PP-ShiTu: A Practical Lightweight Image Recognition System

**Shengyu Wei, Ruoyu Guo, Cheng Cui, Bin Lu, Shuilong Dong, Tingquan Gao, Yuning Du,
Ying Zhou, Xueying Lyu, Qiwen Liu, Xiaoguang Hu, Dianhai Yu, Yanjun Ma**

Baidu Inc.
{weishengyu, zhouying15}@baidu.com

Abstract

In recent years, image recognition applications have developed rapidly. A large number of studies and techniques have emerged in different fields, such as face recognition, pedestrian and vehicle re-identification, landmark retrieval, and product recognition. In this paper, we propose a practical lightweight image recognition system, named PP-ShiTu, consisting of the following 3 modules,mainbody detection, feature extraction and vector search. We introduce popular strategies including metric learning, deep hash, knowledge distillation and model quantization to improve accuracy and inference speed. With strategies above, PP-ShiTu works well in different scenarios with a set of models trained on a mixed dataset. Experiments on different datasets and benchmarks show that the system is widely effective in different domains of image recognition. All the above mentioned models are open-sourced and the code is available in the GitHub repository PaddleClas[1] on PaddlePaddle[2].

1 Introduction

Image recognition is a general task of computer vision. Benefiting from the rapid improvement of deep learning and the huge market demand, image recognition has grown rapidly, and developed into several sub-fields. Many applications use similar pipelines, such as face recognition, pedestrian and vehicle re-identification, landmark recognition, product recognition, etc. In addition, strategies that work in one application often prove to be effective for other applications as well. For example, ArcMargin loss (Deng et al. 2019) and Triplet loss (Wen et al. 2016) were proposed for face recognition, which was shown to be helpful for many other tasks such as pedestrian re-identification (Hermans, Beyer, and Leibe 2017) and landmark recognition (Yokoo et al. 2020).

However, researchers always focus on one or a few applications, hence, as open source projects. When migrating a strategy to another application,it costs a lot. In view of this, we propose the PP-ShiTu image recognition system to solve similar problems in one pipeline. Figure 2 illustrates the framework of PP-ShiTu. PP-ShiTu contains three modules, namely mainbody detection, feature extraction and vector search. The pipeline of PP-ShiTu is very simple. When getting an image, we first detect one or several mainbody areas
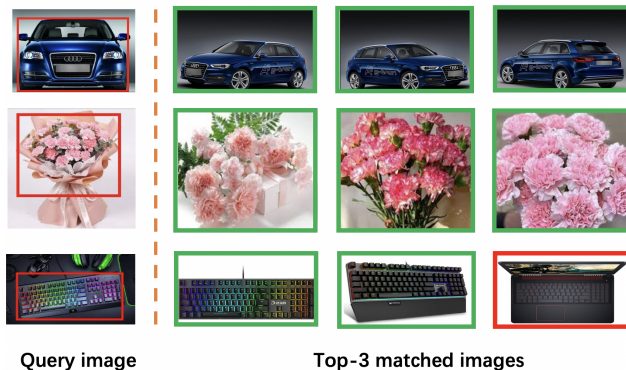

Figure 1: The top-3 results returned by the the proposed PP-ShiTu system for different queries. The third returned result in the third line is wrong because there are only two keyboards in the gallery dataset.

to find the main regions of the image. Then we use a CNN model to extract the features from these regions. A feature is a floating point vector or a binary vector. According to metric learning theory, features imply the similarity of two objects. The shorter the distance between two features, the more similar the original two objects are. Finally we use a vector search algorithm to find the feature in the gallery that are closest to the features we extracted from the image and use the corresponding labels as our recognition results.

In addition to building a generic pipeline, we also introduce a number of effective strategies in the system. We use PP-Picodet with PP-LCNet (Cui et al. 2021a) as the backbone of the mainbody detection model. And PP-LCNet with metric learning strategy, such as ArcMargin (Deng et al. 2019) used as the feature extraction model. Knowledge distillation strategies are widely used in the system. We use backbones trained with SSLD distillation strategy (Cui et al. 2021b) and we also use SSLD to train feature extraction models. Model quantization (Choi et al. 2018) helps us to reduce the storage size of the model. And DeepHash (Wu et al. 2019a) strategy was used to compress the features so as to accelerate the vector search. The ablation experiments show the effectiveness of the strategies above.

