www.multicorewareinc.com

Targeting CNNs for Embedded Platforms

Anshu Arya, Solution Architect @ MulticoreWare

MULTICORE WARE

Company Introduction

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

<table>
<tr><th colspan="2">BUSINESS UNITS</th></tr>
<tr><td>Machine Learning/Neural Networks</td><td rowspan="4">Performance Optimization Services</td></tr>
<tr><td>Image Processing [OpenCV]</td></tr>
<tr><td>Video Codecs [x265]</td></tr>
<tr><td>Compilers [LLVM, OpenCL]</td></tr>
</table>

Global Customers and Partners

ALTERA, AMD, ARM, Bright Visionary Technologies

cadence, Google, Imagination, Microsoft

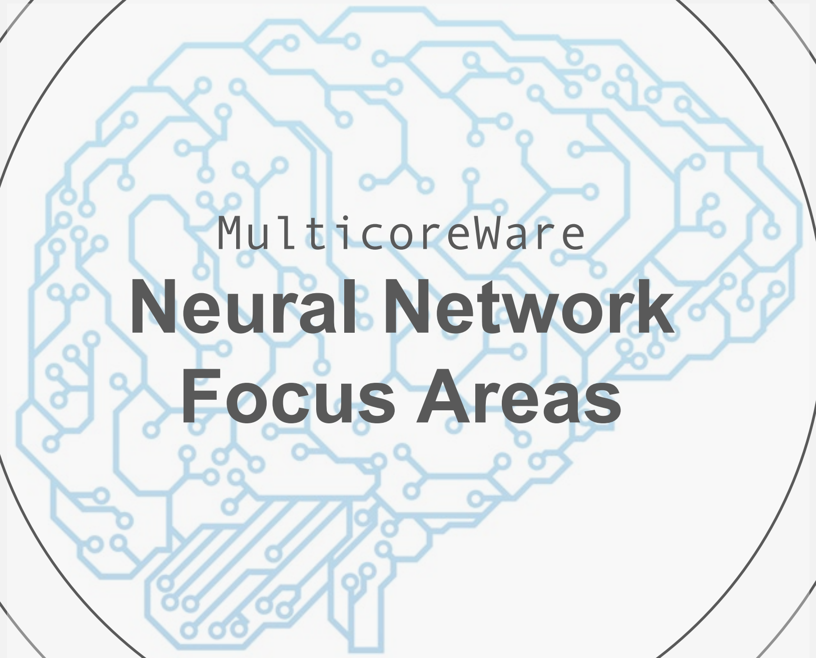
Movidius, NVIDIA, PEGASYS, QUALCOMM

sorenson MEDIA, SYNOPSYS, telestream, XILINX

MULTICORE WARE

MulticoreWare

Neural Network Focus Areas



Automotive & ADAS

- Pedestrian Detection
- Vehicle Detection
- Traffic Sign Recognition

Video Quality

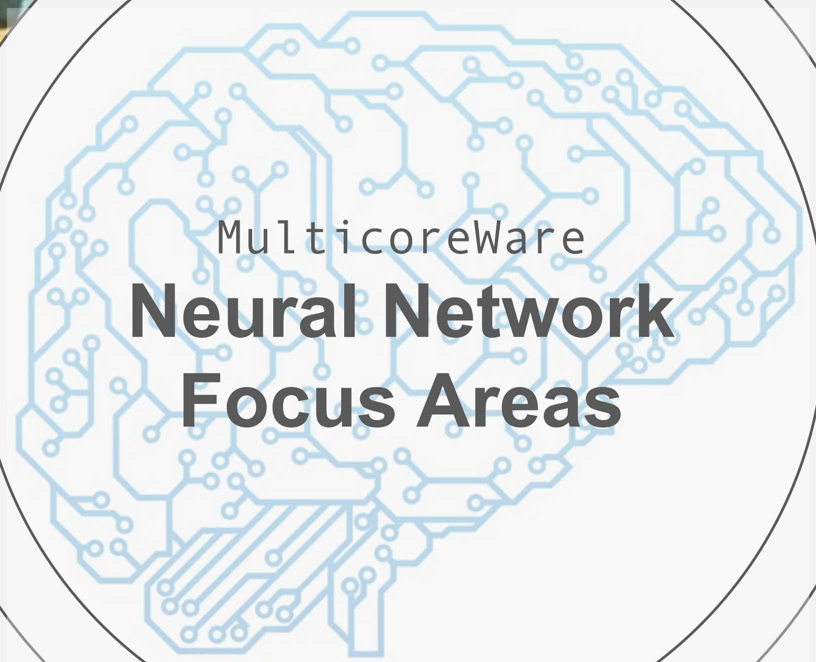
- Audio/Video Lip Sync
- Subtitle Sync
- Text ROI Detection

