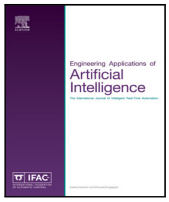
Engineering Applications of Artificial Intelligence 117 (2023) 105621

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Imbalanced data classification: Using transfer learning and active sampling

Yang Liu [a,1], Guoping Yang [a,1], Shaojie Qiao [a,b,*,1], Meiqi Liu [c], Lulu Qu [a], Nan Han [d,**], Tao Wu [e], Guan Yuan [f], Tao Wu [g], Yuzhong Peng [h]

[a] School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China
[b] Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu 610021, China
[c] Sichuan Spaceon Aerospace Information Technology Co., Ltd, Chengdu 611731, China
[d] School of Management, Chengdu University of Information Technology, Chengdu 610103, China
[e] School of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China
[f] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
[g] School of Cybersecurity and Information Law, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[h] Guangxi Key Lab of Human–Machine Interaction and Intelligent Decision, Nanning Normal University, Nanning 530001, China

ARTICLE INFO

Keywords:
Class-imbalanced data
Transfer learning
Active sampling
DenseNet
Real-time data

ABSTRACT

Recently, deep learning models have made great breakthroughs in the field of computer vision, relying on large-scale class-balanced datasets. However, most of them do not consider the class-imbalanced data. In reality, the class-imbalanced distribution can lead to the degradation of model performance, reducing the generalization of these models. In addition, in the era of big data, many applications need to use real-time visual data. These data come from different mobile devices, which continuously generate a huge number of visual data. However, there are few studies using real-time data from information systems, real-time data is easy to capture but difficult to use. In order to solve the above problems, we propose a new model (Transfer Learning Classifier, TLC) based on transfer learning to deal with class-imbalanced data. The model includes active sampling module, real-time data augmentation module and DenseNet module. Among them, (1) the newly proposed active sampling module can dynamically adjust the number of samples with skewed distribution; (2) the data augmentation module can expand the real-time data to avoid over-fitting and insufficient data; (3) the DenseNet module is a standard DenseNet network pre-trained on the ImageNet dataset and transferred to TLC for relearning, and then we adjust the memory usage of the standard DenseNet to make it more efficient. In addition, we have applied a new end-to-end real-time data storage and analysis system. A large number of experiments have been carried out on four different long mantissa data sets. Experimental results show that the proposed TLC model can effectively deal with the static data as well as the real-time data, and the classification effect of imbalanced data is better than that of existing models.

1. Introduction

Visual, textual, and aural data belong to information media data, which have the characteristics of large capacity, fast speed, and diversity. With the development of big data, a large number of network mobile devices (e.g., network cameras) have been deployed in public places, tourist attractions and other areas. These network mobile devices do not require manual intervention, and can continuously capture and send real-time information media data, including pictures and videos, through the network. Humans can use a wealth of data to provide services such as predicting tourist visiting trend and detecting street facility breakdown. However, for users, it is difficult for these real-time data to be directly used for analysis, therefore, we apply an end-to-end (user to analysis) real-time data analysis system, and then the run-time environment allows analysis programs to process the data obtained from thousands of devices simultaneously.

However, most real-world data has a long-tailed distribution (Hamidzadeh et al., 2020), as shown in Fig. 1, that is, the training set follows an exponential decay distribution in the sample size of each class (Kim et al., 2021), which tends to learn a certain type of knowledge. The relevant definition of long-tailed is a distribution with polynomial tail behavior, defined with Eq. (1):

$$P(X > x) \sim cx^{-\alpha} \tag{1}$$

where X is a random variable, c is a local variable, and α is a shape parameter. When α is less than 2, the distribution has infinite

* Corresponding author at: School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China.
** Corresponding author.
E-mail addresses: sjqiao@cuit.edu.cn (S. Qiao), hannan@cuit.edu.cn (N. Han).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.engappai.2022.105621
Received 13 February 2022; Received in revised form 8 October 2022; Accepted 7 November 2022
Available online 24 November 2022

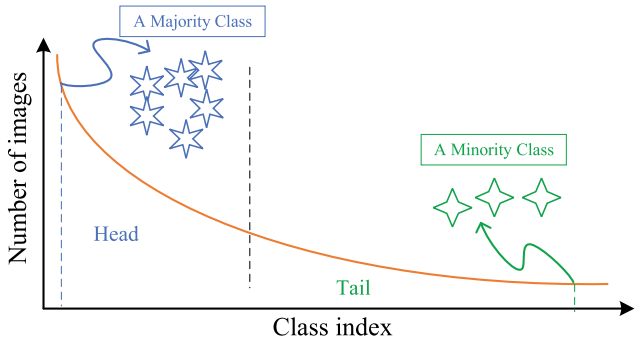


Fig. 1. The phenomenon of long-tailed distributions.

variance, which is also required for these models to produce self-similarity (Downey, 2001).

Challenges. This phenomenon is called imbalanced data problem, and it does exist in several applications such as medical, industrial, and monitoring systems. The problem is to accurately predict the minority classes in the dataset by using the majority classes. Although deep learning algorithms have made great progress in recent years, few studies focus on imbalanced data classification. Compared with traditional classification algorithms, existing deep neural networks can achieve very high performance in class-balanced datasets. However, as we have aforementioned, deep neural networks perform worse in handling the imbalanced data classification problem because they are more inclined to learn from the majority classes instead of the minority classes. In addition, there are a few performance measurements that evaluate the prediction quality of minority classes by deep learning models.

In order to handle the aforementioned challenges, we propose a model based on transfer learning (called TLC) which can effectively deal with class-imbalanced data in multiple class datasets. Specifically, we propose an active sampling algorithm that dynamically updates the number of samples in each class in each iteration based on the F1-score measurement. We combine the newly proposed active sampling algorithm with the standard DenseNet (Huang et al., 2018) model, and use data augmentation and transfer learning techniques to avoid over-fitting as well as generalize the model. This method will significantly improve the performance of discovering minority class and take into account the performance of the majority class. We also improve the memory utilization of the standard DenseNet to make it more efficient. Fig. 2 contains a real-time data augmentation module, a DenseNet transfer learning module and a dynamic sampling module. After training, the transformed images are generated by the real-time data augmentation module. The DenseNet transfer learning module is used to tune the model, and the dynamic sampling module can automatically generate new samples according to the performance of the contrast set, which is based upon F1-scores of the confusion matrix. When we use the TLC model, we freeze the data augmentation and active sampling modules, leaving only the transfer learning module. We input the training samples to DenseNet, after the above operations, we finally obtain accurate classification of samples.

In summary, we make the following original contributions in this study:

(1) We propose an active sampling algorithm, which dynamically adjusts the number of samples in each class according to the F1-score of the confusion matrix w.r.t. the class after each iteration. This algorithm can handle the problems of the imbalanced multiple classifications after several iterations of training, and make accurate predictions.

(2) We propose a novel model (called TLC) based on transfer learning, which performs real-time data augmentation, and then inputs the generated samples into the transfer learning model for training. TLC includes an active sampling algorithm, which can adjust the number of samples of the class. In addition, we improve the memory utilization of DenseNet to make it consume less memory. We also apply an end-to-end (i.e., user-to-analysis) real-time data analysis system to efficiently store and analyze real-time data collected by mobile devices.

(3) We evaluate the proposed method on four different long-tailed datasets. We set different balance rates for the datasets, even as high as 100 (that is, the ratio of the class with the most samples to the class with the least samples are 100). Compared with previous methods, the performance of our proposed method is always better than competitive methods.

The rest of this paper is organized as follows. We briefly review the related work in Section 2. We introduce the architecture of the proposed model and its components in Section 3. Experimental studies are presented in Section 4. Lastly, we conclude our work and discuss the future directions in Section 5.

2. Related work

Visual data analysis is challenging, which is large-scale and heterogeneous (Pouyanfar et al., 2018). Analyzing large-scale data requires expensive computing overhead. Some studies have been proposed to effectively analyze the visual data. Inception (Szegedy et al., 2016), DenseNet (Huang et al., 2018) and ResNet (He et al., 2016) are commonly used models in image classification. All these models need to be trained on large-scale datasets, which are usually balanced.