What's more, we train the mainbody detection model and

---

[1]https://github.com/PaddlePaddle/PaddleClas

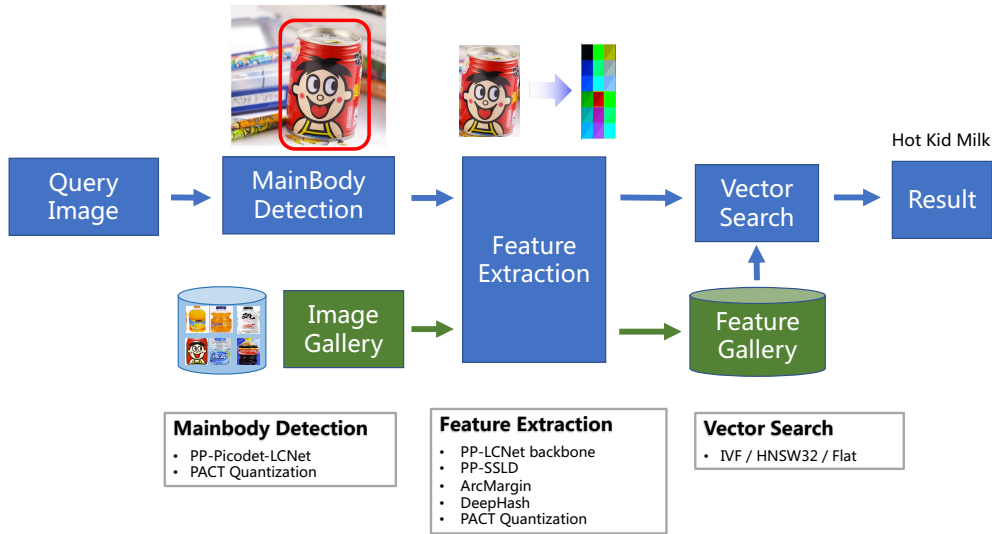[2]https://github.com/PaddlePaddle/Paddle

Figure 2: The framework of the proposed PP-ShiTu.

the feature extraction model with a hybrid dataset by mixing several datasets together. PP-ShiTu works well in different scenarios and obtains competitive results on many datasets and benchmarks.

The rest of paper is organised as follows. In section 2, we present the details of the modules and strategies of PP-ShiTu. Then in section 3, we present and discuss the results of the ablation experiments. Finally, we summarize the conclusions in section 4.

2 Modules and Strategies

2.1 Mainbody Detection

The mainbody detection technology is a widely used detection technology, which refers to detecting all the foreground objects in an image. Mainbody detection is the first step of the whole recognition task, which can effectively improve the recognition accuracy.

There are a wide variety of object detection methods, such as the commonly used two-stage detectors (Faster-RCNN series, etc.), single-stage detectors (YOLO, SSD, etc.), anchor-free detectors (PP-PicoDet, FCOS, etc.) and so on. PaddleDetection developed the PP-YOLOv2 models for server-side scenarios and the PP-PicoDet models for end-side scenarios (CPU and mobile), both of which are SOTA in their respective scenarios. By combining many optimization tricks such as Drop Block (Ghiasi, Lin, and Le 2018), Matrix NMS (Wang et al. 2020c), IoU Loss (Rezatofighi et al. 2019) and so on, PP-YOLOv2 exceeds YOLOv5 on both accuracy and efficiency. PP-PicoDet takes PP-LCNet as its backbone and combines many other detector training tricks such as PAN FPN (Liu et al. 2018), CSP-Net (Wang et al. 2020a), SimOTA (Ge et al. 2021), eventually becoming the first object detector with mAP(0.5:0.95) surpassing 0.30+ on

the COCO dataset within 1M parameters when the input size is 416.

A lightweight mainbody detection model is required for a lightweight image recognition system.Therefore, we utilize PP-PicoDet in PP-ShiTu.

Lightweight CPU Network (PP-LCNet) In order to get a better accuracy-speed trade-off on Intel CPU, we obtained a BaseNet based on comparing different network inferences and added different methods to further improve the accuracy of this network, resulting in the PP-LCNet, which provides a faster and more accurate recognition algorithm with mkldnn enabled. The structure of the entire network is shown in Figure 3. A brief description of the improvements of the network is given below. For more details and ablation experiments, please refer to the original paper (Cui et al. 2021a).

Better activation function. In order to improve the fitting ability of BaseNet, we replaced the activation function in the network with H-Swish from the original ReLU, which can bring a significant improvement in accuracy with only a slight increase in inference time.

