Deep Learning On Mobile
A Practitioner’s Guide
Siddha Ganju
September 2020
Deep Learning On Mobile

A Practitioner’s Guide

Siddha Ganju

September 2020
© 2020 Practical Deep Learning for Cloud, Mobile and Edge; http://practicaldeeplearning.ai/
Why Deep Learning on Mobile?

- Privacy
- Reliability
- Cost
- Latency
Latency Is Expensive!

[Amazon 2008]
Latency Is Expensive!

Load time >3 sec

Bounce 53%

[Google Research, Webpagetest.org]
Power of 10

0.1s  Seamless
1s  Uninterrupted flow of thought
10s  Limit of attention

[Miller 1968; Card et al. 1991; Nielsen 1993]
High Quality Dataset + Hardware + Efficient Mobile Inference Engine + Efficient Model = DL App
How do I train my model?
Learn to Play Melodica

3 Months
Already Play Piano?
FINE TUNE
Your skills

3 Months 1 Week
Fine tuning

Assemble a dataset

Find a pre-trained model

Fine-tune a pre-trained model

Run using existing frameworks

Don’t Be A Hero
— Andrej Karpathy
CustomVision.ai

Upload Images
Bring your own labeled images, or use custom vision to quickly add tags to any unlabeled images

Train
Use your labeled images to teach custom vision the concepts you care about

Evaluate
Use simple REST API calls to quickly tag images with your new custom computer vision model

89% 93% 91%

Use Fatkun Browser Extension to download images from Search Engine, or use Bing Image Search API to programmatically download photos with proper rights
<table>
<thead>
<tr>
<th>Tag</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>afghan_hound</td>
<td>87.5%</td>
<td>92.0%</td>
</tr>
<tr>
<td>airedale</td>
<td>96.0%</td>
<td>92.5%</td>
</tr>
<tr>
<td>basenji</td>
<td>97.4%</td>
<td>93.0%</td>
</tr>
<tr>
<td>bernese_mountain_dog</td>
<td>91.3%</td>
<td>91.0%</td>
</tr>
<tr>
<td>entlebucher</td>
<td>97.2%</td>
<td>87.5%</td>
</tr>
<tr>
<td>great_pyrenees</td>
<td>87.7%</td>
<td>85.0%</td>
</tr>
<tr>
<td>irish_wolfhound</td>
<td>87.8%</td>
<td>85.0%</td>
</tr>
<tr>
<td>leonberg</td>
<td>98.9%</td>
<td>87.0%</td>
</tr>
<tr>
<td>maltese_dogs</td>
<td>96.4%</td>
<td>91.5%</td>
</tr>
<tr>
<td>pomeranian</td>
<td>97.4%</td>
<td>91.5%</td>
</tr>
</tbody>
</table>
How do I run my models?
Core ML

TF Lite

ML Kit
Apple Ecosystem

<table>
<thead>
<tr>
<th>Metal</th>
<th>BNNS + MPS</th>
<th>Core ML</th>
<th>Core ML 2</th>
<th>Core ML 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>2016</td>
<td>2017</td>
<td>2018</td>
<td>2019</td>
</tr>
</tbody>
</table>

- Tiny models (~ KB)!
- 1-bit model quantization support
- Batch API for improved performance
- Conversion support for MXNet, ONNX
- tf-coreml

Apple Ecosystem

- On-device training
- Personalization
- Create ML UI
Core ML Benchmark

Core ML Runtime Speed by OS

Relative speed across devices

![Bar chart comparing Core ML runtime speed for iOS 11 and iOS 12](https://heartbeat.fritz.ai/ios-12-core-ml-benchmarks-b7a79811aac1)

Execution Time (MS) ON Apple Devices

![Bar chart showing execution time for different models and devices](https://heartbeat.fritz.ai/ios-12-core-ml-benchmarks-b7a79811aac1)

- **Inception v3**
  - iPhone 5s (2013): 727
  - iPhone 6 (2014): 637
  - iPhone 6s (2015): 538
  - iPhone X (2017): 557
  - iPhone XS (2018): 129
  - iPhone 11 Pro (2019): 129

