An AI-based Android Application for Ancient Documents Text Recognition

Tomoki Morioka[1], Aravinda C V[2], and Lin Meng[1]

[1] Dept.of Electronic and Computer Engineering, Ritsumeikan University.
Kusatsu, Shiga, Japan.
{ri0086ih@ed,menglin@fc} .ritsumei.ac.jp
[2] Dept of Computer Science and Engineering, NMAM Institute of Technology,
NITTE, Karkala UDUPI, India
aravinda.cv@nitte.edu.in

Abstract

Advances in technology have led to the development and widespread use of high-performance mobile devices. One such device is the smartphone which is a kind of cell phone with operating systems such as IOS and Android and has the highest penetration rate in the world. The use of deep learning in smartphones leads to the spreading of AI-based services. In addition, the demand for image recognition in edge devices and mobile devices is increasing due to examples such as face recognition. This paper proposes an ancient documents text recognition system for Android, using communication with a recognition AI model equipped server. An image recognition application without uses a server is equipped by using TensorFlow Lite for the performance comparison. The results show that the recognition time of the TensorFlow Lite application is shorter than that of the system using a server. However, the CPU and memory usage of the proposal is lower, and the operation is more stable. Hence, the proposed system is more stable in considering the hardware usage.

1 Introduction

Recent technological advances have led to the development and widespread use of high-performance mobile devices. One of them is the smartphone, which is a kind of cellphones equipped with the world's commons OS such as IOS and Android. Featuring touch and flick operation on the touch panel offers a variety of services through unique applications in smartphones.

Furthermore, smartphones employ deep learning technologies, letting the application field expand even further. In the field of image recognition, Deep Learning technologies use a multi-layered neural network for learning, which has higher accuracy than conventional image processing methods, and various studies are being conducted[1, 2, 3]. Usually, the deep learning models are equipped on high-specification PCs with GPUs. Currently, with the researchers' hard work, some slight deep learning models are realized to be applied on resource-limited edge devices, mobile devices, and so on [4, 5, 6].

Our team has also been studying ancient documents for some time. One of them is the Oral Bone Inscriptions, which were used in ancient China 3000 years ago and became the origin of Chinese Characters. OBIs are carved on animal bones and turtle shells, which are cracked and deteriorated, making them difficult to recognize. In our previous research, we have succeeded in achieving a high recognition rate by building a system using multiple deep learning models [7]. However, since this system is published as a PC or web API, we need to prepare a high-spec PC and browser environment. Hence, we consider that we can analyze OBIs more easily by using

an Android application to do these things. For example, an OBIs image taken by a camera can be recognized instantly. As there are no implementation examples of OBIs recognition in Android, we research the implementation of deep learning models and the use of deep learning in smartphones to realize this application.

Implementing deep learning on Android devices using frameworks such as TensorFlow[8] and PyTorch[9], are possible to create standalone applications which perform image recognition only using Android devices.

However, not all Android devices have identical specifications, and it is difficult to guarantee that the application can work on all devices because the application depends on the specifications. Therefore, in this paper, we develop a recognition system that guarantees the operation of applications by using a server to perform image recognition instead of the Android device. Specifically, the trained deep learning models are equipped on the server previously.

In the recognition processing, the target images in the Android device are sent to the server firstly. Then the image recognition is performed in the server using a trained model. Finally, the android device the recognition results. To prove the effectiveness of our proposal, we compare the system performance with the standalone application, implementing a trained model using TensorFlow Lite.

In section 2 describes related work. Section 3 introduces the recognition flow of the proposed system. Section 4 reports and discusses the experimental results, then the paper is concluded in section 5.

