# HonestNN: An Honest Neural Network "Accelerator"

Tim Linscott Vidushi Goyal Andrew McCrabb Pete Ehrett*
University of Michigan, Ann Arbor

## ABSTRACT

It seems like everybody has been making their own neural network accelerator recently [1][2][3][4][5][6][7][8][9]. We figured we could do better. We kind of do. Probably. Anyway, neural networks are giant approximate systems with lots of repeated operations. We tried to have the hardware do some approximations of its own to make the computation go faster. And it does. Except that sometimes it will misclassify a picture of your cat as being a previously unknown painting by Van Gogh or something like that. I'd tell you about how much better we do, but our method for calculating performance is a little convoluted so that we look better. You should just skip to the Methodology section to read about it. But when we do use that method, we demonstrate a 15% "neural-adjusted performance" increase. Look, just go read the methodology, okay?

**Keywords:** Accelerators, Architecture, Machine Learning, ML, AI, DNN, CNN, Graph Processing, Blockchain, Cryptocurrency, 3D Printing, Security, Crowdsourcing, Computer, Electronics, Science, Engineering, #NovelResearch2019, #BuzzwordsGetCites, #BetterScience

## 1 INTRODUCTION

$10^{-12}$s after the big bang, a cooling universe allowed for the formation of the electron. Eventually, some of these electrons would find a new vocation when, around 550-600 million years ago, early bilaterians evolved a simple nerve cord. This opened the door to the process known as encephalization in which animals evolved brains and various sense organs. Most recently, an up-jumped primate at the forefront of the encephalization process figured out a way to compel those electrons to do neuron-like calculations, and branded it the "Artificial Neural Network." Recently, researchers have used this most recent turn of events for the 13.8 billion year-old electron to automatically generate memes and classify laundry [10, 11].

But in order to keep up with the universe's blindingly fast rate of progress, we all decided that we need to make those electrons *even more* efficient at generating memes and sorting laundry. This gave rise to the neural network accelerator. A bunch of researchers have made versions of these things.

*Pete is a CMU alum, which is how he knew we should send this to SIGBOVIK. He even has the ACH mug to prove it. Tim is currently affiliated with a company that would probably be embarrassed to know that he wrote this paper, so we'll just call it "Shmoogle".

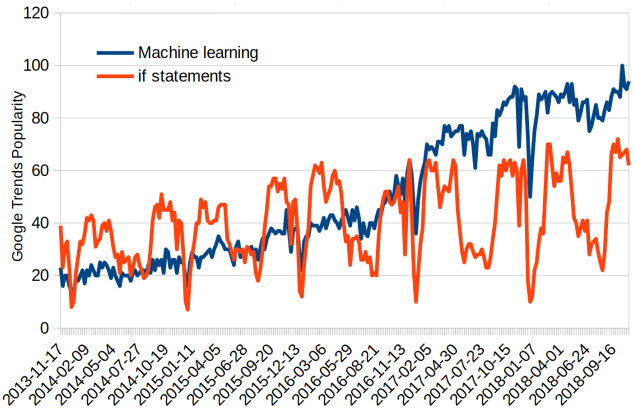


**Figure 1: Ever since late 2016, Machine Learning has become more popular than if statements, according to Google Trends search data.**

Though honestly, only a few have actually made the things. Mostly, we just run a synthesis job and do a simulation and call it a day. It's only because nobody else wants to put up with taping chips out that we can get this paper in.

We present HonestNN, a neural network accelerator where for once the grad students who actually worked on the project actually tell you what they actually did and didn't do. Our contributions are as follows:

- We tried to build and test a neural network accelerator.
- We develop a novel metric for evaluating accelerator performance that also happens to be the only metric in which we beat the state-of-the-art.
- We wrote a paper detailing our implementation and results. This is the big one.
- We cited a bunch of your papers, including one by the TPC chair.
- We express unparalleled honesty in the reporting of our results.