- **Resnet-50**
  - iPhone 5s (2013): 114
  - iPhone 6 (2014): 90
  - iPhone 6s (2015): 77
  - iPhone X (2017): 74
  - iPhone XS (2018): 26
  - iPhone 11 Pro (2019): 44

- **MobileNet**
  - iPhone 5s (2013): 28
  - iPhone 6 (2014): 24
  - iPhone 6s (2015): 20
  - iPhone X (2017): 18
  - iPhone XS (2018): 14

**GPUs became a thing here!**
TensorFlow Ecosystem

- **TensorFlow**: Smaller
- **TensorFlow Mobile**: Faster
- **TensorFlow Lite**: Minimal dependencies
- **2015**: Allows running custom operators
TensorFlow Lite is small

1.5MB
TensorFlow Mobile

300KB
Core Interpreter + Supported Operations
TensorFlow Lite is Fast

- Takes advantage of on-device hardware acceleration
- FlatBuffers
  - Reduces code footprint, memory usage
  - Reduces CPU cycles on serialization and deserialization
  - Improves startup time
- Pre-fused activations
  - Combining batch normalization layer with previous convolution
- Static memory and static execution plan
  - Decreases load time
TensorFlow Ecosystem

- TensorFlow
  - 2015
- TensorFlow Mobile
  - 2016
- TensorFlow Lite
  - 2018

- Smaller
- Faster
- Minimal dependencies
- Allows running custom operators
$ tflite_convert --keras_model_file = keras_model.h5 --output_file=foo.tflite
TensorFlow Ecosystem

2015

Trained TensorFlow Model

2016

TF Lite Converter

. tflite model

2018

Android App

iOS App
var vision = Vision.vision()
let faceDetector = vision.faceDetector(options: options)
let image = VisionImage(image: uiImage)
faceDetector.process(visionImage) { // callback }
How do I keep my IP safe?
Full fledged mobile lifecycle support

Deployment, instrumentation, etc. from Python

- Image Labeling
- Image Segmentation
- Object Detection
- Style Transfer
- Pose Estimation
- Analytics + Monitoring
- Model Management
- Model Protection
Does my model make me look fat?

Apple does not allow apps over 200 MB to be downloaded over cellular network

Download on demand, and interpret on device instead
What effect does hardware have on performance?
Big things come in small packages

<table>
<thead>
<tr>
<th>Geekbench Browser</th>
<th>Geekbench Browser</th>
<th>Geekbench Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Google Pixel 3</strong></td>
<td><strong>iPhone 11 Pro</strong></td>
<td><strong>MacBook Air (Late 2018)</strong></td>
</tr>
</tbody>
</table>

### System Information

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Model</th>
<th>Motherboard</th>
<th>Memory</th>
<th>Processor Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android 9</td>
<td>Google Pixel 3</td>
<td>Stellele</td>
<td>3549 MB</td>
<td>Qualcomm Qualcomm</td>
</tr>
</tbody>
</table>

**Single-Core Score** | **Multi-Core Score**
---|---
2377 | 8356

**System Information**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Model</th>
<th>Motherboard</th>
<th>Memory</th>
<th>Processor Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS 13.1</td>
<td>iPhone 12,3</td>
<td>D421AP</td>
<td>3759 MB</td>
<td>ARM</td>
</tr>
</tbody>
</table>

**Single-Core Score** | **Multi-Core Score**
---|---
5472 | 13840

**System Information**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Model</th>
<th>Motherboard</th>
<th>Memory</th>
<th>Processor Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>macOS 10.14.5 (Beta 18F132)</td>
<td>Macbook Air (Late 2018)</td>
<td>Apple Inc.</td>
<td>8150 MB 2133 MHz LPDDR3</td>
<td>Intel Core i5-8210Y</td>
</tr>
</tbody>
</table>

**Single-Core Score** | **Multi-Core Score**
---|---
4213 | 7855

**System Information**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Model</th>
<th>Motherboard</th>
<th>Memory</th>
<th>Processor Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Inc.</td>
<td>Apple Inc.</td>
<td>Apple Inc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Single-Core Score** | **Multi-Core Score**
---|---
| |
Effect of Hardware