SE modules at appropriate positions SE (Hu, Shen, and Sun 2018) module has been used by a large number of networks since it was proposed. It is a good way to weight the network channels to obtain better features, and is used in many lightweight networks such as MobileNetV3 (Howard et al. 2019). However, on Intel CPUs, the SE module increases the inference time, so we cannot use it for the whole network. In fact, through extensive experiments, we have found that the closer to the tail of the network, the more effective the SE module is. So we just add the SE module to the blocks near the tail of the network, which results in a better accuracy-speed balance. The activation functions of the two layers in the SE module are ReLU and H-Sigmoid
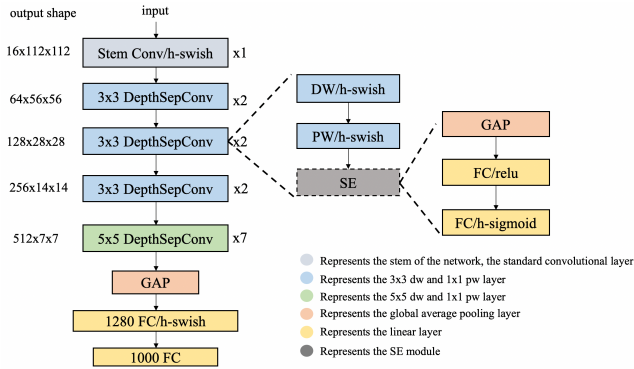
Figure 3: PP-LCNet network structure. The dotted box represents optional modules. The stem part uses standard 3×3 convolution. DepthSepConv means depthwise separable convolutions, DW means depthwise convolutions, PW means pointwise convolutions, GAP means global average pooling.

respectively.

Larger convolution kernels The size of the convolution kernel tends to affect the final performance of the network. In mixnet (Tan and Le 2019), the authors analyzed the effect of different sizes of convolution kernels on the performance of the network, and ended up mixing different sizes of convolution kernels in the same layer of the network. However, such a mixture slows down the inference speed of the model, so we tried to increase the size of convolution kernels, but minimize the increase in inference time. In the end, we set the size of the convolution kernel at the tail of the network as 5×5.

Larger dimensional 1 × 1 conv layer after GAP In PP-LCNet, the output dimension of the network after GAP is small, and directly connecting the final classification layer will lose the combination of features. In order to give the network a stronger fitting ability, we insert a 1280-dimensional size 1x1 conv between GAP layer and final classification layer, which allows more storage space for the model with little increase of inference time.

With these four changes, our model performs well on ImageNet, and table 5 lists the metrics compared to other lightweight models on Intel CPUs.

PP-PicoDet PP-PicoDet is a new series of object detection models developed by PaddleDetection[3].

Specifically, for the backbone model, we utilize the PP-LCNet described above, which helps to improve the detector's inference speed as well as mAP. For the neck, we combine CSP-Net (Wang et al. 2020a) with PAN FPN (Liu et al. 2018) to develop a new FPN structure called CSP-PAN, which helps to enhance the feature map extraction ability. SimOTA (Ge et al. 2021) is adopted and an improved G-Focal loss (Li et al. 2020) function is used in the network training process. What's more, some empirical improvement

---

[3]https://github.com/PaddlePaddle/PaddleDetection

is also used for PP-PicoDet. For example, we use a depth-wise convolution with 5x5 kernel rather than 3x3 in the FPN since it can improve mAP by 0.5% and almost do no harm to the inference speed. More details can be seen on PaddleDetection.

For the mainbody detection task, all objects are seen as foreground, so there is only one class named **foreground** in the label list.

2.2 Feature Extraction

The main problem of image recognition is how to extract better features from the model. Therefore, the ability of feature extraction directly affects the performance of image recognition. In the training phase of feature extraction, we use the metric learning method to learn the features of an image. The metric learning is briefly described below.

Additive Angular Margin Loss (ArcMargin loss) Metric learning attempts to map data into an embedding space, where similar data are close together and dissimilar data are far apart. In metric learning, the quality of features depends on loss, backbone, data quality and quantity, and training strategy. Loss is the most important part of metric learning. The loss of metric learning is divided into two types, namely loss classification-based and pair-based. In recent years, this type of loss has been increasingly used due to improved version of the classification-based loss are more robust. In PP-ShiTu, we use ArcMargin (Deng et al. 2019) loss, which is an improvement based on Softmax loss. The loss of Arcmargin is shown in the formula, and the loss maximizes the classification limit in the angular space, so that the features are better extracted and organized.

$$L = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j=1,j\neq y_i}^{n} e^{s \cos \theta_j}}.$$
(1)

where θ_j is the angle between the weight and the feature. The batch size and the class number are N and n, respectively. m is an angular margin penalty m on the target angle θ_{y_i}, and s is feature scale.

Unified-Deep Mutual Learning (U-DML) Deep mutual learning (Zhang et al. 2017) is a method in which two student networks learn from each other, and a larger teacher network with pre-trained weights is not required for knowledge distillation. In DML, for image classification task, the loss functions contains two parts: (1) loss function between student networks and groundtruth. (2) Kullback–Leibler divergence (KL-Div) loss among the student networks' output soft labels.

In the work of OverHaul (Heo et al. 2019), in which feature map distance between student network and teacher network is used for the distillation process. Transform is carried out on student network feature map to keep the feature map aligned.

