Targeting CNNs for Embedded Platforms

Anshu Arya, Solution Architect @ MulticoreWare
- Founded 2009
- Core Competency: Heterogeneous Computing
- HQ: Silicon Valley
- Seven Location [US, China, India, and Taiwan]
- 225+ Employees

<table>
<thead>
<tr>
<th>BUSINESS UNITS</th>
<th>Performance Optimization Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning/Neural Networks</td>
<td></td>
</tr>
<tr>
<td>Image Processing [OpenCV]</td>
<td></td>
</tr>
<tr>
<td>Video Codecs [x265]</td>
<td></td>
</tr>
<tr>
<td>Compilers [LLVM, OpenCL]</td>
<td></td>
</tr>
</tbody>
</table>

Global Customers and Partners

- Altera
- AMD
- ARM
- BBright
- Cadence
- Google
- Imagination
- Microsoft
- Movidius
- NVIDIA
- Pegasys
- Qualcomm
- Sorenson
- Synopsys
- Telesstream
- Xilinx
Neural Network Focus Areas

Automotive & ADAS
- Pedestrian Detection
- Vehicle Detection
- Traffic Sign Recognition

Video Quality
- Audio/Video Lip Sync
-Subtitle Sync
- Text ROI Detection

Action Detection
- Facial Expressions
- Sports Pose Detection

Other
- Medical Tool Recognition
Automotive & ADAS

- Pedestrian Detection
- Vehicle Detection
- Traffic Sign Recognition

Video Quality
- Audio/Video Lip Sync
- Subtitle Sync
- Text ROI Detection

Neural Network
Focus Areas

- Action Detection
  - Facial Expressions
  - Sports Pose Detection

Other
- Medical Tool Recognition
Automotive & ADAS

- Pedestrian Detection
- Vehicle Detection
- Traffic Sign Recognition

Video Quality

- Audio/Video Lip Sync
- Subtitle Sync
- Text ROI Detection

Action Detection

- Facial Expressions
- Sports Pose Detection

Other

- Medical Tool Recognition
Automotive & ADAS
- Pedestrian Detection
- Vehicle Detection
- Traffic Sign Recognition

Video Quality
- Audio/Video Lip Sync
- Subtitle Sync
- Text ROI Detection

Action Detection
- Facial Expressions
- Sports Pose Detection

Other
- Medical Tool Recognition
[ Neural Network Services ]

Data Labeling
Curate and label image or video data for input into neural network training.

- In-house team of data labelers to perform any type of labeling task confidentially
- Proprietary machine-assisted labeling tool to increase productivity by an order of magnitude

Design & Training
Design a neural network architecture and train it using labeled data.

- Neural network architecture chosen to fit within constraints of target hardware platform
- Training is done on MulticoreWare GPU-accelerated workstations

Deployment & Upgrades
Integrate the neural network engine into your application and receive on-going upgrades.

- Build an application or integrate a neural network engine into your existing code
- Improve the accuracy of the neural network as you collect more training data

Platform Optimization
Iterate on the neural network architecture and perform hardware-specific optimizations.

- Performance and memory optimizations for target hardware platform
- Code rewrites using hardware intrinsics, assembly, RTL, OpenCL, CUDA, etc.
ADAS Detection Classes
[ Mobile/Embedded Platforms ]

What platform(s) will dominate?

- GPUs
- FPGAs
- Vision DSPs
- Custom ASICs

Current Examples

- NVIDIA Drive PX 2 & Xavier
- Xilinx Zynq UltraScale+
- Cadence VP5 & VP6
- Synopsys DesignWare EV6x
- MobileEye EyeQ 4
[ Challenges for Embedded CNNs ]

Performance / Power / Memory

- Need fast detection (not just classification)

Quality

- Predict accurate bounding boxes
[ Challenges for Embedded CNNs ]

Performance / Power / Memory

- Need fast detection (not just classification)
- Need “smaller” CNN architectures
  - Fewer parameters
  - Fewer operations
  - Lower intermediate memory usage

Quality

- Predict accurate bounding boxes
- General advice: use a network with as many parameters/layers as you can reliably train
Need Fast Detection
Read image

Create object proposals (e.g. Pyramidal, Sliding Window, etc.)