However, the above researches do not focus on the problem of data imbalance in image fields. The recent researches for class imbalance problems can be divided into two categories: data-driven and algorithm-driven (Pouyanfar et al., 2018). The data-driven approach is to modify data distribution of datasets. The specific operation is to undersample majority classes or oversample minority classes to change the data distribution to balance the data. However, oversampling may lead to over-fitting, and undersampling may lose some useful information. Chawla et al. (2002) proposed *Synthetic Minority Oversampling Technique* (called SMOTE), which uses linear interpolation between nearest neighbors at the feature level instead of the data level, and generates a small number of samples to deal with the overfitting problem. Mullick et al. (2019) proposed a method of generating synthetic samples, which use the idea of generative adversarial networks (GAN) to perform oversampling on minority classes in deep learning systems. On the other hand, algorithm-driven schemes try to solve the data imbalance problems by improving the learning process. Cao et al. (2013) proposed an effective wrapper framework, which combines the evaluation measurements with cost-sensitive SVM directly to improve the performance of classification. However, these techniques usually require the expensive computational resources and training time to generalize the model.

The recent studies combine the existing class imbalance solutions with deep learning algorithms. Yan et al. (2015) propose a bootstrapping method based on CNNs (Convolutional Neural Networks), which can deal with a few classes for binary classification tasks. In order to better learn the feature representation from CNN, Khan et al. (2018) propose a cost sensitive learning. Cao et al. (2019) proposed a label-distribution-aware margin loss scheme to deal with the over-fitting for the minority classes by regularizing the margins. Shu et al. (2019) proposed a method which can adaptively learn an explicit weighting function directly from data, and this method can be applied to many weighting functions including those assumption-based traditional functions. Gao et al. (2021) proposed a sampling ensemble method for imbalanced learning, which divided all samples into safe samples, boundary samples, rare samples and abnormal samples according to the sample neighborhood information. In each iteration, the majority and minority classes in each training dataset are undersampled or oversampled, respectively. Tahvili et al. (2020) regarded software testing optimization as an imbalanced learning task based on the imbalanced distribution of testing cases, and proposed a new method that consists

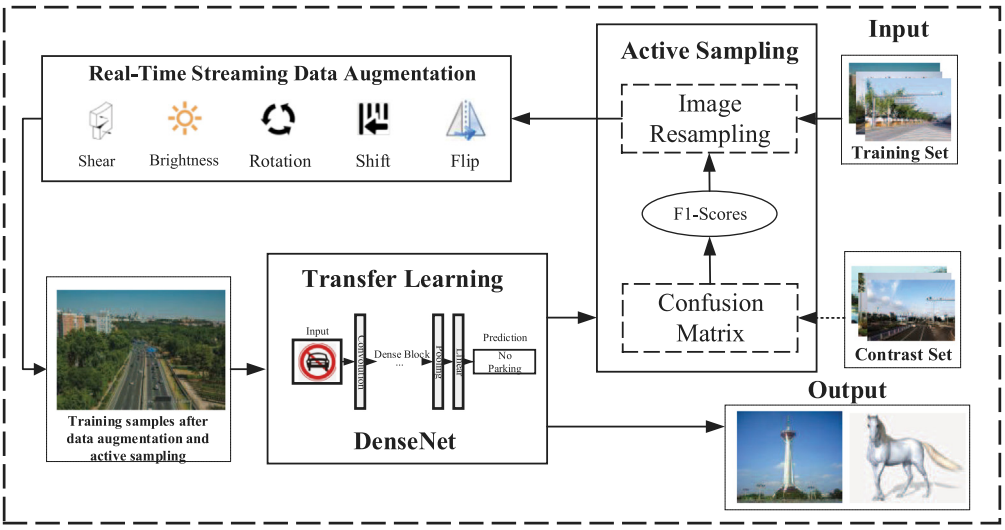


Fig. 2. Structure of the proposed TLC model.

of two main steps. First, they analyzed the specifications of the testing cases and convert them to vectors. Second, by using the obtained vectors, each testing case is classified into a dependent class or an independent class, and then a supervised learning method is performed to deal with the imbalanced dataset. Yuan et al. (2021) proposed an ensemble approach for imbalanced classification that takes into account the presence of overlap in imbalanced data and create a new coefficient called Hard To Learn (HTL), designed to measure the importance of each training instance.

Zhao et al. (2021) we proposed a deep active semi-supervised learning framework, called DSAL (Deeply Supervised Active Learning), by combining active learning and semi-supervised learning strategies. Kim and Kim (2022) propose an active learning model based on an informative experience memory that is populated with meaningful retraining data by evaluating the uncertainty of the data. Dablain et al. (2022) concluded that an important advantage of DeepSMOTE over generative adversarial network (GAN)-based oversampling is that DeepSMOTE does not require a discriminator, and it can generate high-quality artificial images that are information-rich and is suitable for visual inspection. Mohammadian et al. (2020) addressed the digital transformation in academic society and innovative ecosystems in the world beyond Covid19-Pandemic by using the 7PS model and the 5th wave theory. Koziarski et al. (2020) propose a novel oversampling method, a Multi-Class Combined Cleaning and Resampling (MC-CCR) algorithm. Koziarski (2020) examined the possibility of using the concept of mutual class potential, by which to guide the oversampling process in RBO (radial-based oversampling) in the phase of undersampling.

3. Transfer Learning Classifier (TLC)

The proposed model is shown in Fig. 2, including real-time data augmentation module, DenseNet transfer learning module and dynamic sampling module. The transformed images of after training are generated by the real-time data augmentation module, the DenseNet transfer learning module is used to tune the model, and the dynamic sampling module can automatically generate new samples according to the performance of the contrast set, which based on F1-scores of the confusion matrix. The details of the proposed TLC will be introduced in Section 3.

3.1. Real-time data augmentation

Data augmentation can improve the generalization ability of the model (i.e., avoid over-fitting), and reduce the required number of data. Data augmentation, the artificial creation of training data for machine learning by transformations, is a widely studied research field across machine learning disciplines. Although it is important to increase the generalization capabilities of models, it will address other challenges and problems, from overcoming a limited number of training data, to regularizing the objective, to limiting the data used to protect privacy (Bayer et al., 2021). Complex data augmentation is a scheme which can artificially extend the data set by using domain specific synthesization to generate more training data (Taylor and Nitschke, 2018). Our model has two data augmentation modes: (1) off-line mode. Data augmentation is performed before training the model. (2) Online mode. The process is completed in real-time during each iteration of learning. In off-line mode, we need to recreate the dataset before training, and then the whole training process is in a static dataset. However, in online mode, we only augment a small number of images required for each iteration of training. And we obtain the batches of image data through real-time data augmentation, which is dynamic. Real-time data augmentation directly augments the input data for the model in the data space.

3.2. Transfer learning in standard DenseNet

Definition 1 (*Transfer Learning for Image Classification*)**.** In terms of transfer learning (TL for short), there are two elements: pre-trained source domain $D_S = \{Z, P(Z)\}$ and source task $T_S = \{U, f(\cdot)\}$ where Z represents the feature space, U represents the label space, $P(Z)$ is the marginal probability distribution and $f(\cdot)$ is the target function. The goal is to transfer the knowledge from the source domain D_S to the target domain D_T, and finally improve the learning ability of the target function $f_T(\cdot)$ in the target task T_T. It is worth noticing that $D_S \neq D_T$ or $T_S \neq T_T$ where T_S represents the image classification task,[2] and T_T represents the network data classification task. The task is defined with Eq. (2).

$$T = \{y, P(Y|X)\} \tag{2}$$

The condition $T_S \neq T_T$ implies that either $y_S \neq y_T$ or $P(Y_s|X_s) \neq P(Y_T|X_T)$. Therefore, the ultimate goal is to use the knowledge of image data to strengthen the prediction capability of the target function $f_T(\cdot)$.

In this study, we chose the standard DenseNet (Huang et al., 2018), of which high-level structure of standard DenseNet is shown in Fig. 3 for transfer learning. The proposed model is initially trained on the ImageNet dataset, and the weighting parameters is then fine-tuned.

[2] https://paperswithcode.com/dataset/imagenet.

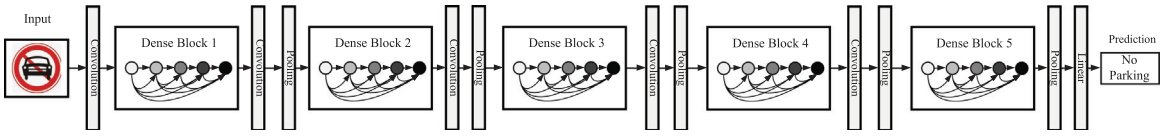


Fig. 3. High-level structure of the standard DenseNet.