To avoid too time-consuming teacher model training process, in this paper, based on DML, we proposed U-DML, in which feature maps are also supervised during the distillation process. Figure 4 shows the framework of U-DML.
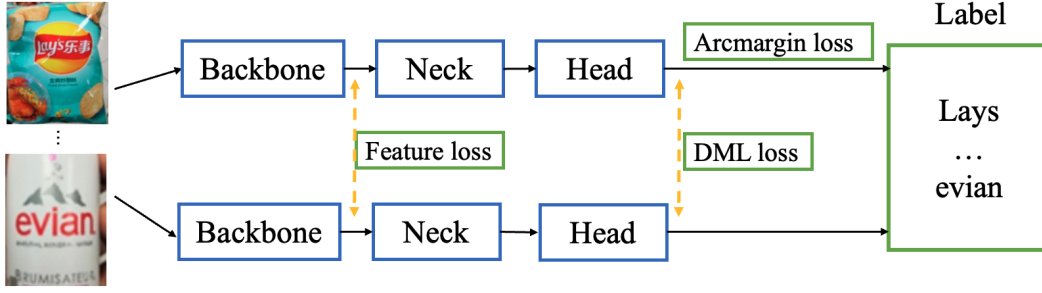
Figure 4: U-DML framework

There are two networks for the distillation process: the student network and the teacher network. They have exactly the same network structures with different initialized weights. The goal is that for the same input image, the two networks can get the same output, not only for the prediction result but also for the feature map.

The total loss function consists of three parts: (1) Arcmargin loss. Since the two networks' neck and head are trained from scratch, Arcmargin loss can be used for the networks' convergence; (2) DML loss. The final output distribution of the two networks is expected to be the same, so DML loss is needed to ensure the consistency of the distribution between the two networks; (3) Feature loss. The architectures of the two networks are the same, so their feature maps are expected to be the same, feature loss can be used to constrain the two networks' intermediate feature map distance.

$$Loss_{arc} = Arcmargin(S_{hout}, gt) + Arcmargin(T_{hout}, gt) \tag{2}$$

in which S_{hout} denotes head output of the student network and T_{hout} denotes that of the teacher network. gt donates the groudtruth label of the input image.

DML loss. In DML, parameters of each sub-network are updated separately. Here, to simplify the training process, we calculate the KL divergence loss between the two sub-networks and update all the parameter simultaneously. The DML loss is as follows.

$$Loss_{dml} = \frac{KL(S_{pout}||T_{pout}) + KL(T_{pout}||S_{pout})}{2} \tag{3}$$

in which $KL(p||q)$ denotes KL divergence of the p and q. S_{pout} and T_{pout} can be calculated as follows.

$$S_{pout} = Softmax(S_{hout})$$
$$T_{pout} = Softmax(T_{hout}) \tag{4}$$

Feature loss. During the training process, we hope that the backbone output of the student network is the same as that of the teacher network. Therefore, similar to Overhaul, feature loss is used for the distillation process. The loss can be calculated as follows.

$$Loss_{feat} = L2(S_{bout}, T_{bout}) \tag{5}$$

in which S_{bout} means backbone output of the student network and T_{bout} means that of teacher network. Mean square error Loss is utilized here. It is noted that for the feature loss, feature map transformation is not needed because the two feature maps used to calculate the loss are exactly the same.

Finally, the total loss for the U-DML training process is shown as follows.

$$Loss_{total} = Loss_{arc} + Loss_{dml} + Loss_{feat} \tag{6}$$

DeepHash In practical applications, the massive retrieval images and videos feature database will not only consume huge storage space, but also result in long retrieval time, which is unacceptable in some application scenarios. In addition, the storage space on the mobile device is very limited, which puts forward further challenges for massive feature storage. With this in mind, we integrate DeepHash (Luo et al. 2020) functionality in addition to Metric Learning capabilities in PP-Shitu which can help to decrease storage and speed up retrieval

DeepHash studies how to obtain representative binary features using deep neural networks, which uses bit to store binary features, and adopt Hamming distance to measure the distance between two feature vectors. Specifically, we have open sourced three of the most common DeepHash algorithms: DLBHC (Lin et al. 2015), LCDSH (Cao et al. 2017), DSHSD (Wu et al. 2019b).

In PP-ShiTu, we adopt the optimized DSHSD (Wu et al. 2019b) algorithm to get a practical binary model for identifying goods. Compared with the real-valued model, the retrieval precision may slightly drop, but the storage space of gallery features can be reduced 32 times, and the retrieval speed can be increased by more than 5 times when the size of the search library larger than 100 thousand.

2.3 Vector Search

In order to run on multiple operating system, including Linux, Windows and MacOS, we use Faiss [4] as the vector search module, which is a library for efficient similarity search and contains algorithms that support search in sets

---
[4]https://github.com/facebookresearch/faiss

of vectors of any size, up to ones that possibly do not fit in RAM. In PP-ShiTu, we choose three algorithms, HNSW32 (Malkov and Yashunin 2018), IVF (Babenko and Lempitsky 2012) and FLAT to meet the demand of different scenarios.