For each proposal:
- Crop from frame
- Pre-process (e.g. warp)
- Run CNN classifier

Post-process (NMS)
Potentially hundreds to thousands of proposals needed to get tight bounding boxes
[ Typical Detection - Cycles ]

- CNN Classifier: 53%
- Generate Proposals: 39%
- Crop Proposals: 8%
- Change the pipeline
  - No need to run classifier on each proposal
  - Re-use convolution feature map across proposals

- Requires RoI Pooling layer
  - Extracts proposal features from full frame map

- Still need to generate proposals via Selective Search (SS)
  - Now even more limited by SS speed

*Image from Girshick, “Fast R-CNN”*
[ Fast Detection – “Faster R-CNN” ]

- State-of-the-Art Localization + Classification
  - 26+ implementations by 2015 & 2016 ImageNet competitors
    - All winners use some variation

- Use a CNN to create proposals
  - RPN (region proposal network)
  - Re-use convolution feature map for localization & classification
  - Uses pre-defined “anchor” boxes to determine bounding box dimensions

- Must be trained in 4 stages

- Eliminates the need for prior object proposals
  - No more Selective Search or EdgeBoxes

*Image from Ren et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”*
## Fast Detection – Pipeline Evolution

<table>
<thead>
<tr>
<th>Typical Detection</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Read image</td>
<td>- Read image</td>
<td>- Read image</td>
</tr>
<tr>
<td>- Create object proposals</td>
<td>- Create object proposals</td>
<td>- Create object proposals</td>
</tr>
<tr>
<td>- <strong>For each</strong> proposal:</td>
<td><strong>For each proposal:</strong></td>
<td><strong>Run CNN classifier</strong></td>
</tr>
<tr>
<td>- Crop from frame</td>
<td>- Crop from frame</td>
<td>- Take as input a list of proposals</td>
</tr>
<tr>
<td>- Pre-process</td>
<td>- Pre-process</td>
<td>- Generate proposals using feature map and region proposal network</td>
</tr>
<tr>
<td>- Run CNN classifier</td>
<td>- Run CNN classifier</td>
<td>- Re-use feature map for classification</td>
</tr>
<tr>
<td>- Run CNN classifier</td>
<td>- Take as input a list of proposals</td>
<td>- Post-process (NMS)</td>
</tr>
<tr>
<td>- Post-process (NMS)</td>
<td>Use RoI pooling layer</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
R-FCN

- Adopts RPN network from Faster-RCNN
- Replaces all fully-connected layers
- Inherits training difficulty of Faster-RCNN
- Comparable quality, but better inference speed
- Showed RPNs can outperform Selective-Search and EdgeBox proposals
- Can be integrated with existing network architectures

*Image from Dai et al., “R-FCN: Object Detection via Region-based Fully Convolutional Networks”*
YOLO/YOLOv2

- No explicit region proposals or RPN
- v2 is fully-convolutional
- Uses k-means to determine best shapes for bounding boxes
- Multi-scale training allows trade-off for lower resolution input and speed vs. higher resolution and accuracy
- Has problems detecting small/overlapping objects

Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts $B$ bounding boxes, confidence for those boxes, and $C$ class probabilities. These predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor.

*Image from Redmon et al., “You Only Look Once: Unified, Real-time Object Detection”*
SSD
- Adds convolutional layers to predict bounding boxes of various scales/aspect ratios
- Fully convolutional
- Performance & Quality > YOLO & < YOLOv2

SqueezeDet
- Uses “convdet” layers inspired by YOLO
- Uses anchor boxes inspired by Faster R-CNN, but uses k-means to improve them
- Fully convolutional
- Performance & Quality comparable to YOLOv2

YOLO/YOLOv2

Faster R-CNN
R-FCN
- Explicit Proposals [ACCURATE]
- Simultaneous Classification and Detection [FAST]
Need “Smaller” Architectures
“Don’t be a hero”
- Sage advice from cs231, Andrej Karpathy
Top-Down Design

- Start with top architectures on ILSVRC
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