Transfer learning does not randomly initialize the weights of different layers, instead, it uses a pre-trained network for initialization. When training TLC, we first train the last layer of DenseNet, which is the weighting parameter of the classification layer. The purpose is to fix the feature extractor and make the classifier more accurate. Then, we freeze the first two dense blocks, and train the latter two dense blocks and classifiers. This is because the first two dense blocks of DenseNet extract generalized features (including shape, contour, color, etc.) in ImageNet, while the latter two dense blocks extract more specific information, e.g., textures. After updating the weighting parameters of the latter two dense blocks and the classifier, TLC uses the generalized features of the transfer learning to learn. So, the proposed transfer learning training method is very efficient.

3.2.1. Active sampling

The training data used for transfer learning play an important role in adjusting the weighting parameters, and learning imbalanced classes may be very difficult. In this study, we propose an active sampling algorithm to solve this problem. We use the F1-scores on the contrast dataset to adjust the class distribution of the training samples, as shown in Algorithm 1, which indirectly affects the training process.

Algorithm 1 Active Sampling Algorithm

Input: The training set Y_t, a contrast set Y_c, an initial densenet $DenseNet$, and a class list C.
Output: The $DenseNet^1$ after active sampling.

1. $DenseNet^1_0 \leftarrow DenseNet$, $i \leftarrow 1$, $N^* \leftarrow |Y_t|/|C|$;
2. **for** *each class* $c_k \in C$ **do**
3. $N_{i,k} \leftarrow N^*$;
4. **end for**
5. **while** $NotFullTrained(DenseNet^1_{i-1})$ **do**
6. $Y_i \leftarrow ReSampling(Y_t, N_i)$;
7. $DenseNet^1_i \leftarrow Train(DenseNet^1_{i-1}, Y_i)$;
8. $F1_i \leftarrow UpdateF1(DenseNet^1_i, Y_c)$;
9. **for** *each class* $c_k \in C$ **do**
10. $N_{i+1,k} \leftarrow UpdateSize(F1_i, c_k)$;
11. **end for**
12. $i \leftarrow i + 1$;
13. **end while**
14. $DenseNet^1 \leftarrow DenseNet^1_{i-1}$;
15. **return** $DenseNet^1$;

The target model $DenseNet^1$ is obtained based on $DenseNet$, is trained by the training set Y_t, and performs active sampling based on the contrast set Y_c. All classes are in the list $C = \{c_k\}$. Algorithm 1 shows the detail of active sampling for training the model, where $|Y_t|$ represents the number of samples in the training set Y_t and $|C|$ represents the number of classes.

The working mechanism of Algorithm 1 is that: (1) we initialize the model and variables (line 1). (2) We initialize the number of each class in the batch data (lines 2–4). (3) We calculate the F1-score of each class, and then dynamically adjust the number of classes of the batch data in the next iteration based on the F1-score of each class. In the ith iteration, the image Y_i is sampled from Y_t, and the real-time data augmentation module in Section 3.1 augments the sample Y_i. As shown in Fig. 2, according to the F1-score of the confusion matrix from the previous iteration, the number of images in each class can be determined. We use the training samples to train the model and the updated model classifies these images in the contrast dataset Y_c (lines 5–13). Lastly, (4) we obtain and output a new model $DenseNet^1$ which can be used to accurately classify imbalanced data (lines 14–15).

It is worth noting that the images in the contrast set Y_c are different from images in the testing dataset, because they use images from different cameras or the same camera at different times. In Algorithm 1, we calculate the F1-score of class c_k in the ith iteration by Eq. (3).

$$f1_{i,k} = \frac{2 \cdot P_{i,k} \cdot R_{i,k}}{P_{i,k} + R_{i,k}} \tag{3}$$

where $P_{i,k}$ represents the precision and $R_{i,k}$ represents the recall of class c_k in the ith iteration. For all classes in the ith iteration, $F1_i = \{f1_{i,k}\}$ is the vector of F1-scores. Generally speaking, when the F1-score of a class is very high, it is easy to be recognized when compared to the classes with low F1-score in C. Therefore, improving the performance of classes with lower F1-score is important, and the samples from the class with low F1-score will be chosen in the next iteration. The total number of trained images keeps unchanged in each iteration. And, we calculate the number of images from class c_k in the next iteration by Eq. (4). For $\forall c_k$, the number of images is N^* initially, that is, $f1_{0,k} = N^*$.

$$UpdateSize(F1_i, c_k) = \frac{1 - f1_{i,k}}{\sum_{c_s \in C} 1 - f1_{i,s}} \times N^* \tag{4}$$

3.3. Efficient DenseNet (E-DenseNet)

3.3.1. Naive implementation

As shown in Fig. 4, in modern deep learning models, solid arrows of a computation graph represent layer operations. The nodes of computation graph represent intermediate results of feature maps stored in memory (Pleiss et al., 2017; Gao et al., 2019). As shown in Fig. 4(left), we show a simple computation graph of the DenseNet layer. The input of each layer is the output from the previous layer, that is, the feature map (color boxes). These feature maps of different layers are not stored consecutively in memory. The first operation (i.e., *Concatenation*) of the naive implementation aggregates them into a contiguous memory. Assuming the number of previous layers is k, and each layer produces m features, then contiguous memory should contain $k * m$ feature maps. Naive implementation performs batch normalization (represented by *Batch Norm*) on the above $k * m$ feature maps, and allocates a new $k * m$ continuous memory. At last, a convolution operation (*Convolution*) generates m new features according to the above *Batch Norm* operation. We find that the intermediate operations require a large amount of memory to reallocate, because contiguous memory is required to contain all the previous feature maps. However, the output feature of each layer only needs $O(m)$ memory.

In deep learning frameworks (e.g., PyTorch), more memory is allocated to store gradients in the phase of back propagation. As shown in Fig. 4(right), the computation graph operations of the gradients are represented by the inverted solid arrow. The normalized feature maps of the forward propagation (dotted arrow) and the gradients of the output are used to calculate the gradients of *Batch Norm* features. Storing these gradients requires additional memory allocation of $O(k * m)$ space.

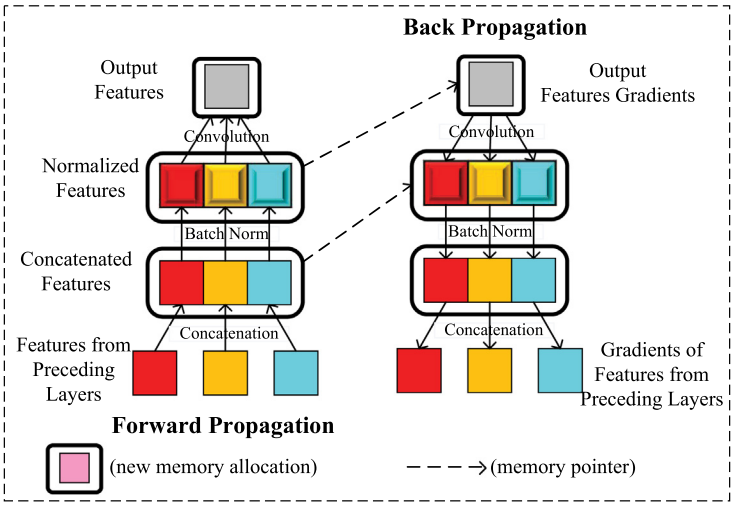


Fig. 4. Naive implementation of DenseNet.

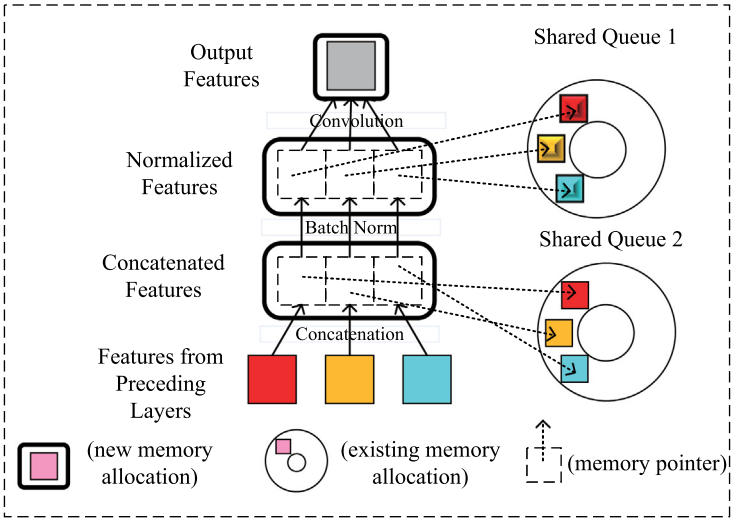


Fig. 5. Efficient implementation of DenseNet.