Action Detection

- Facial Expressions
- Sports Pose Detection

Other

- Medical Tool Recognition



MulticoreWare

Neural Network Focus Areas

Video Quality

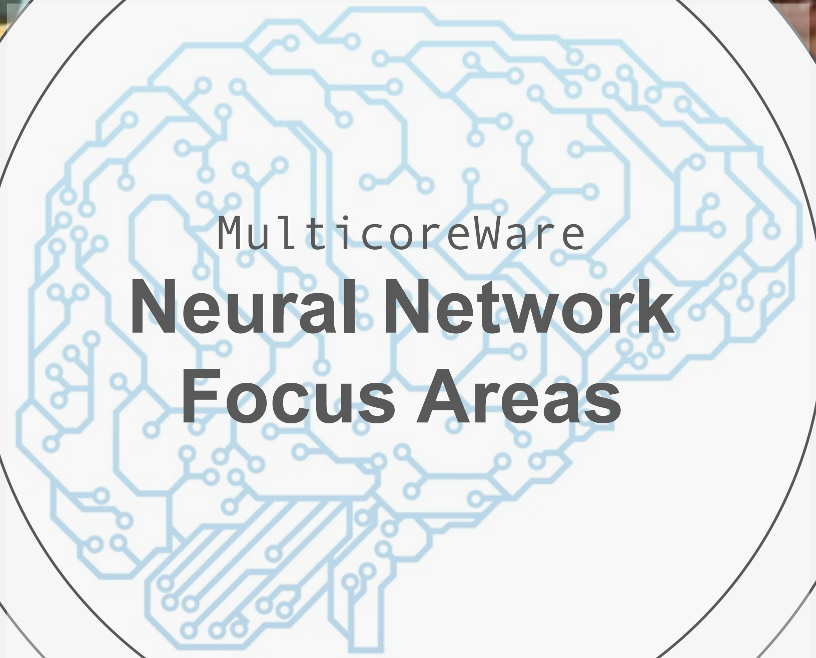
- Audio/Video Lip Sync
- Subtitle Sync
- Text ROI Detection

