Deep Learning and Computer Vision in Low Power Devices

Lokesh Vadlamudi

San Jose State University

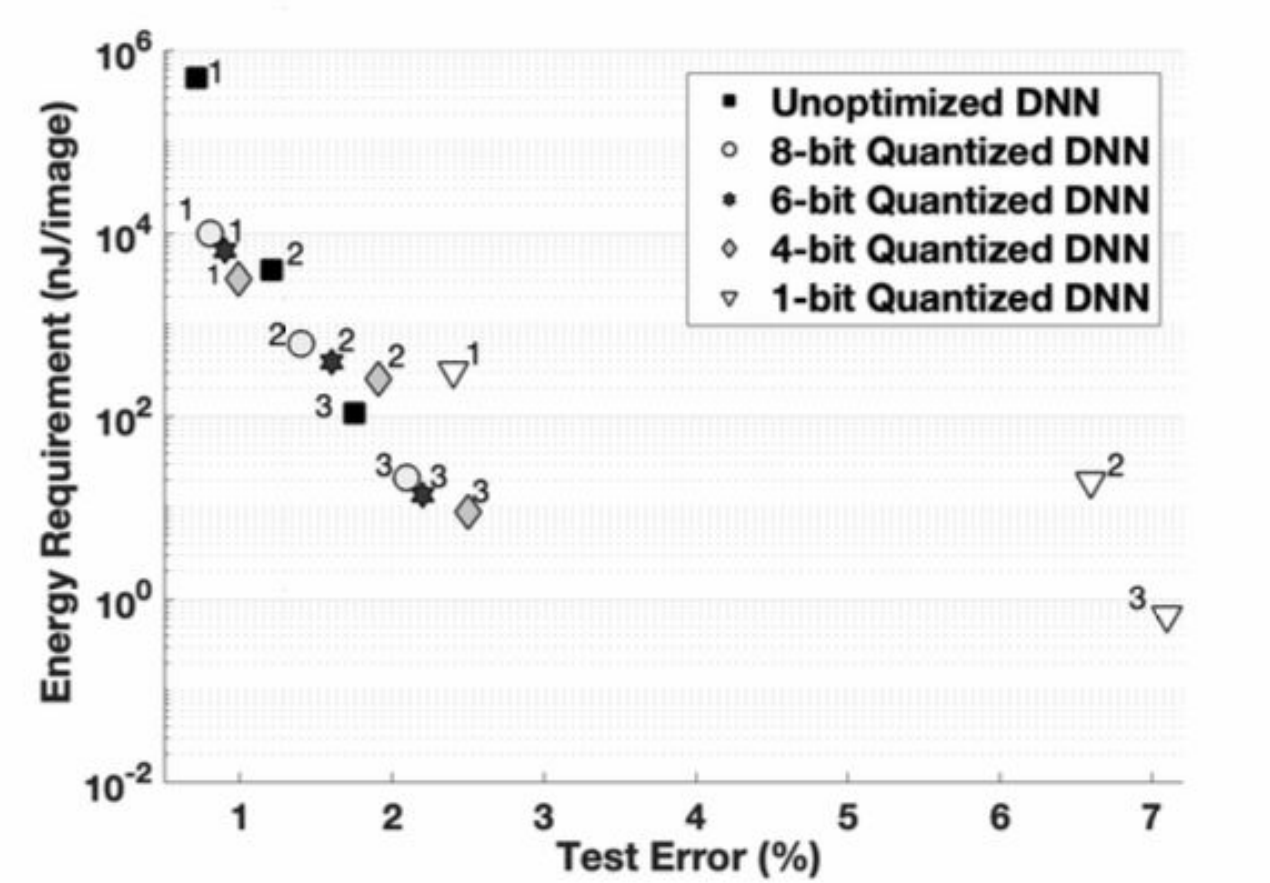
Deep Learning

Introduction

- Deep neural networks (DNNs) are successful in many computer vision tasks. However, the most accurate DNNs require millions of parameters and operations, making them energy, computation and memory intensive. This impedes the deployment of large DNNs in low-power devices with limited compute resources.
- Hence, we are going to talk about few techniques for reducing memory and energy requirements.

