ECE 477 Project

Comparing Modern Compression Techniques & DNN Compression

Midterm Report

Authors:
Stephen More
Felipe Orrico Scognamiglio
Jacob Gillette
Emilio Magaña
Hassan Alabdulaziz

May 5, 2021
Instructor: Ben Lee

Contents

1 Introduction

Compression is a necessary technique required for the transmission of video across any type of platform as creating videos requires an enormous amount of data for storage. Compression, in all of its forms, allows for more efficient delivery of digital information. There are two main types of compression: Lossless compression, and Lossy compression. Different types of videos in varying situations requiring different types of compression and unique approaches. For example, modern compression techniques, such as HEVC (High-efficiency video coding, also known as H.265) are Lossy compression techniques. This survey will analyze and evaluate the multiple available modern standards of video compression, and disclose the trade-offs and limitations of each of the formats studied.

Compression is needed to support the many multimedia applications used by engineers across diverse fields in ECE (Electrical and Computer Engineering and Computer Science). Understanding the trade-offs between cutting-edge compression techniques is important if one intends to learn from past design choices when trying to implement new designs under the tight engineering constraints of the industry.

Over the last decade, the use of Machine Learning (ML) has become the main solution for a wide range of modern computing applications. The key concept is that it uses data-driven programming, rather than a strictly defined algorithm. Architecture to use such data must be created, such as within hardware components, but once this is done the only thing needed to get an ML application to perform is sufficient training data. In the last two years, 90% of the world's data has been created. This means that data is becoming increasingly available, at the same time ML computing technologies are rapidly accelerating. The end goal of an ML application is to get it to interpret a result from unseen data. However, the bottleneck then becomes which hardware is being used to run inference, and the number of resources needed to satisfy an application.

This survey will look at ways in which a branch of ML applications (DNNs) can be compressed to be less resource-intensive, as well as some of the trade-offs that come with such compression, and modern video compression techniques that have been the standard over the past 20 years.

2 Background

2.1 DNNs Compression Techniques

Deep Neural Networks (DNNs) have become extremely popular for solving diverse sets of challenges including natural language processing, image recognition, autonomous vehicles, and classification problems which are harder for non-data-driven algorithms to solve. The availability of the datasets needed to train such networks has also increased vastly in recent years, which in turn has created a demand in DNNs to extract information from such datasets. Since DNNs are desired to be used in diverse environments with vastly different hardware and bandwidth constraints, compression is crucial. Strict hardware requirements such as the amount of memory available are a huge constraint when attempting to

apply DNN applications to embedded systems. A classical example of an object detection DNN is AlexNet [1] which requires 240MB of space. While 240MB may be available on higher-end embedded systems, there is still more that is needed. What if multiple networks need to be used simultaneously? What if one wants to deploy DNN applications across many lower-cost embedded devices with less memory resources? What if such embedded devices are communicating with servers over the internet to execute their applications? There will always be a demand for more computers, and the memory required to use DNN applications either over the cloud or on the edge is still a bottleneck for many applications. This is why it is important to look at the cutting edge of DNN compression and compare the engineering tradeoffs between techniques.

This section will use the paper "Deep Compression" [2] as an example illustrating common DNN compression techniques 1-3, many different methodologies will use variations of pruning, quantization, and encoding techniques discussed. When the term "losslessly" is used it is intended to mean "no loss of network accuracy".

2.1.1 Pruning

In a general sense pruning is the act of looking at a trained network and evaluating it to find the weights and neurons which affect the output the least and removing them. The tradeoff for this action is typically the accuracy of the network, however, there are typically a large number of ineffective connections and neurons which can be removed with little accuracy drawbacks. Using the VGG16 network as a reference [3], 90% of weights are in fully connected layers, however, the same percentage only makes up for 1% of all operations [4]. If the weights that are barely being used are removed, storage can be saved with little sacrifice. There are a variety of methodologies to evaluate the impact of neurons and connections. One implementation used is to mask out all weights below a certain threshold (Ex. [-0.1,0.1]), eliminating them from impacting the neurons of subsequent layers, and the final output [5]. The process is typically done iteratively as the inner workings of DNNs are black boxes, and improper pruning can cause large drops in accuracy.

The iterative process could be the following [6]:

1. Evaluation of neuron/connection importance
2. Removal of least important neuron/connection
3. Fine Tuning (shown in quantization section)
4. Has the desired accuracy/memory utilization tradeoff been met?