Action Detection

- Facial Expressions
- Sports Pose Detection

Other

- Medical Tool Recognition



MulticoreWare

Neural Network Focus Areas

Action Detection

- Facial Expressions
- Sports Pose Detection

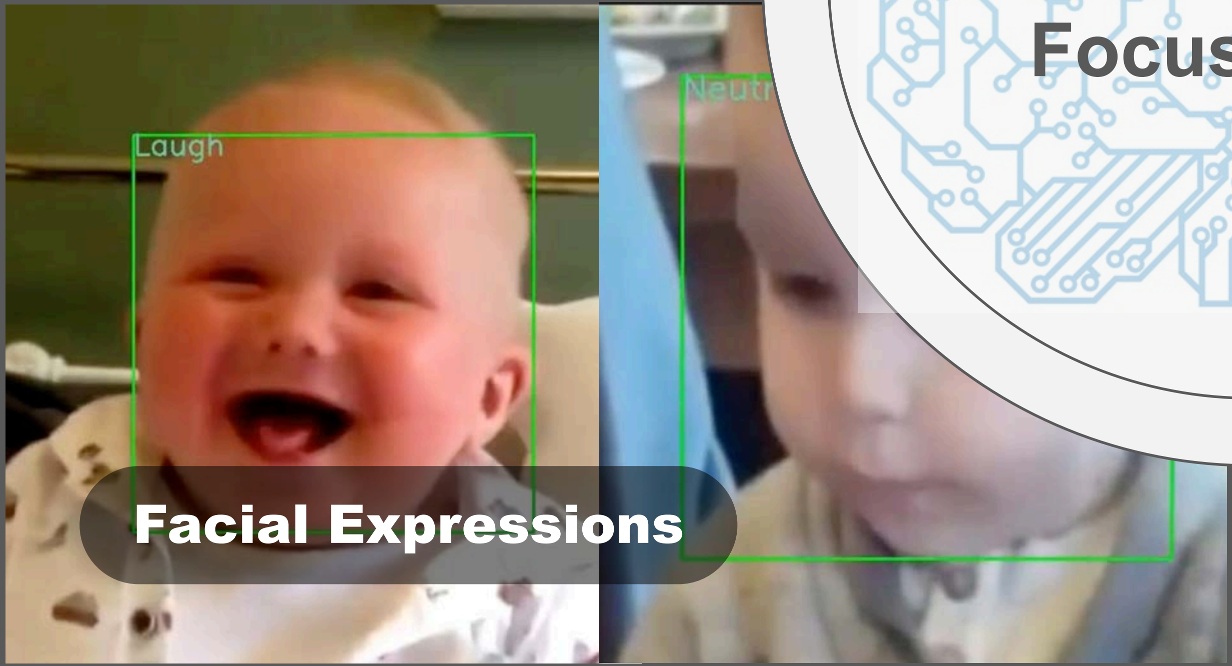
Other

- Medical Tool Recognition

MULTICORE WARE

MulticoreWare

Neural Network Focus Areas



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

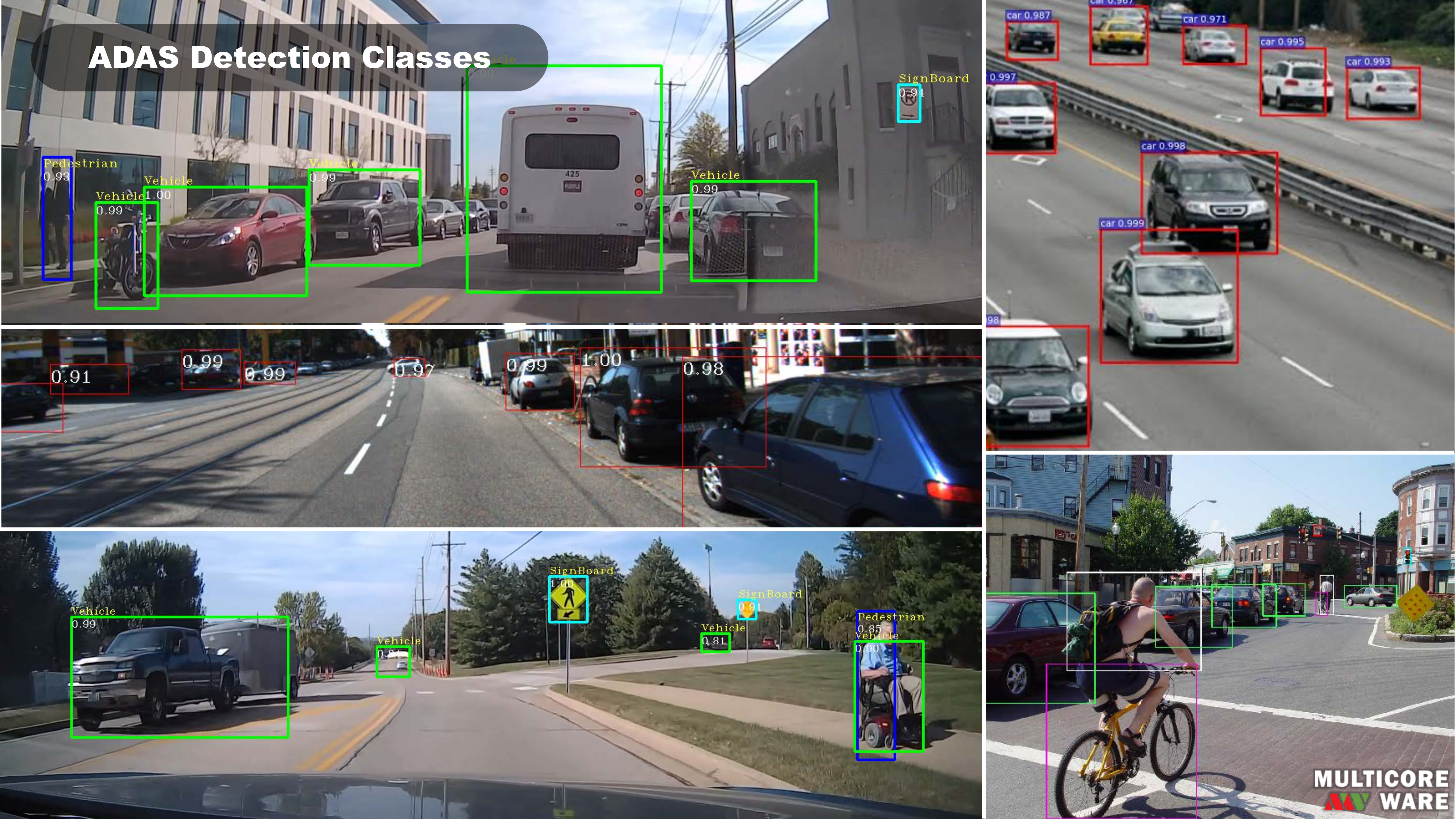
Platform Optimization

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.

