Medical Cost Estimation Using Hardware-Accelerated Neural Networks

Seth Polsley
Texas A&M University
spolsley@tamu.edu

Yayun Liu
Texas A&M University
Liuyayun915@tamu.edu

Pu Chen
Texas A&M University
pu_chen@tamu.edu

ABSTRACT
Big data is becoming increasingly important in the modern world due to the amount of technology used in everyday life. Yet, it's potential for learning new information has been relatively untapped due to the complexity of dealing with such large amounts of data. This work explores the implementation of an integer neural network in hardware, demonstrating the advantages that can be achieved both by reducing the complexity of machine learning computations and using these simplified algorithms as the basis of dedicated, high-speed machine learning components. The classifier is applied to big data from an analytics perspective by trying to predict care costs from healthcare data, which was selected in order to show the breadth of information that can be gathered from big data. The hardware implementation of the neural network does show dramatic speed advantages, and the cost predictor indicates that there is a great deal of relevant data to be used for expenditure prediction, although the specific model used could be improved.

1. INTRODUCTION
Big data is everywhere. Increasingly, there are overwhelming amounts of data being generated each day due to technologies like electronic record-keeping, mobile cameras, and a wealth of digital sensors. In the field of healthcare alone, a vast collection of data about individuals' health and activities is provided by all kinds of digital equipment, and healthcare organizations are attempting to leverage this data by interpreting their records with machine learning techniques. There is a huge potential for knowledge in healthcare analytics, knowledge that can keep doctors and patients better informed. For instance, Google Flu Trends can be used to track the spread of conditions like influenza across the country [1].

However, all kinds of big data analytics face challenges. On massive datasets, machine learning algorithms may run very slowly. For example, many algorithms in speech processing, image processing, and pattern recognition exist but are slow due to computational complexity [2]. The software is being pushed to the limit, and in order to meet faster processing demands, hardware implementations are becoming necessary. In this work, the authors explored the hardware implementation of a specific machine learning tool: the neural network. Specifically, an integer neural network (INN) was designed and tested; INNs have several advantages over traditional neural networks, among them being performance and cost advantages due to reduced complexity. Their lower complexity also makes INNs excellent candidates for dedicated hardware components that can be included in low-cost systems which would benefit from built-in machine learning capabilities. To demonstrate a practical application of this component in a field that could gain much from high-performance learning, it is used as the basis for a healthcare expenditure predictor.

2. PREVIOUS WORK
As the field of machine learning grows, researchers have begun to experience software's limitations. Several groups have published work on implementing machine learning classifiers in hardware due to the outstanding performance gains. Mahmoodi et al. [3] proposed a simple hardware architecture for implementation of pairwise SVM classifier on FPGA. In the architecture, vector multiplication and pairwise classification are designed to run in parallel. Reyna-Rojas et al. [4] chose to implement the SVM classifier on an SoC platform in order to exploit the parallel nature of the SVM algorithm.

Other classifiers have also been studied for their benefits as hardware components. Most relevant to this paper, Eberhardt et al. [5] created a hardware neural network to gain advantages from the algorithm's parallelism as done in [3]. In Jung and Kim [6], the authors showed that their hardware neural networks were well-suited to real-time applications. There is even a commercially-available neural network chip provided by Cognimem, which has been used effectively in hardware-software co-designed systems [7]. Sun and Cheng [8] demonstrated a fast and efficient neural network in hardware within the field of healthcare, using their system to analyze patient's cardiograms to detect abnormalities that could lead to heart attacks or disease.

Additionally, many hardware neural networks are variants of the standard algorithms which use integer calculations only, called integer neural networks, or INNs [9]. INNs have been becoming increasingly popular in recent years for their numerous advantages. There is an inherent speedup available to INNs because of the removal of all the floating point calculations [10]. FPUs are also quite expensive, so these neural networks are ideal for low-cost solutions [11]. Unfortunately, training integer-based networks is more difficult because weight updates may not propagate well using a discretized function, a similar problem experienced in deep learning where weight updates may underflow and not propagate. While some new algorithms are being developed for training integer weights [12], most systems work by first performing offline training on a standard neural network and then scaling the weights and activation function to integers using a defined bit resolution.