## 2 BACKGROUND

Machine Learning is an important paradigm in computer programming. According to Figure 1, since December 2016, there have been more Google searches for "Machine Learning" than for "if statements." Clearly, we should expect more ML in the future and fewer conditional branches. After all, when is the last time you read an "if statement accelerator" paper? Unless you count branch predictors, probably never. But what actually is machine learning? This section addresses that question.

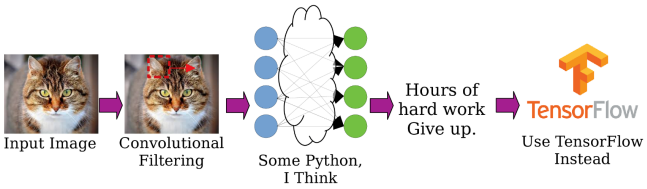


**Figure 2: As far as I can tell, this is how state-of-the-art inference is done.**

## 2.1 Modern ML Algorithms

There are a lot different ways to do machine learning nowadays. Mostly, though, we just use Deep Neural Networks (or "Machine Learning" in popular nomenclature) usually in the form of TensorFlow, though the research community tends to focus on how to use MLPerf as if that were a real thing. Most ML code tends to be of the form

```
import tensorflow

tensorflow.process("my_data.bin")
print("This is a picture of a cat.")
```

As the above code snippet shows, we can take advantage of the fact that almost every CNN or image recognition presentation that has a figure for image processing uses a picture of cat. Seriously, it's like a fundamental law of nature. To break down modern CNNs even further, Figure 2 shows how the process usually works. First, a picture of a cat is provided to the CNN. The CNN "convolves" the image using a processes called "convolution." Then there's some more code, which is probably written in Python, just like everything else is these days. You can either work really hard to create your own neural network, or you can just give up and use TensorFlow instead.

# 3 ARCHITECTURE

Maybe you guessed this from the last section, but I'm not that familiar with machine learning (seriously, when I review ML papers, I only put my expertise as a 2 if I'm feeling *really* confident.) But from what I hear, there are a lot of matrix and vector multiplications going on. And it's iterative or something, so you reuse the results from one step in the next step. So we have one cache to store the weights that we multiply with the features and another one to store the results. Then there are a ton of compute units. I was sure that I would need to do vectorized signed floating-point multiplication and some kind of threshold function, but I wasn't sure what else needed to be done in each step, so I thought I'd cover my tracks and just insert a complete RISC-V core in each of the compute units.

## 3.1 Optimizations

Our architecture takes advantage of real-time DNN pruning to problematically eliminate low-impact nodes. Real talk, though? We kind of forgot to connect a few of the compute units. But what are the odds that an important node is going to find itself in the broken pipeline? Pretty low, it turns out [12]. And the power we save not processing those probably unimportant nodes can get turned into increased clock speeds.

To improve transfer speeds between HonestNN and its peripherals, we use USB3.0 for maximum data rates.

I really wish this section was longer. I barely got HonestNN working in time to write this. There were bugs right up to the deadline, so I definitely did not have time to optimize anything. I think one of my source files has an `#ifdef USE_OPTIMIZATION` directive, and I'm 100% sure that `#define USE_OPTIMIZATION` was commented out by the end of my last debugging session.

## 3.2 Security Considerations

Everybody has been talking about security recently, so I assume that I need to address any potential reviewer comments for this architecture, too. The good news is that a lot of really smart people are coming up with a lot of novel security solutions. So for our threat model, I assume that we are to secure the part of the cloud that we want to perform our computation in. If the cloud is unhackable, then by deploying our chip in the cloud, we will be unhackable as well.

We plan on further ensuring the security of our solution using 1,048,576-bit, thousand-round RSA secured via the Blockchain, because we assume quantum computing will never really become a Thing. Also, because we haven't released our code yet (see Section 4.1), we also have something called "security through obscurity," which despite not being that good, is still something.