MULTICORE WARE

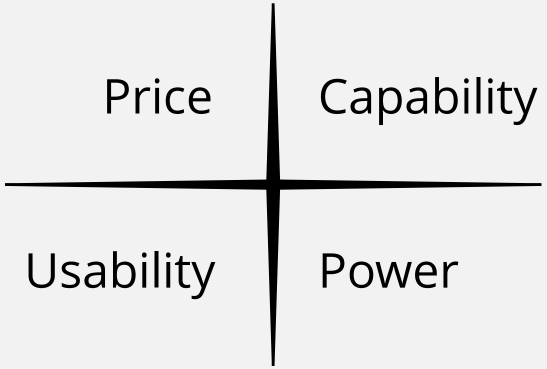
ADAS Detection Classes



[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

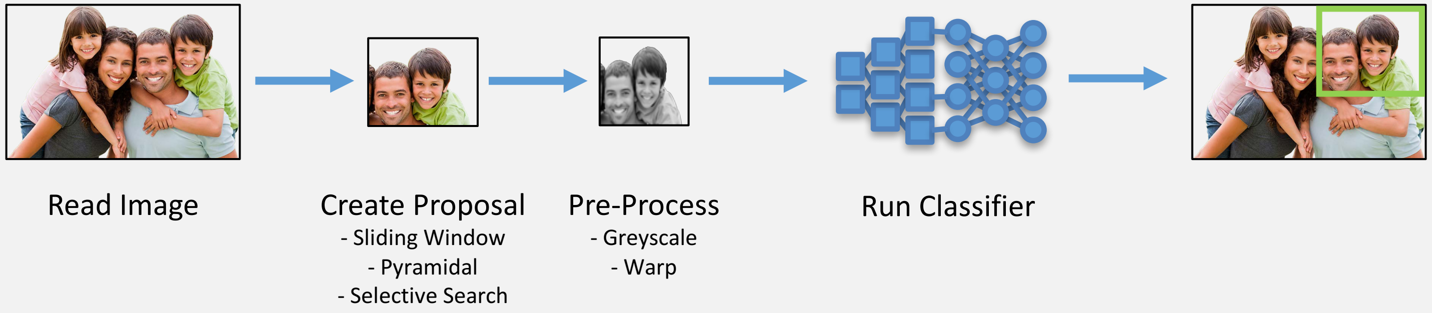
- Need fast detection (not just classification)
- Need “smaller” CNN architectures
  - Fewer parameters
  - Fewer operations
  - Lower intermediate memory usage

Quality

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

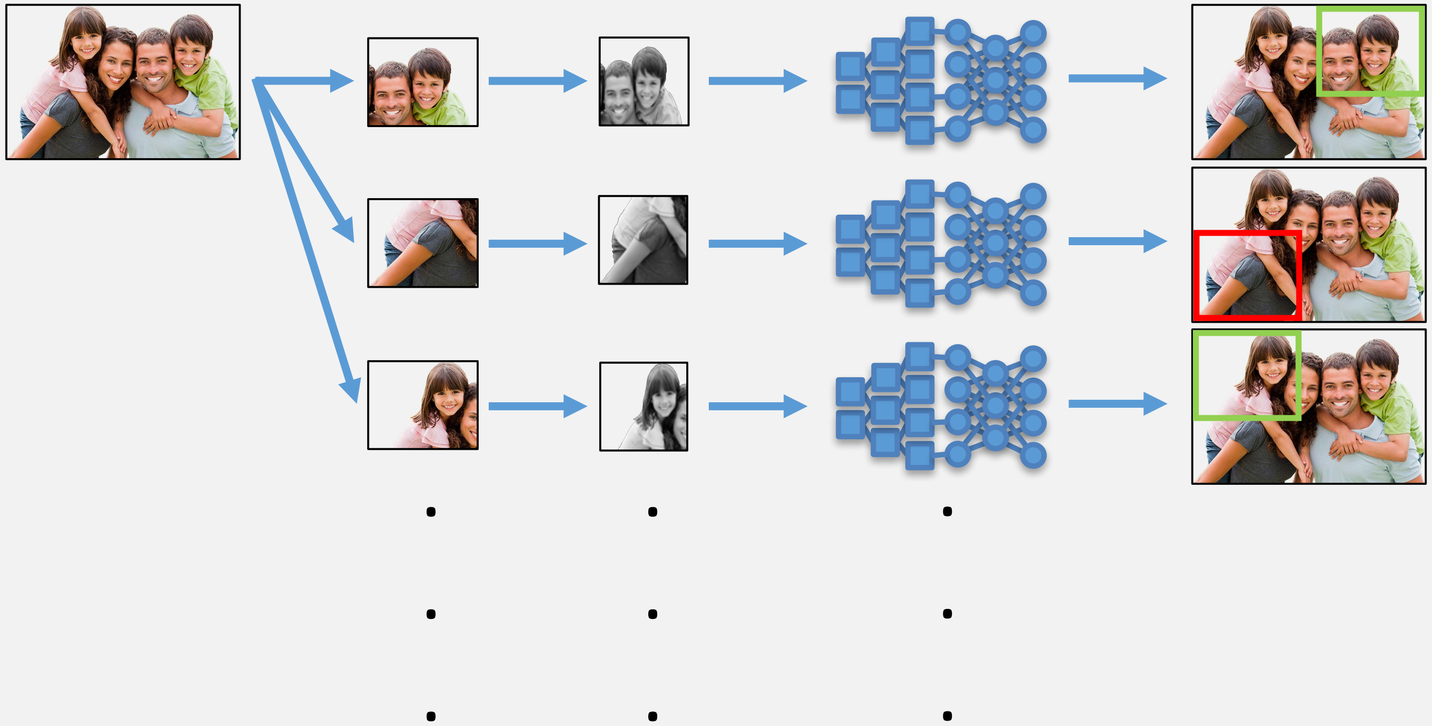
Need Fast Detection

[Typical Detection]



