implementations, including: the MSRA checkpoint of training on ImageNet from scratch; the Google's checkpoint of training on ImageNet from scratch [5]; the checkpoint of pre-training on JFT-300M and fine-tuning on ImageNet with layer-consistent learning rates [5]. Please refer to Table 3.

### 5.1.3 Results

All compared results are shown in Table 3. **(1)** In terms of the baseline (*i.e.*, train on ImageNet from scratch), our ckpt-1 is higher than both MSRA ckpt [16] and Google ckpt-1 [5]. Our implementation and MSRA implementation are same at the model architecture and the size of the input image (*i.e.*, $224 \times 224$). The main difference is the pre-processing of the input image (see Section 4.1.2) and hyper-parameters, which should be the main reasons of different performance of this two baselines. In contrast, the details of the model architecture and the image pre-processing are not demonstrated in [5], and the size of the image size is $299 \times 299$. But our ckpt-1 still performs better than Google ckpt-1 on the evaluation of $299 \times 299$ validation images. It demonstrates the good quality of our implementation of the baseline. **(2)** Moreover, the comparison between our baseline checkpoint (*i.e.*, our ckpt-1 in Table 3) and our fine-tuning checkpoints (*i.e.*, our ckpt-2/3/4/5 in Table 3) demonstrate two points. The accuracy of our ckpt-2 is much lower than that of our ckpt-1. As analyzed in Section 5.1.1, the significant difference between Tencent ML-Images and ImageNet, as well as the label noises in Tencent ML-Images,
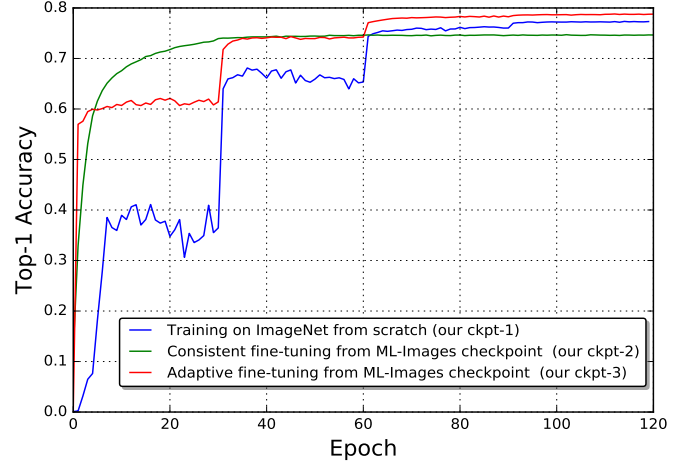


Fig. 6. The curves of top-1 accuracy of different checkpoints along the training progress, evaluated on the validation set of ImageNet.

could bring in negative influence to the model performance. But the fine-tuning with layer-consistent learning rate fails to alleviate this negative influence. In contrast, our ckpt-3 with the layer-wise adaptive fine-tuning learning rate shows the improvements of 1.0% at top-1 accuracy and 0.6% at top-5 accuracy. This demonstrates that the fine-tuning with layer-wise adaptive learning rate can not only utilize the good visual representation encoded in bottom layers of the Tencent ML-Images checkpoint, but also alleviate the significant difference between two databases, which is encoded in top layers. It tells us that the Tencent ML-Images checkpoint includes good visual representation, but it should be carefully fine-tuned to help other vision tasks. We also evaluate the checkpoints of different epochs using top-1 accuracy, on the validation set of ImageNet, as shown in Fig. 6. **(3)** The results of Google ckpt-2 are 79.2% top-1 accuracy and 94.7% top-5 accuracy. The improvements over Google ckpt-1 are 1.7% at top-1 accuracy and 0.8% top-5 accuracy. It demonstrates the good quality of the initial checkpoint pre-trained on JFT-300M. In contrast, all of our ckpt-3/4/5 show higher accuracies than Google ckpt-2 when the image size of validation set is set as $299 \times 299$. And, our ckpt-4 with adaptive image sizes in fine-tuning achieves the highest 80.73% top-1 and 95.5% top-5 accuracy, and the improvements over our ckpt-1 are 1.73% at top-1 and 1% at top-5 accuracy. Our checkpoints exceed Google ckpt-2 on both the accuracy and the accuracy improvement over baseline. Considering that the size of JFT-300M is about 17× of Tencent ML-Images, it verifies the high quality of Tencent ML-Images and our training and fine-tuning. However, the accuracies of the fine-tuning from JFT-300M are still higher than that of the fine-tuning from our checkpoint.