3.3.2. Efficient implementation

To solve the memory reallocation problem, we propose two circular sequential queues as pre-allocated shared memory space to avoid rapid memory growth, as shown in Fig. 5. In the phase of forward propagating, we put all intermediate output results into these two shared queues. In the phase of back propagating, we can recompute concatenated features and normalize features as needed in a real-time fashion, because the concatenation and normalized operations are economical to achieve.

Much work has been done to explore the recomputation schemes. Chen et al. (2016) trained 1000 layers of ResNet by the re-computation scheme of a single GPU, and they developed recomputation in MxNet (Chen et al., 2015) to support arbitrary networks. Generally speaking, recomputation of intermediate results takes memory into consideration. Pleiss et al. (2017) discovered that the recomputation scheme is very practical for DenseNet. *Concatenation* and *Batch Norm* operations consume costly memories but require very little computation time. Therefore, recomputation of intermediate results can save a lot of memories with few time costs.

Shared Queue for Concatenation. Feature concatenation operation requires $O(k * m)$ of memory storage, where m represents the number of features added in per layer, and k represents the number of previous layers. Instead of reallocating new memory for all concatenation operations, we put the output results of concatenation into a pre-allocated shared queue ("Shared Queue 1"). As shown in Fig. 5, the concatenation feature maps point to the shared queue. It takes much less time to copy a feature map to pre-allocated memory than the event that it does reallocate new memory, so shared memory makes concatenation more efficient. Then, it reads these feature maps directly from "Shared Queue 1" as input to the *Batch Norm* operation. The data of "Shared Queue 1" are temporary, because they are used by all network layers, which can be effectively recalculated when concatenation features are used in the phase of back propagation. We use a simple circular queue as the storage structure, because it is sequential and can effectively use memory resources, in order to overcome the "false overflow" problem.

Shared Queue for Batch Norm. In the similar way, the *Batch Norm* operation also requires $O(k * m)$ memory, so the *Batch Norm* results are copied to the "Shared Queue 2". Since the data in "Shared Queue 2" are temporary and will be covered by the next layer, we must recompute the results of the *Batch Norm* in the phase of back propagation. *Batch Norm* only includes scalar multiplication and addition operations, so

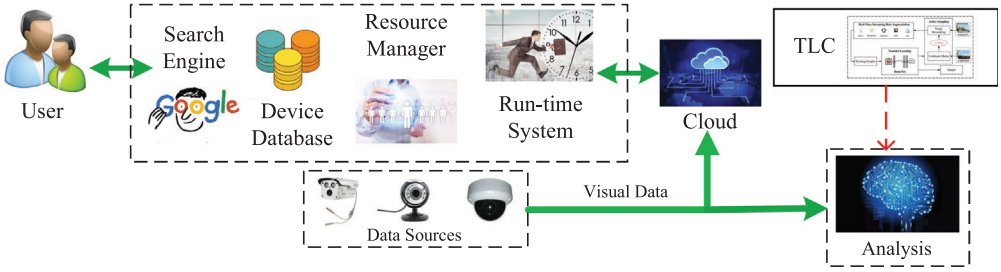


Fig. 6. Working mechanism of the applied end-to-end system.

the cost of recomputing is economical. Similarly, the shared queue can effectively use memory resources.

3.4. End-to-end system

We applied an end-to-end real-time data processing system (i.e., user-to-analysis system). This system can utilize the real-time visual data captured by a large number of network mobile devices. The system architecture is shown in Fig. 6, and the main components include: (1) web user interface. A user inputs the location of the network devices, and then finds the specific information of the network devices, and analyzes the data of these devices. (2) Device database. This device database contains real-time data from many industries, including banks, governments, and schools. (3) Run-time environment. The system can retrieve real-time visual data, and then analyze these real-time data. (4) Resource manager. The system utilizes the cloud to retrieve and analyze the real-time data, which also employ the cloud to store the real-time data. (5) Analyzer. This module analyzes the real-world image/video data, and the proposed TLC model can be applied to the analysis module of this system.

Although many network devices can be accessed publicly, it is difficult to find a specific device among different brands or types. Data of different network devices are collected on network servers, and there are many different ways to organize data streams. The process of retrieving data from different network devices depends on the device type. Some devices have built-in servers to allocate and send data, and each device has an independent and unique IP address. Users can directly connect to the device through its IP address, and exquisite data by using the HTTP GET requests. For many government agencies, although network devices are deployed and data are available to the public, accessing these data must pass through the specified servers of these agencies, rather than being directly sent to the device. The proposed system has the same programming interface to analyze the data from different devices, which can effectively process and analyze the real-time data.

4. Experiments and discussions

In this section, we first introduce the imbalanced data in experiments and the experimental setting, and then discuss the experimental results on the imbalanced datasets.

4.1. Imbalanced datasets

In order to evaluate the performance of the proposed TLC model in different imbalanced scenarios, we choose the most commonly-used CIFAR10 and CIFAR100[3] balanced datasets. The CIFAR10 and CIFAR100 datasets include 50,000 training images and 10,000 testing images with 10 and 100 classes, respectively. And then, we randomly delete samples in each class from these two datasets to construct imbalanced datasets. We only consider the long-tailed distribution scenario, that is, the training set follows an exponential decay distribution in the number of samples in each class (Kim et al., 2021). For the proposed E-DenseNet introduced in Section 3.3.2, we use ImageNet dataset to pre-train, and then input it into the TLC model for transfer learning. It is worthwhile to notice that we use N_i/N_K to represent the imbalance ratio, where N_i represents the number of training samples from class i, K represents the index of the class with the largest number of samples, and the class index is arranged in a descending order according to the number of samples. We express the maximum imbalance ratio of CIFAR10 and CIFAR100 with N_1/N_K, and specify two imbalance scenarios: $N_1/N_K = 10$ and $N_1/N_K = 100$. In addition, we also select the originally imbalanced dataset Caltech101[4] and HAM10000[5] ("Human Against Machine with 10 000 training images") (Tschandl et al., 2018). The Caltech101 dataset is composed of 9146 images, the images of objects belong to 101 classes, and there is around 40 to 800 images in each class where the maximum imbalance ratio is 25.8. Moreover, the HAM10000 dataset contains images of population in two different sites during twenty years, i.e., the Department of Dermatology of the Medical University of Vienna, and the skin cancer practice of Cliff Rosendahl in Queensland. This dataset includes 10,015 RGB color images and 7 classes (i.e., akiec, bcc, bkl, df, mel, nv, vasc), as shown in Fig. 7(f), where the maximum imbalance ratio is 58.3.

4.2. Experimental setup

Baselines:

(1) softmax cross entropy (CE). This loss function is the classic multi-class loss function. The calculation of gradients of multi-class cross entropy loss is very simple, and the loss is only related to the probability of the correct class, but the class imbalance problem is not considered.

(2) softmax cross entropy and supplementary entropy (COT) (Chen et al., 2019). This loss function iteratively updates neural network parameters by alternating between the primary objective and the complement objective.

(3) focal loss (FL) (Lin et al., 2017). This loss function addresses this class imbalance by reshaping the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples, and focuses on training a sparse set of hard examples.

(4) DeepSMOTE (Dablain et al., 2022). A novel oversampling algorithm for deep learning models based on the highly popular SMOTE method. DeepSMOTE bridges the advantages of metric-based resampling approaches that use data characteristics to leverage their performance, with a deep architecture capable of working with complex and high-dimensional data.