- Read image
- Create object proposals (e.g. Pyramidal, Sliding Window, etc.)
- <u>For each</u> proposal:
  - Crop from frame
  - Pre-process (e.g. warp)
  - Run CNN classifier
- Post-process (NMS)

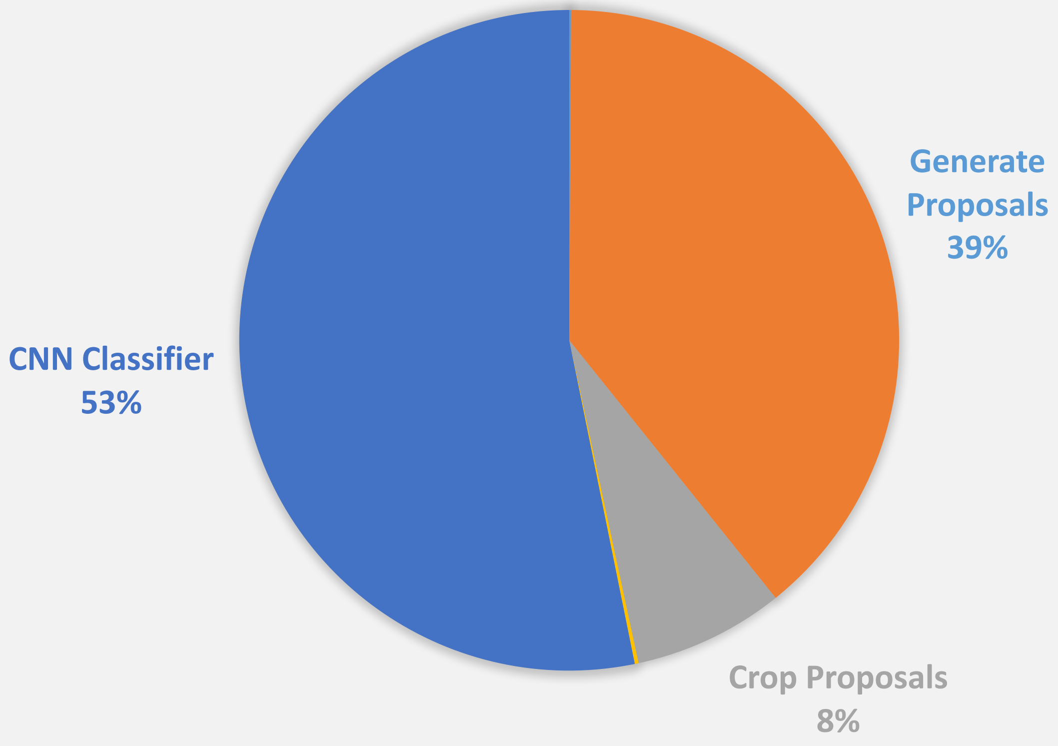
[Typical Detection]



[Typical Detection]

Potentially hundreds to thousands of proposals needed to get tight bounding boxes

[Typical Detection - Cycles]



[Fast Detection – “Fast R-CNN”]

- 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

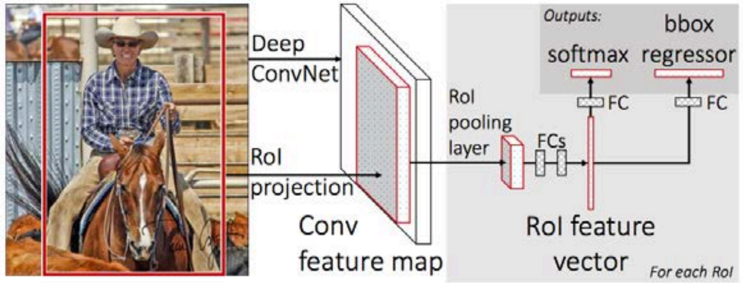


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

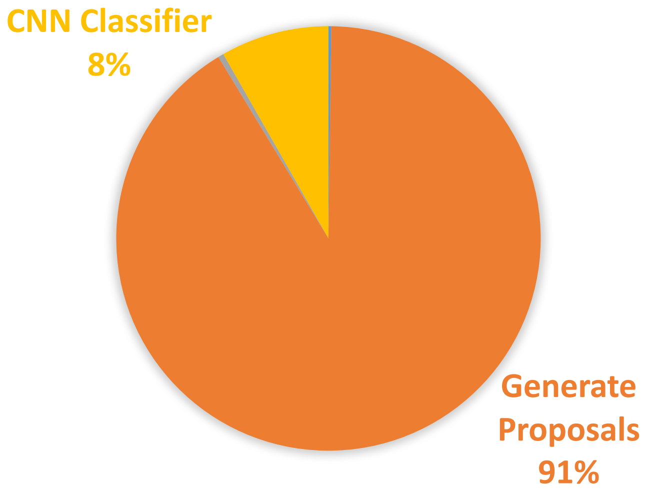


*Image from Girshick, “Fast R-CNN”

