ED Final Project

Mango Classifier Using Raspberry Pi 3

Author: 資工三 406410035 秦紫頤

Abstract. Deep learning is a really popular topic right now. But most of the work is done and deploy on a normal computer. Deploying deep learning on edge devices is rising recently. We think we can make a little application of deep learning on Raspberry Pi. We are working on Mango Classification recently so we proposed a way to deploy the already trained ResNeST model after pruning into Raspberry Pi to classify the quality of the mango. The ultimate goal for this is to replace humans by Raspberry Pi to classify the mango.

I. INTRODUCTION

Deep learning is really popular right now and implemented in many edge devices recently. Due to the Covid-19 that is happening worldwide social distancing becomes a must. Just a few weeks ago researchers in Massachusetts Institute of Technology (MIT) develop a deep learning model that can monitor social distancing in the workplace in edge device. From this, we are curious about if Raspberry Pi is capable of doing anything related to deep learning. Also, currently, we're working on a competition Mango Classification in AICUP 2020. We want to combine my work with embedded system see if this is possible. We make a simple mango grading classifier using Raspberry Pi 3 at the end. Due to the hardware limitation, our work can only test one mango image at a time and output it's grading on the screen. This is a big step in deep learning implementing in Raspberry Pi 3.

II. RELATED WORK

ResNeSt: Split-Attention Networks[1] presents a simple and modular Split-Attention block that enables attention across feature-map groups. By stacking these Split-Attention blocks ResNet-style, this work obtains a new ResNet variant called ResNeSt. ResNeSt is the current state of the art network in image classification on ImageNet.

Deep Learning on a Raspberry Pi for Real-Time Face Recognition[2] describes a fast and accurate pipeline for real-time face recognition that is based on a convolutional neural network (CNN) and requires only moderate computational resources. After training CNN on a desktop PC we employed a Raspberry Pi for the classification procedure.

Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices[3] introduces deploying neural networks on commercial edge devices such as Raspberry Pi and Jetson. This paper showed that Pytorch, Tensorflow, and some popular deep learning libraries are feasible in these edge devices.

III. METHOD

A. *Package Installation in Raspberry Pi 3*

In order to let our Raspberry Pi 3 run Pytorch which is a machine learning framework that accelerates the path from research prototyping to production deployment, we need to install some packages in Raspberry Pi 3. Below is the detailed procedure

1. Download NOOBS for Raspberry Pi and install it (takes less than an hour)
2. Python 3.7 should already be installed in the system already by checking with `python3 --version`
3. Make a new project folder for our final project
4. Make a new python virtual environment in the project folder and activate the environment
   a. `python3 -m venv env`
   b. `source env/bin/activate`
5. Install Pytorch dependencies:
   a. `sudo apt install libopenblas-dev libblas-dev m4 cmake cython python3-dev python3-yaml python3-setuptools python3-wheel python3-pillow python3-numpy`
   b. `sudo apt install libatlas3-base`
   c. `sudo pip3 install numpy`
   d. `python3 -m pip install Pillow==6.1`
6. Download wheel files of Pytorch 1.3 and torchvision 0.4 online
7. Install Pytorch and torchvision
   a. `pip3 install torch-1.3.0a0+deadc27-cp37-cp37m-linux_armv7l.whl`
   b. `pip3 install torchvision-0.4.0a0+d31eafa-cp37-cp37m-linux_armv7l.whl`
8. Create a requirements.txt include: beautifulsoup4, bottleneck, fastprogress>=0.2.1, matplotlib, numexpr, nvidia-ml-py3, packaging, pandas, pyyaml, requests, scipy
9. Install all the packages inside the requirements.txt by `pip3 install -r requirements.txt`

B. *Deep Learning Network for Mango Classification*

We use ResNeST-101 as our backbone network because ResNeST is the state of the art network in image classification in ImageNet. I use ResNeST rather than ResNeST-50, ResNeST-200 or ResNeST-269 is because ResNeST-50's accuracy isn't enough and ResNeST-200 and ResNeST-200 and ResNeST-269 is too deep. ResNeST is an attention version of ResNet by stacking split-attention blocks in ResNet-style. It splits the original ResNet into various cardinality groups and there are various splits in each cardinal group. We do split attention in each split and each cardinal group can be obtained by fusing via an element-wise summation across multiple splits. At last, we concatenate all the cardinal groups and add a softmax layer at the end to return the classification probabilities for each class. Since

this network just published two months ago so doing transfer learning by using pre-trained isn't available right now and this did affect a bit to our accuracy.