2 Related work

2.1 TensorFlow Lite

TensorFlow is an open software library for numerical computation released by google. It is also a numerical library specialized for deep learning that can perform calculations on mul-

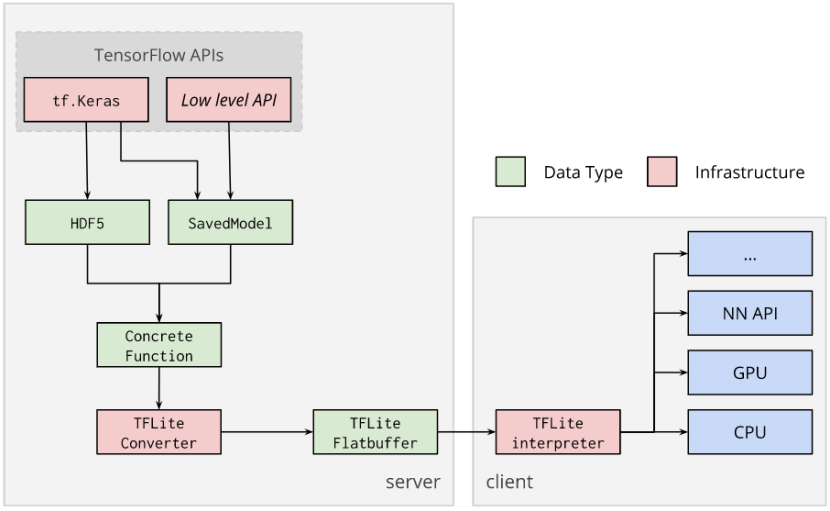

Figure 1: Deploying TensorFlow Lite[8](https://www.tensorflow.org/lite/convert/index?hl=ja)

tidimensional arrays of "Tensor". Almost all of the major operating systems are applied to TensorFlow, such as Linux, macOS, and Windows, for the reason of TensorFlow's extensive library and customization features. Currently, TensorFlow possesses a larger number of users among deep learning frameworks.

TensorFlow Lite is a toolset for running TensorFlow models on the edge and mobile devices, which provides APIs to enable inference on mobile devices, the transformation of TensorFlow models for mobile devices, and model optimization through quantization and other means.

The flow of deployment to edge devices and mobile devices is shown in Figure 1 by using TensorFlow Lite. Firstly, TensorFlow Lite Converter converts a model trained with TensorFlow APIs. Secondly, the model is deployed using the publicly available TensorFlow Lite API. It also has an API for deploying to Android.

2.2 System target of Oral Bone Inscriptions Recognition

The goal of our system aims to recognize and organize the ancient documents, including OBIs [10], rubbings[11], Japanese Kuzushiji documents[12] and so on. This paper only focuses on OBIs recognition and organization.

OBIs are one kind of ancient character, used in China more than 3,000 years ago and are the origin of Chinese characters. They were carved on turtle shells and animal bones because there was no ink and paper. Deciphering OBIs is very important for the study of history and Chinese characters. However, they are difficult to be deciphered due to severe deterioration. Hence, researchers have tried to recognize the OBIs by image processing and deep learning [13, 10, 14]. However, these methods should cut OBIs images manually. Furthermore, some researchers have tried to realize detecting OBIs from Oracle Bone and recognizing the OBIs in the same system [14, 7].

For the researchers are easy to recognize the OBIs online, the goal of this paper is to build an online OBI recognition system based on Android using two deep Learning models shown in the paper [7].