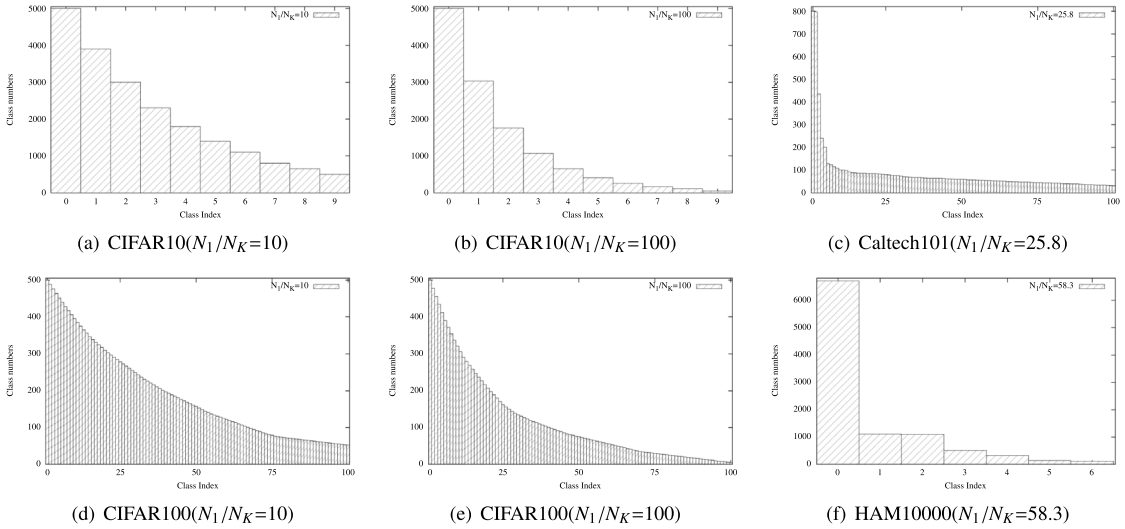
(5) MC-CCR (Koziarski et al., 2020). MC-CCR utilizes an energy-based approach to modeling the regions suitable for oversampling, less affected by small disjuncts and outliers than SMOTE.

(6) MC-RBO (Koziarski, 2020). MC-RBO is a radial-based oversampling method for imbalanced data.

[3] http://www.cs.toronto.edu/kriz/cifar.html.

[4] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[5] https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000.



(a) CIFAR10(N_1/N_K=10) (b) CIFAR10(N_1/N_K=100) (c) Caltech101(N_1/N_K=25.8)

(d) CIFAR100(N_1/N_K=10) (e) CIFAR100(N_1/N_K=100) (f) HAM10000(N_1/N_K=58.3)

Fig. 7. Distributions of classes w.r.t. training samples.

Table 1
Baseline loss functions.

Baseline	Calculation method	Introduction
CE	$-\sum_i y_i^* log(y_i)$	y_i^* is the ground truth value, y_i is the predicted value
COT	$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1,j\neq g}^{K}(\frac{y_{i,j}}{1-y_{i,g}})log(\frac{y_{i,j}}{1-y_{i,g}})$	N is the total number of samples, K is total number of classes, $y_{i,j}$ is the ith sample which is predicted in the jth class, $y_{i,g}$ is the ith sample which is predicted in the gth class
FL	$-\sum_i \alpha_i y_i^{*\gamma} log(y_i)$	y_i^* is the ground truth value, y_i is the predicted value, α_i is a balanced factor which need to be learned, γ is a hyperparameter

We select some powerful loss functions, which can be simply applied to the deep neural network model without additional learning procedures or hyperparametric adjustment. We combine the above three loss functions with different models (e.g., DenseNet Huang et al., 2018; Gao et al., 2019, ResNet He et al., 2016, and etc.), to obtain different baselines.

The details of these three loss functions are shown in Table 1.

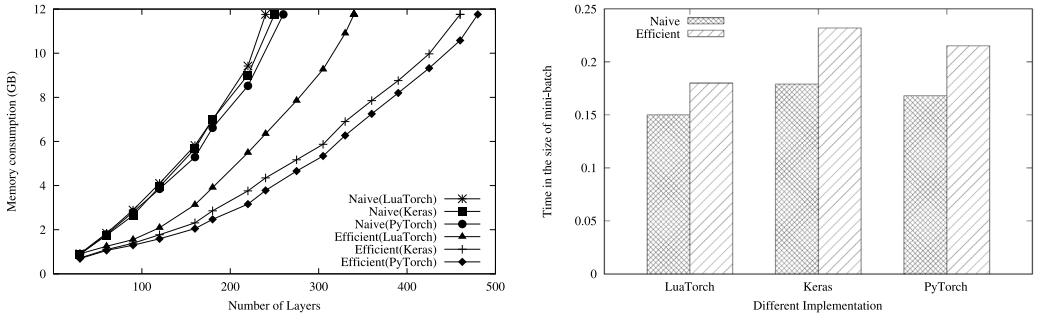
Model training. We use 80% of the data in the datasets for training, 20% of the data for validating. The training set strictly divides the samples in each class according to the imbalance ratio of the long-tailed distribution. We train the standard DenseNet and E-DenseNet on the CIFAR10 balanced dataset and observe their memory consumption and the number of parameters, respectively. We set their growth rate (the number of filters in each convolutional layer) to 12 and the batch size to 64. The standard DenseNet and E-DenseNet are implemented with LuaTorch,[6] Keras[7] and Pytorch[8] (Gao et al., 2019), respectively. We pre-train E-DenseNet on the ImageNet dataset, and then put the pre-trained E-DenseNet and standard DenseNet into the transfer learning module of TLC, respectively. When TLC performs transfer learning (i.e., transferring the pre-trained model to the target dataset), the weights of all convolutional layers are frozen, and the weights of the classifier are randomly initialized. We warm up the pre-trained model by only training the classifier, because the convolutional layer retains the weights from the original dataset, and the significant part of these weights can be reused. We set the warm-up epochs to 10, the batch size to 128, and the optimizer to Adam (Kingma and Ba, 2015). When the step of warm-up is completed, we freeze the shallow dense blocks by only training the deep layer parameters, because the deep layers can learn the knowledge from the target dataset. Then, we start to train the complete TLC model and the number of epochs is set to 200, but the batch size is not directly specified. The batch size is determined by the hyperparameter $\mathcal{N}$, that is, the initial batch size is $\mathcal{N}*C$ where $\mathcal{N}$ is the number of samples of each class in each batch and C is the number of classes. Subsequently, TLC will execute the active sampling algorithm according to the initial setting, by dynamically changing the number of samples from each class in each batch according to F1-score. We choose the SGD (Stochastic Gradient Descent) (Lei and Tang, 2021) optimizer with a learning rate of 0.0006, and the momentum is 0.9. The experiments are conducted in the Ubuntu 20.04 operation system in the hardware environment of NVIDIA GeForce RTX 3080 Ti, CPU memory with 32 GB.

4.3. Evaluation of the proposed E-DenseNet

We use the aforementioned LuaTorch, Keras and PyTorch to implement standard DenseNet and E-DenseNet, and compare their performance in different frameworks. The naive implementation of DenseNet allocates memory for each *Concatenation* and *Batch Norm* operation, and we compare it with E-DenseNet. The tensors stored in memory in the phase of training are feature maps and network parameters.

As shown in Fig. 8(a), we observe that the Naive implementations of these three frameworks all take up memory very quickly, and then training a network with more than 220 layers requires more than 12 GB of memory. However, E-DenseNet employs the memory shared queues that can significantly reduce the memory consumption. In the LuaTorch implementation, the Naive implementation of 220-layer model uses about 6 GB of memory. Under the same memory cost (12 GB), E-DenseNet can train about 340-layer models. Considering the PyTorch-based E-DenseNet, we can train a 500-layer network on a single GPU. According to Fig. 8(a), we can also see that the efficiency of LuaTorch based on the Lua language is lower than that of Keras and PyTorch based on the Python language. In addition, we can see that Pytorch

[6] http://github.com/torch/torch7/.
[7] https://github.com/keras-team/keras.
[8] http://github.com/pytorch/pytorch/.



(a) GPU memory consumption (b) Computation time in the size of mini-batch

Fig. 8. Comparison of memory and time consumption.

Table 2
Flops and MACs (MAdds) for the end-to-end system.

Model	Flops	MACs (MAdds)
End-to-end system (No Aug.)	9921	19,836
End-to-end system (No Warm-up.)	9824	19,681
End-to-end system (No Samp.)	9856	19,710
End-to-end system (No Samp. and No Aug.)	10,235	20,440
End-to-end system	**9601**	**19,235**

Table 3
Classification accuracy (%) comparison of different baselines on the imbalanced CIFAR dataset.