   4.1 Yes -> Stop pruning

   4.2 No -> goto step 1

The way in which "Deep Compression" [2] performs pruning is by using the a threshold to eliminate ineffective weights, then retraining the network to find the correct weights for the final pruned network. For AlexNet pruning reduced the number of parameters by a factor of 9 [2], without any accuracy loss.

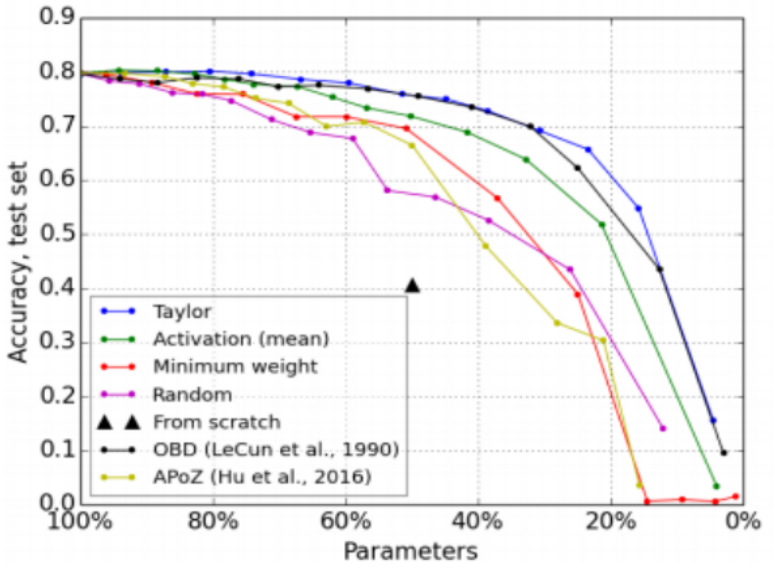

Figure 1: AlexNet, across different pruning techniques, accuracy vs % of parameters after pruning [6]

Figure 1 shows lossy pruning and that as one prunes more aggressively, the accuracy of the network can be significantly affected. However, there is a sweet spot in which one can get a large drop in the number of parameters, while maintaining lesser accuracy losses. A drawback with pruning is the manual setup of the iterative fine tuning parameters, which have to be considered layer by layer [7].

2.1.2 Weight Quantization

Quantization is the process of grouping weights into similar estimated values in order to use fewer bits to represent every weight in the network.

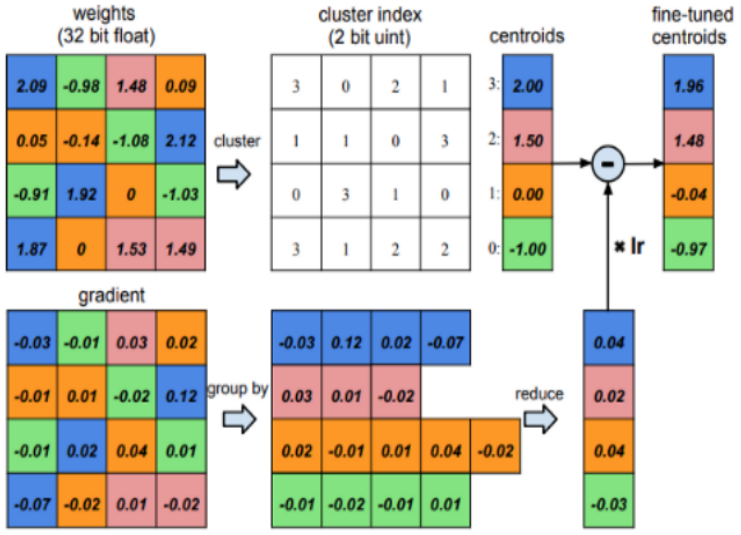

Figure 2: Quantization and weight sharing in "Deep Compression" methodology [2]

Figure 2 shows how the initial weights (upper left) are grouped together based on their closest estimated floating point value. This process allows for the number of

bits used to represent weights to be reduced from 32 to 2. The values in the centroids column map the 2-bit ints in cluster index to estimated values (similar to a lookup table). To account for the error of using estimated values, fine-tuning is then performed on the shared weights. This is done via using gradients corresponding to weight values then summing, multiplying by learning rate, and subtracting them from the centroids column [2]. The next iteration would use the previously calculated fine-tuned values as the centroids column in the next iteration [2].