When we deploy our model in Raspberry Pi 3 the first time, Raspberry Pi 3 crash before outputting the result, we think this is because Pi 3 isn't powerful enough to execute this deep network. We don't want to give up this network so we use a pruning algorithm while training to decrease the model parameters in order to reduce the model size. The idea is coming from this paper **Comparing Rewinding and Fine-Tuning in Neural Network Pruning[4]** from Renda et al. The concept of this pruning algorithm is really simple "Train the model, prune its weakest connections, retrain the model at its fast, early training rate, and repeat, until the model is as tiny as you want." This did reduce the model size without sacrificing accuracy.

C. Testing Program

We write a simple testing program in Raspberry Pi 3 that can get a mango image and forward it to the classification model and predicts its' grading level and at last output the result on the screen. Evaluating an image takes about 5-10 minutes and also when it starts to evaluate the second image, Raspberry Pi 3 will crash. We think it is because although we have done pruning to the model, but the model isn't small enough. Maybe done more pruning can solve the problem but we haven't tried this. As for now, this testing program can only evaluate an image a time.

The detail of the testing program is as follows:

1. Load the model
2. Load the image
3. Data transform (Resize, ToTensor, Normalize)
4. Forward the image to the model
5. The output is the probability of the image in each class
6. Get the highest probability class as the classification result
7. Print the result on the screen

D. Detail Procedure

1. Install NOOBS Linux system that works on Raspberry Pi 3
2. Install Pytorch dependencies on Raspberry Pi 3
3. Install Pytorch and torchvsion on Raspberry Pi 3
4. Train our mango classification model with the pruning algorithm on the computer
5. Put the model.pth and test image in Raspberry Pi 3
6. Write our testing program on Raspberry Pi 3
7. Execute the testing program
8. Classification result will show on the screen

E. Experiments

The training phase will be conducted in the Ubuntu Linux system and using Nvidia GTX1080Ti as our GPU device. We are training with 270 epochs, with batch size 24, learning rate 0.009375, and

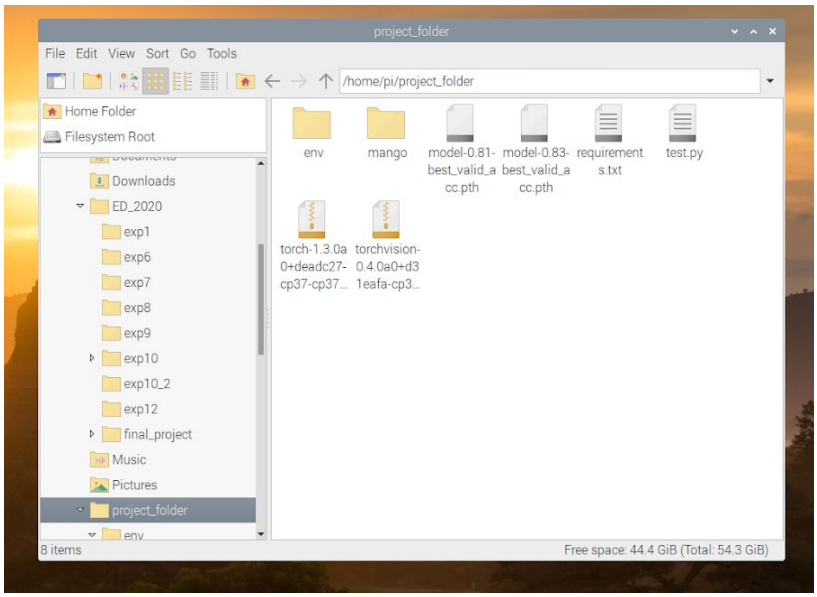
CosineAnnealingLR as our learning rate scheduler. We will test our result on Raspberry Pi with NOOBS Linux system installed in it.