MULTICOREWARE

[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

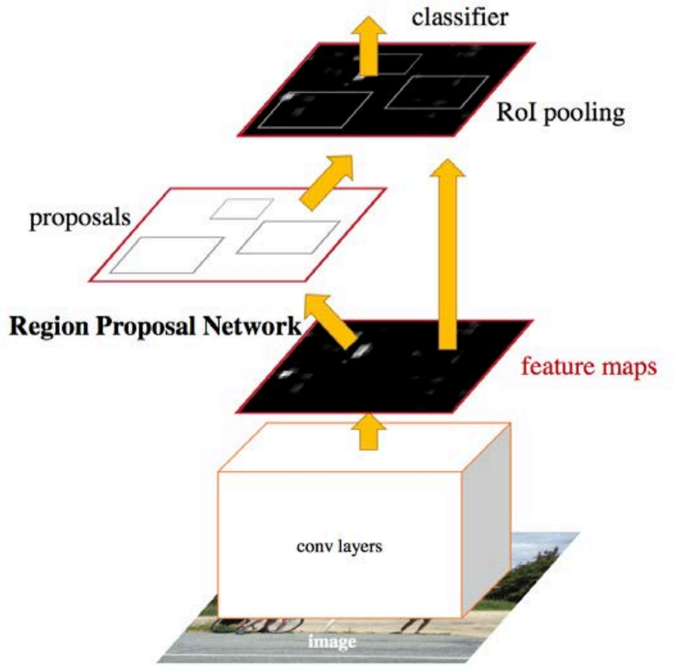


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

*Image from Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"

[Fast Detection – Pipeline Evolution]

Typical Detection

- Read image
- Create object proposals
- For each proposal:
  - Crop from frame
  - Pre-process
  - Run CNN classifier
- Post-process (NMS)

Fast R-CNN

- Read image
- Create object proposals
- ~~For each proposal:~~
  - ~~Crop from frame~~
  - ~~Pre-process~~
  - ~~Run CNN classifier~~
- Run CNN classifier
  - Take as input a list of proposals
  - Use RoI pooling layer
- Post-process (NMS)

Faster R-CNN

- Read image
- ~~Create object proposals~~
- Run CNN classifier
  - Take as input a list of proposals
  - Generate proposals using feature map and region proposal network
  - Re-use feature map for classification
- Post-process (NMS)

[Fast Detection – Fully Convolutional]

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

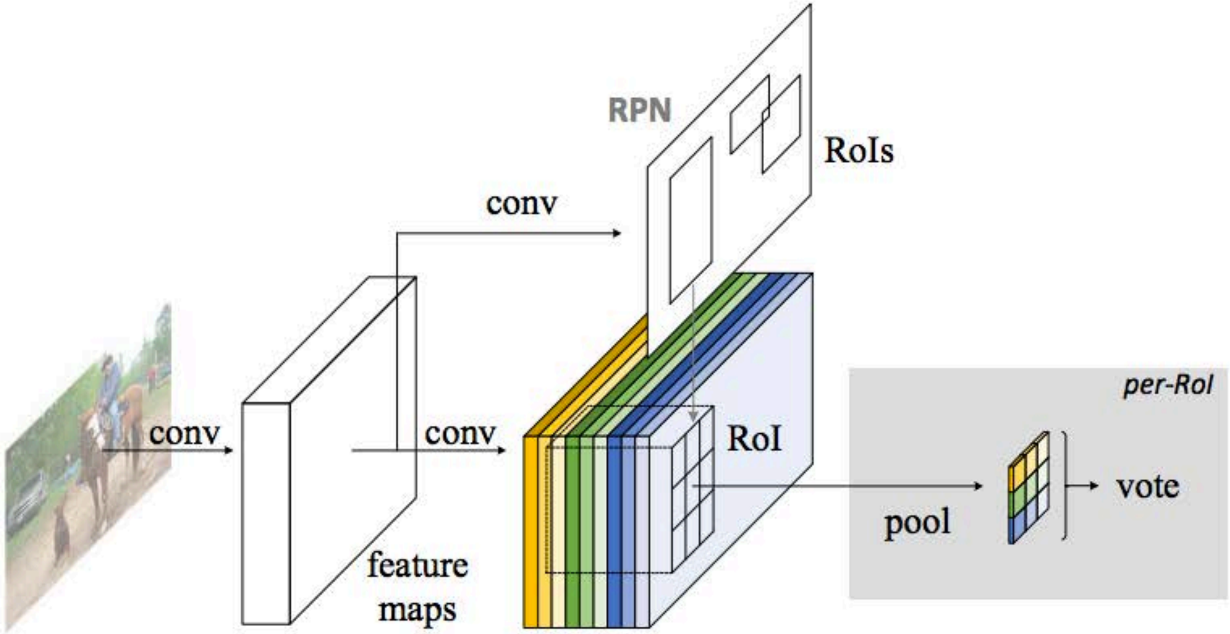


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

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

[Even Faster Detection]