Using the example of AlexNet with methodology in [2], quantization using 8 bits for convolutional, and 5 bits for fully connected layers has decreased the number of bits needed to represent weights from 32, all losslessly [5]. The compression rate of quantization shown in figure 2 is given by the following equation:

$$r = \frac{nb}{nlog_2(k) + kb}$$

Figure 3: Equation for compression rate of "Deep Compression" implementation [2]

- n = # of connections in the network
- b = # of bits in each connection
- k = # of clusters (# of colors figure 3)

After the quantization stage AlexNet was found to be reduced by a factor of 27 from its original size [2].

2.1.3 Huffman Encoding

Huffman encoding is useful when dealing with gaussian-like distributions, meaning there are certain values which have a much higher probability of occurrence than others. Huffman encoding further reduces the amount of space needed to store the weight values by using variable length encoding. For example the weight that has the highest probability of occurrence would be stored in the value '0'.

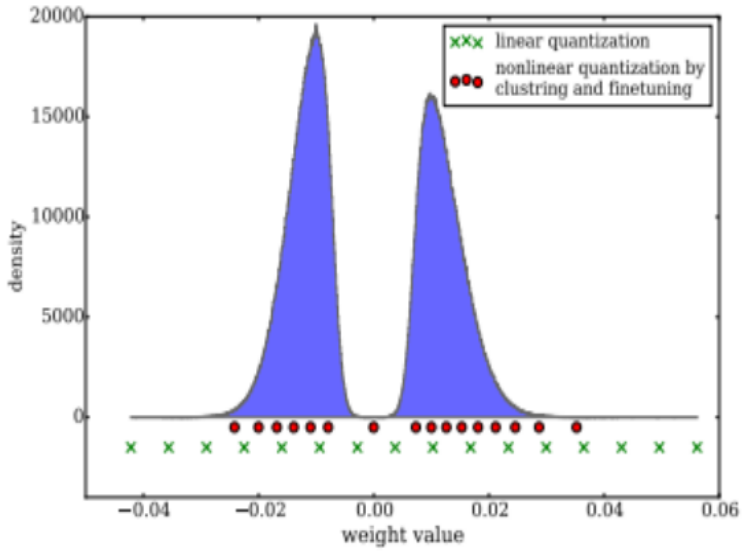

Figure 4: "Deep Compression" weight distributions after quantization [2]

Huffman encoding was found to save 20-30% of network storage size, causing the final reduction of AlexNet to be 35x [2]. This is the final stage of the algorithm used in "Deep Compression" [2].

2.2 Video Compression Techniques

2.2.1 H.264 and H.265

H.264 was standardized in 2003, as a video compression technology (codec), and is currently the most widely used format for compression with 91% of video industry developers using it as of September 2019 [8]. The greatest changes transitioning from H.264 to H.265, are "the expansion of the pattern comparison and difference-coding areas from 16×16 pixel to sizes up to 64×64, improved variable-block-size segmentation, improved "intra" prediction within the same picture, improved motion vector prediction and motion region merging, improved motion compensation filtering, and an additional filtering step called sample-adaptive offset filtering"[9]. H.264 uses 16x16 macroblocks, macroblocks being, "a processing unit in image and video compression formats based on linear block transforms", usually through DCT (discrete cosine transform). In comparison H.265 uses 64x64 macroblocks, allowing for better compression efficiency at all resolutions especially at high resolutions since there is much more data to compress. H.256 supports resolutions up to 8192x4320, while H.264 supports a max resolution of 4096x2048 pixels. When a macroblock is encoded in intra mode, "a prediction block is formed based on previously encoded and reconstructed (but unfiltered) blocks" [10]. H.265 uses 33 directions of motion for motion/frame prediction, while H.264 only uses nine directions of motion. H.265 supports parallel processing as well. Intel has produced processors with instruction sets for encoding and decoding H.265 video, showing a promising future for H.265 even though H.264 is still more commonly used.