## 5.2 Transfer Learning to Caltech-256

We also conduct transfer learning to another small-scale single-label image database, *i.e.*, Caltech-256 [9], which includes 30,607 images with 256 object categories. We utilize a pre-trained checkpoint of the ResNet-101 model to extract features for each image of Caltech-256. Specifically, we adopt the output of the global average pooling in ResNet-101 as the feature vector (2,048 dimensions). Then we train

| ResNet-101 checkpoints | Training and fine-tuning settings | Validation size 224 top-1 acc  top-5 acc | | Validation size 299 top-1 acc  top-5 acc | |
|---|---|---|---|---|---|
| MSRA ckpt [16] | Train on ImageNet, 224 | 76.4 | 92.9 | – | – |
| Google ckpt-1 [5] | Train on ImageNet, 299 | – | – | 77.5 | 93.9 |
| Our ckpt-1 | Train on ImageNet, 224 | 77.8 | 93.9 | 79.0 | 94.5 |
| Google ckpt-2 [5] | Train on JFT-300M, fine-tune on ImageNet, 299 | – | – | 79.2 | 94.7 |
| Our ckpt-2 | Pre-train on Tencent ML-Images, fine-tune on ImageNet with layer-wise consistent learning rate | 74.7 | 92.6 | – | – |
| Our ckpt-3 | Pre-train on Tencent ML-Images, fine-tune on ImageNet | **78.8** | **94.5** | 79.5 | 94.9 |
| Our ckpt-4 | Pre-train on Tencent ML-Images, fine-tune on ImageNet 224 to 299 | 78.3 | 94.2 | **80.73** | **95.5** |
| Our ckpt-5 | Pre-train on Tencent ML-Images, fine-tune on ImageNet 299 | 75.8 | 92.7 | 79.6 | 94.6 |

TABLE 3
Results (%) of single-label image classification on the validation set of ImageNet, using single model and single crop inference. In the column of "Train and fine-tune settings", '224' indicates that the size of training image is $224 \times 224$, while '299' indicating $299 \times 299$; '224 to 299' means that the size of training image in early epochs is $224 \times 224$, and that in later epochs is $299 \times 299$. For Our ckpt-3/4/5, we adopt the layer-wise adaptive learning rate in fine-tuning.

| Checkpoints | Checkpoint 1 | Checkpoint 2 | Checkpoint 3 |
|---|---|---|---|
| Accuracy | 86.0 | 83.6 | 86.5 |

TABLE 4
Classification accuracy (%) of transfer learning to Caltech-256. Checkpoints 1 to 3 respectively indicate: training on ImageNet from scratch; training on Tencent ML-Images from scratch; pre-training on Tencent ML-Images, and fine-tuning on ImageNet with a layer-wise adaptive learning rate.

a multi-category SVM classifier to predict the label of each category. We compare with three checkpoints, including: training on ImageNet from scratch; training on Tencent ML-Images from scratch; pre-training on ML-Image, and fine-tuning on ImageNet with layer-wise adaptive learning rate. The results of different checkpoints are shown in Table 4. The result of the ImageNet checkpoint is higher than that of the Tencent ML-Images checkpoint. It again demonstrates that the significant difference of distribution between Tencent ML-Images and single-label image databases. The checkpoint of adaptive fine-tuning from Tencent ML-Images gives 86.5% accuracy, which is higher than 86% accuracy of the ImageNet checkpoint. It also verifies that the good generalization of visual representation of the Tencent ML-Images could be well explored through adaptive fine-tuning.

### 5.3 Transfer Learning to Object Detection

We conduct transfer learning to object detection on the benchmark PASCAL VOC database, including 20 categories. To fairly compare with [5], we also adopt the 'trainval' images from both PASCAL VOC 2007 and 2012 as the training set of fine-tuning, including 16,551 training images. All models are evaluated on the testing set of PASCAL VOC 2007, including 4,952 images, using mean average precision at 50% IOU threshold (mAP@.5) metric.

**Comparisons.** We compare with the transfer learning did in [5], including their baseline checkpoint pre-trained on ImageNet, and their checkpoints pre-trained on JFT-300M and JFT-300M+ImageNet. Our first checkpoint (*i.e.*, our ckpt-1) is also pre-trained on ImageNet from scratch, then fine-tuned on VOC. Our second checkpoint (*i.e.*, our ckpt-2) is firstly pre-trained on Tencent ML-Images, then

fine-tuned on ImageNet (using the same setting with "Our ckpt-3" in Table 3), and further fine-tuned on VOC.