Using machine learning, it is possible to predict much more from healthcare data than just patient risks for certain illnesses. In [13], Shang and Goldman used age and life expectancy to predict a patient's care costs. Others have studied the effects of conditions ranging from rheumatoid arthritis to morbid obesity, as seen in [14] and [15], on healthcare costs. There are many other possible factors that may be considered when using healthcare data to predict financial information, such as patient gender or race [16][17]. This work considers several common features that are present in most health records, many of which have been shown in previous studies to be contributing factors to cost.

3. PROPOSED WORK

As seen in the previous section, there has been considerable research over the past few years studying the benefits of hardware implementations of machine learning classifiers. Also, much research has been done in the field of healthcare analytics to study care costs. In the proposed work, we seek to combine these two ideas using hardware-software co-design to find a fast and efficient solution. That is, we hope to gain the performance boosts afforded by the hardware-based classifiers and tie them to the practical information gained by statistical machine learning. To our knowledge, while hardware-based classifiers and software-based care prediction have been studied in some depth separately, they have not been combined in a co-designed system to improve the performance of these often-slow software operations. It is also our hope that the healthcare cost estimation technique will be novel in its simplicity and capable of performing nearly as well as methods published in other research.

The proposed work is divided into three phases. The first phase consisted entirely of the software model. This includes the implementation of a standard neural network and a less-complex integer neural network. Both networks were tested on a widely-used digit recognition dataset before being used to build the data model.

The second phase focused on re-implementing the software INN as a hardware component. The machine learning model created in the first phase is unchanged, using the same data and feature set. However, this phase did include evaluation of the hardware implementation, using timing gathered from both the Carbon Design System's virtual prototyping tools[1] and Xilinx's ModelSim[2].

The final phase was developed concurrently with the other two. It was focused on acquiring and cleaning the healthcare data. A model to represent the most relevant data for the desired problem of cost estimation also had to be constructed.

4. IMPLEMENTATION

Phase I – Software Implementation

The first task was the development of a software neural network. This neural network was written in C++ for system portability and included implementations of standard floating-point algorithms like backpropagation for training. To test the validity of the implementation, it was trained and tested on a well-known machine learning dataset—the MNIST handwritten digit database[3], a subset of which is shown in Figure 1. It was tested alongside the open-

[1] http://www.carbondesignsystems.com

[2] http://www.xilinx.com

[3] http://yann.lecun.com/exdb/mnist/

source Weka toolkit (Waikato Environment for Knowledge Analysis). Weka is a flexible tool developed by the University of Waikato that can be used for testing a variety of machine-learning related analyses. As found using Weka and supported by a number of publications, neural networks perform very well on the MNIST dataset, often achieving accuracies close to 100%. The neural network written by the authors of this work performed favorably as well, achieving 96.94% accuracy from a single training session.

Next, an INN was built by modifying the existing network to use integer-based calculations. As discussed in the previous work, backpropagation is not typically used for INNs because of bit resolution limitations leading to poor weight updates. While some newer training methods for INNs are being developed, the INN implemented in this case was designed to convert the trained network created from the floating-point calculations into one that could be used with an integer-based feed forward method for classification.