### Challenges:
- Reduce weight space ~400x
- Reduce compute by ~100x
- Retain high accuracy

<table>
<thead>
<tr>
<th>Parameter</th>
<th>AlexNet Requirement</th>
<th>Available on Embedded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight Space</td>
<td>~250 MB</td>
<td>128 – 512 Kb</td>
</tr>
<tr>
<td>Operations for 30FPS VGA</td>
<td>~2400 GMAC/s</td>
<td>24 - 32 GMAC/s</td>
</tr>
</tbody>
</table>
Top-Down Design

- Start with top architectures on ILSVRC
- “Shrink” it:
  - Remove layers (especially FC layers)
  - Reduce # of convolution filters
  - Decrease convolution filter size
  - Made easier if fewer detection classes
Toy Example: Detecting Faces

- 1 class
- AlexNet is clearly overkill
- Similar accuracy with smaller network

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AlexNet Type</th>
<th>Shrunken Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight Space</td>
<td>~250MB (500x)</td>
<td>&lt;512Kb (1x)</td>
</tr>
<tr>
<td>Layers</td>
<td>10 (7 CV + 3 FC)</td>
<td>5 (3 CV + 2 FC)</td>
</tr>
<tr>
<td>Compute Time</td>
<td>640x</td>
<td>1x</td>
</tr>
<tr>
<td>Operations per input</td>
<td>832 MMACs</td>
<td>1.3 MMACs</td>
</tr>
</tbody>
</table>
Top-Down Design

- Start with top architectures on ILSVRC
- “Shrink” it:
  - Remove layers (especially FC layers)
  - Reduce # of convolution filters
  - Decrease convolution filter size
  - Made easier if fewer detection classes
- Reduce until it fits into your target compute/memory constraints
- Structured approaches:
  - SVD
  - Pruning
Top-Down Design

- Start with top architectures on ILSVRC

- “Shrink” it:
  - Remove layers (especially FC layers)
  - Reduce # of convolution filters
  - Decrease convolution filter size
  - Made easier if fewer detection classes

- Reduce until it fits into your target compute/memory constraints

- Structured approaches:
  - SVD
  - Pruning

Bottom-Up Design
“Don’t be a hero”
- Sage advice from cs231, Andrej Karpathy
### Top-Down Design

- Start with top architectures on ILSVRC
- “Shrink” it:
  - Remove layers (especially FC layers)
  - Reduce # of convolution filters
  - Decrease convolution filter size
  - Made easier if fewer detection classes
- Reduce until it fits into your target compute/memory constraints
- Structured approaches:
  - SVD
  - Pruning

### Bottom-Up Design

- Consider target hardware and its compute/memory constraints
- Assemble architecture layer-by-layer
- Use top ILSVRC architectures as a guideline
  - Mimic structure/layer patterns
  - Inception modules (Szegedy, et al.)
- Error-prone, could end up with something “untrainable”, leave to the experts
### CNN Architecture Design

**SqueezeNet**
- Back-bone of “SqueezeDet”
- Fully convolutional
- “Fire Modules” use 1x1 convolutions to ”squeeze” a layer before feeding into a mixed 1x1 and 3x3 layer

**Darknet-19**
- Backbone of “YOLOv2”
- Fully convolutional
- Uses 1x1 convolutions to compress feature maps between 3x3 convolutions

<table>
<thead>
<tr>
<th>Classifier</th>
<th>SqueezeNet-Type</th>
<th>Darknet-Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight Space</td>
<td>4MB</td>
<td>100MB</td>
</tr>
<tr>
<td>Runtime Memory</td>
<td>100MB</td>
<td>500MB</td>
</tr>
<tr>
<td>Operations per input</td>
<td>3.6 GMACs</td>
<td>1.4 GMACs</td>
</tr>
</tbody>
</table>
- Energy-Aware Pruning (Yang, et al.)

- Deep Compression (Han, et al.)
  - SqueezeNet shown to compress effectively

- Shortcut Connections (He, et al.)
  - DenseNets, Highway
### Challenges for Embedded CNNs

<table>
<thead>
<tr>
<th>Performance / Power / Memory</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Need fast detection (not just classification)</td>
<td>- Simultaneous classification and detection</td>
</tr>
</tbody>
</table>
| - Need “smaller” CNN architectures  
  - Fewer parameters  
  - Fewer operations  
  - Lower intermediate memory usage | - Pruning, 1x1 convolutions, Bottom-up design with hardware considered |
Contact Me

Anshu Arya
anshu@multicorewareinc.com