SqueezeNet and SqueezeNext

Ammar Ahmad Awan

CSE 5194.01 – Fall 2018

Network Based Computing Laboratory
Agenda

- Recap: DNN Architectures
- Introduction: SqueezeNet Architecture
- SqueezeNext
- What’s next?
Recap

• What DNN architectures have we covered so far?
  • AlexNet
  • GoogLeNet (Inception)
  • VGG
  • ResNet
AlexNet

- Five Convolution Layers – 61 million parameters

https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637
GoogLeNet (Inception v4)

- GoogLeNet
- Inception v1—v4
- Very complex network
- 25 million parameters

VGG

- VGG-16 has 13 convolution layers
- VGG-19 has 138 million parameters

ResNet(s)

- Very deep architectures
- But still relatively lower number of parameters – why?
  - 25 million parameters only

Summary

- AlexNet, GoogLeNet, VGG, ResNet, and Inception-v4

https://openreview.net/pdf?id=Bygq-H9eg
Recap

- What DNN architectures have we covered so far?
  - AlexNet
  - GoogLeNet (Inception)
  - VGG
  - ResNet

- What have we not seen?
  - NASNet
  - MobileNet
  - SqueezeNet
NASNet – Google AutoML

• Three keys things
  • AutoML- machine generated models
  • Architecture search on CIFAR-10
  • Transfer the learned architecture to ImageNet

MobileNet - Google

- Smaller Models are needed too!

- Two new hyper-parameters: 1) Width Multiplier: Thinner Models and 2) Resolution Multiplier: Reduced Representation

---

Table 8. MobileNet Comparison to Popular Models

<table>
<thead>
<tr>
<th>Model</th>
<th>ImageNet Accuracy</th>
<th>Million</th>
<th>Million Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 MobileNet-224</td>
<td>70.6%</td>
<td>569</td>
<td>4.2</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>69.8%</td>
<td>1550</td>
<td>6.8</td>
</tr>
<tr>
<td>VGG 16</td>
<td>71.5%</td>
<td>15300</td>
<td>138</td>
</tr>
</tbody>
</table>

Table 9. Smaller MobileNet Comparison to Popular Models

<table>
<thead>
<tr>
<th>Model</th>
<th>ImageNet Accuracy</th>
<th>Million</th>
<th>Million Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50 MobileNet-160</td>
<td>60.2%</td>
<td>76</td>
<td>1.32</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>57.5%</td>
<td>1700</td>
<td>1.25</td>
</tr>
<tr>
<td>AlexNet</td>
<td>57.2%</td>
<td>720</td>
<td>60</td>
</tr>
</tbody>
</table>

https://arxiv.org/abs/1704.04861
Agenda

• Recap: DNN Architectures

• Introduction: SqueezeNet Architecture

• SqueezeNext

• What’s next?
Introduction

- So why to make a DNN small?
  - Smaller CNNs require less communication across servers during distributed training.
  - Smaller CNNs require less bandwidth to export a new model from the cloud to an autonomous car.
  - Smaller CNNs are more feasible to deploy on FPGAs and other hardware with limited memory.
SqueezeNet

• One of the first attempts to make DNNs smaller
  • And one of the most high-impact efforts too!
• *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*
SqueezeNet

- Benefits of Smaller models
  - More efficient distributed training
  - Less overhead when exporting new models to clients
  - Feasible FPGA and embedded deployment
Alternatives?

• Network Pruning (NP) and Re-training
  • Make the network sparse by zero-ing connections below a threshold
  • Train the new sparse network for some iterations
• Make the large DNN smaller by compressing it
  • Deep Compression – combine NP + 8-bit quantization + Huffman coding
Concept of Micro-architecture

• Selecting filter dimensions manually is cumbersome

• The so-called “module” in a neural network define its “micro-architecture”
  
  • Example: Inception module in GoogLeNet
  
  • Combination of 1x1 or 3x3 conv. filters

• Proposed: Fire module for SqueezeNet
Concept of Macro-architecture

• What defines a Deep Neural Network?
  • Depth – no. of layers
  • Type of Connections – Residual (skip) networks
  • ResNet is a good example for *A/B comparisons*
  • A- Linear connected layers like in VGG
  • B- Skip connections as in ResNet
Neural Network Design Space

- Main idea is to find out a disciplined approach
  - Many papers and many strategies
  - A/B comparison is one strategy to find out the best DNN
    - Use the old structure and change something
    - Compare the accuracy and adjust the network
Three Strategies to make models smaller