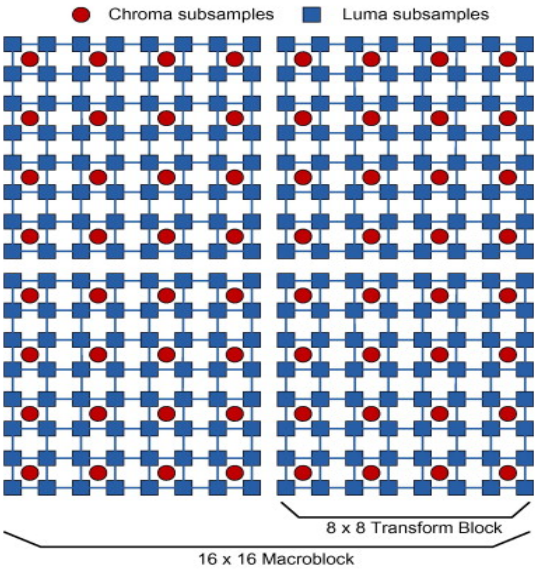

Figure 5: 16x16 4:2:0 Macroblock [11]

In figure 5 shown above is a macroblock in a 4:2:0 format, comprising of a 16x16 array of luma samples and 2 subsampled 8x8 arrays of chroma samples (luma represents the brightness of an image and chroma represents the color difference).

Since H.265 is much more processor intensive than H.264, a hardware decoder is necessary to be able to decode using H.265, making it so that a "simple firmware" upgrade wouldn't be sufficient to handle H.265. Since H.265 is able to compress using less bandwidth a greater range of consumers will be able to consume videos of higher resolution that weren't beforehand available given stricter bandwidth constrictions.

2.2.2 AV1

AV1 by the Alliance for Open Media (AOM) is an open-source video compression service that would be available to use over the internet for free. AV1 would be a royalty-free alternative to HEVC. With motion prediction, AV1 is not yet as efficient as HEVC.

By 2023, 66% of internet-connected TVs will be set to stream at 4k quality [12]. 4k video streaming takes more than double the bit rate of standard HD video. Video accounts for around 75-80% of global internet traffic [13]. With the number of connected devices expected to increase from 24 billion now to 29 billion in 2023, it's clear that there is a need for more optimized encoding algorithms for video streaming. When H.264 was created, the amount of media consumption through internet streaming was not as widespread as today. Since its creation, it has become very clear that the amount of content that is consumed on a daily basis will only increase. However, the growth of content consumption is exponential compared to the increase in connection speed. This calls for a way to decrease the number of bits that need to be transferred when someone is consuming content. From that need, multiple compression standards have been created to try to solve this bandwidth "shortage". AV1, even if it is not as widespread as H.264/265, is said to be the long-term solution while it is 30% more efficient than H.265 and

over twice as efficient as H.264.

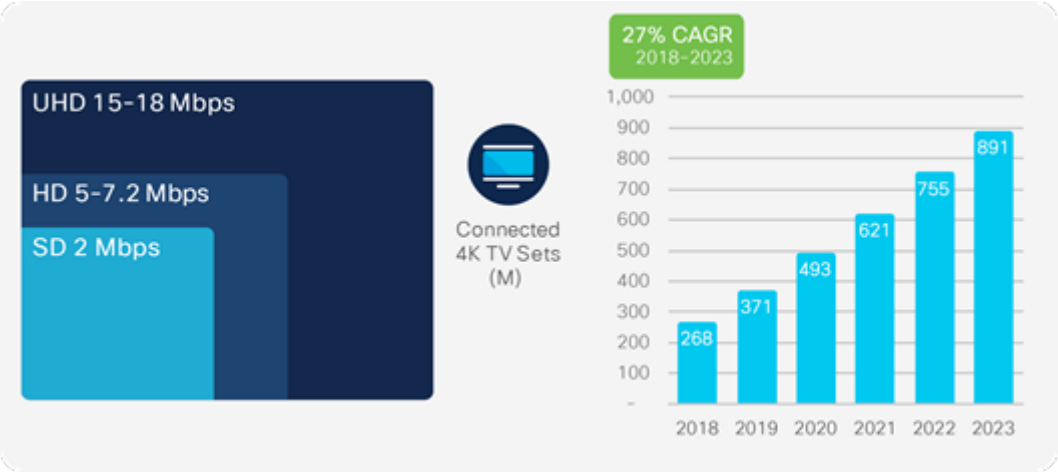

Figure 6: Increasing use 4k TVs

Encoding of 4K 15 sec clip	SW or HW	Time
H.264	Software	1 min
H.264	Hardware	20 secs
H.265	Software	5 mins
H.265	Hardware	20 secs
AV1	Software	10 mins

Figure 7: Amount of time to encode a 4K 15-second video using multiple compression algorithms

3 Comparison

3.1 H.265 and AV1

Initially, in order to be able to measurably differentiate between H.265 and the royalty-free open-source alternative AV1, we must understand the applications of each of those formats, and understand the thought process behind the algorithms.