1.1. Parameter Quantization

- One method to reduce the number of memory accesses is to decrease the size of DNN parameters.
- Some methods show that it is possible to have negligible accuracy loss even when the precision of the DNN parameters is reduced.
- LightNN, CompactNet, and FLightNN find the optimal bit-width for different parameters of a DNN, given an accuracy constraint.



- Here, as the parameter bit-width decreases, the energy consumption decreases at the expense of increasing test error.

CompactNet

- CompactNet automatically optimizes a pre-trained CNN model on a specific resource-limited platform given a specific target of inference speedup. Guided by a simulator of the target platform, CompactNet progressively trims a pre-trained network by removing certain redundant filters until the target speedup is reached and generates an optimal platform-specific model while maintaining the accuracy.

- **Advantage:**
- When the bit-widths of the parameters decrease, the prediction performance of DNNs remains constant. This is because constraining parameters has a regularization effect in the training process.

- **Disadvantage:**
- DNNs employing quantization techniques need to be retrained multiple times, making the training process very expensive.

1.2. Pruning *Parameters and Connections:*

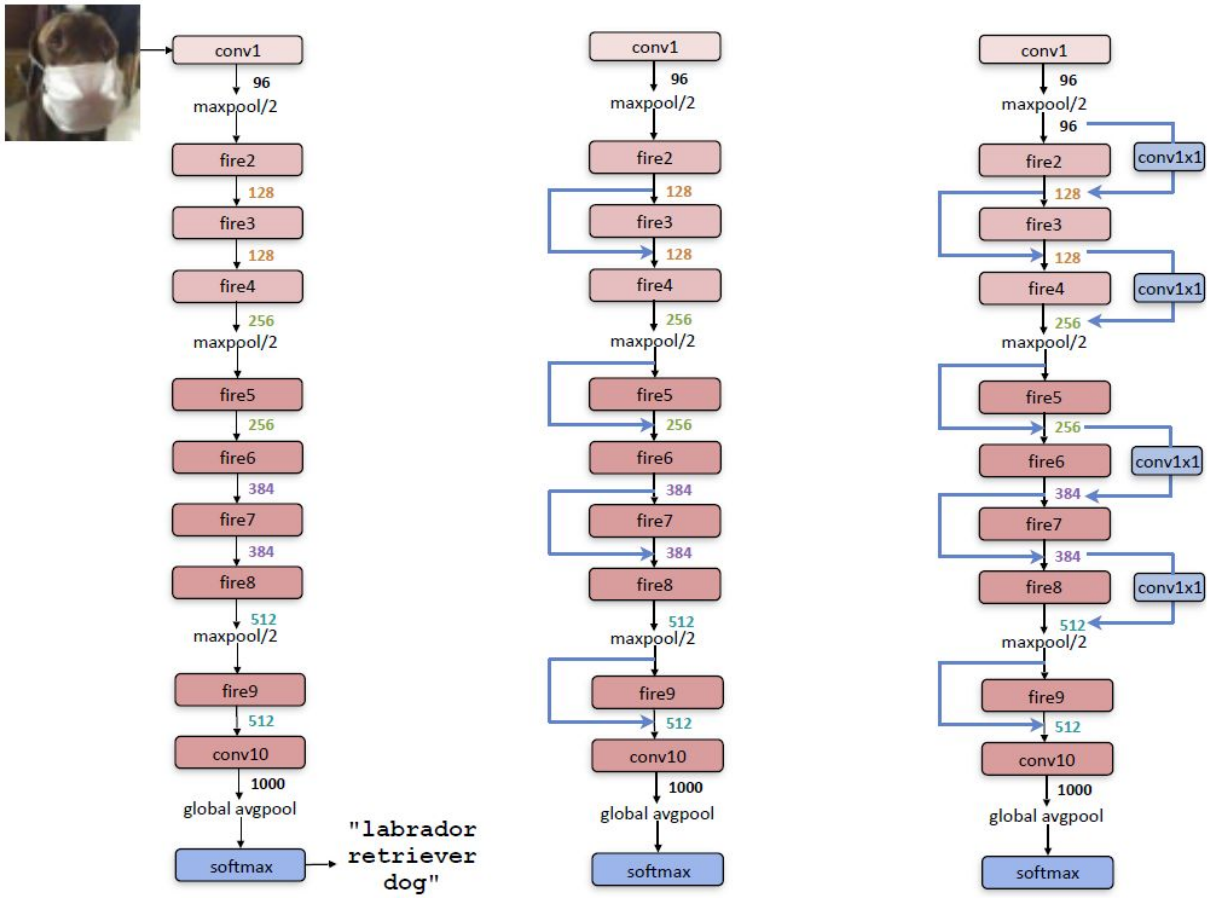
- The Hessian-weighted distortion measure can be used to identify the importance of parameters in a DNN. These measure-based pruning methods only operate on the fully-connected layers. By performing pruning, quantization, and encoding, Deep Compression reduces the model size by 95%.
- Path-level pruning is also seen in tree-based hierarchical DNNs. Although these techniques can identify the unimportant connections, they create unwanted sparsity in DNNs. Sparse matrices require special data structures (unavailable in deep learning libraries), and are difficult to map onto modern GPUs.

