Computer Vision based Coin Counter using DL

ChienHsuan Yen
NTUT EE
t109318040@ntut.org.tw

***Abstract*—this paper proposes to use a deep learning-based computer vision method to solve this annoying problem, speed up the checkout, and even reduce the occurrence of miscalculated amounts.**

***Index Terms*—coin detection, coin classification**

I. Introduction

There are many retail markets or stores in Taiwan that use banknotes and change for transactions. Although there are electronic payment services, most people still choose cash for transactions. Unless it is a large amount of goods, everyone will consider using electronic payment tools. Especially when a lot of change is used for transactions at the same time, it will take a long time to confirm the total amount of all the change and cause trouble for the cashier. Therefore, this paper proposes to use a deep learning-based computer vision method to solve this annoying problem, speed up the checkout, and even reduce the occurrence of miscalculated amounts. According to the experimental results, my method can achieve good coin amount calculation accuracy.

II. Related work

Because this article is only a final project, not a rigorous academic research, so I will not compare it with other methods that have been proposed, so as not to cause too much unnecessary trouble.

Detection model. Coins are circular objects on the picture, so *Hough Transform* can be used to detect circular objects and extract the coin picture. This method is simple and effective. But I only use deep learning methods, so I use *Single Shot Detector* [1] detection algorithm, a one-stage algorithm, which has faster inference speed without losing too much accuracy.

Classification model. Common classification models such as *VGG16* [4] and *ResNet-50* [2] are models with a large number of parameters. It is difficult to use such complex models in the actual field. Therefore, this paper uses lightweight models such as *MobileNet v2* [3] and the complex models mentioned above for some performance comparisons.

III. Method

The entire architecture process is to use the detection model to extract the coin image, then use the classification model to classify the coin image, and then calculate the total amount of the coin. The detailed flow chart is in Fig. 1. Some examples of dataset images are shown in Fig. 2.

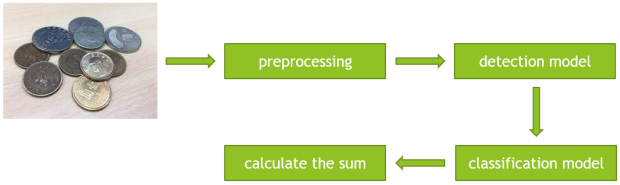

Fig. 1: Architecture

Fig. 2: Dataset image examples

A. Pre-processing

The pre-processing methods used during training are random erasing, random croping, random fliping and normalize. Just use normalize when testing.

B. Detection model

I use MobileNet v2 as backbone and detection head using SSDLite. As a feature extraction network, MobileNet v2 has less computation and computing time, and use Lite version of SSD to reduce computing time. The detailed architecture is shown in Fig. 3.

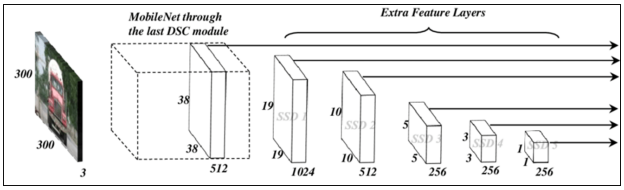

Fig. 3: SSD architecture

C. Classification model

The research that has been proposed so far has many complex and huge models, which need to use the gpu server to run the model, which has no practical value in the actual field. Therefore, this paper compares the performance differences between the complex model and the lightweight model,

including the accuracy and inference time, as a reference for practical use. The objects to be compared are ResNet-50 and MobileNet v2.

IV. Experiments

I use the Average Precision (AP) metric to measure the performance of the detection model and the classification accuracy metric to measure the performance of the classification model.

A. Object detector

I use MobileNet v2 as backbone and detection head using SSDLite. As a feature extraction network, MobileNet v2 has less computation and computing time, and use Lite version of SSD to reduce computing time. The total training epochs are 120. The detection Average Precision (AP) score is in Table I. Some detection results are shown in Fig. 4.

	AP[IoU=0.5]	AP[IoU=0.7]	AP[IoU=0.9]
MobileNet v2 SSDLite	1.0	1.0	0.736

TABLE I: Detection performance

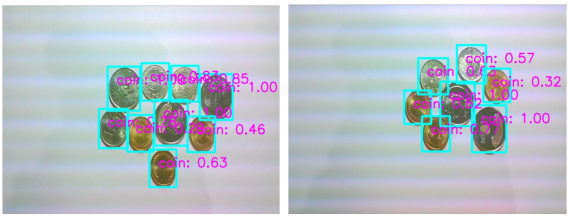

Fig. 4: Detection results

B. Classifier

I compares the performance differences between the complex model and the lightweight model, including the accuracy and inference time, as a reference for practical use. The objects to be compared are ResNet-50 and MobileNet v2.