It is worth noting that a feature gallery needs to be built before vector search. The feature gallery is composed of features extracted from labeled images. In PP-ShiTu pipeline, the label of query is obtained from feature gallery after similarity search.

3 Experiments

3.1 Experimental Setup

Datasets We used a lot of public datasets for different scenarios to train our model, the specific introduction is as follows.

Mainbody Detection For mainbody detection training process, we use 5 public data sets: Objects365 (Shao et al. 2019), COCO2017 (Lin et al. 2014), iCartoonFace (Zheng et al. 2020), LogoDet-3k (Wang et al. 2020b) and RPC (Wei et al. 2019). More detailed information can be seen in Table 1.

Dataset	Number of images	Images used	Scenarios
Objects365	1700K	6K	General Scenarios
COCO2017	120K	5K	General Scenarios
iCartoonFace	2k	2K	Cartoon Face
LogoDet-3K	3k	2K	Logo
RPC	3K	3K	Product

Table 1: Statement of mainbody detection training dataset

General Recognition For general recognition, we use 7 public datasets including Aliproduct (Cheng et al. 2020), GLDv2 (Weyand et al. 2020), VeRI-Wild (Lou et al. 2019), LogoDet-3K (Wang et al. 2020b), iCartoonFace (Zheng et al. 2020), SOP (Song et al. 2016), Inshop (Liu et al. 2016). These data sets are also used for general training, model distillation and deephash. The detail of each dataset is described in the Table 2.

Dataset	Number of images	Number of classes	Scenarios
Aliproduct	2498771	50030	Product
GLDv2	1580470	81313	Landmark
VeRI-Wild	277797	30671	Vehicle
LogoDet-3K	155427	3000	Logo
iCartoonFace	389678	5013	Cartoon
SOP	59551	11318	Product
Inshop	25882	3997	Product
Total	**5M**	**185K**	——

Table 2: Statement of recognition training dataset

Test set To evaluate the speed of PP-ShiTu, we release a test set consisting of a query image set and a feature library.

We collect 200 images from Internet as the query set. The query dataset can be found in the GitHub repository PaddleClas. For gallery, we used a combination of Products-10k(Bai et al. 2020) and some common products downloaded from the Internet, and build feature library with image gallery. Each image in the query set contains at least one object from gallery. Figure 5 shows the examples of images in test set.

Implementation Details The following briefly describes our implementation details, which are critical to the result of the final system.

Mainbody Detection For mainbody detection, we use PP-PicoDet architecture as our mainbody detection architecture. To improve the backbone's feature extraction capabilities, we use PP-LCNet-2.5x as the detector backbone, and the pretrained weights of PP-LCNet-2.5x is trained with SSLD distillation method (Cui et al. 2021b) using PaddleClas. The total epoch number for mainbody detection is 100, the batch size and learning rate are reduced to half of the standard configuration provided in PaddleDetection because of the GPU memory limit.

General Recognition For general recognition, we use all 7 datasets mentioned above to train our PP-LCNet-2.5x for 100 epochs. We follow standard practice and perform data augmentation with random-size cropping to 224×224 pixels and random horizontal flipping. Optimization is performed using SGD with momentum 0.9 and a mini-batch size of 2048. Weight decay is empirically set to 1e-5 and initial learning rate is 0.1. In Arcmargin Loss, we set the scale to 30 and the margin to 0.2. In order to further improve the indicators in small datasets, we have increased the sampling rate of LogoDet, Inshop, and VeRI-Wild by four times. The hyper-parameters of U-DML distillation training are the same with those of the general recognition except the loss function. In the evaluation phase, we use each single dataset for evaluation, and the metric we use is recall@1. In DSHSD Loss, the weight of contrastrive loss is set to 0.05, and the margin is set to 1024 (twice of the feature embedding size).

3.2 Mainbody Detection Result

Table 3 shows the mAP results on 5 public data sets. PP-YOLO is the mainbody detection model we provided in the last version, which is suggested for server-side usage. PP-PicoDet is 14 times faster than PP-YOLO on CPU latency while just has a slight mAP reduction (-2.4%), To further reduce the model size, we quantize PP-PicoDet model using PACT and the final model size is reduced to 6.9M, which is suitable for mobile deployment.