- Replace 3x3 filters with 1x1 filters
  - 1x1 filters have 9X fewer parameters than 3x3 filters
- Decrease the number of input channels to 3x3 filters
  - \#parameters = \textbf{(number of input channels)} \times (number of filters) \times (3 \times 3)
  - Decrease “number of input channels” using “Squeeze” layers
- Late Down-sampling to make convolution layers have large activation maps and thus better accuracy -- (late max-pooling)
Micro-architecture: The Fire Module

- Organization of convolution filters in the Fire module
- Squeeze
- Expand
SqueezeNet Macro-architecture

- Consists of conv1 + 8 “Fire” modules + conv10
- Two proposed designs
  - 1) With and 2) Without compression
- Simple design with no exotic sub-structure
Are there better designs?

• Vanilla SqueezeNet (as per the prior sections).

• SqueezeNet with simple bypass connections between some Fire modules

• SqueezeNet with complex bypass connections between the remaining Fire modules.
Details of SqueezeNet

Table 1: SqueezeNet architectural dimensions. (The formatting of this table was inspired by the Inception2 paper (Ioffe & Szegedy, 2015).)

<table>
<thead>
<tr>
<th>layer name/type</th>
<th>output size</th>
<th>filter size / stride (if not a fire layer)</th>
<th>depth</th>
<th>$s_{1\times1}$ (#1x1 squeeze)</th>
<th>$e_{1\times1}$ (#1x1 expand)</th>
<th>$e_{3\times3}$ (#3x3 expand)</th>
<th>$s_{1\times1}$ sparsity</th>
<th>$e_{1\times1}$ sparsity</th>
<th>$e_{3\times3}$ sparsity</th>
<th># bits</th>
<th>#parameter before pruning</th>
<th>#parameter after pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>input image</td>
<td>224x224x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv1</td>
<td>111x111x96</td>
<td>7x7/2 (x96)</td>
<td>1</td>
<td>100% (7x7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6bit</td>
<td>14,208</td>
<td>14,208</td>
</tr>
<tr>
<td>maxpool1</td>
<td>55x55x96</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire2</td>
<td>55x55x128</td>
<td>2</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>11,920</td>
<td>5,746</td>
</tr>
<tr>
<td>fire3</td>
<td>55x55x128</td>
<td>2</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>12,432</td>
<td>6,258</td>
</tr>
<tr>
<td>fire4</td>
<td>55x55x256</td>
<td>2</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>45,344</td>
<td>20,646</td>
</tr>
<tr>
<td>maxpool4</td>
<td>27x27x256</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire5</td>
<td>27x27x256</td>
<td>2</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td></td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>49,440</td>
<td>24,742</td>
</tr>
<tr>
<td>fire6</td>
<td>27x27x384</td>
<td>2</td>
<td>48</td>
<td>192</td>
<td>192</td>
<td></td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>6bit</td>
<td>104,880</td>
<td>44,700</td>
</tr>
<tr>
<td>fire7</td>
<td>27x27x384</td>
<td>2</td>
<td>48</td>
<td>192</td>
<td>192</td>
<td></td>
<td>50%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>111,624</td>
<td>46,236</td>
</tr>
<tr>
<td>fire8</td>
<td>27x27x512</td>
<td>2</td>
<td>64</td>
<td>256</td>
<td>256</td>
<td></td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>6bit</td>
<td>188,992</td>
<td>77,581</td>
</tr>
<tr>
<td>maxpool8</td>
<td>13x12x512</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire9</td>
<td>13x13x512</td>
<td>2</td>
<td>64</td>
<td>256</td>
<td>256</td>
<td></td>
<td>50%</td>
<td>100%</td>
<td>30%</td>
<td>6bit</td>
<td>197,184</td>
<td>77,581</td>
</tr>
<tr>
<td>conv10</td>
<td>13x13x1000</td>
<td>1x1/1 (x1000)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>20% (3x3)</td>
<td></td>
<td></td>
<td>6bit</td>
<td>513,600</td>
<td>103,400</td>
</tr>
<tr>
<td>avgpool10</td>
<td>1x1x1000</td>
<td>13x13/1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

activations: | parameters: | compression info: | 1,248,424 (total) | 421,098 (total)
SqueezeNet vs. Model Compression

- Smaller model size but similar accuracy!