Real talk, though? I don't know what an attack on a neural-network accelerator would look like. It seems to me that attacks would all be higher up in the stack since HonestNN only exists to do math really fast. It would be like attacking a calculator. That said, malware targeting calculators is a real thing, so maybe I should be worried. [13]

# 4 METHODOLOGY

## 4.1 Framework

We wanted to implement HonestNN in SystemVerilog, but were told by people paid more than us (who may or may not be last authors of this paper), to implement it in VHDL. A week later, these same so-called "authors" changed the requirement to C++, then Python, then probably HTML (I fell asleep during that meeting), and then finally SystemVerilog.

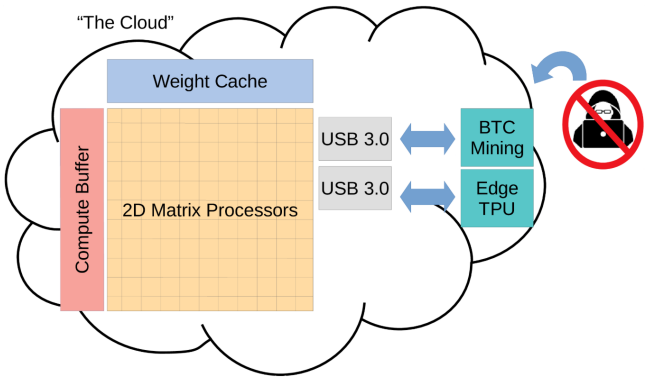


**Figure 3: A diagram of how our accelerator works. Our architecture lives in an "unhackable" "cloud." The Edge TPU is included in the event that the rest of the accelerator isn't working, but if Google won't give us one of those, we will fall back on mining Bitcoin to pay for this project.**

We're pleased to report that we started in SystemVerilog from the beginning, so there was no lost time.

In the end, we implemented HonestNN in SystemVerilog. By "we" I obviously just mean the grad students on this paper, *i.e.,* the entire remaining author list since one of us got annoyed and took off everyone who didn't actually do the work. When's the last time you've ever heard of a professor writing code for a research project? And by "implemented" I mean prototyped. By "SystemVerilog," I mean to say that I wrote the various components so that I could synthesize everything in the end. Okay, not "everything," per se; we used CACTI [14] for the caches and stuff, because I couldn't figure out how to get those to synthesize from the Verilog. For testing, I wrote this great simulator that shows how everything is supposed to work in a mostly cycle-accurate way. I did that in Python, so it took about a day to write and five days to run. I think I used a little bit of BookSim as well [15]. Or at least, I opened it once. And how can I forget gem5? Any architecture paper that doesn't mention a gem5 simulation is crap. *You* may have used real hardware but nothing beats "cycle-accurate simulation." And if you believe I actually even went as far as doing any of the above, I have a bridge to sell you. Why do you think my "code" isn't on GitHub? Hey, I'm citing your papers; don't judge me.

So how do we turn this pile of overlapping implementations into meaningful, comparable results? If it weren't for the fact that I promised unrivaled honesty in this paper, I'd probably just mumble something about "benchmarking" before suggesting we take the conversation offline. The short version is that we use that Python simulator I mentioned. But I'll have you know that I wrote that simulator as a proof-of-concept within the first month of the project and only touched it again before the deadline when I realized that nothing else was going to give me meaningful performance numbers. The simulator assumes an idealized neural network perfectly fitted to our design and fully utilizing all its compute without thrashing the cache. Then it outputs a single number representing the number of operations performed by each compute unit, which gets fed into another script that I wrote last night to turn that into neural-adjusted performance, which we discuss below.