3.3 General Recognition Result

Table 4 shows the recall@1 of ResNet50_vd and PP-LCNet-2.5x on different dataset. Among them, ResNet50_vd is trained on their respective datasets. PP-LCNet-2.5x is trained on the entire merged dataset. As can be seen from the table, our training strategy allows our lightweight model to approach or even exceed the accuracy of ResNet50_vd. Moreover, because our recognition model is trained on a
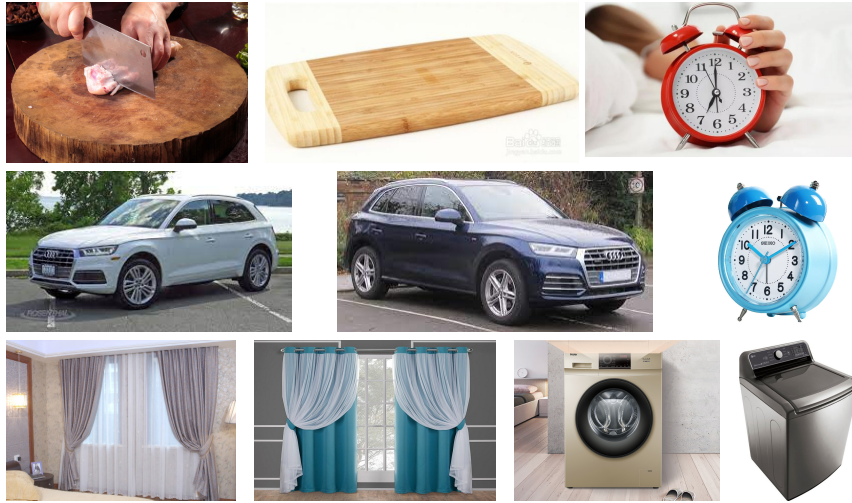
Figure 5: Examples of images in test set.

-	PP-YOLO R50-vd-DCNv2	PP-PicoDet	PP-PicoDet Int8
Objects365	0.427	0.338	0.359
COCO2017	0.480	0.392	0.416
iCartoonFace	0.399	0.340	0.375
LogoDet-3K	0.507	0.501	0.564
RPC	0.312	0.432	0.406
Average	0.425	0.401	0.424
Model size (MB)	250	29.2	6.9
Latency on CPU (ms)	466.0	29.8	-

Table 3: Ablation experiments of mainbody detection models in PP-ShiTu.Here **PP-PicoDet Int8** means the model is quantized using PACT provided by PaddleSlim. Latency tested on Intel® Xeon® Gold 6148 Processor with batch size of 1 and MKLDNN enabled, the number of thread is 10.

large number of different types of data, it can be applied to different scenarios.

3.4 Ablation study

Ablation study for PP-LCNet   To test the generalization ability of the model, we use challenging datasets like ImageNet-1k throughout the process of designing the model. Table 5 shows the accuracy-speed comparison between PP-LCNet and other competive lightweight models. It can be seen that our PP-LCNet has a significant advantage both on speed and accuracy, even when compared to a very competitive network like MobileNetV3.

Ablation study for U-DML   Based on the general PP-LCNet-2.5x recognition model, we carry out some experiments. The training strategy is almost the same with that for base PP-LCNet-2.5x recognition model training process, except that we adopt distillation loss. Table 7 shows some results for different methods.

Model recall@1 on 6 data sets is improved by 0.5% on average using DML, and after using U-DML, the model recall@1 is improved by 0.7%. Since the model is generated using knowledge distillation training process, it will not take any extra time cost during inference.

It is noted that we did not fine-tune any hyper-parameters for the model training process such as loss ratio. What's more, compared to the standard training process (the final train loss is 2.64), the distillation model tends to generalize better and takes longer time to converge (the final train loss of the student is 2.75), which is consistent with the conclusion in (Beyer et al. 2021). We believe that with more epochs, the distillation model can perform better.

Ablation study for DeepHash   Based on DSHSD algorithm, we also carry out some experiments combined with insights of other DeepHash methods. Table 6 shows some results for different strategies based on MobileNetv3. It can be seen that, with label smoothing, we can further improve the binary feature representation capacity.

3.5 Inference time latency on test set

We compare inference speed of PP-ShiTu and the server model, which was released on PaddleClas earlier and use ResNet 50 backbone for detection and feature extraction, on

Model	Recall@1						Latency (ms)
	Aliproduct	VeRI-Wild	LogoDet-3k	iCartoonFace	SOP	Inshop	
ResNet50_vd	0.836	–	–	–	–	–	
ResNet50_vd*	–	0.938	–	–	–	–	15.2
ResNet50_vd*	–	–	0.898	–	–	–	
ResNet50_vd	–	–	–	0.832	–	–	
PP-LCNet-2.5x	**0.839**	0.888	0.861	**0.841**	**0.793**	**0.892**	5.0

Table 4: Comparison of ResNet50_vd and PP-LCNet-2.5x in PP-ShiTu on each dataset. * indicates that the stride of the last block in the backbone is 1, which means larger FLOPs. Latency tested on Intel® Xeon® Gold 6148 Processor with batch size of 1 and MKLDNN enabled, the number of thread is 10.