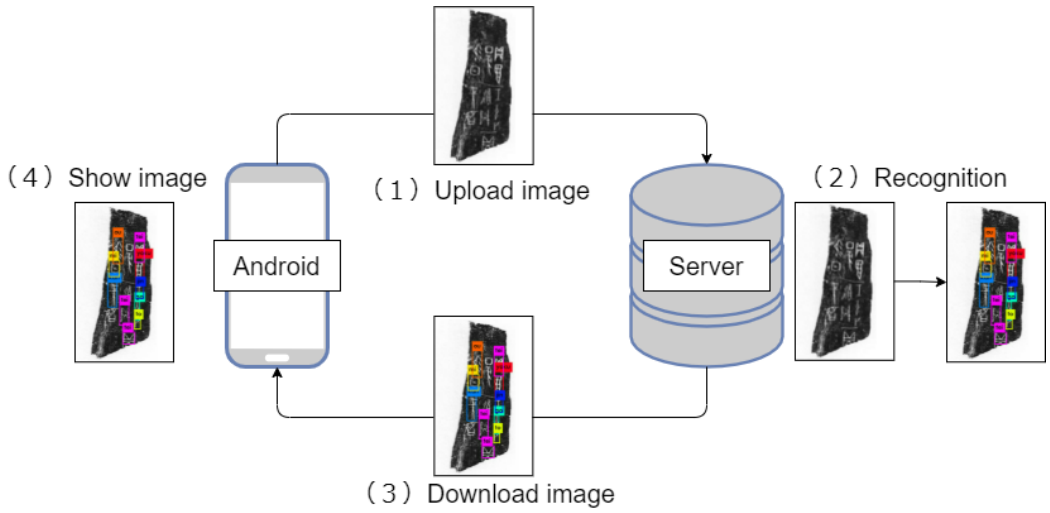

Figure 2: System flow

3 System flow

The system flow of the proposed method, a server-based image recognition system, is shown in Figure 2.

In the first step, YOLOv3[15], an object detection and recognition model, is applied to recognize multiple characters simultaneously. However, detecting all characters without errors is very difficult. In the second step, the user trims the undetected characters manually and applies MobiliNet to recognize the undetected characters. The system is installed in an application programming interface and is open to researchers and other interested users. The Android application in this paper is challenging work of this API. Hence, only YOLOv3 is applied for detecting and recognizing OBIs.

(1) Image uploading

The first step is to upload the image from the Android device to the server. We use the POST method of the HTTP protocol to communicate, requesting the URL on the Android side and writing the image data in the request body. The MIME type is "multipart/form-data", which sends three pieces of data. The type attribute of each input element is "*submit*","*file*", and "*text*".

"*file*" is an image file.
"*text*" contains a number and indicates the threshold of the image recognition result.
"*submit*" is the trigger for Upload, and the server-side starts processing when it receives "*submit*".

(2) Image recognition

The second step is to perform image recognition on the server. YOLOv3 is applied in this work, which runs on DarkNet, a framework built-in C programming language, to recognize OBIs. DarkNet refers to the directory path of the image stored in the server and the received threshold value and processes it. The output images of the recognition results are saved in the same directory in the server.

(3) Image Downloading

The third step is to download the image. At the same time as the recognized recognition result is stored in the server, the Android side sends a request and starts downloading the image using the GET method of the HTTP protocol.

(4) Show image

The fourth step is to display the downloaded image of the recognition result on the Android application.

4 Evaluation

We equip the proposal system on android, uses Xperia XZ1 Compact with 4GB memory and 32GB storage. Five OBI rubbing images are employed as the testing image, which is selected from book [16].

For comparing the performance, and TensorFlow Lite system is also equipped on android. The proposal results and the TensorFlow Lite results are shown in Figure 3 and Figure 4, respectively. The detected and recognized OBIs are crossed by a bounding box, which proves two of the system are equipped very well.

