High Performance SqueezeNext for CIFAR-10

Jayan Kant Duggal and Mohamed El-Sharkawy
jduggal@purdue.edu, melshark@iupui.edu

IoT Collaboratory IUPUI, Department of Electrical and Computer Engineering,
Purdue School of Engineering and Technology, Indianapolis
Outline

- Problem Statement
- Background
- Proposed Architecture
- Results
- Conclusion
Problem Statement

CNN Requires **HIGH COMPUTATION**, CNN Needs **LARGE MEMORY**.

- **DSE**
- Arch. Modification.
- CIFAR-10 dataset specific training.

- Computationally intensive models are not feasible for embedded or autonomous systems.
- Require:
  - Good model size (less parameters).
  - Decent model accuracy.
  - Flexible CNN model (tradeoffs with less penalty).

---

*DSE- Design Space Exploration, CNNs- Convolutional Neural Networks, DNNs- Deep Neural Networks*
Abstract

- CNN is a memory and computationally and memory intensive.
- Need an efficient & small sized architecture with competitive accuracy.
- Perform Design Space Exploration (DSE) of CNN/ DNN.
- Methods to improve CNN performance.
- Proposed High Performance SqueezeNext for CIFAR-10.
- Results comparison between the proposed arch. and the baseline SqueezeNext arch.
SqueezeNet

SqueezeNext

Methods to improve CNN architecture
SqueezeNet Architecture

- Initial and final Convolution layers.
- 8 fire modules.
- 1 x 1 Squeeze layer.
- 1 x 1 & 3 x 3 Expand layer.
- Lack of FCC layer.
- Optimizer used: SGD

Model accuracy: 78.1%

Original Model size: 4.80 MB

Modified Model size: 2.75 MB

Model Speed: 7 sec / epoch

SqNet baseline model tweaked (improved) for fair comparison.
SqueezeNext Architecture

- 23 layer depth and 1.0 width.
- Use bottle neck modules.
- Element-wise addition skip connection.
- Four stage configuration implementation.
- Optimizer used: SGD+ momentum+ nestrov.
- 4 LR time step decay with exponential update.

CIFAR-10 Model accuracy: 87.56%
CIFAR-10 Model size: 2.59 MB
CIFAR-10 Model speed: 23 sec / epoch

Bottle Neck Module
Hardware and Software framework

**Hardware & Software used:**

- Intel i7 8th generation processor with 32 GB RAM.
- Required memory for dataset and results: 4GB.
- NVIDIA GTX 1080 GPU.
- Spyder version 3.6.
- Pytorch version 1.1.
- Livelossplot (Loss and accuracy visualization).

**Livelossplot package**: Package for live model loss and model accuracy, CNN model visualization tool.
Method to improve CNN Architecture

- More Data or (data augmentation)
- Architecture Tuning
  - Learning Rate Schedule methods.
  - Different optimizers.
  - Different activation functions.
  - Save and Load checkpoint method (Model state dictionary method).
- Architecture modification
  (Proposing a new arch.)
- Model Ensembles

We Want More Data

Step (Green) & Exponential (brown) decay are both better than others.

SGD + MOMENTUM + NESTROV (pink) performs better than other
Proposed
High Performance SqueezeNext Architecture
Proposed High Performance SqueezeNext for CIFAR-10

- RELU sandwiched b/w conv & batch normalization performs better.
- ReLU in place performs better than ReLU, sigmoid, tanh and leakyRelu.
- No need to reintroduce non-linearity or activation function after the basic block.
- Converges to good accuracy.
Forms the 1st block of each stage in 4 stage implementation.

Four different colors that are dark blue, blue, orange and yellow represent 4 stage implementation.

Consists of Skip connection and one separate HPSqnxt basic block on the left, preserves the important feature maps in the initial stages.

Initially does not consist of max pooling in the beginning.
Forms the remaining blocks of the four stage implementation.

Consists of Skip connection only on the left.

In the end consists of an average pooling.

Due to absence of extra basic block module on the left in the remains blocks attains a better model size & speed without affecting accuracy.
Proposed High Performance SqueezeNext for CIFAR-10

- First block of each color is High Performance SqueezeNext Structure 1.
- Remaining blocks of each color is High Performance SqueezeNext Structure 2.
- No max pooling layer used.
- Xavier- uniform initialization.
- One spatial and one FC layer.
- SGD+ momentum +nestrov is used.
- 4 LRs with step decay and exponential update is used.
## Results

*Proposed High Performance SqueezeNext for CIFAR-10*

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy%</th>
<th>Model Size(MB)</th>
<th>Model speed (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqueezeNet-v1-0 (baseline)</td>
<td>78.1</td>
<td>2.75</td>
<td>7</td>
</tr>
<tr>
<td>SqueezeNext-23-1x-v1 (baseline)</td>
<td>87.56</td>
<td>2.59</td>
<td>23</td>
</tr>
<tr>
<td>Proposed modified implementation of SqueezeNext-23-1x-v1</td>
<td>92.25</td>
<td>5.14</td>
<td>48</td>
</tr>
<tr>
<td>Proposed HP-SqueezeNext-21-1x-v2</td>
<td>92.05</td>
<td>2.60</td>
<td>18</td>
</tr>
<tr>
<td>Proposed HP-SqueezeNext-06-0.50x-v1</td>
<td>82.44</td>
<td>0.370</td>
<td>7</td>
</tr>
<tr>
<td>Proposed HP-SqueezeNext-06-1x-v1</td>
<td>86.82</td>
<td>1.24</td>
<td>8</td>
</tr>
<tr>
<td>Proposed HP-SqueezeNext-06-0.75x-v1</td>
<td>82.86</td>
<td>1.24</td>
<td>8</td>
</tr>
</tbody>
</table>

*All results are 3 average runs with SGD, LR is 0.1*
Results

High Performance SqueezeNext for CIFAR-10

1) Baseline SqueezeNext
2) Pytorch SqueezeNext
3) Proposed Modified Implementation of SqueezeNext
4) High Performance SqueezeNext

4 Step decay with exponential update LR schedule.

More dynamic and reaches a late accuracy saturation point.

All architectures are trained with same hyperparameters for fair comparison.

Less over fitting between training and testing accuracy.
Conclusion

Proposed High Performance SqueezeNext for CIFAR-10

- Propose HIGH PERFORMANCE SQUEEZENEXT architecture.
- CIFAR-10 specific dataset based training and testing improves DNN performance.
- *Livelossplot package* assists in rapid training of small sized CNN/ DNN models.
- Prefer SAVE AND LOAD model checkpoint with model state dictionary method.
- High Performance SqueezeNext attained a best model size and model speed with a **minimum tradeoff (less penalty)**.
- BEST MODEL ACCURACY: **92.05%** (14 % better than SqNet (Baseline) & 4.5 % better than SqNxt (Baseline)).
- BEST MODEL SIZE: **0.370 MB**
- BEST MODEL SPEED: **7 secs/ epoch** (16 secs better than SqNxt baseline and equivalent to SqNet baseline).
THANK YOU
Questions ??