Model	Params(M)	FLOPs(M)	Top-1 Acc.(%)	Top-5 Acc.(%)	Latency(ms)
MobileNetV2-0.25x	1.5	34	53.21	76.52	2.47
MobileNetV3-small-0.35x	1.7	15	53.03	76.37	3.02
ShuffleNetV2-0.33x	0.6	24	53.73	77.05	4.30
PP-LCNet-0.25x	**1.5**	**18**	**51.86**	**75.65**	**1.74**
MobileNetV2-0.5x	2.0	99	65.03	85.72	2.85
MobileNetV3-large-0.35x	2.1	41	64.32	85.46	3.68
ShuffleNetV2-0.5x	1.4	43	60.32	82.26	4.65
PP-LCNet-0.5x	**1.9**	**47**	**63.14**	**84.66**	**2.05**
MobileNetV1-1x	4.3	578	70.99	89.68	3.38
MobileNetV2-1x	3.5	327	72.15	90.65	4.26
MobileNetV3-small-1.25x	3.6	100	70.67	89.51	3.95
ShuffleNetV2-1.5x	3.5	301	71.63	90.15	-
PP-LCNet-1x	**3.0**	**161**	**71.32**	**90.03**	**2.46**

Table 5: PP-LCNet ablation study. Comparison of state-of-the-art light networks over classification accuracy. Latency tested on Intel® Xeon® Gold 6148 Processor with batch size of 1 and MKLDNN enabled, the number of thread is 10.

Method	Embedding-size	Recall@1
DLBHC	512	0.773
DSHSD	512	0.798
DSHSD+Label Smoothing	512	0.802

Table 6: Ablation experiments of DeepHash strategies of binary recognition model based on Aliproducts dataset

the test set. F1-scores of two models are both 0.3306. Table 8 shows the latency of two models tested on CPU and GPU server.

4 Conclusions

In this paper, we propose a practical lightweight image recognition system, PP-ShiTu. We introduce a general pipeline and practical strategies including PP-PicoDet, U-DML, Arcmargin loss and DeepHash. Experiments show that PP-ShiTu works very well on different tasks and datasets. The corresponding ablation experiments are also provided.

References

Babenko, A.; and Lempitsky, V. 2012. The inverted multi-index. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3069–3076. doi:10.1109/CVPR.2012.6248038. 2.3

Bai, Y.; Chen, Y.; Yu, W.; Wang, L.; and Zhang, W. 2020. Products-10k: A large-scale product recognition dataset. *arXiv preprint arXiv:2008.10545* . 3.1

Beyer, L.; Zhai, X.; Royer, A.; Markeeva, L.; Anil, R.; and Kolesnikov, A. 2021. Knowledge distillation: A good teacher is patient and consistent. *arXiv preprint arXiv:2106.05237* . 3.4

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. HashNet: Deep Learning to Hash by Continuation. *IEEE Computer Society* . 2.2

Cheng, L.; Zhou, X.; Zhao, L.; Li, D.; Shang, H.; Zheng, Y.; Pan, P.; and Xu, Y. 2020. Weakly Supervised Learning with Side Information for Noisy Labeled Images. 3.1

Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P. I.-J.; Srinivasan, V.; and Gopalakrishnan, K. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085* . 1

Model	Recall@1						
	Aliproduct	VeRI-Wild	LogoDet-3k	iCartoonFace	SOP	Inshop	Average
PP-LCNet-2.5x	0.839	0.888	0.861	0.841	0.793	0.892	0.852
PP-LCNet-2.5x + DML	0.841	0.892	0.863	0.848	0.803	0.894	0.857
PP-LCNet-2.5x + U-DML	**0.841**	**0.901**	**0.863**	**0.849**	**0.803**	**0.896**	**0.859**

Table 7: Ablation experiments of knowledge distillation of recognition model

Processor Info.	Model	Latency (ms)			
		Detection	Feature Extraction	Vector Search	Overall
Intel®Xeon®Gold 6148	Server Model	923	79	44	1051
	PP-ShiTu	**94**	**45**	51	**194**
Intel®Core™ i5-6300U	Server Model	3305	475	3.3	3790
	PP-ShiTu	**282**	**150**	4.4	**442**

Table 8: Inference time of PP-ShiTu and server model on test set. Latency tested on models with batch size of 1 and MKLDNN enabled, the number of thread is 10.