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

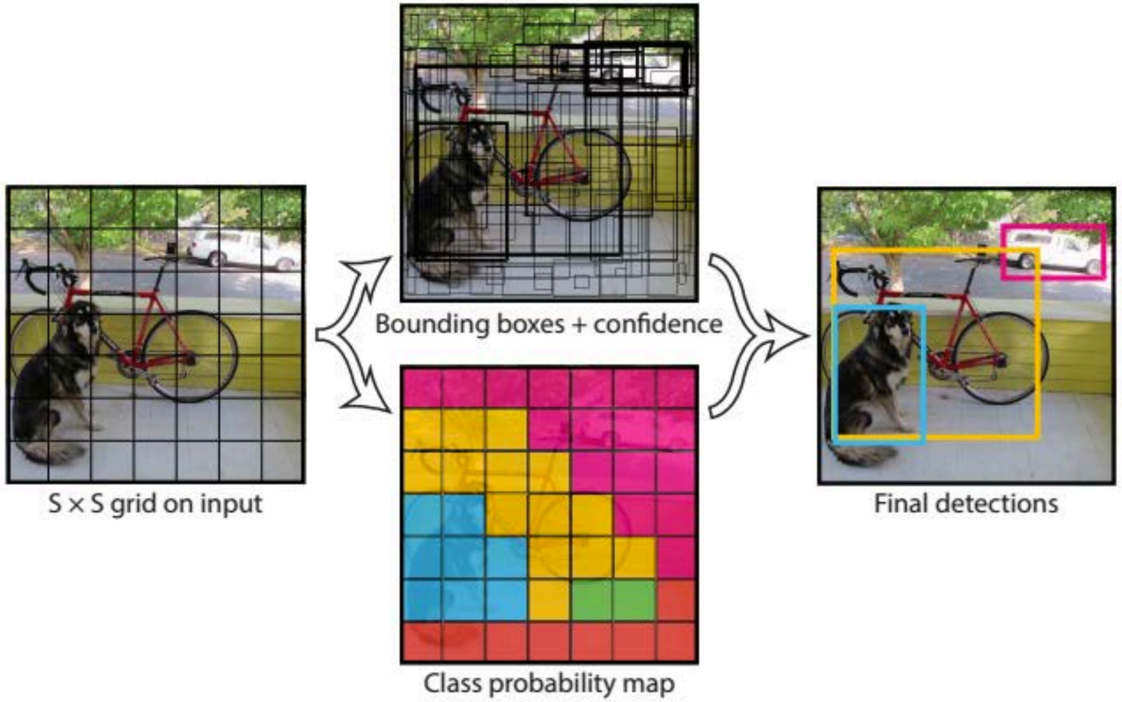


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

*Image from Redmon et al., “You Only Look Once: Unified, Real-time Object Detection”

[Even Faster 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

Simultaneous Classification and Detection [FAST]

Faster R-CNN
R-FCN

Explicit Proposals [ACCURATE]

Need “Smaller” Architectures

“Don’t be a hero”

\- Sage advice from cs231, Andrej Karpathy

Top-Down Design

- Start with top architectures on ILSVRC

[CNN Architecture Design – What Already Works?]