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

<table>
<thead>
<tr>
<th>CNN architecture</th>
<th>Compression Approach</th>
<th>Data Type</th>
<th>Original → Compressed Model Size</th>
<th>Reduction in Model Size vs. AlexNet</th>
<th>Top-1 ImageNet Accuracy</th>
<th>Top-5 ImageNet Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>None (baseline)</td>
<td>32 bit</td>
<td>240MB</td>
<td>1x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>SVD (Denton et al., 2014)</td>
<td>32 bit</td>
<td>240MB → 48MB</td>
<td>5x</td>
<td>56.0%</td>
<td>79.4%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>Network Pruning (Han et al., 2015b)</td>
<td>32 bit</td>
<td>240MB → 27MB</td>
<td>9x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>Deep Compression (Han et al., 2015a)</td>
<td>5-8 bit</td>
<td>240MB → 6.9MB</td>
<td>35x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>None</td>
<td>32 bit</td>
<td>4.8MB</td>
<td>50x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>Deep Compression</td>
<td>8 bit</td>
<td>4.8MB → 0.66MB</td>
<td>363x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>Deep Compression</td>
<td>6 bit</td>
<td>4.8MB → 0.47MB</td>
<td>510x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
</tbody>
</table>
Microarchitectural design space exploration

(a) Exploring the impact of the squeeze ratio (SR) on model size and accuracy.

(b) Exploring the impact of the ratio of 3x3 filters in expand layers (pct_{3x3}) on model size and accuracy.
Macroarchitectural design space exploration

- Different macro-architectures can help improve the accuracy!

Table 3: SqueezeNet accuracy and model size using different macroarchitecture configurations

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla SqueezeNet</td>
<td>57.5%</td>
<td>80.3%</td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet + Simple Bypass</td>
<td>60.4%</td>
<td>82.5%</td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet + Complex Bypass</td>
<td>58.8%</td>
<td>82.0%</td>
<td>7.7MB</td>
</tr>
</tbody>
</table>
Conclusion

• Many models have focused on complex architectures for better accuracy

• This work is on accuracy-preserving but smaller DNN architectures

• Main motivation for smaller DNNs is speed and portability

• Fire modules in SqueezeNet provide accuracy similar to AlexNet (a very large model)
Agenda

- Recap: DNN Architectures
- Introduction: SqueezeNet Architecture
- SqueezeNext
- What’s next?
SqueezeNext

- SqueezeNext: Hardware-Aware Neural Network Design

- A new family of neural network architectures

- Design guided by
  - Previous architectures such as SqueezeNet
  - Simulation results on a neural network accelerator
SqueezeNext

• So what is new in SqueezeNext?

• AlexNet's accuracy with $\text{112}\times$ fewer parameters

• Deeper variants achieve VGG-19 accuracy with only 4.4 Million parameters, ($\text{31}\times$ smaller than VGG-19).
Versus MobileNet?

- SqueezeNext achieves better top-5 classification accuracy with 1.3× fewer parameters compared to MobileNet.
- Avoids depthwise-separable convolutions that are inefficient on some mobile processors.
Key Contributions

- Wide range of accuracy - the ability to make speed-accuracy tradeoffs
- Hardware simulation results for power and inference speed on an embedded system
- 2.59×/8.26× faster and 2.25×/7.5× more energy efficient as compared to SqueezeNet/AlexNet without any accuracy degradation.
SqueezeNext vs. others

ResNet

SqueezeNet

SqueezeNext
SqueezeNext

- 0.5 Million model parameters only
- 112x smaller than AlexNet
- 2x smaller than SqueezeNet
- Variations, in terms of width and depth, span a wide range
- Deeper variation of SqueezeNext == VGG-19 accuracy
- 4.4 Million parameters only
### Performance: Baseline SqueezeNext