Cui, C.; Gao, T.; Wei, S.; Du, Y.; Guo, R.; Dong, S.; Lu, B.; Zhou, Y.; Lv, X.; Liu, Q.; Hu, X.; Yu, D.; and Ma, Y. 2021a. PP-LCNet: A Lightweight CPU Convolutional Neural Network. 1, 2.1

Cui, C.; Guo, R.; Du, Y.; He, D.; Li, F.; Wu, Z.; Liu, Q.; Wen, S.; Huang, J.; Hu, X.; Yu, D.; Ding, E.; and Ma, Y. 2021b. Beyond Self-Supervision: A Simple Yet Effective Network Distillation Alternative to Improve Backbones. 1, 3.1

Deng, J.; Guo, J.; Xue, N.; and Zafeiriou, S. 2019. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4690–4699. 1, 1, 2.2

Ge, Z.; Liu, S.; Wang, F.; Li, Z.; and Sun, J. 2021. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430* . 2.1, 2.1

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2018. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890* . 2.1

Heo, B.; Kim, J.; Yun, S.; Park, H.; Kwak, N.; and Choi, J. Y. 2019. A Comprehensive Overhaul of Feature Distillation. *CoRR* abs/1904.01866. URL http://arxiv.org/abs/1904.01866. 2.2

Hermans, A.; Beyer, L.; and Leibe, B. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* . 1

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q. V.; and Adam, H. 2019. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2.1

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2.1

Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; and Yang, J. 2020. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv preprint arXiv:2006.04388* . 2.1

Lin, K.; Yang, H. F.; Hsiao, J. H.; and Chen, C. S. 2015. Deep learning of binary hash codes for fast image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2.2

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer. 3.1

Liu, S.; Qi, L.; Qin, H.; Shi, J.; and Jia, J. 2018. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8759–8768. 2.1, 2.1

Liu, Z.; Luo, P.; Qiu, S.; Wang, X.; and Tang, X. 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3.1

Lou, Y.; Bai, Y.; Liu, J.; Wang, S.; and Duan, L.-Y. 2019. VERI-Wild: A Large Dataset and a New Method for Vehicle Re-Identification in the Wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3235–3243. 3.1

Luo, X.; Chen, C.; Zhong, H.; Zhang, H.; and Hua, X. 2020. A Survey on Deep Hashing Methods . 2.2

Malkov, Y. A.; and Yashunin, D. A. 2018. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. 2.3

Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; and Savarese, S. 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Pro-

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 658–666. 2.1

Shao, S.; Li, Z.; Zhang, T.; Peng, C.; Yu, G.; Zhang, X.; Li, J.; and Sun, J. 2019. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8430–8439. 3.1

Song, H. O.; Xiang, Y.; Jegelka, S.; and Savarese, S. 2016. Deep Metric Learning via Lifted Structured Feature Embedding. In *Computer Vision and Pattern Recognition (CVPR)*. 3.1

Tan, M.; and Le, Q. V. 2019. MixConv: Mixed Depthwise Convolutional Kernels . 2.1

Wang, C.-Y.; Liao, H.-Y. M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; and Yeh, I.-H. 2020a. CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 390–391. 2.1, 2.1

Wang, J.; Min, W.; Hou, S.; Ma, S.; Zheng, Y.; and Jiang, S. 2020b. LogoDet-3K: A Large-Scale Image Dataset for Logo Detection. 3.1, 3.1

Wang, X.; Zhang, R.; Kong, T.; Li, L.; and Shen, C. 2020c. Solov2: Dynamic, faster and stronger. *arXiv e-prints* arXiv–2003. 2.1

Wei, X.-S.; Cui, Q.; Yang, L.; Wang, P.; and Liu, L. 2019. RPC: A large-scale retail product checkout dataset. *arXiv preprint arXiv:1901.07249* . 3.1

Wen, Y.; Zhang, K.; Li, Z.; and Qiao, Y. 2016. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, 499–515. Springer. 1

Weyand, T.; Araujo, A.; Cao, B.; and Sim, J. 2020. Google Landmarks Dataset v2 – A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. 3.1

Wu, L.; Ling, H.; Li, P.; Chen, J.; Fang, Y.; and Zhou, F. 2019a. Deep supervised hashing based on stable distribution. *IEEE Access* 7: 36489–36499. 1

Wu, L.; Ling, H.; Li, P.; Chen, J.; and Zou, F. 2019b. Deep Supervised Hashing Based on Stable Distribution. *IEEE Access* 7(1): 36489–36499. 2.2

Yokoo, S.; Ozaki, K.; Simo-Serra, E.; and Iizuka, S. 2020. Two-stage discriminative re-ranking for large-scale landmark retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 1012–1013. 1

Zhang, Y.; Xiang, T.; Hospedales, T. M.; and Lu, H. 2017. Deep Mutual Learning. *CoRR* abs/1706.00384. URL http://arxiv.org/abs/1706.00384. 2.2

Zheng, Y.; Zhao, Y.; Ren, M.; Yan, H.; Lu, X.; Liu, J.; and Li, J. 2020. Cartoon Face Recognition: A Benchmark Dataset. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2264–2272. 3.1, 3.1