## 4.2 Neural-Adjusted Performance

We define "Neural-Adjusted Performance" as GOPs per Watt per dollars of NRE normalized against process node, algorithm, and years since the release of TensorFlow v1 (lower is better). Let's walk through each of these in turn. First, GOPs are counted by the number of irreducible mathematical operations performed by the accelerator in the best case where the data exactly fits the pipeline width and all nodes are in use and no lanes have accidentally been disconnected from the rest of the architecture. Second, we normalize with respect to power, because that's the limiting factor in new architectures according to my advisor [16]. Third, we normalize against the Non-Recurring Engineering expenses that went into developing this chip. Since the chip was designed entirely by underpaid grad and undergrad students desperate to get the project done before the last of the grant money dried up, it cost less than $100,000—much less than the average industry-made chip. We normalize against process node to balance out the fact the we spent 10 months trying to file paperwork that would get us access to real-world standard cell libraries and getting ignored by university and industry bureaucrats so many times that we finally gave up and used a really terrible open-source standard technology library. Not that I'm bitter or anything. We also normalize against the algorithm used. While we working on HonestNN, some people came out with a bunch of cool techniques in software that would make neural networks way faster. We figure that if we'd known about them, our core would be a lot faster, so we factor in the speed and accuracy of the fastest state-of-the-art neural network at the time the paper came out. But we also need to give credit where credit is due. Later researchers are really just poaching good ML ideas from TensorFlow, so the earlier researchers had a leg up on us in that they were forced to be creative and smart and come up with novel ML techniques on their own. If we weren't afflicted by analysis paralysis due to being surrounded by so much ML literature and so many ML APIs and ML papers, we'd probably have come up with some even better ideas of our own. So, by normalizing against the number of years since TensorFlow v1 was released, we attempt to level the playing field between us and the researchers of yesteryear.

All in all, this yields a broad perspective on the "is it worth it to spend this amount of time and money on improving

automated laundry categorization by 0.1%?" question that all developers and researchers ask themselves at some point.

### 4.3 Benchmarks

We tested our neural network accelerator using SPEC2017 as well as a basic Hello World program. We also tried to use MiBench, but it was way too hard to port without knowing the rationale behind the hard coded values (yes, you heard it right), so we gave up there.

Next, we used LinPack. Have you heard of this one? It's like a ten million by ten million matrix-matrix multiply. No one in the real world would ever need to do a ten million by ten million matrix-matrix multiply. But this is academia and not even my advisor knows what's going on inside real-world machine learning, so I guess this means that this is our best guess. But, hey, LinPack is great. It rewards ~~stupid~~ elegant architectures like HonestNN that are pretty much just bundles of compute cores. No one should be the least bit surprised that we did pretty well here.

Cloud-based neural network workloads demand that providers balance handling large volumes of requests with the need to serve responses at high speeds. Thus, neural network accelerators need to be designed around a conscious bandwidth-latency trade-off. We designed our accelerator to be easily customizable to optimize for one design point or the other. We analyzed a comprehensive selection of benchmarks targeting bandwidth and latency goals with a wide range of possible configurations of the accelerator. Figure 4 presents a pareto curve of the optimal accelerator configurations.

Look. My advisor wrote that paragraph just a few minutes before the deadline. I don't want to take it out, but in the interests of honesty, I'm going to repeat what was just said with a few corrections. Here goes:

Cloud-based neural network workloads demand that providers balance handling large volumes of requests with the need to serve responses at high speeds (*pretty true*). Thus, neural network accelerators need to be designed around a conscious bandwidth-latency trade-off (*but this is definitely not a knob you can easily tune*). We designed our accelerator (*my advisor still doesn't remember the project's name*) to be "easily" customizable to optimize for one design point or the other (*I literally had an undergrad in tears trying to configure this thing*). We analyzed a comprehensive selection of benchmarks (*Hello World and LinPack*) targeting bandwidth and latency goals with a wide range of possible configurations of the accelerator (*five configurations in total*). Figure 4 presents a pareto curve of the "optimal" accelerator configurations (*yeah, right*). One of my co-authors (I'm not sure who, because we're all editing this at once) keeps editing this section to complain how that's not really a pareto curve, but it's 2 hours before deadline and I literally don't care anymore.