- **Advantage:** Pruning can be combined with quantization and encoding for significant performance gains. When the three techniques are used together, the VGG-16 model size decreases to 2% of its original size.
- **Disadvantage:** The training effort associated with DNN pruning is considerable because DNNs have to be pruned and trained multiple times. The advantages of pruning are noticed only when using custom hardware or special data structures for sparse matrices.

2.1. ***Convolutional Filter Compression***

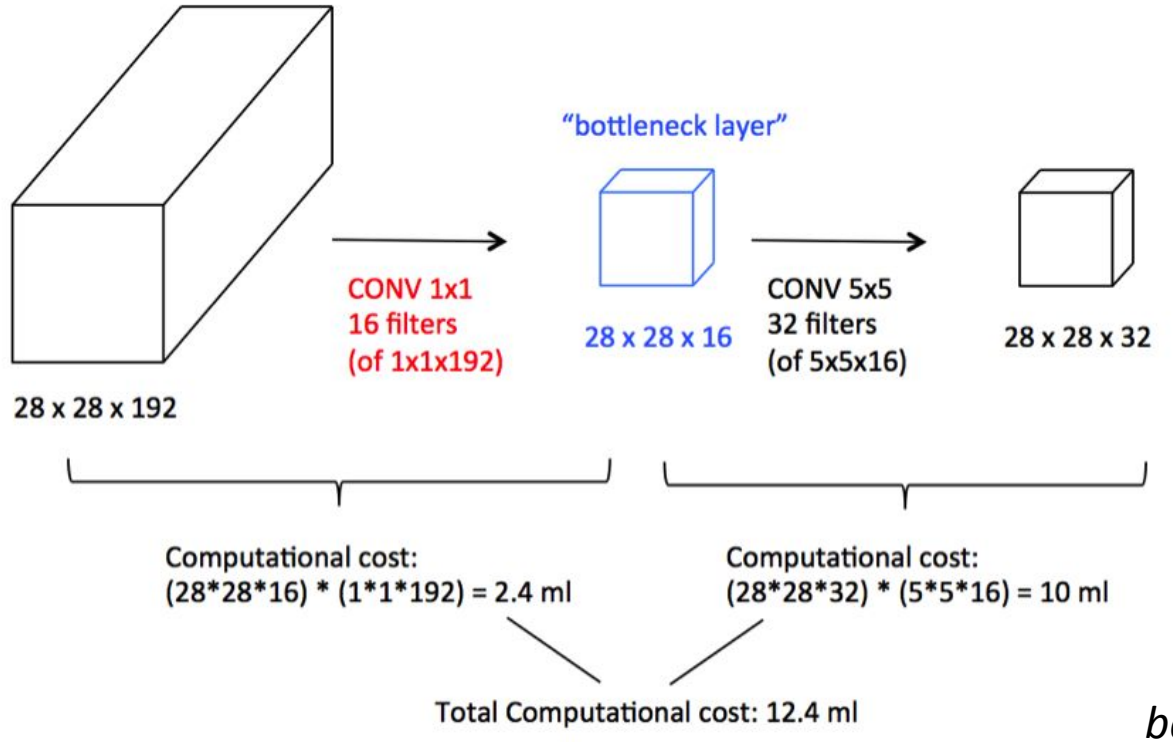
- Smaller convolution filters have considerably fewer parameters and lower computation costs than larger filters. For example, a 1×1 filter has only 11% parameters of a 3×3 filter. However, removing all large convolution layers affects and lowers its accuracy.
- SqueezeNet is one such technique that uses three strategies to convert 3×3 convolutions into 1 × 1 convolutions to reduce the number of parameters.