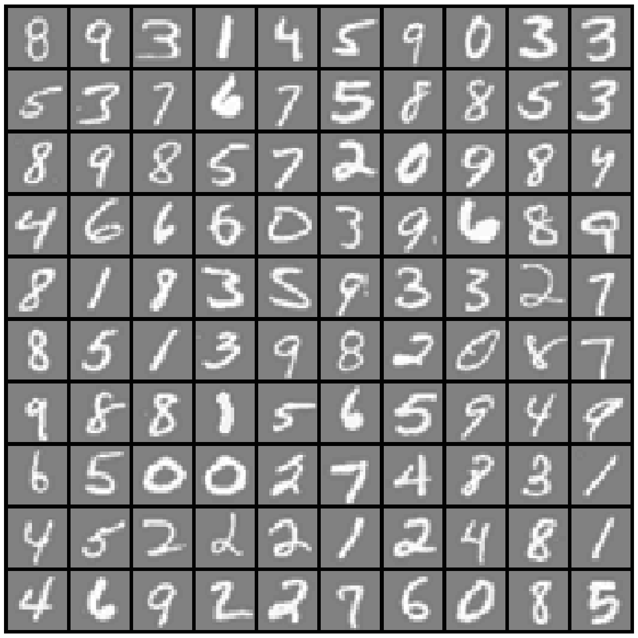
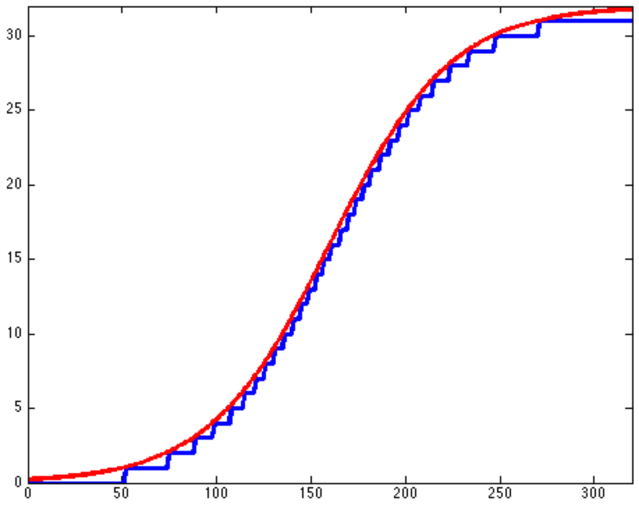

F

Figure 1: Visualization of examples from the MNIST handwritten digit dataset

Some special considerations must be made when building an INN from a standard neural network. Even without using backpropagation for training, the feed forward technique can be complicated. It involves propagating neuron values from layer n - 1 to layer n, multiplying each neuron's value by the weight associated with the edge between the neuron at n - 1 and the neuron at n, and then normalizing all of those values to a new "activation" value of the neuron. As the name implies, these values specify how activated the neuron is, and they propagate forward through the network using weights in the same technique that was just described. Typically, activation values are determined using an activation function. These functions must be able to map any input from negative to positive infinity into a bounded value, usually between -1 and 1 in floating-point networks. Most good activation functions experience the most extreme change near the origin and have little change at very low or high values; the sigmoid or hyperbolic-tangent functions are often used. Because of the non-linear nature of these functions, INNs will often precompute and scale the values to be stored in a look-up-table (LUT). The LUT produced for the healthcare data is shown in Figure 2 along with the actual sigmoid curve. This discretization process demonstrates the issue with using integer-based backpropagation, since weight updates are based on the derivative of the activation function. When scaled for integers, almost all of the weight updates would be zero outside of the rapidly changing center of the sigmoid; floating-point calculations capture such small updates much better, until overflow becomes a problem in deep networks.



F

Figure 2: The standard Sigmoid curve (red) shown against a discrete estimation of the curve saved in a LUT (blue)

The feed forward algorithm also needs integer weights, so these too are scaled from the weights calculated from the original neural network. When determining the scale for the weights and the activation function, it is important to consider the size of the network. In C++, integers are 4 bytes, so the activation function must be able to accept any 32-bit value as an input. Layer sizes, weight resolution, and activation function resolution must all be carefully balanced to ensure there is no overflow, which would corrupt the output at the current layer and prevent proper classification. For any given neuron, the maximum value that may be generated from the feed forward algorithm is the number of neurons at the previous layer times the maximum value for weights times the maximum value of activation. That is, the INN must be defined such that 2^n * 2^w * 2^a does not exceed 2^32. Selecting the bit resolutions of w and a must be done carefully in order to minimize the loss of accuracy that comes from converting to a discrete scale.

The INN was written to be able to convert the weights and inputs of any generic network to any bit resolution. It also automatically generated a LUT for the activation function at the desired resolution. To test the INN, the same digit recognition problem was run using 12 bits for the weights (+/- 2048) and 12 bits for the activation value. The final accuracy was 96.14%, which demonstrated a loss in accuracy of less than a percent.