Dataset	CIFAR10		CIFAR100	
N_1/N_K	10	100	10	100
DenseNet121+CE	83.73	67.13	58.11	40.73
DenseNet121+FL	82.16	66.19	61.76	41.67
DenseNet121+COT	84.33	**69.37**	59.91	42.11
TLC+CE	83.81	67.31	61.12	42.47
TLC+FL	85.15	**68.82**	61.74	43.35
TLC+COT	**85.96**	68.46	**62.05**	**43.79**

is more efficient than Keras. It is worth noting that the total memory consumption of E-DenseNet will not increase linearly with the network depth, because the number of parameters has a quadratic function relationship with the network depth. In addition, as shown in Fig. 8(b), we present the time consumption in the size of mini-batch for the 100-layer standard DenseNet and E-DenseNet, respectively. The time cost of E-DenseNet using LuaTorch, Keras and PyTorch is increased by 20.0%, 29.6% and 27.9%, respectively. This additional cost is due to recalculating the intermediate feature maps in the phase of back propagation. If the GPU memory is limited, the time consumption of shared storage is a reasonable trade-off.

4.4. Real-time performance and computational complexity of end-to-end system

In the experiment, FLOPs (Floating Point Operations) measure the number of operations for forward propagation in the network. The smaller the FLOPs are, the faster the computing speeds. Typically, the value of MACs (multiply-accumulate operations) is twice the value of the FLOPs. The number of floating-point operations and multiply-add cumulative are used to evaluate the performance of the end-to-end system. We perform ablation experiments to evaluate the performance of the end-to-end system, with four comparison groups. The experimental results are shown in Table 2, where end-to-end (No Aug.) denotes that we do not use data augmentation, end-to-end (No Warm-up.) implies that no warm-up is performed in the phase of transfer learning, end-to-end (No Samp.) means that we do not use active sampling, and end-to-end (No Samp. and No Aug.) denotes that we do not use active sampling as well as data augmentation.

In the ablation experiment, in the case without using data augmentation, FLOPs reaches 9921, which is 3.33% higher than end-to-end system with data augmentation and active sampling, indicating that the end-to-end system has the fastest computational speed and the fastest forward propagation in the DenseNet network. The computational performance of the end-to-end system without warm-up in the phase of transfer learning drops, and the FLOPs value reaches 9824.

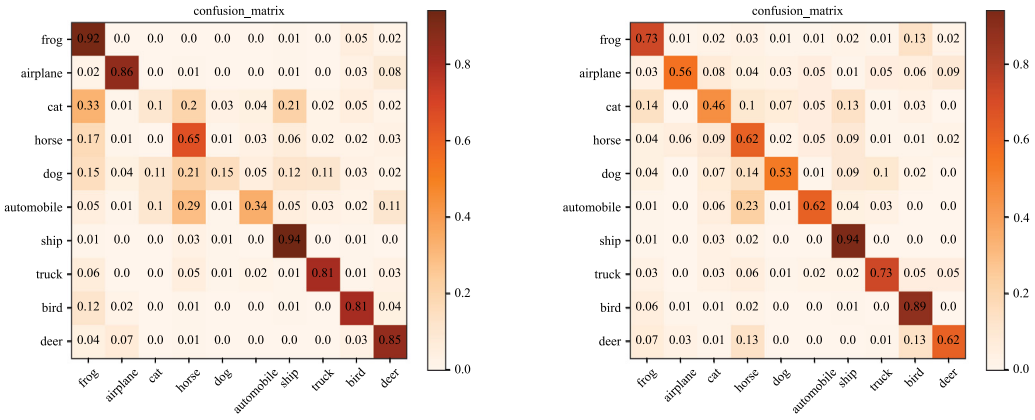
When we do not use active sampling, the FLOPs of the end-to-end system is the largest among the first three models, and we find that the active sampling algorithm has the biggest impact on the runtime performance of the end-to-end system. The comparison group without the active sampling algorithm and data augmentation has the highest FLOPs value, indicating that the computational speed of our end-to-end system is slow when the active sampling algorithm and data augmentation are not used. Therefore, the proposed end-to-end system runs faster by applying the data augmentation and active sampling algorithms.

4.5. Evaluation of TLC on different imbalanced datasets

Result on Imbalanced CIFAR10 and CIFAR100. CIFAR10 and CIFAR100 contain RGB images of real-world things (32*32 pixels). Since the CIFAR dataset is balanced, we construct imbalanced datasets as shown in Figs. 7(a), 7(b), 7(d) and 7(e). We combine CE, FL and COT with DenseNet121, respectively (e.g., DenseNet121+CE, DenseNet121+FL, and DenseNet121+COT.), and then compare them with the proposed TLC model. The results are shown in Table 3. Note that DenseNet121 refers to the DenseNet with 121 layers of convolution, which is viewed as a standard DenseNet. Differently, TLC uses E-DenseNet as the transfer learning module.

We observe that when the imbalance ratio is 10, the best performance is obtained by the TLC+COT model on the CIFAR10 and CIFAR100 datasets, and the accuracy of the testing set reaches 84.96% and 62.05%, respectively. This is because the proposed TLC model not only has the feature extraction capability of DenseNet, but also can augment the images of the dataset to enhance its generalization capability, and the active sampling algorithm can change the distribution of each batch w.r.t. the training data. At the same time, TLC+COT performs best on the CIFAR100 dataset as well, because the proposed model can learn the decision boundaries of imbalanced data in a balanced manner. In particular, on the CIFAR10 dataset, when the imbalance ratio reaches 100, the overall accuracy of TLC in the testing set is lower than that of the standard DenseNet121. This is because the standard DenseNet121 is inclined to learn the majority class. The prediction of the majority class can even reach 0.94, but the prediction accuracy of the minority class is only 0.10.

In Table 4, we compare the classification accuracy of DenseNet121+CE with the TLC model and the state-of-the-art models on the CIFAR-10 dataset. We can observe that when the imbalance ratio of the data is 100, our model wins other models when the



(a) DenseNet121+CE on CIFAR10 (N_1/N_K=100) (b) TLC+CE on CIFAR10 (N_1/N_K=100)

Fig. 9. Confusion matrix of CIFAR10 (N_1/N_K = 100).

Table 4
Classification accuracy (%) of each class on the imbalanced CIFAR-10 dataset.

Class	Ship	Frog	Airplane	Deer	Truck	Bird	Horse	Auto	Dog	Cat
Number of samples	5000	3237	2096	1357	878	568	368	238	154	100
DenseNet121+CE	94.03	**92.16**	86.07	**85.45**	**81.22**	81.61	65.34	34.89	15.67	10.71
TLC+CE	94.72	73.15	56.03	82.21	73.84	**89.11**	62.34	**62.46**	53.13	46.54
TLC+FL	92.21	74.25	80.35	76.65	72.75	66.43	**68.70**	43.85	**61.90**	51.15
TLC+COT	**94.80**	71.21	81.13	74.80	69.85	72.20	66.42	59.73	59.52	**52.90**
DeepSMOTE (Dablain et al., 2022)	87.53	71.15	73.23	69.19	67.83	67.20	64.24	57.32	43.95	51.85
MC-CCR (Koziarski et al., 2020)	89.62	81.33	**87.73**	72.36	61.46	56.26	63.78	55.12	59.04	46.85
MC-RBO (Koziarski, 2020)	91.72	82.83	81.98	72.31	70.34	73.06	62.04	46.77	54.84	42.43

number of samples is small. Although DenseNet121+CE performs well when the number of samples is large, it performs poorly when the number of samples is small. When the number of dog samples is 154, the classification accuracy of DenseNet121+CE is only 10.71%. Because DenseNet121+CE cannot enhance its generalization ability by augmenting the images. Among these three state-of-the-art models, MC-RBO performs the best, with an average accuracy of 67.83% on the testing set. It outperforms DenseNet121+CE on the minority samples, but it does not perform as well as DenseNet121+CE and the TLC model on the majority samples. Because MC-RBO focuses on the classification accuracy of the minority samples, thus sacrificing the accuracy of majority sample classification. The model with the best performance is TLC+FL, which achieves an average accuracy of 68.82%. TLC performs well in the classification accuracy of the minority as well as the majority classes, because it uses an active sampling algorithm, which continuously balances the classification of the minority and the majority samples, thereby enhancing its generalization ability. Additionally, the active sampling algorithm can change the distribution of each batch of data. Because TLC combines the active sampling and data augmentation algorithms, it shows high classification accuracy performance, especially for the minority samples.

