Speeding up Deep Learning with Quantization

SoonYau
Nov 18, 2018 · 5 min read

In last week, Facebook has just open sourced their matrix multiplication library which you can read it here .Readers may quickly find the word “quantized” or “quantization” appear a lot in that article and wonder what is magical about this new hype word that help giving 2.4x performance boost on CPU. I’m going to give some beginner’s introduction to quantization, I may use some simple maths a long the way but don’t worry, I promise it is very gentle.

Recent advancement in AI or more specifically a technique called deep learning (DL) brought a lot of excitement about the type of applications that are possible e.g. object detection, speech synthesis and image generation. Despite of many exciting news about breakthrough in research or applications, the fact is that we still don’t see a lot of products or services (yet, but it is changing rapidly). One of the reasons is that it is costly to run deep neural network which often require expensive and power hungry GPU (graphical processing unit) to run. This became obvious to me when I worked on project to use DL to analyse real-time video back in 2 years ago. The cost of renting GPUs on cloud would easily eat up all the profits. It is no coincidence that dozens of chip companies sprung up to make specialized chip to speed up deep learning. I personally have been interested in speeding up deep learning inference and this led me to studying quantization for neural network.

What is quantization?

Quantization isn’t something new, it has been around for decades since the creation of digital electronics. When we take photo on our cellphone, the real world scene which is analog is captured by the camera and digitized into computer files. Quantization is part of that process that convert a continuous data can be infinitely small or big to discrete numbers within a fixed range, say numbers 0, 1, 2, .., 255 which are commonly used in digital image files. In deep learning, quantization generally refers to converting from floating point (with dynamic range of the order of 1^-38 to 1x10^{38}) to fixed point integer (e.g. 8-bit integer between 0 and 255). Some information will be lost in quantization but researches show that with tricks in training, the loss in accuracy is manageable.

Why quantization?

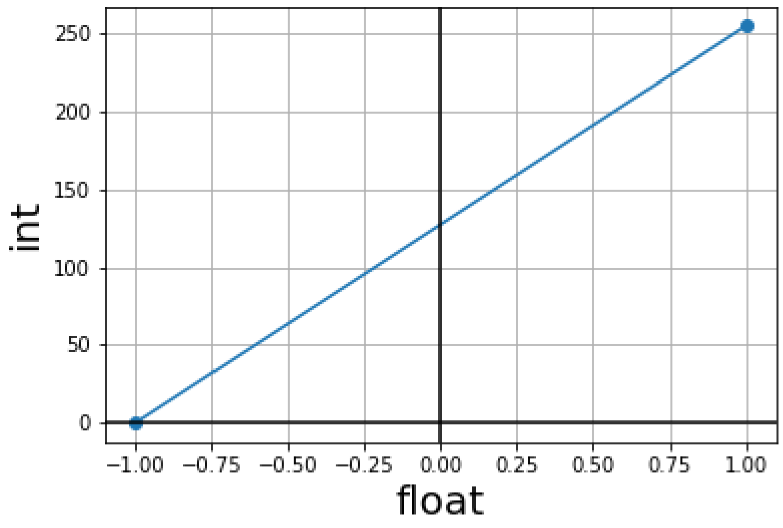
There are two main reasons. Deep neural network consists of many parameters which are known as weights, for example, the famous VGG network has over 100 million parameters!! In many scenarios, the bottleneck of running deep neural network is in transferring the weights and data between main memory and compute cores. Therefore, as opposed to GPU that have one large shared pool of memory, modern AI chips have a lot of local memories around the compute cores to minimize the data transfer latency. Now, by using 8-bit integer rather than 32-bit, we instantly speed up the memory transfer by 4x! This also bring other benefits that accompanies smaller file size i.e. less memory storage, faster download time etc.

How to quantize a pre-trained network?

It is actually quite straight forward (but the training a quantized-aware network is a bit tricky which I will explain in later articles). In FBGEMM, the matrices are quantized per row which has finer granularity. In Tensorflow Lite, the quantization happen per layer which is coarser but same theory applies to both. The mapping from float to int is simple and we only need two parameters — scale and offset.

$$x_{int8} = \frac{x_{float}}{x_{scale}} + x_{offset}$$

You can take out a pen and paper, and draw a x-y plot where x axis is the floating number, and y axis is the integer. Say the trained weights have floating range of between -1.0 and +1.0 and integer has range between 0 and 255. Now mark a point (-1.0, 0) and another point (+1.0, 255), then draw a straight line connecting them. The inverse of the slop of line is the scale and the y-value at the intersection of the line with x=0 is the offset which is 128 (rounded from 127.5) in this example. As you could recall to maths at school, this is exactly linear equation y = mx+c. See? This is no rocket science.



Now when we want to perform calculation, we'll need to convert the integer back into floating point which is simply the reverse of quantization equation i.e. you minus the offset from integer and multiply with scale. The second reason for quantization is that floating point compute core is more complex than fixed-point's counterpart, therefore take up larger silicon area and more power hungry. As we have shown, number format is just a way of representation. In dequantization, we could convert the 8-bit integer into higher precision fixed-point integer, say int32. There are additional processing steps and bitwidth limitation which I'll not go into depth here. If all calculations run using fixed-point operations, then we could probably fit the neural network into small microcontroller or DSP for embedded applications. Or, chip designer could squeeze it higher number of compute cores into the chip hence increasing the computing throughput.

Lastly, after processing a layer, the resulting floating point are converted back into quantized integer, this process is known as re-quantization. This is required because current machine learning framework and GPU need to transfer the layer output back to main memory before transferring it again to start the next layer. New processor and machine learning framework architecture could make this requantization-dequantization step redundant if multiple or all layers can be computed without having to transfer the intermediate layer output back to main memory.

Why 8 bits?

Mainly because it is the smallest bitwidth supported by mainstream computers, going below that will probably deteriorate the performance due to extra steps to unpack the data. Furthermore, currently, the accuracy of network drop rapidly below 8-bit. Quantizing to lower bitwidth is an active research and a breakthrough in research coupled with new type of processor, we may see many times speed improvement in near future.

Want to know more?

Hope you enjoy reading this article. If you are interested in more technical details and software implementation, feel free to visit my Github repo where I have just started writing tutorials about Quantized Neural Network.

Machine Learning Deep Learning