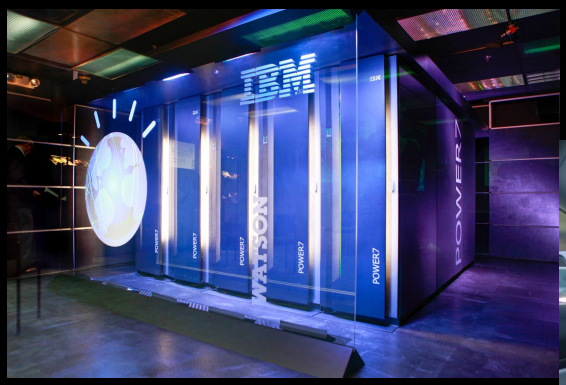
Artificial neural networks, everywhere!



(Ideal)

Just a brief introduction to artificial neural networks, ANNs are digital constructs inspired by our own brains that layer neuron-like components on top of each other and are then trained to perform specific tasks, such as object recognition and identification. In general, due to their versatility, ANNs are transforming how we use and interact with technology. We use neural networks to help autonomous cars drive, to fly autonomous drones and track objects of interest, to handle voice recognition for our cars, phones, and home automation systems, and even to help diagnose medical conditions and cancer. Ideally (assuming we solve the privacy and other ethical implications of the technology), neural networks can power systems to automatically recognize us as we arrive home and turn on lights and other parts of the house, could help us track the amount of food in our pantries and fridges (as well as how much we’re eating), help manage forests by identifying changing conditions and possibly help predict wildfires on site, and so on.

So, what's the catch?

(reality + problem)

So, why aren't they doing all of these wonderful things and making our lives simpler? Well, training and running current ANN implementations require large amounts of matrix operations, and are traditionally run on GPUs that can do the required number of operations on a reasonable timeframe. The problem with this approach is that these GPUs require a lot of power, sometimes many hundreds of Watts, limiting where these can be run to devices with a lot of power available, such as PC's, datacenters, or some modern cellphones. It may be possible to run ANNs on small microcontrollers, but the performance penalty of not using a GPU to parallelize the required computations could be massive (and still power hungry for some really low-power environments). So, are we stuck to relying on the power grid for large power-hungry GPUs? Are there alternatives?

What about FPGAs?

- Low power
- Customizable
- Highly parallel

(Solution)

So in summary, the problem is that conventional implementations of NNs require too much electrical and computational power to allow them to really become ubiquitous. Enter Field Programmable Gate Arrays, or FPGAs. An FPGA is a collection of digital logic circuits that can be rewired to do different tasks. Think of FPGAs as a grid (or fabric) of logic gates (AND, OR, NOT, etc.) that can be re-arranged to form arbitrary circuits (in reality they're a bit more complex than this, but this mental model is close enough to understand what they are doing). Since when working on an FPGA we are working with circuits, it becomes much easier to organize components so that many, many operations can be done in parallel, more than can be done in a CPU and possibly even a GPU. Moreover, because of the level of control over the circuits being implemented granted to use by FPGAs, it is possible to implement only what is needed and nothing more, yielding power savings when compared to a CPU or a GPU.

The reality, though, is that FPGAs are not a silver bullet. If you remember the mental model of a grid of logic gates, one of the primary physical imitations of FPGAs is that the grid is finite-- there are only so many components in total that can be rewired. To make matters worse, most NNs use floating point numbers for their weights (think state), which are expensive to represent on FPGAs, exacerbating the space problem on FPGAs. So, what can we do?

Project Goals

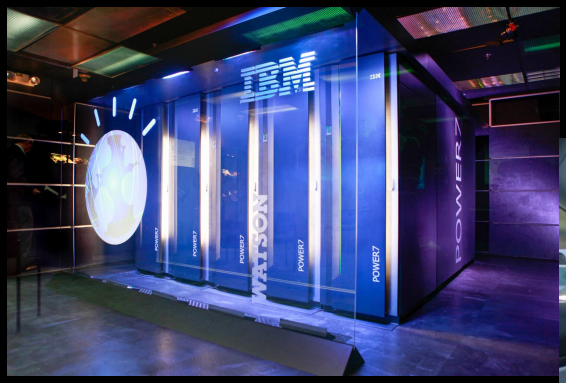
- Optimize ResNet20 for FPGAs
- Retrain with different datasets and explore accuracy
- (stretch) Try applying approach to a larger ResNet

Our proposed solution builds on work already being done here at UCSD. Existing work has taken ResNet20 (a NN with skip connections, or connections between non-adjacent layers) and optimized it for FPGA use by quantizing the weights (what this means is representing them as integers instead of floating point) and by using novel algorithms to reduce the number of skip connections required, as these connections require extra FPGA resources to implement. Our solution will build off of this and investigate if using a technique called knowledge distillation can be effective in further optimizing ResNet20**.** We will also explore if it is better to do KD before quantization or after by comparing the accuracy of the different implementations, and against the original tensorflow software implementation.

So far current work has only used one dataset, so we hope to expand it to use others to demonstrate that it continues to work with different kinds of data.

As a stretch goal, should we complete everything else, we hope to adapt a larger ResNet to the existing framework, train it, and quantize, KD, and trim skip connections.

Conclusion



In conclusion, artificial neural networks are everywhere and have many uses, including automated driving and object detection and identification, but we are currently limited in their use due to the power and compute requirements of NNs on GPUs. Work is being done on leveraging FPGAs as an alternative, and our solution will explore further optimizations to FPGA designs with the hope of making these realizable on hardware and eventually leading to more efficient NNs that can be deployed in more places.

Any questions?