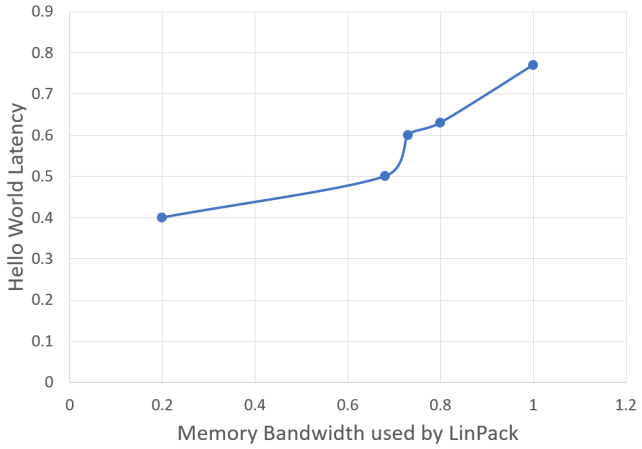


**Figure 4: HonestNN can be configured to prefer latency (time to execute Hello World) or bandwidth (memory bandwidth consumed running LinPack). I'm not really sure what the take-away is here other than "Hey, look! A pareto curve! These guys clearly did a thorough analysis!" (But wait, no, that isn't actually pareto, is it...)**

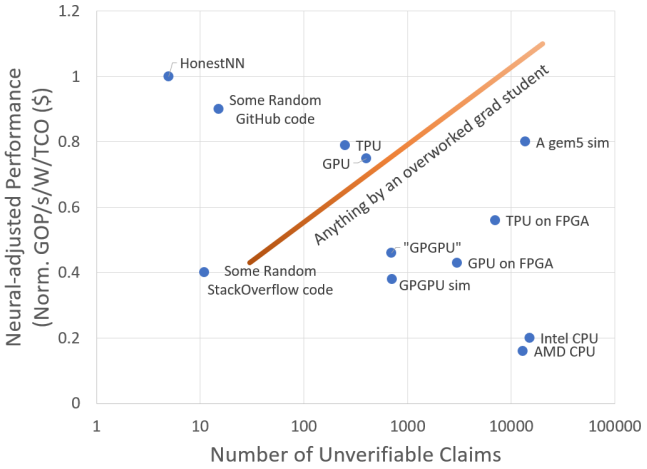


**Figure 5: HonestNN beats the state-of-the-art on both neural-adjusted performance and the honesty with which we present our results.**

## 5 RESULTS

### 5.1 Performance

Figure 5 compares HonestNN to the state-of-the-art research and industry solutions and to random internet code. But did you know that it's really hard to compare your research project to other people's research projects? It's not like they share the data behind their graphs. So all those research papers *that I was going to cite if I'd had access to some useful data for once* all contained large numbers of claims that I can't verify. Oddly, I found that a research paper's performance was directly correlated to the number of dubious

claims made by the authors. As for industry solutions, it's not like NVIDIA is going to give the source code to their latest GPU. Still they have some insanely dense manuals that are less fun to read than your average angry adviser email and which make me wonder whether any of the engineers even understood the design docs at all. On the other hand, we also looked into a number of sources of random internet code some of which has been elevated to the level of divine revelation by academics. gem5, for instance, which brings nightmares and madness to any foolish enough to study its unfathomable eldritch mysteries, is considered the gold standard in computer architecture simulation. However, it is built from 13,716 different commits at last count, none of which make any sense. Meanwhile, you can find some random code on StackOverflow that accelerates neural networks as long as you're willing to assume that "the memory subsystem can't be that complicated" and won't play into performance too much.

Dear reader, we make no such assumptions, which is why HonestNN leads the pack with fewer unverifiable claims than the state-of-the-art and the internet at large. The reason for this is simple: we do not claim that our accelerator actually works at all, nor do we claim that if it did work, you would actually want to use it. In fact, I'll go out on a limb and claim that neither of these is the case at all. But one thing's for certain: its neural-adjusted performance normalizes to one.