F. Dataset

The dataset I am using is released from AICUP 2020 Mango Classification competition. Because this is not an open dataset so it shouldn't be public used by other people. There are 5600 training images and 800 development images. We will use the 5600 training image to train our model and randomly select some images from the development set to test our model in Raspberry Pi.

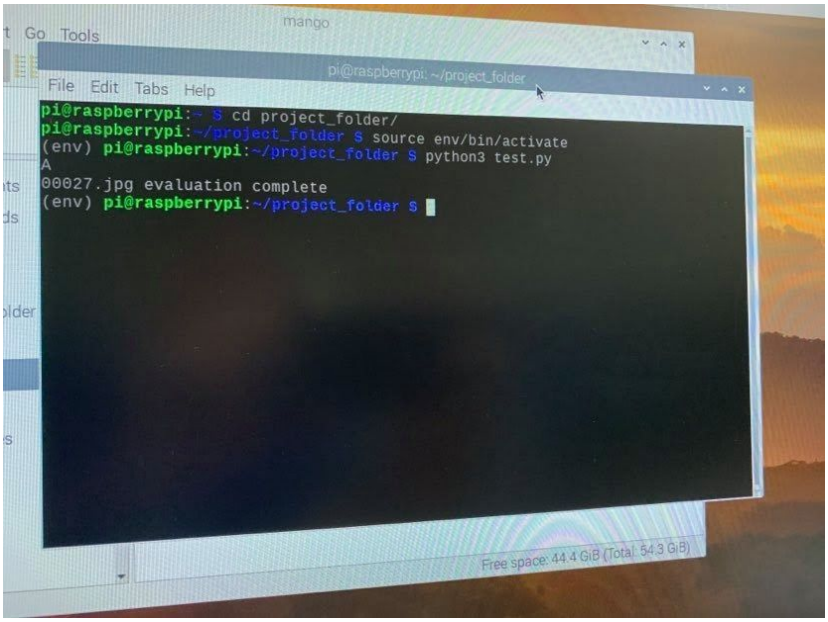
IV. Results

The image below is what it's like in my project folder.



- env folder is the python virtual environment, inside are all the packages I will need to execute the testing program.
- mango folder has one mango image that is for testing
- two model.pth file is the trained model I will use in the testing program
- test.py is the testing program
- the last two files are the wheel files for pytorch and torchvision which can be used in Raspberry Pi

For mango grading classification problem, mango should be classified into high quality (A), medium quality (B), or low-quality (C). After executing the testing program, it will output the mango's quality level based on its raw image. The following image is the result that is shown on the terminal.



You can see the demo video by following this link: [Mango Classifier using Raspberry Pi 3]

REFERENCES

[1] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, Alexander Smola, “ResNeSt: Split-Attention Networks”, 2020

[2] Oliver Dürr, Yves Pauchard, Diego Browarnik, Rebekka Axthelm, “Deep Learning on a Raspberry Pi for Real-Time Face Recognition”, EG conference, 2015

[3] Ramyad Hadidi, Jiashen Cao, Yilun Xie, Bahar Asgari, Tushar Krishna, Hyesoon Kim, “Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices”,IEEE International Symposium on Workload Characterization, (IISWC), 2019

[4] Alex Renda, Jonathan Frankle, Michael Carbin, “Comparing Rewinding and Fine-Tuning in Neural Network Pruning”, International Conference on Learning Representations, (ICLR), 2020

[5]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition, Computer Vision and Pattern Recognition (CVPR), 2016

Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace: https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers-monitor-social-distancing-in-the-workplace/

Quick setup instructions for installing PyTorch and fastai on Raspberry Pi 4: https://medium.com/analytics-vidhya/quick-setup-instructions-for-installing-pytorch-and-fastai-on-raspberry-pi-4-5ffbe45e0ae3