**Implementation details.** Our implementation is based on the TensorFlow implementation of the Faster RCNN framework ( [34], [35]). Specifically, we use stochastic gradient descent with the momentum of 0.9 for training; the initial learning rate is set to be $8 \times 10^{-4}$ and decays by 0.1 at every 80k steps; the batch size is set to be 256; the model is trained for 180k steps; the weight decay is set to be $10^{-3}$. The input image is resized such that the short side is fixed to 600-pixels, while the aspect ratio is maintained.

**Results.** The results are summarized in Table 5. In comparison of the baseline checkpoint, our ckpt-1 achieves 80.1%, while Google ckpt-1 gives 76.3%. With the same databases and models, it demonstrates that our implementation quality is much better than Google's implementation. Our ckpt-2 shows the improvement of 1.4% than our ckpt-1. This verifies that the good visual representation of the checkpoint pre-trained on Tencent ML-Images. In contrast, Google ckpt-2 and ckpt-3 show 81.4% and 81.3%, respectively, which are slightly lower than our ckpt-2. It demonstrates that the checkpoint pre-trained on Tencent ML-Images and fine-tuned on ImageNet has similar generalization with those pre-trained on JFT-300M and JFT-300M+ImageNet. Considering that JFT-300M is about 17-times larger than our Tencent ML-Images, we could claim that Tencent ML-Images is a high-quality database. We also tried the checkpoint that is pre-trained on Tencent ML-Images, then fine-tuned on VOC. But the performance is not as good as other checkpoints, thus we didn't report its results here. We think the reason is that the big gap of the data distribution and task, between Tencent ML-Images and VOC.

### 5.4 Transfer Learning to Semantic Segmentation

We conduct transfer learning to semantic segmentation on the benchmark PASCAL VOC 2012 database, including 20 foreground categories and 1 background category. To fairly compare with [5], we also adopt the augmented training set of PASCAL VOC 2012 as the training set of fine-tuning, which includes 10,582 training images. All models are evaluated on the validation set of PASCAL VOC 2012, including

| Checkpoints | Pre-training and fine-tuning settings | mAP@0.5 |
|---|---|---|
| Google ckpt-1 | Pre-train on ImageNet, fine-tune on VOC | 76.3 |
| Our ckpt-1 | Pre-train on ImageNet, fine-tune on VOC | 80.1 |
| Google ckpt-2 | Pre-train on JFT-300M, fine-tune on VOC | 81.4 |
| Google ckpt-3 | Pre-train on JFT-300M+ImageNet, fine-tune on VOC | 81.3 |
| Our ckpt-2 | Pre-train on Tencent ML-Images, fine-tune on ImageNet, and then fine-tune on VOC | **81.5** |

TABLE 5
Results of object detection on the testing set of PASCAL VOC 2007. Note that "VOC" in "fine-tune on VOC" indicates the combined training set of PASCAL VOC 2007 and 2012.

| Checkpoints | Pre-training and fine-tuning settings | mIOU |
|---|---|---|
| Google ckpt-1 | Pre-train on ImageNet, fine-tune on VOC | 73.6 |
| Our ckpt-1 | Pre-train on ImageNet, fine-tune on VOC | 74.0 |
| Google ckpt-2 | Pre-train on JFT-300M, fine-tune on VOC | 75.3 |
| Google ckpt-3 | Pre-train on JFT-300M+ImageNet, fine-tune on VOC | **76.5** |
| Our ckpt-2 | Pre-train on Tencent ML-Images, fine-tune on ImageNet, and then fine-tune on VOC | 76.3 |

TABLE 6
Results of semantic segmentation on the validation set of PASCAL VOC 2012. Note that "VOC" in "fine-tune on VOC" indicates the augmented training set of PASCAL VOC 2012.

1,149 images, using the mean itersection-over-union (mIOU) metric.

**Comparisons.** We adopt the same setting of checkpoints as did in the above object detection experiments (see Section 5.3). Thus, here we didn't repeat it to keep clarity.

**Implementation details.** Our implementation is based on the semantic segmentation architecture of DeepLab [36]. For fair comparison, our implementation also adopts DeepLab-ASPP-L structure that has four branches after the Conv5 block of ResNet-101 model. All ASPP branchs use $3 \times 3$ kernels but with different atrous rates (*i.e.*, $\{6, 12, 18, 24\}$). During training, we use the "poly" learning rate policy (with power = 0.9) and the initial learning rate is set to be $3 \times 10^{-3}$. The weight decay is set as $5 \times 10^{-4}$. The model is trained for 50k steps using the stocastic gradient descent with momentum of 0.9. The batch size is set to be 6, and the input image is resized to $513 \times 513$.