### 5.2 Design Costs

To determine the area and power of our design and its maximum clock speed, we synthesized our implementation using Synopsys Design Compiler. Turns out that Design Compiler is really, really, *really* slow and that it gives different numbers every time you rerun synthesis. We basically just launched a bunch of different synthesis runs with random edits made to our synthesis script and picked whichever one had completed and gave us the best results by the time this paper deadline rolled around. Honestly, I'm not sure if Design Compiler just stumbled upon a pure genius optimization that time or whether this is an erroneous result. Because there was this one other synthesis job that gave me the same clock frequency but at like 5× more area. Next time we'll just wave the rubber chicken of `compile_ultra` and call it a day.

In the end, though, synthesis results really aren't *that* important since we're going to normalize everything in our performance analysis just like the other cool researchers do [17]. In short, our accelerator is of size 1, consumes 1 unit of power, and operates at a clock speed of 1.

### 5.3 Development Cost

We hired a bunch of underpaid world class labourers (read graduate students), some apprentices (read undergraduate interns), and finally used the latest "up-to-date" free tools provided by the university to build our awesome accelerator–HonestNN–which beats the so-called "state-of-the-art" NN accelerators from industry built by overpaid engineers and researchers with the most sophisticated technology. Such a waste of resources! Not to mention, by the end of this project, the lead author of this paper was roped in by the leading software giant because of his *sheer* expertise in the area. Well, knowing that the fate of the world is in *honest* hands, I can finally rest in peace.

## 6 LIMITATIONS

In the interests of full disclosure, we present the weaknesses associated with HonestNN.

- There's really nothing concrete I can point to and say "this is HonestNN." I mean, there are a bunch of scripts and log files, but there's really no single source of truth here.
- I never really verified the functionality of HonestNN. I ran a few tests on a couple Verilog modules and the simulator works "fine" (see below), but I don't think that really counts.
- I've found bugs in the simulator and reran the results more times than I can count. Sometimes fixing the bugs made HonestNN faster, sometimes it made it slower. Did I catch all the bugs? I really don't know anymore. But the deadline is looming large, so I guess I have to act like I did.
- I was really nervous about the whole "untested chip" thing, so I snuck a TPU into our design. I don't know what the rest of HonestNN is needed for if we have a TPU on hand. Why not just use the TPU?
- If anyone looked at my code, I would either be blacklisted from every major tech company from now until the singularity occurs or forced to memorize every known style guide before I was even allowed to *look* at a computer again.

## 7 RELATED WORK

Okay, so there are a bunch of academics who thought of ideas for neural network accelerators. But there's no way I'm going to be able to cite them all. Seriously, NIPS had like 3,000 submissions this year. So I'm just going to cite the top nine papers from Google Scholar [1–9], and another paper from my advisor's best friend's former postdoc who gave a really memorable talk or something [18]. I'm not totally sure how the implementation on this one worked, but it seemed interesting and I'm including it anyway [19]. But none of these people have a hundred billion dollars. You know who does have a hundred billion dollars? Google. And what do you know? They made their own neural network accelerator,

too. And actually built it. And we can actually use it. But they're not academics, so all I'm giving them is this one lame citation: [20].

If you're one of the blind reviewers and you're trying to find the authors of this paper in related work or metadata, let me spare you the effort and include some egregious and unwarranted citations of our own papers so that it's easy to figure out who we are: [21, 22].

## 8 FUTURE WORK

At some point, we should probably build a working version of HonestNN that really does run from start to finish instead just being a delicately balanced pile of scripts with all the structural integrity of a house of cards. My advisor wants to have the next version use emerging NVM technologies, and our sponsors want to know whether our work has any applications to autonomous driving.

Personally, if this paper gets in, I'm never coming back to this project. I need novel ideas to fill out my thesis, and incremental improvements on this dead horse of a project won't make the cut. Besides, no one really gets around to their future work anyway. Do you?

## 9 CONCLUSIONS

Who cares? By the time you read this, something else will have come along that supersedes everything we built for this paper. Or that did the analysis better. Or that faked their results to make it look better. Or the singularity will have occurred. Whatever.

