DeepXCam: Very Fast CNN-based Mobile Applications: Multiple Style Transfer and Object Recognition

Ryosuke Tanno, Wataru Shimoda, Keiji Yanai  The University of the Electro-Communications, Tokyo

1. Objective
Common Features of All the Apps
- Standalone CNN mobile applications (no external server required)
- Speeding up by multi-threading and fast framework
- Recognizing any size of images by multi-scale Fully Convolutional Network
- Significant reduction in memory requirements
- Being applicable to various kinds of mobile devices

Example: 100-class food recognition
- recognition time: 26.2ms(iPhone7Plus), top-5 accuracy: 91.5%

2. Proposal Contents

Anyone can build very fast CNN-based mobile apps including object recognition apps and style transfer apps.

Prepare a training image data
Train a CNN model by Caffe (or DIGITS)
Generate a C source code by Caffe2C automatically
Prepare a GUI code of mobile app
Generate CNN-based image recognition app by compiling the generated C code and the GUI code

Devolved in our lab

- Caffe2C / Chainer2C
  - convert parameter files to C source codes that run on mobile devices
- Very fast CNN-based mobile recognition/transfer engine
  - speeding up by multi-threading and fast framework
- Adopting NIN architecture for a recognition engine
  - any size of input images
  - the trade-off between accuracy and processing time by changing input image sizes

If you prepare training data, you can create mobile recognition apps in a day!!

3. DeepXCam for recognition (X = Food, Dog, Bird, Flower)

- Training DCNN
  - Use Network-In-Network(NIN)[3] considering mobile implementation
  - Save the size of the network parameters

Network In Network [3]
- only conv layers
- no FC layers
- relatively smaller than the other architectures
  - any image size correspondence

- Pre-trained CNNs with ImageNet 2000 category images
(totally 2.1 million images)

- Speeding up Conv layers ⇒ Speeding up GEMM
  - computation of conv layers is decomposed into “im2col” operation and matrix multiplications
  - BLAS(iOS: Accelerate Framework, Android: OpenBLAS)
  - we use the NEON instruction set which can execute four multiplications and accumulating calculations at once.
  - iOS: 2Core*4 = 8 calculation, Android: 4Core*4 = 16 calculation

4. Accuracy and Recognition Time

UCF-FOOD100 class recognition performance

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>98.5%</td>
<td>96.0%</td>
<td>94.5%</td>
<td>93.0%</td>
<td>91.5%</td>
<td>90.0%</td>
<td>88.5%</td>
<td>87.0%</td>
<td>85.5%</td>
</tr>
<tr>
<td>10.0%</td>
<td>9.5%</td>
<td>9.0%</td>
<td>8.5%</td>
<td>8.0%</td>
<td>7.5%</td>
<td>7.0%</td>
<td>6.5%</td>
<td>6.0%</td>
<td>5.5%</td>
</tr>
</tbody>
</table>

Trade-Off between Accuracy and Recognition Time

Input Image Size  227x227  200x200  180x180  160x160

<table>
<thead>
<tr>
<th></th>
<th>iPhone 7 Plus</th>
<th>iPad Pro</th>
<th>iPhone SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>227x227</td>
<td>55.7ms (56.9ms)</td>
<td>67.6ms (72.1ms)</td>
<td>76.0ms (81.3ms)</td>
</tr>
<tr>
<td>200x200</td>
<td>42.1ms (47.9ms)</td>
<td>49.7ms (54.0ms)</td>
<td>44.0ms (48.3ms)</td>
</tr>
<tr>
<td>180x180</td>
<td>35.5ms (40.2ms)</td>
<td>44.0ms (48.3ms)</td>
<td>36.2ms (40.4ms)</td>
</tr>
<tr>
<td>160x160</td>
<td>26.2ms (30.4ms)</td>
<td>32.6ms (36.7ms)</td>
<td>28.0ms (32.2ms)</td>
</tr>
</tbody>
</table>

Accuracy (top-5)  95.2%  95.1%  94.1%  91.5%

4. DeepStyleCam (Image Style Transfer)

- ConvDeconvNetwork[2] can treat only one fixed style.
  - If transferring ten kinds of styles, we have to train ten different ConvDeconvNetwork independently.
  - This isn’t good for mobile implementation(required memory size)
- We modified [2] can train multiple styles at the same time
  - adding a fusion layer and a style input stream(inspired by [1])
- Training
  - We input sample images to the content stream and style images to the style stream.(The training method is the same as [2])
  - We shrunk the ConvDeconvNetwork compared to [2]
    - added one down-sampling layer and up-sampling layer
    - replaced 9x9 kernels with smaller 5x5 kernels in the first and last convolutional layers
    - reduced five Residual Elements into three

Multi Style Transfer and Object Recognition App

- Food Rec App (both iOS/Android)
  - Our Project page
  - http://foodcam.mobi
  - Please search “DeepFoodCam”

- Multi Style Transfer (only iOS)
  - Our Project page
  - Please search “RealTimeMultiStyleTransfer”

Reference