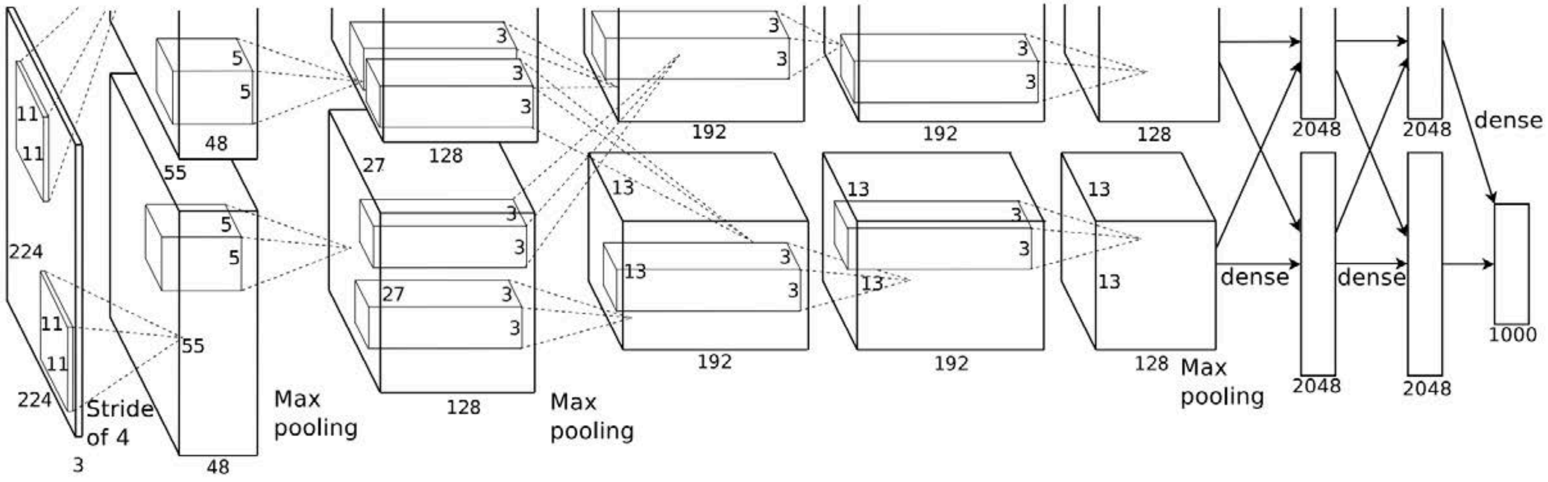


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

Parameter	AlexNet Requirement	Available on Embedded
Weight Space	~250 MB	128 – 512 Kb
Operations for 30FPS VGA	~2400 GMAC/s	24 - 32 GMAC/s

Challenges:

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

Image attribution: Krizhevsky, et. al

MULTICOREWARE

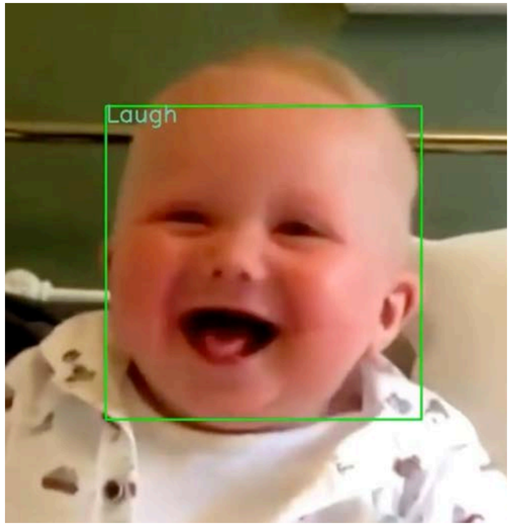
Top-Down Design

- 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

[CNN Architecture Design – Top Down]

Toy Example: Detecting Faces

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



Classifier	AlexNet Type	Shrunk Network
Weight Space	~250MB (500x)	<512Kb (1x)
Layers	10 (7 CV + 3 FC)	5 (3 CV + 2 FC)
Compute Time	640x	1x
Operations per input	832 MMACs	1.3 MMACs

MULTICORE WARE

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”

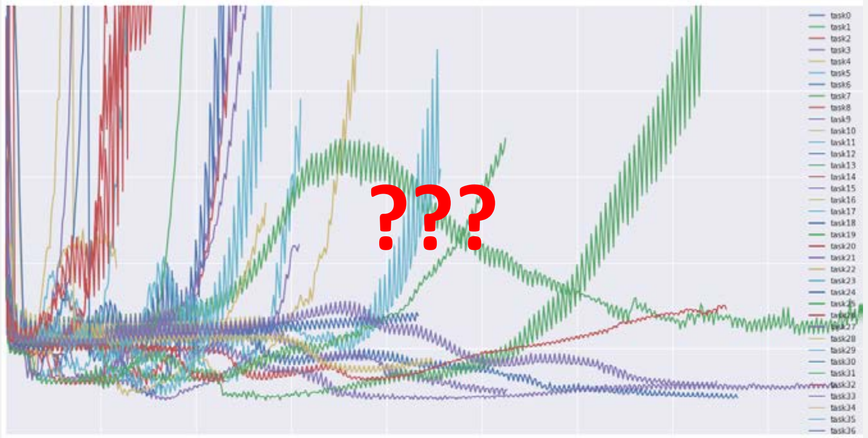
- Sage advice from cs231, Andrej Karpathy

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



MULTICORE WARE

[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

Classifier	SqueezeNet-Type	Darknet-Type
Weight Space	4MB	100MB
Runtime Memory	100MB	500MB
Operations per input	3.6 GMACs	1.4 GMACs

[More To Do]

- 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]

Performance / Power / Memory

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

Progress

- Simultaneous classification and detection
- Pruning, 1x1 convolutions, Bottom-up design with hardware considered

[End]

Contact Me

Anshu Arya
anshu@multicorewareinc.com

MULTICORE WARE