But before you think of rejecting this paper out of hand because of its shoddy implementation and weak evaluation methodology, why don't you take a minute to talk to your grad students and make sure this isn't happening under your nose right now? Go ahead, I'll wait. But you can be sure of one thing: HonestNN is the most honest neural network accelerator you'll ever see published. Please?

## REFERENCES


[1] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.

[2] Ryuichi Sakamoto, Ryo Takata, Jun Ishii, Masaaki Kondo, Hiroshi Nakamura, Tetsui Ohkubo, Takuya Kojima, and Hideharu Amano. Scalable deep neural network accelerator cores with cubic integration using through chip interface. In *SoC Design Conference (ISOCC), 2017 International*, pages 155–156. IEEE, 2017.

[3] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. Yodann: An ultra-low power convolutional neural network accelerator based on binary weights. In *ISVLSI*, pages 236–241, 2016.

[4] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. A high-throughput neural network accelerator. *IEEE Micro*, 35(3):24–32, 2015.

[5] Laura Fick, David Blaauw, Dennis Sylvester, Skylar Skrzyniarz, M Parikh, and David Fick. Analog in-memory subthreshold deep neural network accelerator. In *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4. IEEE, 2017.

[6] Kalin Ovtcharov, Olatunji Ruwase, Joo-Young Kim, Jeremy Fowers, Karin Strauss, and Eric S Chung. Accelerating deep convolutional neural networks using specialized hardware. *Microsoft Research Whitepaper*, 2(11), 2015.

[7] Ryuichi Sakamoto, Ryo Takata, Jun Ishii, Masaaki Kondo, Hiroshi Nakamura, Tetsui Ohkubo, Takuya Kojima, and Hideharu Amano. The design and implementation of scalable deep neural network accelerator cores. In *Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 2017 IEEE 11th International Symposium on*, pages 13–20. IEEE, 2017.

[8] William J Dally, Angshuman Parashar, Joel Springer Emer, Stephen William Keckler, and Larry Robert Dennison. Sparse convolutional neural network accelerator, February 15 2018. US Patent App. 15/458,837.

[9] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. In *ISCA '16*. IEEE, 2016.

[10] V Peirson, L Abel, and E Meltem Tolunay. Dank learning: Generating memes using deep neural networks. *arXiv preprint arXiv:1806.04510*, 2018.

[11] Li Sun, Simon Rogers, Gerardo Aragon-Camarasa, and J Paul Siebert. Recognising the clothing categories from free-configuration using gaussian-process-based interactive perception. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2464–2470. IEEE, 2016.

[12] Who_really_cares. xkcd: Machine learning. In *Does-anyone-read-this-part-anyway?*, 2017.

[13] Piotr Bania. Gaara, 2007.

[14] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. Cacti 6.0: A tool to model large caches. In *International Symposium on Microarchitecture*, 2007.

[15] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *ISPASS '13*, 2013.

[16] Todd Austin. Application-specific design tutorial. In *University of Michigan EECS 573: Microarchitecture*, 2018.

[17] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pages 367–379. IEEE Press, 2016.

[18] Alessandro Aimar, Hesham Mostafa, Enrico Calabrese, Antonio Rios-Navarro, Ricardo Tapiador-Morales, Iulia-Alexandra Lungu, Moritz B Milde, Federico Corradi, Alejandro Linares-Barranco, Shih-Chii Liu, et al. Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. *IEEE transactions on neural networks and learning systems*, pages 1–13, 2018.

[19] Doug Zongker. Chicken chicken chicken: Chicken chicken. *Annals of Improbable Research*, pages 16–21, 2006.

[20] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *ISCA '17*, pages 1–12. IEEE, 2017.

[21] Timothy Linscott, Pete Ehrett, Valeria Bertacco, and Todd Austin. Swan: mitigating hardware trojans with design ambiguity. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2018.

[22] Pete Ehrett, Vidushi Goyal, Opeoluwa Matthews, Reetuparna Das, Todd Austin, and Valeria Bertacco. Analysis of microbump overheads for 2.5 d disintegrated design. 2017.