Image Processing at ISCAS

An Exploration to Develop Neural Networks for Personal Devices

26/10/2019

Preface

We thank our supervisor Dr. Libo Zhang for his invaluable guidance and our seniors at ISCAS for their generous support

Introduction: Existing Method

Existing popular image recognition methods are deep learning based

Otherwise require detailed understanding of the subject

Motivation: potential demand

Privacy, convenience, speed

Federated Learning

Benefit individual developers

Aim: Our project

Time is short (5 weeks)

- Restricted to personal devices

Explore and share some practical methodologies in developing “small” and “fast” neural networks

Task: hand pose detection

Illustrate with a web game

Related work: Some reviewed models

Image classification

- AlexNet, VGG16, ResNet, MobileNet

Object detection

- R-CNN, Fast R-CNN, Faster R-CNN, YOLO, SSD

Practical Concerns: Datasets

Representativeness

Surroundings

Figure 1: different hands under the same background (Bruère-Terreault, 2019)

Figure 2: the same hand under different backgrounds (alish_manandhar, 2019)

Practical Concerns: Model Size

Loading time

Processing time

Training efficiency

Category	No. in training set (~70%)	No. in validation set (~30%)	Total size/MB
Paper	1,075	400	136
Rock	1,144	400	134
Scissors	1,155	400	131

Table 1: Summary statistics of the final dataset

Model	No. of parameters	Depth/No. of layers	Size/MB
MobileNet	3,228,864	88	39
ResNet50	23,587,712	52	283
VGG16	14,714,688	23	177

Table 2: Size statistics of MobileNet, ResNet50 and VGG16 as Keras application (keras-team, 2019)

Methodology: Depthwise Separable Convolution

Factorize a standard convolution into a depthwise convolution and a 1×1 pointwise convolution

- use 8 to 9 times less computation under a 3×3 kernel

Transfer learning

- Common practice in computer vision

Methodology: NEtwork architecture

Layer/type	Input size	Output size	No of parameters
MobileNet (backbone)	224× 224 × 3	7 × 7 × 1024	3228864
Global average pooling	7 × 7 × 1024	1 × 1 × 1024	0
Dropout	1024	1024	0
Softmax	1024	3	3075

Table 3: Architecture of our hand pose classification network

MobileNet architecture (Howard et. al., 2017)

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Experiment

Preprocess: resize and random flipping

Validation set: 400x3 images (equally split between sources)

Underlying network	Accuracy	Speed/fps	Size/MB	Environment
Vanilla 5-block CNN	66.67%	124.15	334	NVIDIA Tesla V100 SXM2 32 GB
MobileNet	98.56%	131.36	39	
ResNet50	89.22%	123.06	283	
VGG16	66.67%	111.61	177	

Table 4: Performance statistics of our model with different backbones

Online performance

Try the game here

API loading time problem

Integrity problem