L-R: iPhone XS, iPhone X, iPhone 5

https://twitter.com/matthieuroff/status/1126575118812110854?s=11
TensorFlow Lite benchmarks

Alpha Lab releases Numericcal: http://alpha.lab.numericcal.com/
TensorFlow Lite benchmarks


<table>
<thead>
<tr>
<th>Model</th>
<th>CPU</th>
<th>RAM</th>
<th>Year</th>
<th>Android</th>
<th>Test 1. ms (C)</th>
<th>Test 2. ms (F)</th>
<th>Test 3. ms (C)</th>
<th>Test 4. ms (F)</th>
<th>Test 5. ms (C)</th>
<th>Test 6. ms (F)</th>
<th>Test 7. ms (F)</th>
<th>Test 8. ms (F)</th>
<th>Test 9. px (F)</th>
<th>AI-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huawei P20 Pro</td>
<td>HiSilicon Kirin 970 (970PULI)</td>
<td>6GB</td>
<td>2018</td>
<td>8.1</td>
<td>144</td>
<td>120</td>
<td>2634</td>
<td>279</td>
<td>241</td>
<td>4390</td>
<td>779</td>
<td>193</td>
<td>6</td>
<td>6532.57</td>
</tr>
<tr>
<td>Huawei Honor 10</td>
<td>HiSilicon Kirin 970 (970PULI)</td>
<td>4GB</td>
<td>2018</td>
<td>8.1</td>
<td>107</td>
<td>140</td>
<td>2293</td>
<td>277</td>
<td>239</td>
<td>4770</td>
<td>606</td>
<td>194</td>
<td>6</td>
<td>6532.57</td>
</tr>
<tr>
<td>Huawei P20</td>
<td>HiSilicon Kirin 970 (970PULI)</td>
<td>4GB</td>
<td>2018</td>
<td>8.1</td>
<td>126</td>
<td>133</td>
<td>2541</td>
<td>273</td>
<td>242</td>
<td>5104</td>
<td>742</td>
<td>195</td>
<td>6</td>
<td>6532.57</td>
</tr>
<tr>
<td>Mediatek P80 Dev Platform</td>
<td>Mediatek Helio P80</td>
<td>4GB</td>
<td>2018</td>
<td>8.1 proto</td>
<td>21</td>
<td>439</td>
<td>2230</td>
<td>846</td>
<td>1419</td>
<td>4499</td>
<td>394</td>
<td>1582</td>
<td>5</td>
<td>2259.34</td>
</tr>
<tr>
<td>OnePlus 6</td>
<td>Snapdragon 845</td>
<td>8GB</td>
<td>2018</td>
<td>9.0 proto</td>
<td>24</td>
<td>802</td>
<td>1356</td>
<td>928</td>
<td>1999</td>
<td>3886</td>
<td>303</td>
<td>1244</td>
<td>5</td>
<td>2061.74</td>
</tr>
<tr>
<td>Google Pixelbook</td>
<td>Intel Core i7-7700K</td>
<td>8GB</td>
<td>2017</td>
<td>7.1</td>
<td>75</td>
<td>613</td>
<td>1420</td>
<td>1357</td>
<td>3368</td>
<td>3886</td>
<td>288</td>
<td>1489</td>
<td>14</td>
<td>1734.98</td>
</tr>
<tr>
<td>HTC U12+</td>
<td>Snapdragon 845</td>
<td>8GB</td>
<td>2018</td>
<td>8.0</td>
<td>60</td>
<td>620</td>
<td>1433</td>
<td>1229</td>
<td>2792</td>
<td>3542</td>
<td>329</td>
<td>1485</td>
<td>11</td>
<td>1708.49</td>
</tr>
<tr>
<td>Asus Zenfone 5z</td>
<td>Snapdragon 845</td>
<td>8GB</td>
<td>2018</td>
<td>8.0</td>
<td>60</td>
<td>620</td>
<td>1401</td>
<td>1198</td>
<td>2788</td>
<td>3477</td>
<td>326</td>
<td>1439</td>
<td>10</td>
<td>1638.03</td>
</tr>
<tr>
<td>Samsung Galaxy S9+</td>
<td>Exynos 9810 Octa</td>
<td>8GB</td>
<td>2018</td>
<td>8.0</td>
<td>148</td>
<td>1208</td>
<td>1572</td>
<td>958</td>
<td>1872</td>
<td>2430</td>
<td>612</td>
<td>1230</td>
<td>8</td>
<td>1628.07</td>
</tr>
<tr>
<td>Samsung Galaxy S9</td>
<td>Snapdragon 845</td>
<td>8GB</td>
<td>2018</td>
<td>8.0</td>
<td>65</td>
<td>661</td>
<td>1459</td>
<td>1219</td>
<td>2081</td>
<td>3120</td>
<td>311</td>
<td>1592</td>
<td>8</td>
<td>1590.00</td>
</tr>
<tr>
<td>Samsung Galaxy S9</td>
<td>Snapdragon 845</td>
<td>4GB</td>
<td>2018</td>
<td>8.0</td>
<td>63</td>
<td>690</td>
<td>1518</td>
<td>1273</td>
<td>2856</td>
<td>3587</td>
<td>319</td>
<td>1592</td>
<td>7</td>
<td>1539.00</td>
</tr>
</tbody>
</table>
Alchemy by Fritz

```bash
$ fritz model benchmark <path to keras model.h5>
...
Fritz Model Grade Report
Core ML Compatible: True
Predicted Runtime (iPhone X): 31.4 ms (31.9 fps)
Total MFLOPS: 686.90
Total Parameters: 1,258,580
Fritz Version ID: <Version UID>

$ fritz model benchmark --version-uid <Version UID>
```