1) MobileNet v2: MobileNet v2 is used as the classifier, random horizontal flip and random erasing are used for pre-processing of training data, Adam optimizer is used for optimizer, cross entropy loss is used for loss function, cosine annealing learning rate is used for learning rate decay, and the initial learning rate is 3E-03, the lowest The learning rate is 5E-06 and the number of training epochs is 30. The training curves are shown in Fig. 5. The classification accuracy of the model for each coin in the test set is shown in Table II.

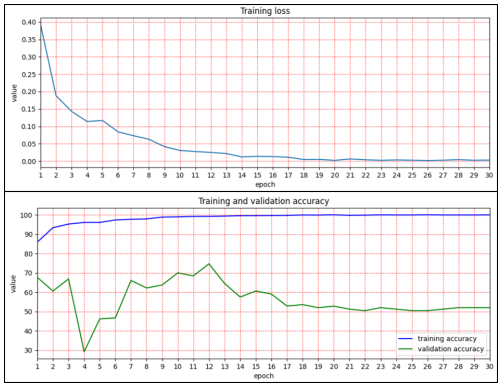

Fig. 5: MobileNet v2 training curve

coin type	accuracy
coin1	100%
coin5	100%
coin10	100%
coin50	100%

TABLE II: Performance of MobileNet v2 on testset

2) ResNet-50: ResNet-50 is used as the classifier, random horizontal flip and random erasing are used for pre-processing of training data, Adam optimizer is used for optimizer, cross entropy loss is used for loss function, cosine annealing learning rate is used for learning rate decay, and the initial learning rate is 3E-03, the lowest The learning rate is 5E-06 and the number of training epochs is 30. The training curves are shown in Fig. 6. The classification accuracy of the model for each coin in the test set is shown in Table III.

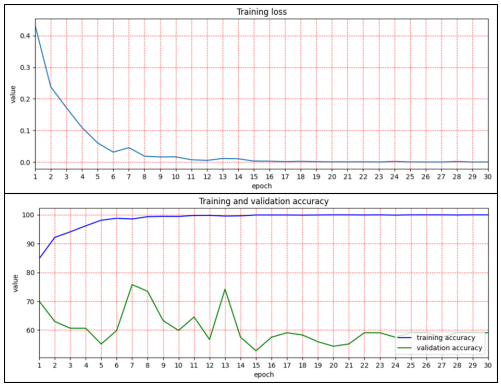

Fig. 6: ResNet-50 training curve

coin type	accuracy
coin1	100%
coin5	100%
coin10	100%
coin50	100%

TABLE III: Performance of ResNet-50 on testset

C. Comparison

Detection models all use MobileNet v2 SSDLite. Compare the performance differences between MobileNet v2 and

ResNet-50 as classifiers. The comparison items include training accuracy, training loss, validation accuracy, inference time, and actual running results. Deep learning framework uses PyTorch to run models on CPU. The training curves are compared in Fig. 7. Actual results are shown in Fig. 8. Inference time comparison is in Table IV.

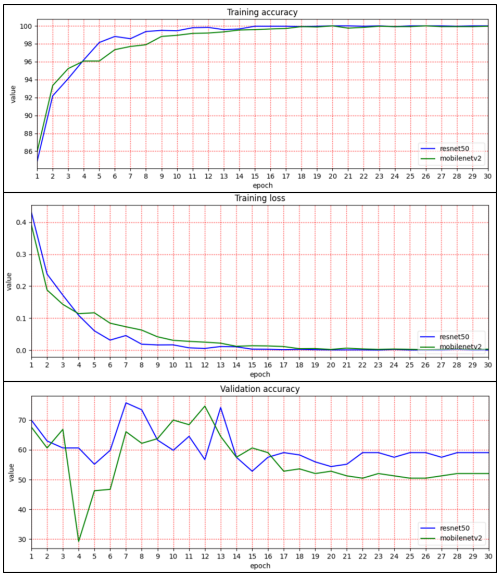

Fig. 7: Comparison

Fig. 8: Actual running result. On the left is the result of MobileNet v2. On the right is the result of ResNet-50.

classifier	inference time (s)
MobileNet v2	1.2
ResNet-50	2.9

TABLE IV: Inference time comparison

V. Conclusion

In this paper, I proposes to use a deep learning-based computer vision method to solve this annoying problem, speed up the checkout, and even reduce the occurrence of miscalculated amounts. According to the experimental results, my method can achieve good coin amount calculation accuracy. Moreover, according to the actual situation, a model with a suitable size can be selected to achieve a balance under different conditions.

Acknowledgment

Thanks to many developers who are willing to expose some excellent projects on GitHub.

References

[1] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. SSD: Single shot multibox detector. In ECCV, 2016.
[2] He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. In CVPR, 2016.
[3] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. Mobilenetv2: Inverted residuals and linear bottlenecks. In CVPR, 2018.
[4] Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.