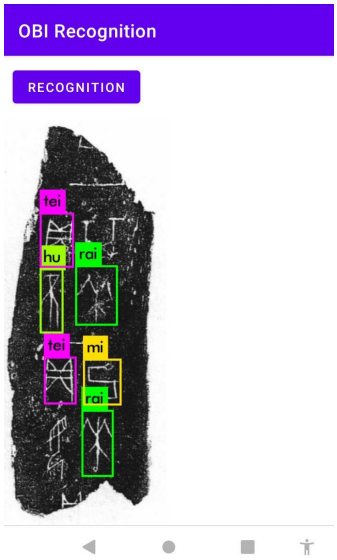

Figure 3: Evaluation results of the proposal

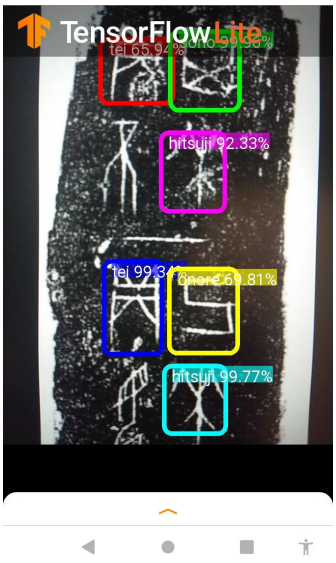

Figure 4: Evaluation results of TensorFlow Lite application

Table 1: Performance of the proposed system

Image name	Image size	Recognition time	Average CPU usage	Average memory usage
00004.jpg	45.07KB	3.078s	5%	48.2MB
2.jpg	46.52KB	2.969s	2%	51.9MB
30.jpg	49.99KB	2.980s	3%	52.4MB
00009.jpg	38.71KB	2.981s	6%	52.2MB
2426-36-1.jpg	256KB	3.571s	6%	51.2MB

4.1 Evaluation results

The performance of evaluation results are shown in Table 1, Table 2.

Recognition time

In terms of recognition time, the TensorFlow Lite app is generally shorter than the proposed system. However, the recognition time of 2426-36-1.jpg is almost the same, indicating that the recognition time of the TensorFlow Lite application is highly dependent on the file size.

CPU usage and memory usage

In terms of average CPU usage, the proposed system uses less than CPU 10% of the CPU in all five images. While the TensorFlow Lite app uses approximately 40 ˜50%.

About the average memory usage, the TensorFlow Lite application uses about 530MB, while the proposed system uses about 50MB which is only 10% of the TensorFlow Lite application used.

Table 2: Performance of TensorFlow Lite system

Image name	Image size	Recognition time	Average CPU usage	Average memory usage
00004.jpg	45.07KB	1.204s	38%	526.8MB
2.jpg	46.52KB	1.044s	50%	521.5MB
30.jpg	49.99KB	1.233s	50%	538.5MB
00009.jpg	38.71KB	1.434s	39%	535.7MB
2426-36-1.jpg	256KB	3.556s	42%	528.9MB

The CPU and memory usage of the proposed system is reduced because Android's tasks are only communication and image display.

Application size and startup time

In terms of Application size, TensorFlow Lite utilizes 289MB of memory. However, the proposed application utilizes 13.72MB is only 4.7% of TensorFlow Lite. Hence, the proposal is a slight application that has a great advantage in memory utilization.

In terms of startup time, the proposed system is 0.572s. However, the TensorFlow Lite application costs 3.147s which is 5.5 times longer than the proposal. It also proves the advantage of our proposal in startup time.

4.2 Discussion

4.2.1 Discussion in Performance Comparison

In terms of the recognition time, the TensorFlow Lite application is shorter than the proposed application. This is because the proposed method relies on communication. The communication flow consists of establishing a connection, waiting for server processing, and receiving the resulting image. The average waiting time for server processing was about 2 seconds. This is because the model of the TensorFlow Lite application starts running immediately after the start of inference, while the model of the proposed method starts running only after the image transmission is completed.

In terms of CPU and memory usage, the TensorFlow lite application is more expensive. Furthermore, Android devices use about 1 to 1.5 GB of memory to run the OS, causing the system cannot provide too much memory to run the other applications. Otherwise, the proposed system only uses a few memory and more stable for users.

Since Android devices currently in widespread use range from those with high specifications to those with low specifications, we consider that a proposed system with stable operation is superior for providing as a service. However, if the specifications of Android devices increase with the development of technology and Android devices with high specifications become common, the operation of the TensorFlow Lite app, which is a stand-alone application, will be guaranteed. It is also possible that research on optimization and efficiency of deep learning models will progress and applications will become even lighter. The stability of the operation of the stand-alone application is an issue for the future, and we conclude that the proposed system using communication, which is currently stable, is better.