Phase II – Hardware Implementation

With the software already written, this portion of the project was focused on porting the INN to the Carbon Design tool and building a Verilog representation that could be synthesized into a standalone INN component. Running the INN on the ARM Cortex A9 was accomplished by modifying the existing code base for the sort algorithms. The feed forward and activation lookup functions replaced the sort algorithms and the semihost controller was used to read necessary values from the hard drive into the simulator (including the network parameters and the test data).

Once the software had been fully tested, a Verilog INN was written. This iteration of the INN operated in much the same way as the C++ version, but at this point, the parameters were hard-coded in the network. Loading the parameters could be very slow using a serial protocol like APB, and since that stage was unimportant for the feed forward timing, parameter loading was bypassed.

The Verilog INN was tested first using Xilinx's ModelSim. Test data was written to the INN chip in the testbench and the result was read at its output. After testing, a Carbon component was built using ModelStudio that could be imported into Carbon Design.

Phase III – Healthcare Data Processing

In order to apply the INN to cost prediction, a model for the healthcare data had to be established. Before that, the data had to be acquired and cleaned. The data used in this application was selected from the Texas Inpatient Public Use Data File (PUDF). This is a large data set with patient records from all over Texas, detailing hundreds of features. A portion of data from the first quarter of 2012 from the base one of the PUDF set was used. While this trimmed down the number of patient records from millions to about 250,000, there were still over a hundred features per record. This would yield poor results in many machine learning applications, so a subset of features was selected. Based on work from [9] and [12], common features such as gender and age were chosen. Additional features included information directly tied to healthcare charges, such as patient diagnosis and length of hospital stay. The complete list of selected features, both the PUDF name and field description, is shown in Table 1. An additional record was kept from the data set, which was the total charges column. This was later taken as the desired output for supervised training of the neural network for cost estimation.

PUDF Field Name	**Description**
TYPE_OF_ADMISSION	Type of admission, e.g. emergency, urgent, etc
PUBLIC_HEALTH_REGION	Public health region of patient's address
SEX_CODE	Patient's gender
LENGTH_OF_STAY	Length of stay
PAT_AGE	Patient's age

PUDF Field Name	**Description**
PRINC_DIAG_CODE	ICD-9-CM diagnosis code for the principal diagnosis
POA_PRINC_DIAG_CODE	Code identifying whether Principal Diagnosis code was present at the patient was admitted to the hospital
MS_MDC	Major Diagnostic Category
MS_DRG	Diagnosis Related Group
APR_MDC	MCC as assigned by 3M APR-DRG Grouper
APR_DRG	DRG as assigned by 3M APR-DRG Grouper
RISK_MORTALITY	Risk of mortality score
ILLNESS_SEVERITY	Severity of illness score

Table 1: List of selected fields from the PUDF database and their descriptions

Following selection of data and features, there were still anomalies such as blank fields and different data types. For classification, all records must contain numeric features with no missing information. To clean the data, it was imported into Matlab. Simple scripting operations were performed in Matlab that converted all the values to decimal numbers (including mapping from string identifiers) and assigned empty fields default values. After ensuring there was no missing information, all of the fields were normalized to the same scale of 0 to 1. Thus, for each column, the maximum value was now 1 instead of a random number. Such scaling ensures uniformity for the network as well as simplifying the task of scaling inputs to the INN later at the appropriate bit depth.

In order to allow the classifier to estimate costs, the total charges needed to be grouped into classes that specified ranges of cost. Since this was the output to be used in supervised learning, the size of a range would give the granularity of the classifier's estimation capabilities. By studying the cost densities in the data, a set of bin edges was determined through experimentation that yielded a fairly even distribution of records. The selected bin edges were (in dollars): 10000, 20000, 35000, 60000, 100000+. Each record was grouped into the bin for the edge it was nearest to but not exceeding. Figure 3 shows the final distribution for the data.