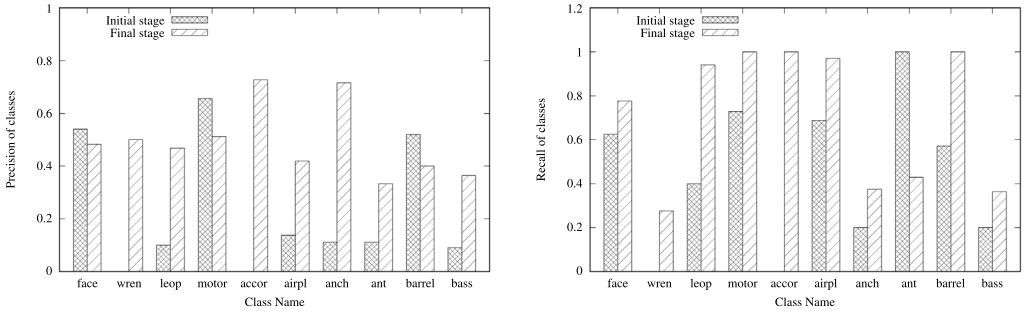
In order to further demonstrate the effectiveness of the proposed algorithm, we present a confusion matrix with an imbalance ratio of 100 on CIFAR10 for the original DenseNet+CE and TLC+CE, as shown in Figs. 9(b) and 9(a). We observe these two confusion matrices and find that the largest value of the main diagonal of Fig. 9(a) is 0.94 and the smallest is 0.10, but the value of the main diagonal of Fig. 9(b) is relatively uniform. It is interesting to find that the average values of the main diagonals in these two confusion matrices are almost the same, indicating that the proposed TLC model based on active sampling can guarantee the classification accuracy as well as the average accuracy of each class.

Furthermore, we conduct experiments on the real imbalanced dataset Caltech101 and HAM10000. The results and discussions are given as follows.

Results on Caltech101 dataset. Results on Caltech101 dataset. In the first group of comparison experiments, we compares five state-of-the-art models for dealing with imbalanced data. ResNet18 and DenseNet121 cannot overlearn most classes, so these two models performed poorly in the experiment. Among the state-of-the-art models, the performance of DeepSMOTE is relatively poor, and MC-CCR (Koziarski et al., 2020) and MC-RBO (Koziarski, 2020) can handle the spatial properties of data with advanced instance generation modules, so they outperform the DeepSMOTE model. In addition, they outperform ResNet18 and DenseNet121. The performance of these three models combined with three different loss functions is worse than that of the TLC model, because the active sampling algorithm can balance the classification accuracy of the majority and minority samples, and data augmentation can enhance the generalization capability of the model, TLC obtains the best prediction accuracy on the testing set.

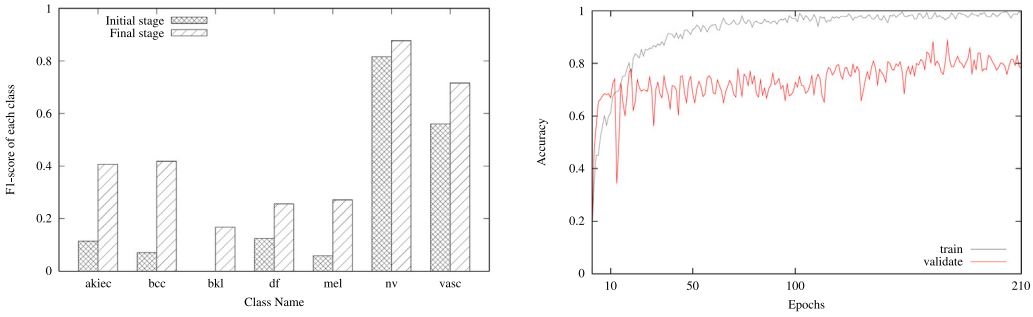
In the second group of experiments, after the warm-up phase, we freeze the first three dense blocks, because we need to use the generalized information of the shallow network, and the specifically deeper information needs to be relearned. The experimental results are shown in Table 5, where TLC (No Aug.) implies that we do not use data augmentation, TLC (No Warm-up.) denotes that no warm-up is performed during transfer learning, TLC (No Samp.) means that we do not use active sampling, and TLC (No Samp. and No Aug.) denotes that we do not use active sampling and data augmentation methods.

In the ablation experiment, we do not use data augmentation or active sampling. The drop of prediction accuracy indicates that data augmentation and active sampling can improve the accuracy of the TLC model. In addition, we observe that in the case of no active sampling (other conditions remain unchanged). The prediction accuracies of TLC (No Samp.) with three loss functions are 82.24% (CE), 83.95% (FL), 84.87% (COT), which are all lower than the case without warm-up or data augmentation, indicating that active sampling has an important impact on improving the classification accuracy of the TLC model. If we do not use data augmentation and active sampling, the classification accuracy of the TLC model is lower than other TLC methods. The proposed TLC model is combined with three different loss functions,



(a) Comparison of the initial and finished precision (b) Comparison of the initial and final recall

Fig. 10. Comparison of classification precision and recall w.r.t classes on the Caltech101 data.



(a) F1-score of each class (b) Classification accuracy

Fig. 11. F1-score and classification accuracy on the HAM10000 data.

Table 5
Classification accuracy (%) comparison of different models on the Caltech101 data.

Model	CE	FL	COT
ResNet18 (He et al., 2016)	79.63	81.45	82.05
DenseNet121 (Huang et al., 2018)	80.13	81.11	82.15
DeepSMOTE (Dablain et al., 2022)	81.22	82.41	82.65
MC-CCR (Koziarski et al., 2020)	82.13	83.69	83.45
MC-RBO (Koziarski, 2020)	82.94	83.77	84.03
TLC (No Aug.)	83.75	84.01	85.74
TLC (No Warm-up.)	83.12	84.37	85.57
TLC (No Samp.)	82.24	83.95	84.87
TLC (No Samp. and No Aug.)	81.21	83.13	84.01
TLC	**84.45**	**85.57**	**87.96**

and we find that different models with the COT loss function performs the best, and the accuracy on the testing set reaches 87.96%. Because data augmentation can make the model generic, and the warm-up operation can make the model be adaptive to the target data before training.

Furthermore, we randomly select 10 classes to observe the changes in their precision and recall in the initial and final stages of the experiment, as shown in Fig. 10. We found that the precision of face, motor and barrel has decreased slightly, while the precision of other classes is greatly improved, as shown in Fig. 10(a). In particular, the initial precision of wren and accor is 0. Through TLC learning, the final precision is increased by 50.00% and 72.72%, respectively. In terms of the recall, the recall on the ant data is dropped by 57.15%, while the recall of other classes is improved, as shown in Fig. 10(b). In particular, the recall on the accor data is improves by 100%. By comparing Figs. 10(a) and 10(b), we find that TLC improves the recall more than the precision.

Results on HAM10000 dataset. After the warm-up phase, we freeze the first two dense blocks. Because the HAM10000 dataset is quite different from the ImageNet dataset, we only need the color, shading and other commonly-used information that the shallow network has extracted. For specific information such as texture, the deep layers of the network need to relearn them.

As shown in Fig. 11(a), the initial F1-score of akiec, bkl, bcc, df, and mel is low, even reaching 0, indicating that these classes cannot be well learned. After TLC training, the F1-score of each class has been significantly improved, up to 0.34. Because the active sampling algorithm can dynamically adjust the number of each class in each batch of training data according to the F1-score, the model can take into account the learning of minority classes. At the same time, data augmentation improves the robustness of the model. In addition, we give the accuracy curve, as shown in Fig. 11(b). We set the warm-up to 10 epochs, and train the classifier in advance, and set it to 200 epochs in the phase of training. We find that the classification accuracy on the training set converges to 98.0%–99.0% at the 100th epochs, while the accuracy of the validation set converge to 78.0%–79.0% at the 200th epochs. For the training set, although the accuracy is high, the learning of minority classes is not sufficient. For the validation set, because the number of each class in each batch of data is dynamically adjusted according to the previous F1-score, the number of majority classes in each batch may decrease while the minority classes will increase in the next iteration of learning. However, the minority class may not be learned well in a short period of time. The number of majority classes in each batch is reduced and the original accuracy of the majority class will also be affected by the minority class, so the convergence speed is slow.

5. Conclusion and future work

In this paper, we propose an imbalanced data classification model based on transfer learning, including data augmentation module, transfer learning module, and active sampling module, which can effectively

process the static data as well as the real-time data, and cope with the problem of class imbalance. The proposed active sampling can iteratively adjust the class distribution of each batch of data according to the F1-score w.r.t. each class, in order to make the model learn the balanced knowledge. Ablation experiments show that the active sampling has a great impact on improving the accuracy of TLC model. In addition, to handle the problem of low utilization of real-time data, we design an end-to-end analysis system, and the proposed model can be directly applied in the analysis module of the system.