4.2.2 Discussions in OBIs Recognition

OBIs Recognition using deep learning has its problems. It is the lack of data sets and the resulting low character detection rate in the object detection model. In the previous study [7], another inference model was prepared and reinforced to compensate for the low detection rate. The data sets for the object detection model has to be created manually, which is not an easy task because it requires knowledge of the crustacean characters and errors occur. The proposed method in this paper uses communication to recognize OBIs; OBIs images are sent from Android devices so that the images can be stored in the server. We can then use the images and recognition results to create a data sets to enhance the recognition model on the server. This is an advantage of the proposed method for OBIs recognition because in a standalone application, it is very difficult to re-train the recognition model because it is implemented in the application and the specification of the device is too low to train it. However, there are many problems that need to be solved, such as how to modify the collected data and how to retrain it. These are our future tasks.

5 Conclusion

Smartphones are cell phones equipped with an operating system and have the highest penetration rate in the world. Therefore, the use of deep learning in smartphones will lead to the further spread of AI-based services. In addition, the demand for image recognition in edge devices and mobile devices has been rising in recent years. This paper proposes an ancient documents text recognition system for Android, using communication with a server. The comparison of performance is done with the image recognition application using TensorFlow Lite. The results show that the recognition time of the proposal is longer than that of the TensorFlow Lite application. However, in the viewpoint of CPU and memory usage, the proposed system can be guaranteed to work even on devices with low performance. Also, in the viewpoint of OBIs recognition, the proposed method was shown to be superior because of the possibility of expanding the data sets. Our future work is to build a better system for relearning using the proposed method and to improve the stability of the operation of the standalone application.

6 Acknowledgments

Part of this work was supported by the Art Research Center of Ritsumeikan University and Key Laboratory of Oracle Bone Inscriptions Information Processing, Anyang Normal University.

References

[1] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffeand Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Vincent Vanhoucke Christian Szegedy, Sergey Ioffe and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4278 – 4284, 2017.

[3] Laurens van der Maaten Gao Huang, Zhuang Liu and Kilian Q. Weinberge. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.

[5] Ziyi Liu, Junfeng Gao, Guoguo Yang, Huan Zhang, and Yong He. Localization and classification of paddy field pests using a saliency map and deep convolutional neural network. Technical report, 02 2016.

[6] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

[7] Yoshiyuki Fujikawa, Hengyi Li, Xuebin Yue, Lin Meng, et al. Recognition of oracle bone inscriptions by using two deep learning models. *arXiv preprint arXiv:2105.00777*, 2021.

[8] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[10] Lin Meng. Two-stage recognition for oracle bone inscriptions. In *Image Analysis and Processing - ICIAP 2017*, pages 672–682. Springer International Publishing, 2017.

[11] Hiroyuki Tomiyama Zhiyu Zhang, Zhichen Wang and Lin Meng. Deep learning and lexical analysis combined rubbing character recognition. In *The 2019 International Conference on Advanced Mechatronic Systems (ICAMechS 2019)*, 2019.

[12] Lyu Bing, Hiroyuki Tomiyama, and Lin Meng. Frame detection and text line segmentation for early japanese books understanding. volume 1, pages 600–606, 01 2020.

[13] Guoying Liu, Jici Xing, and Jing Xiong. Spatial pyramid block for oracle bone inscription detection. pages 133–140, 02 2020.

[14] Lin Meng, Bing Lyu, Zhiyu Zhang, C. V. Aravinda, Naoto Kamitoku, and Katsuhiro Yamazaki. Ocrale bone inscription detector based on ssd. *Proceedings of New Trends in Image Analysis and Processing - ICIAP 2019*, pages 126 – 136, 2019.

[15] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. In *Computer Vision and Pattern Recognition*, 2018.

[16] P.M Zuo. Shanghai bo wu guan cang jia gu wen zi. 2009.