https://alchemy.fritz.ai/

Python library to analyze and estimate mobile performance

No need to deploy on mobile
Which devices should I support?

To get 95% device coverage, support phones released in the last 4 years.

For unsupported phones, offer graceful degradation (lower frame rate, cloud inference, etc.).
Could all of this result in heavy energy use?

Glitches & Battery!!!
Recently, I haven’t been able to watch videos properly at ALL. The video glitches within the first few seconds, freezing. Then the only way for me to actually get the video to get out of my screen is the skip the video. If this app’s “a way to keep connected with friends”, then at least let me see what my friends are saying!! Also, it’s a mega battery drainer and using it while I’m on trips usually ends up being a pain for me.

Burning through my battery
Your space-time continuum update broke the app. Watching videos burns through the battery and the phone gets very hot. Lost 20% battery watching a 10min video on my iPhone 7 Plus. Other video streaming apps work as usual.
Energy considerations

You don’t usually run AI models constantly; you run it for a few seconds

With a modern flagship phone, running MobileNet at 30 FPS should burn battery in 2–3 hours

Bigger question — do you really need to run it at 30 FPS? Could it be run at 1 FPS?
Energy reduction from 30 FPS to 1 FPS

Percentage GPU utilization with varying frames per second

- SqueezeNet
- MobileNet
- ResNet-50

iPod Pro 2017

© 2020 Practical Deep Learning for Cloud, Mobile and Edge; http://practicaldeeplearning.ai/
What exciting applications can I build?
Audible Barcode recognition

Aim: Help blind users identify products using barcode

Issue: Blind users don’t know where the barcode is

Solution: Guide user in finding a barcode with audio cues

Sarah Russell
@blindbat84

Holy shit, the Seeing AI reading of barcodes is fast as hell. Love that.

10:42 · 7/13/17 · TW Blue
AR Hand Puppets,
Hart Woolery from 2020CV, Object Detection (Hand) + Key Point Estimation

[https://twitter.com/2020cv_inc/status/1093219359676280832]
Object Detection (Ball, Hoop, Player) + Body Pose + Perspective Transformation
Remove objects

Brian Schulman, Adventurous Co.