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
<th># Params</th>
<th>Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>57.10</td>
<td>80.30</td>
<td>60.9M</td>
<td>1×</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>57.50</td>
<td>80.30</td>
<td>1.2M</td>
<td>51×</td>
</tr>
<tr>
<td>1.0-SqNxt-23</td>
<td>59.05</td>
<td>82.60</td>
<td>0.72M</td>
<td>84×</td>
</tr>
<tr>
<td>1.0-G-SqNxt-23</td>
<td><strong>57.16</strong></td>
<td><strong>80.23</strong></td>
<td><strong>0.54M</strong></td>
<td><strong>112×</strong></td>
</tr>
<tr>
<td>1.0-SqNxt-23-IDA</td>
<td>60.35</td>
<td>83.56</td>
<td>0.9M</td>
<td>68×</td>
</tr>
<tr>
<td>1.0-SqNxt-34</td>
<td>61.39</td>
<td>84.31</td>
<td>1.0M</td>
<td>61×</td>
</tr>
<tr>
<td>1.0-SqNxt-34-IDA</td>
<td>62.56</td>
<td>84.93</td>
<td>1.3</td>
<td>47×</td>
</tr>
<tr>
<td>1.0-SqNxt-44</td>
<td><strong>62.64</strong></td>
<td><strong>85.15</strong></td>
<td><strong>1.2M</strong></td>
<td><strong>51×</strong></td>
</tr>
<tr>
<td>1.0-SqNxt-44-IDA</td>
<td>63.75</td>
<td>85.97</td>
<td>1.5M</td>
<td>41×</td>
</tr>
</tbody>
</table>
Design: SqueezeNext Block

- Use SqueezeNet as baseline architecture and use:
  1) Low Rank Filters, 2) Bottleneck Module, and 3) Fully Connected Layers
1.0-SqNxt-23

- 23 modules in SqueezNet-23
- Notice the FC layer!
Hardware Simulator

- 16 × 16 or 8 × 8 array of PEs,

- 128KB or 32KB global buffer, and a DMA controller to transfer data between the DRAM and the buffer.

- PE has a 16-bit integer multiply-and-accumulate (MAC) unit and a local register file.

**Figure 5:** Block diagram of the neural network accelerator used as the reference hardware for inference speed and energy estimation of various neural networks.
Performance

- Note the different metrics being used for Performance
  - Inference Time
  - Energy
- Compared to?
  - ...
SqueezeNext vs. Mobilenet

- 1.5× wider and 2× wider channels as compared to the respective baseline.

- 2.0- SqNxt-44 network is able to match VGG-19’s performance with 31× less parameters.