AV1 is nearly twice as efficient as H.264, works with high-quality video, has no licensing fee, and works with bandwidth constraints. H.265 has uncertainties around its licensing fees making it untenable for an average consumer. AV1's goal is to replace H.264 and compete with H.265 so the high-quality video can be shared freely and efficiently on the internet. Companies and creators have to pay a royalty to compress and decompress with H.264 and H.265.

The AV1 algorithm comes from the need for a free alternative to other compression methods that use a royalty-based distribution system. The four main points of advantage that AV1 possesses against other methods are The royalty-free

ecosystem, cutting edge technology, collaborative, open-source development, and a claimed 30 percent better compression than H.265. As it has various big names supporting it (with the likes of Amazon, Google, Netflix, and Microsoft), it is nearly impossible to turn AV1 into a royalty-based system. AV1 has two major downsides. The first is that there is little to no hardware encoding support even though its decode complexity is lower than H.265; nevertheless, AV1's more advanced video encoding algorithms require more powerful hardware. At this current time AV1 encoding is done by CPUs and AV1 decoding is supported by NVIDIA's latest GeForce RTX 30 series GPUs [14]. The other downside is that AV1 is too slow for any practical use, which encodes a single 4k frame that takes around 100 seconds [15]. This means that while H.264 takes 1 minute to encode a 15-second 4k clip (software), and H.265 takes 5 minutes, AV1 takes 10 minutes to perform the same task. To put it in perspective, a one-hour video converted to H.265 using hardware acceleration will take 80 minutes to render, while when using AV1, the same file will take over 40 hours.

References

[1] Jerry Wei. *AlexNet: The Architecture that Challenged CNNs*. URL: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[2] S. Han H. Mao and W. J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning Trained Quantization and Huffman Coding*. URL: https://arxiv.org/abs/1510.00149.

[3] Muneeb ul Hassan. *VGG16 – Convolutional Network for Classification and Detection*. URL: https://neurohive.io/en/popular-networks/vgg16/.

[4] *Pruning deep neural networks to make them fast and small*. URL: https://jacobgil.github.io/deeplearning/pruning-deep-learning.

[5] C. Shorten. *Deep Compression*. URL: https://towardsdatascience.com/deep-compression-7b771b3aa773.

[6] Pavlo Molchanov Stephen Tyree Tero Karras Timo Aila Jan Kautz. "PRUNING CONVOLUTIONAL NEURAL NETWORKS FOR RESOURCE EFFICIENT INFERENCE". In: *NVIDIA* (2017). DOI: https://arxiv.org/pdf/1611.06440.pdf.

[7] Y. Cheng D. Wang P. Zhou and T. Zhang. *A Survey of Model Compression and Acceleration for Deep Neural Networks*. URL: https://arxiv.org/abs/1710.09282v9.

[8] Wikipedia. *Advanced Video Coding*. URL: https://en.wikipedia.org/wiki/Advanced_Video_Coding.

[9] Wikipedia. *High Efficiency Video Coding*. URL: https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding.

[10] Iain Richardson. *H.264/AVC Intra Prediction*. URL: https://www.vcodex.com/h264avc-intra-precition/.

[11] Minhua Zhou & Raj Talluri. *Macroblock*. URL: https://www.sciencedirect.com/topics/engineering/macroblock.

[12] Cisco Annual Internet Report. *Cisco Annual Internet Report (2018–2023) White Paper*. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[13] J. Vieron and I. Trow. *AV1: Implementation Performance and Applications*. URL: https://www.ibc.org/download?ac=6506.

[14] NVIDIA. *rtx-30-series-av1-decoding*. URL: https://www.nvidia.com/en-us/geforce/news/gfecnt/202009/rtx-30-series-av1-decoding/.

[15] C. Liu. *AOM AV1 vs HEVC/H.265 Will AV1 Dominate in 4K/8K Processing?* URL: https://www.macxdvd.com/mac-dvd-video-converter-how-to/aom-av1-vs-hevc-h265.htm.