Object Segmentation + Image Inpainting

https://twitter.com/smashfactory/status/1139461813710442496
Magic Sudoku App

Edge Detection + Classification + AR Kit

https://twitter.com/braddwyer/status/910030265006923776
Snapchat

Face Swap  GANs
Can I make my model even more efficient?
How To Find Efficient Pre-Trained Models

Papers with Code
https://paperswithcode.com/sota

Model Zoo
https://modelzoo.co

© 2020 Practical Deep Learning for Cloud, Mobile and Edge; http://practicaldeeplearning.ai/
Model Pruning

Aim: Remove all connections with absolute weights below a threshold

Song Han, Jeff Pool, John Tran, William J. Dally, "Learning both Weights and Connections for Efficient Neural Networks", 2015

© 2020 Practical Deep Learning for Cloud, Mobile and Edge; http://practicaldeeplearning.ai/
Pruning in Keras

```python
model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(),  
    tf.keras.layers.Dense(512, activation=tf.nn.relu),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation=tf.nn.softmax)  
])

model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(),  
    prune.Prune(tf.keras.layers.Dense(512, activation=tf.nn.relu)),  
    tf.keras.layers.Dropout(0.2),  
    prune.Prune(tf.keras.layers.Dense(10, activation=tf.nn.softmax))  
])
```
So many techniques — So little time!

01 Channel pruning

02 Model quantization

03 ThiNet (Filter pruning)

04 Weight sharing

05 Automatic Mixed Precision

06 Network distillation
Tencent AI Labs created an Automatic Model Compression (AutoMC) framework.
AutoML – Let AI Design an Efficient Arch

- Neural Architecture Search (NAS) — An automated approach for designing models using reinforcement learning while maximizing accuracy.
- Hardware Aware NAS = Maximizes accuracy while minimizing run-time on device
- Incorporates latency information into the reward objective function
- Measure real-world inference latency by executing on a particular platform
- 1.5x faster than MobileNetV2 (MnasNet)
- ResNet-50 accuracy with 19x less parameters
- SSD300 mAP with 35x fewer FLOPs
### Evolution of Mobile NAS Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 Acc (%)</th>
<th>Pixel-1 Runtime</th>
<th>Search Cost (GPU Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV1</td>
<td>70.6</td>
<td>113</td>
<td>Manual</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>72.0</td>
<td>75</td>
<td>Manual</td>
</tr>
<tr>
<td>MnasNet</td>
<td>74.0</td>
<td>76</td>
<td>40,000 (4 years+)</td>
</tr>
<tr>
<td>ProxylessNas</td>
<td>74.6</td>
<td>78</td>
<td>200</td>
</tr>
<tr>
<td>Single-Path NAS</td>
<td>74.9</td>
<td>79.5</td>
<td>3.75 hours</td>
</tr>
</tbody>
</table>

Han Cai and Ligeng Zhu and Song Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware", ICLR 2019
Can I improve my model without accessing user data?
On-Device Training in Core ML

• Core ML 3 introduced on device learning
• Never have to send training data to the server with the help of `MLUpdateTask`
• Schedule training when device is charging to save power

```swift
let updateTask = try MLUpdateTask(
    forModelAt: modelUrl,
    trainingData: trainingData,
    configuration: configuration,
    completionHandler: {
        [weak self] in
        self?.model = context.model
        context.model.write(to: newModelUrl)
    }
)
```
Federated Learning!!!

On-device data can be used to train a smarter central model and improve our users' experience.

But since there's no way we'd wanna bring that data to the server...

...the training can be brought to the device!

https://federated.withgoogle.com/
Train a global model using 1000s of devices without access to data

Encryption + Secure Aggregation Protocol

Can take a few days to wait for aggregations to build up

https://github.com/tensorflow/federated
Mobile AI Development Lifecycle

1. Collect Data
2. Label Data
3. Train Model
4. Convert Model
5. Optimize Performance
6. Deploy
7. Monitor

© 2020 Practical Deep Learning for Cloud, Mobile and Edge; http://practicaldeeplearning.ai/
| 01 | Why deep learning on mobile? |
| 02 | Building a model |
| 03 | Running a model |
| 04 | Hardware factors |
| 05 | Benchmarking |
| 06 | State-of-the-art applications |
| 07 | Making a model more efficient |
| 08 | Federated learning |
How do I access the slides instantly?

http://PracticalDeepLearning.ai

@PracticalDLBook

Icons credit - Orion Icon Library https://orioniconlibrary.com
That’s all, folks!