- 2.0-SqNxt23v5 has 1.3× fewer parameters than MobileNet-1.0-224 – but comparable performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-1</th>
<th>Top-5</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5-SqNxt-23</td>
<td>63.52</td>
<td>85.66</td>
<td>1.4M</td>
</tr>
<tr>
<td>1.5-SqNxt-34</td>
<td>66.00</td>
<td>87.40</td>
<td>2.1M</td>
</tr>
<tr>
<td>1.5-SqNxt-44</td>
<td>67.28</td>
<td>88.15</td>
<td>2.6M</td>
</tr>
<tr>
<td>VGG-19</td>
<td>68.50</td>
<td>88.50</td>
<td>138M</td>
</tr>
<tr>
<td>2.0-SqNxt-23</td>
<td>67.18</td>
<td>88.17</td>
<td>2.4M</td>
</tr>
<tr>
<td>2.0-SqNxt-34</td>
<td>68.46</td>
<td>88.78</td>
<td>3.8M</td>
</tr>
<tr>
<td>2.0-SqNxt-44</td>
<td>69.59</td>
<td>89.53</td>
<td>4.4M</td>
</tr>
<tr>
<td>MobileNet</td>
<td>67.50 (70.9)</td>
<td>86.59 (89.9)</td>
<td>4.2M</td>
</tr>
<tr>
<td>2.0-SqNxt-23v5</td>
<td>67.44 (69.8)</td>
<td>88.20 (89.5)</td>
<td>3.2M</td>
</tr>
</tbody>
</table>
## Overall: Simulated Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Params ($\times$ 1E+6)</th>
<th>MAC</th>
<th>Top-1</th>
<th>Top-5</th>
<th>Depth</th>
<th>8x8, 32KB Time</th>
<th>Energy</th>
<th>16x16, 128KB Time</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>60.9</td>
<td>725M</td>
<td>57.10</td>
<td>80.30</td>
<td>—</td>
<td>x5.46</td>
<td>1.6E+10</td>
<td>x8.26</td>
<td>1.5E+10</td>
</tr>
<tr>
<td>SqueezeNet v1.0</td>
<td>1.2</td>
<td>837M</td>
<td>57.50</td>
<td>80.30</td>
<td>—</td>
<td>x3.42</td>
<td>6.7E+09</td>
<td>x2.59</td>
<td>4.5E+09</td>
</tr>
<tr>
<td>SqueezeNet v1.1</td>
<td>1.2</td>
<td>352M</td>
<td>57.10</td>
<td>80.30</td>
<td>—</td>
<td>x1.60</td>
<td>3.3E+09</td>
<td>x1.31</td>
<td>2.4E+09</td>
</tr>
<tr>
<td>Tiny Darknet</td>
<td>1.0</td>
<td>495M</td>
<td>58.70</td>
<td>81.70</td>
<td>—</td>
<td>x1.92</td>
<td>3.8E+09</td>
<td>x1.50</td>
<td>2.5E+09</td>
</tr>
<tr>
<td>1.0-SqNxt-23</td>
<td>0.72</td>
<td>282M</td>
<td>59.05</td>
<td>82.60</td>
<td>[6,6,8,1]</td>
<td>x1.17</td>
<td>3.2E+09</td>
<td>x1.22</td>
<td>2.5E+09</td>
</tr>
<tr>
<td>1.0-SqNxt-23v2</td>
<td>0.74</td>
<td>228M</td>
<td>58.55</td>
<td>82.09</td>
<td>[6,6,8,1]</td>
<td><strong>x1.00</strong></td>
<td>2.8E+09</td>
<td>x1.13</td>
<td>2.4E+09</td>
</tr>
<tr>
<td>1.0-SqNxt-23v3</td>
<td>0.74</td>
<td>228M</td>
<td>58.18</td>
<td>81.96</td>
<td>[4,8,8,1]</td>
<td>x1.00</td>
<td>2.7E+09</td>
<td>x1.08</td>
<td>2.3E+09</td>
</tr>
<tr>
<td>1.0-SqNxt-23v4</td>
<td>0.77</td>
<td>228M</td>
<td>59.09</td>
<td>82.41</td>
<td>[2,10,8,1]</td>
<td>x1.00</td>
<td>2.6E+09</td>
<td>x1.02</td>
<td>2.2E+09</td>
</tr>
<tr>
<td>1.0-SqNxt-23v5</td>
<td>0.94</td>
<td><strong>228M</strong></td>
<td><strong>59.24</strong></td>
<td>82.41</td>
<td>[2,4,14,1]</td>
<td>x1.00</td>
<td><strong>2.6E+09</strong></td>
<td><strong>x1.00</strong></td>
<td><strong>2.0E+09</strong></td>
</tr>
<tr>
<td>MobileNet</td>
<td>4.2</td>
<td>574M</td>
<td>67.50(70.9)</td>
<td>86.59(89.9)</td>
<td>—</td>
<td><strong>x2.94</strong></td>
<td>9.1E+09</td>
<td>x2.60</td>
<td>5.8E+09</td>
</tr>
<tr>
<td>2.0-SqNxt-23</td>
<td>2.4</td>
<td>749M</td>
<td>67.18</td>
<td>88.17</td>
<td>[6,6,8,1]</td>
<td>x3.24</td>
<td>8.1E+09</td>
<td>x2.72</td>
<td>5.9E+09</td>
</tr>
<tr>
<td>2.0-SqNxt-23v4</td>
<td>2.56</td>
<td>708M</td>
<td>66.95</td>
<td>87.89</td>
<td>[2,10,8,1]</td>
<td>x3.17</td>
<td>7.5E+09</td>
<td>x2.55</td>
<td>5.4E+09</td>
</tr>
<tr>
<td>2.0-SqNxt-23v5</td>
<td><strong>3.23</strong></td>
<td>708M</td>
<td>67.44</td>
<td><strong>88.20</strong></td>
<td>[2,4,14,1]</td>
<td>x3.17</td>
<td><strong>7.4E+09</strong></td>
<td><strong>x2.55</strong></td>
<td><strong>5.4E+09</strong></td>
</tr>
</tbody>
</table>
Conclusion

• SqueezeNext, a new family of neural network architectures

• Achieve AlexNet’s top-5 performance with 112× fewer parameters.

• MobileNet is a very novel network for Mobile applications, but SqueezeNext was able to exceed MobileNet’s top-5 accuracy by 1.6%, with 1.3× fewer parameters.

• Tight coupling between neural net design and performance modeling on a neural net accelerator architecture was essential to get our results
Agenda

- Recap: DNN Architectures
- Introduction: SqueezeNet Architecture
- SqueezeNext
- What’s next?
A race for newer architectures

- **Architectures** is an overloaded term
- Computer **Architectures** are evolving
- Neural Network **Architectures** are evolving too!
- What will be the next break-through?
Slides on SqueezeNet Impact

• Will use the Author Deck from:

• https://aspire.eecs.berkeley.edu/wiki/_media/eop/2017/bichen_talk_slides.pdf