Squeeze Net



SqueezeNet (Left): begins with a standalone convolution layer (conv1), followed by 8 Fire modules (fire2–9), ending with a final conv layer (conv10).

- **Advantage:** Bottleneck convolutional filters reduce the memory and latency requirements of DNNs significantly. For most computer vision tasks, these techniques obtain state-of- the-art accuracy.



based on Andrew Ng's lectures at DeepLearning.ai

2.2. ***Matrix Factorization***

* This technique factorizes multi-dimensional tensors (in convolutional and fully-connected layers) into smaller matrices to eliminate redundant computation.

- Some factorizations accelerate DNNs up to 4×.
- To minimize the accuracy loss, matrix factorizations are performed one layer at a time. The layer-by-layer optimization approach makes it difficult to scale these techniques to large DNNs because the number of factorization hyper-parameters increases exponentially with DNN depth.

- **Advantages:** Matrix factorizations are methods to reduce the computation costs in DNNs. The same factorizations can be used in both convolutional layers and fully connected layers.
- **Disadvantages:** The computation costs associated with matrix factorization often offset the performance gains obtained from performing fewer operations. Matrix factorizations are also difficult to implement in large DNNs because the training time increases exponentially with increasing depth.

3 . Network Architecture Search:

- Network Architecture Search (NAS) is a technique that automates DNN architecture design for various tasks. NAS uses a Recurrent Neural Network (RNN) controller and uses reinforced learning to compose candidate DNN architectures.
- These candidate architectures are trained and then tested with the validation set. The validation accuracy is used as a reward function to then optimize the controller's next candidate architecture.
- MNasNet is 2.3× faster than NASNet with 4.8× fewer parameters and 10× fewer operations. Moreover, MNasNet is also more accurate than NASNet.

- **Advantages:** NAS automatically balances the trade-offs between accuracy, memory, and latency by searching through the space of all possible architectures without any human intervention. NAS achieves state-of-the-art performance in terms of accuracy and energy efficiency on many mobile devices.
- **Disadvantage**: The computational demand of NAS algorithms makes it difficult to search for architectures that are optimized for large datasets.

4 . Knowledge Distillation:

- Trains a compact DNN that mimics the outputs, features, and activations of a more computation-heavy DNN.
- Training small DNNs to learn such functions is challenging with the conventional back propagation algorithm.
- some methods train small DNNs to learn complex functions by making the small DNNs mimic larger pre-trained DNNs. These techniques transfer the “knowledge” of a large DNN to a small DNN through a process called Knowledge Transfer (KT).

- **Advantages:** KT- and KD-based techniques can reduce the computation costs of large pre-trained DNNs significantly.

- **Conclusion:**
- DNNs are powerful tools for performing many computer vision tasks. However, because the best performing (most accurate) DNNs are designed for situations where computational resources are readily available, they are difficult to deploy on embedded and mobile devices.