In terms of our future work, we will design new loss functions that can be used for imbalanced learning, e.g., by differently weighting the loss of each sample, and propose more advanced model architecture to deal with the classification of the imbalanced data.

CRediT authorship contribution statement

Yang Liu: Conceptualization, Methodology, Writing & revising. **Guoping Yang:** Conceptualization, Methodology, Writing – original draft. **Shaojie Qiao:** Supervision, Validation, Writing – review & editing. **Meiqi Liu:** Methodology, Investigation, Writing – review. **Lulu Qu:** Writing – original draft, Investigation. **Nan Han:** Investigation, Methodology, Writing – review. **Tao Wu:** Formal analysis, Investigation. **Guan Yuan:** Supervision, Validation. **Tao Wu:** Formal analysis. **Yuzhong Peng:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China under grants 62272066, 61962006; Sichuan Science and Technology Program under grants 2021JDJQ0021, 2022YFG0186; Planning Foundation for Humanities and Social Sciences of Ministry of Education of China under grant 22YJAZH088; Chengdu Technology Innovation and Research and Development Project under grant 2021-YF05-00491-SN, 2021-YF05-02413-GX, 2021-YF05-02414-GX; Chengdu Major Science and Technology Innovation Project under grant 2021-YF08-00156-GX; Chengdu "Take the Lead" Science and Technology Project under grants 2022-JB00-00002-GX, 2021-JB00-00025-GX; Chengdu Soft Science Research Project under grants 2021-RK00-00065-ZF, 2021-RK00-00066-ZF; Digital Media Art, Key Laboratory of Sichuan Province, Sichuan Conservatory of Music, Chengdu, China under grant 21DMAKL; The 54th Research Institute of China Electronics Technology Group Corporation-University Cooperation Project under grant SKX212010057; Si Chuan Network Culture Research Center Project under grant WLWH22-1; Science and Technology Innovation Capability Improvement Project of Chengdu University of Information Technology under grant KYTD202222; Sichuan Social Science High-level Team Project under grant 2015Z177.

References

Bayer, M., Kaufhold, M., Reuter, C., 2021. A survey on data augmentation for text classification. CoRR, abs/2107.03158 arXiv:2107.03158.

Cao, K., Wei, C., Gaidon, A., Aréchiga, N., Ma, T., 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In: Proceedings of Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, , Vancouver, BC, Canada, December 8-14. pp. 1565–1576.

Cao, P., Zhao, D., Zaïane, O.R., 2013. An optimized cost-sensitive SVM for imbalanced data learning. In: Proceedings of Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17. In: Lecture Notes in Computer Science, vol. 7819, Springer, pp. 280–292.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artificial Intelligence Res. 16, 321–357.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z., 2015. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. CoRR, abs/1512.01274 arXiv:1512.01274.

Chen, H., Wang, P., Liu, C., Chang, S., Pan, J., Chen, Y., Wei, W., Juan, D., 2019. Complement objective training. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, la, USA, May 6-9, 2019. OpenReview.net.

Chen, T., Xu, B., Zhang, C., Guestrin, C., 2016. Training deep nets with sublinear memory cost. CoRR, abs/1604.06174 arXiv:1604.06174.

Dablain, D., Krawczyk, B., Chawla, N.V., 2022. DeepSMOTE: Fusing deep learning and SMOTE for imbalanced data. IEEE Trans. Neural Netw. Learn. Syst. 1–15.

Downey, A.B., 2001. Evidence for long-tailed distributions in the internet. In: Paxson, V. (Ed.), Proceedings of the 1st ACM SIGCOMM Internet Measurement Workshop, IMW 2001, San Francisco, California, USA, November 1-2. ACM, pp. 229–241.

Gao, X., He, Y., Zhang, M., Diao, X., Jing, X., Ren, B., Ji, W., 2021. A multiclass classification using one-versus-all approach with the differential partition sampling ensemble. Eng. Appl. Artif. Intell. 97, 104034.

Gao, Huang, Zhuang, Liu, Geoff, Pleiss, Laurens, Van, Der, Maaten, 2019. Convolutional networks with dense connectivity. IEEE Trans. Pattern Anal. Mach. Intell..

Hamidzadeh, J., Kashefi, N., Moradi, M., 2020. Combined weighted multi-objective optimizer for instance reduction in two-class imbalanced data problem. Eng. Appl. Artif. Intell. 90, 103500.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30. IEEE Computer Society, pp. 770–778.

Huang, G., Liu, S., van der Maaten, L., Weinberger, K.Q., 2018. CondenseNet: An efficient DenseNet using learned group convolutions. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. Computer Vision Foundation/IEEE Computer Society, pp. 2752–2761.

Khan, S.H., Hayat, M., Bennamoun, M., Sohel, F.A., Togneri, R., 2018. Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans. Neural Netw. Learn. Syst. 29 (8), 3573–3587.

Kim, Y.-J., Kim, W.-T., 2022. Uncertainty assessment-based active learning for reliable fire detection systems. IEEE Access 10, 74722–74732.

Kim, Y., Lee, Y., Jeon, M., 2021. Imbalanced image classification with complement cross entropy. Pattern Recognit. Lett. 151, 33–40.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Koziarski, M., 2020. Radial-based undersampling for imbalanced data classification. Pattern Recognit. 102, 107262.

Koziarski, M., Woźniak, M., Krawczyk, B., 2020. Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise. Knowl.-Based Syst. 204, 106223.

Lei, Y., Tang, K., 2021. Learning rates for stochastic gradient descent with nonconvex objectives. IEEE Trans. Pattern Anal. Mach. Intell. 43 (12), 4505–4511.

Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. IEEE Computer Society, pp. 2999–3007.

Mohammadian, H.D., Shahhoseini, H., Castro, M., Merk, R., 2020. Digital transformation in academic society and innovative ecosystems in the world beyond covid19-pandemic with using 7PS model for IoT. In: Proceedins of 2020 IEEE Learning with MOOCS (LWMOOCS), Antigua Guatemala, Guatemala. pp. 112–117.

Mullick, S.S., Datta, S., Das, S., 2019. Generative adversarial minority oversampling. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27-November 2. IEEE, pp. 1695–1704.

Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., Weinberger, K.Q., 2017. Memory-efficient implementation of DenseNets. CoRR, abs/1707.06990 arXiv:1707.06990.

Pouyanfar, S., Tao, Y., Mohan, A., Tian, H., Kaseb, A.S., Gauen, K., Dailey, R., Aghajanzadeh, S., Lu, Y., Chen, S., Shyu, M., 2018. Dynamic sampling in convolutional neural networks for imbalanced data classification. In: Proceedings of IEEE 1st Conference on Multimedia Information Processing and Retrieval, MIPR 2018, Miami, FL, USA, April 10-12. IEEE, pp. 112–117.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D., 2019. Meta-Weight-Net: Learning an explicit mapping for sample weighting. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, December 8-14. pp. 1917–1928.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30. IEEE Computer Society, pp. 2818–2826.

Tahvili, S., Hatvani, L., Ramentol, E., Pimentel, R., Afzal, W., Herrera, F., 2020. A novel methodology to classify test cases using natural language processing and imbalanced learning. Eng. Appl. Artif. Intell. 95, 103878.

Taylor, L., Nitschke, G., 2018. Improving deep learning with generic data augmentation. In: Proceedins of 2018 IEEE Symposium Series on Computational Intelligence, Bangalore, India, November 18-21. pp. 1542–1547.

Tschandl, P., Rosendahl, C., Kittler, H., 2018. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 1–9.

Yan, Y., Chen, M., Shyu, M., Chen, S., 2015. Deep learning for imbalanced multimedia data classification. In: Proceedings of 2015 IEEE International Symposium on Multimedia, ISM 2015, Miami, FL, USA, December 14-16. IEEE Computer Society, pp. 483–488.

Yuan, B., Zhang, Z., Luo, X., Yu, Y., Zou, X., Zou, X., 2021. OIS-RF: A novel overlap and imbalance sensitive random forest. Eng. Appl. Artif. Intell. 104, 104355.

Zhao, Z., Zeng, Z., Xu, K., Chen, C., Guan, C., 2021. DSAL: Deeply supervised active learning from strong and weak labelers for biomedical image segmentation. IEEE J. Biomed. Health Inf. 25 (10), 3744–3751.