Project Based on Objection Detection and Image Classification: *Writing Using Air Gestures*

Yufei Wang

I. Introduction

This project is a combination of object detection and image classification launched on the iOS edge devices[1]. Through the device's camera, detecting the user's fingertip and tracking the movement of the fingertip with a black line. And then using image classification to recognize the digits that users have written.

II. Objection detection

The method to train the model is using Google Cloud AutoML and export an edge Tensorflow lite file for the trained model and a text file for storing labels. The dataset is composed of 418 pictures of the index finger of my right hand with different backgrounds.

A. First dataset

I used the dataset Fingers[3] by Pavel Koryakin from Kaggle. This dataset contains pictures with one, two, three, four and five fingers. Since what I need is only the picture of fingertips, I used the pictures of one finger from the dataset Fingers as my first dataset. After labeling the positions of fingertips on Google console and training, the result was not as good as I expected. Since the pictures from the dataset Fingers have been preprocessed to grayscale, the model cannot detect fingertips with RGB picture, which caused that the model cannot detect the fingertips from the pictures that I took by my phone.

B. Second dataset

For the second dataset, I decided to take about 100 photos using continuous shooting mode by my phone since I realized that the movement of the fingertip in the final project would be continuous. I used the pictures of my own fingers and the pictures from dataset Fingers as my second dataset. The result was not good to be used in the final project because not only the model cannot detect fingertips but also the model would recognize totally unrelated parts of the picture for test as a fingertip. The reason why the model could not detect fingertips was that I used continuous shooting mode which results in a similar background in the train set. Since object detection depends on the features in the specific areas, the model was possible to consider a similar background as the object that I wanted to detect. With different backgrounds, it was difficult to detect fingertips. In addition, the reason why the model recognized unrelated things as fingertips is that the resolution of the pictures from the dataset Fingers is small, so I had to draw a bigger bounding box including the unrelated black background when I labeled the positions of fingertips on the picture. As a consequence, the model had chances to recognized shadow as fingertips.

C. Third dataset

I removed the pictures from the dataset Fingers and also took more pictures of my finger to increase the number of pictures to 418. I did not use continuous shooting mode for new pictures. I took photos with different backgrounds and different positions of fingertips on the photos. After labeling and training, the model had a high accuracy of test images.

D. Trade-off between computation time and accuracy

I chose the model with the best trade-off. In the project, the movement of fingertips should be smooth, so the response time of object detection is important. Since the size of the dataset for training is not large, I have to choose the model with the best trade-off. Under an ideal situation in which the dataset for training is large, I would choose a model with fast computation. The accuracy can be improved by increasing the size of the dataset, but the computation time depends on the model itself. So, the computation time is more important than the accuracy in this situation. In order to pursue a better writing experience, a fast model should be chosen.

III. Image classification

The method to train the model is resizing the picture for prediction to 112 x 112 before reshaping to 28 x 28. This can reduce the information that loses during reshaping before prediction.

A. Problem with digit classification

There are plenty of tutorials about how to construct a model classifying handwritten digits, but most of them are dealing with the most common dataset whose pictures have a resolution of 28 x 28, which is a very low resolution. The project was tested on iPhone XS with a resolution of 2436 x 1125. The huge gap between the two resolutions results in that the model that trained on pictures with a resolution of 28 x 28 cannot be used on this project. With the model trained on 28 x 28 pictures, the project either had a low confidence

on the right number or had a high confidence on the wrong number (recognize an 8 as 9 with confidence over 80%). The problem is that before the digit classifier makes its prediction, the picture for prediction needs to be preprocessed. The picture needs to be formatted into a 28 x 28 picture with grayscale. After testing, a picture with high resolution (especially without 1:1 aspect ratio) will lose too much information after being reshaped into a 28 x 28 picture.

B. Failed solution

Instead of drawing digits by myself, which is not an efficient way, I found a neural network[2] that can generate larger ideal images based on Compositional Pattern Producing Networks (CPPN) combined with Variation Autoencoder (VAE) and Generative Adversarial Networks (GAN). CPPN can generate images at any resolution. GAN can be trained using the handwritten digits that we already have now to generate new handwritten digits. With CPPN and GAN, the model is capable of generating larger pictures of handwritten pictures, but the model can only generate two or three digits only. VAE is used to penalize the generator if it could not generate all the digits. With this neural network, I decided to generate the dataset of handwritten digits with high resolution that can be used for training new digit classifier for this project. First, I trained the neural network to generate pictures. And then I put pictures in the corresponding directory (e.g. digit 1 in a directory named '1'). Upload the dataset to Google AutoML for training.
However, the thickness of digits also increases as the resolution increases. The thickness of the line for drawing on the phone cannot be too thick, otherwise it will be difficult to control fingers to draw a proper digit.

C. Solution

After the failure of generating a new dataset for training, I tried to change the size of the picture generated after drawing. If I can resize twice rather than resizing from a high resolution to a low resolution directly. I can minimize the probability of losing information. So I resize the image for prediction from 1125 x 2436 to 112 x 112 first and then resize it to 112 x 112 to 28 x 28. As a result, I got a perfect picture for prediction.

D. Further problem to be solved

With traditional digit classification, digits must be in the middle of the picture to get a good confidence for prediction. But it is hard to make every digit be in the middle of the canvas. The problem can be solved by replacing traditional digit classification by digit detection. Using object detection to detect the location of digits and what digits are is an available method to improve the current model.

IV. Extensions

This project can also classify letters, sentences and so on whatever can be generated by writing as long as there are corresponding models. And this project can also be combined with holographic projection. Then people can write things without the restrictions of pens, papers or portable devices. The main idea of this project is fingertips detection. With fingertips detection, we can manipulate smart devices with computer vision. For example, through finger detection, we can adjust the sound volume of televisions remotely without any extra controller. And I think fingertips detection can also be applied to VR games. With a more accurate and faster finger detection or hands detection, users can get rid of heavy controller grips to have a better gaming experience.

V. Conclusion

This paper has introduced the project of writing using air gestures. This project composes of two main technologies in computer vision, object detection, and image classification. Using object detection to detect the movement of fingertips and draw lines corresponding to the movement of fingertips. Using image classification to classify what users have written. This paper also includes problems and solutions during development. This project can be extended to many other applications such as VR games and smart devices. The restrictions of this project are the accuracy and computation time of detecting fingertips. Higher accuracy and faster computation can significantly improve the writing experience of this project.

References

[1] *Edge TF Lite iOS tutorial — Cloud AutoML Vision Object Detection*. en. URL: https://cloud.google.com/vision/automl/object-detection/docs/tflite-ios-tutorial (visited on 01/31/2020).

[2] David Ha. "Generating Large Images from Latent Vectors". In: *blog.otoro.net* (2016). URL: http://blog.otoro.net/2016/04/01/generating-large-images-from-latent-vectors/.

[3] Pavel Koryakin. *Fingers*. en. Apr. 2019. URL: https://kaggle.com/koryakinp/fingers (visited on 01/31/2020).