**Results.** The results are summarized in Table 6. We could obtain the similar observations with the transfer learning to object detection (see Section 5.3 and Table 5). 1) Our implementation of the baseline checkpoint is better than that of Google, *i.e.*, our ckpt-1 74.0% vs. Google ckpt-1 73.6%. 2) Our ckpt-2 shows the improvement up to 2.3% over our ckpt-1, which verifies the good visual representation of the checkpoint pre-trained on Tencent ML-Images. 3) Google ckpt-2 shows 75.3% and Google ckpt-3 gives 76.5%. It demonstrates that the checkpoint pre-trained on Tencent ML-Images and fine-tuned on ImageNet has better generalization than that pre-trained on JFT-300M, while is similar with the checkpoint pre-trained on JFT-300M+ImageNet.

## 6 CONCLUSIONS

In this work, we built a large-scale multi-label image database, dubbed Tencent ML-Images, including about 18M images and 11K categories. It is the largest-scale public multi-label image database until now. We presented the lage-scale visual presentation learning of deep convolutional neural networks on Tencent ML-Images, employing a distributed training framework with MPI and NCCL. A novel loss function was carefully designed to alleviate the side-effect of the severe class imbalance in the large-scale multi-label database. Extensive experiments of transfer learning to other visual tasks, including single-label image classification, object detection, and semantic segmentation, verify that Tencent ML-Images is of very high quality, and the pre-trained checkpoint has very good visual representation. We hope that this work could provide a new benchmark of largre-scale visual representation, to promote other visual tasks in the research and industry community. The Tencent ML-Images database, the complete code from data preparation, pre-training to fine-tuning and feature extraction, and the pre-trained and fine-tuned checkpoints of the ResNet-101 model have been released at *https://github.com/Tencent/tencent-ml-images*.

## REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. IEEE, 2009, pp. 248–255.

[3] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy, "Openimages: A public dataset for large-scale multi-label and multi-class image classification." *Dataset available from https://github.com/openimages*, 2017.

[4] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, and S. Belongie, "Learning from noisy large-scale datasets with minimal supervision," in *CVPR*, 2017.

[5] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *ICCV*. IEEE, 2017, pp. 843–852.

[6] C. Fellbaum, *WordNet*. Wiley Online Library, 1998.

[7] NVIDIA, "Nvidia collective communications library (nccl)." *https://developer.nvidia.com/nccl*, 2017.

[8] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[9] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[10] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[11] W. Li, L. Wang, W. Li, E. Agustsson, and L. Van Gool, "Webvision database: Visual learning and understanding from web data," *arXiv preprint arXiv:1708.02862*, 2017.

[12] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *CVPR*. IEEE, 2010, pp. 3485–3492.

[13] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[15] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *BMVC*, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[17] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *ECCV*. Springer, 2002, pp. 97–112.

[18] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 319–326.

[19] M. Grubinger, P. Clough, H. Müller, and T. Deselaers, "The iapr tc-12 benchmark: A new evaluation resource for visual information systems," in *International Workshop OntoImage*, 2006, pp. 13–23.

[20] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nuswide: A real-world web image database from national university of singapore," in *CIVR*, 2009.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[23] B. Wu, S. Lyu, and B. Ghanem, "Constrained submodular minimization for missing labels and class imbalance in multi-label learning." in *AAAI*, 2016, pp. 2229–2236.

[24] B. Wu, Z. Liu, S. Wang, B.-G. Hu, and Q. Ji, "Multi-label learning with missing labels," in *ICPR*, 2014.

[25] B. Wu, S. Lyu, and B. Ghanem, "Ml-mg: Multi-label learning with missing labels using a mixed graph," in *ICCV*, 2015, pp. 4157–4165.

[26] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji, "Multi-label learning with missing labels for image annotation and facial action unit recognition," *Pattern Recognition*, vol. 48, no. 7, pp. 2279–2289, 2015.

[27] Y. Li, B. Wu, B. Ghanem, Y. Zhao, H. Yao, and Q. Ji, "Facial action unit recognition under incomplete data based on multi-label learning with missing labels," *Pattern Recognition*, vol. 60, pp. 890–900, 2016.

[28] B. Wu, F. Jia, W. Liu, B. Ghanem, and S. Lyu, "Multi-label learning with missing labels using mixed dependency graphs," *International Journal of Computer Vision*, pp. 1–22.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*. Springer, 2016, pp. 630–645.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[31] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[32] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 117–124, 2009.

[33] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 6517–6525.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[35] X. Chen and A. Gupta, "Spatial memory for context reasoning in object detection," in *ICCV*, pp. 4086–4096.

[36] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.