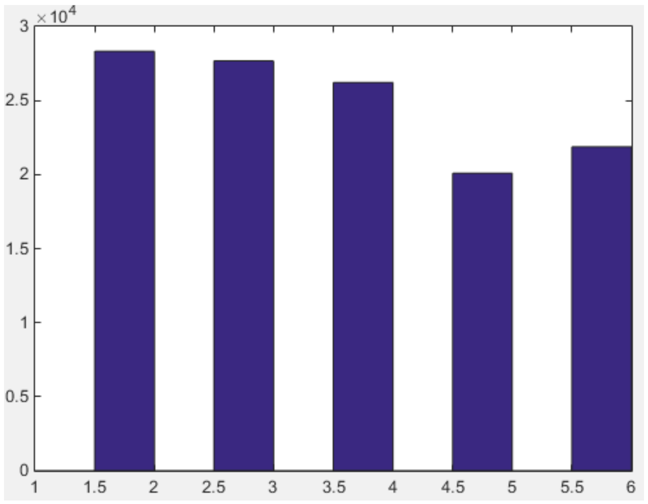


Figure 3: Distribution of healthcare data along the defined cost categories

5. RESULTS

The initial testing results from the MNIST handwritten digit dataset were reported in the previous section, but to reiterate, the standard neural network and the INN implementations achieved 96.94% and 96.14%, respectively. The loss in accuracy was minimal for this problem.

The final model for the healthcare data was first run through the standard neural network in order to train weights. The constructed network had 13 input neurons (corresponding directly to the cleaned and

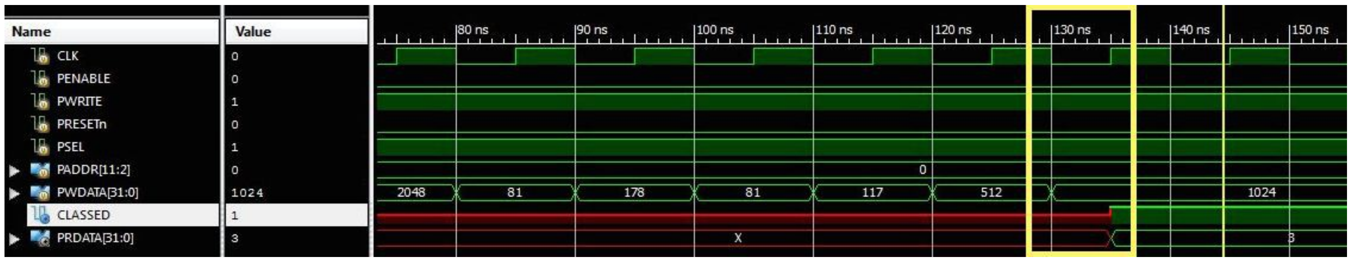


Figure 4: ModelSim timing diagram for the Verilog INN

scaled 13 input features), 25 hidden neurons, and 5 output neurons. The final accuracy was 43.43%. While there is certainly room for improvement, this accuracy shows that some important information is being captured. As Figure 3 shows, the data is roughly uniformly distributed, so a random classifier might be expected to get approximately 20% probability over the five cost regions. This model achieves over twice that accuracy.

All of the network parameters were then saved and scaled to be compatible with an INN with 13x25x5 neurons. The weights were given 12 bits of resolution for the healthcare data, but the activation LUT was only given 6 bits. This reduction in resolution was motivated by the goal of reducing the memory space required to store the LUT in memory. Unfortunately, reducing the bit resolution did come at the cost of reduced accuracy. The INN achieved only 37.13% accuracy on the same data using the same parameters with integer scales. While the 6% loss in accuracy is not as insignificant as the loss experienced in the first test dataset, it did come with the advantage of reducing the activation LUT by nearly two orders of magnitude in size. The size was an important consideration for using the Carbon Design tool, so the reduced resolution was ultimately selected over the higher accuracy.

All of the previous results focused on showing the validity of the networks' implementations and that some of the features of the healthcare data could be used to predict costs. Finally, some timing information was gathered to compare the performance of the software INN to the Verilog one. The standard neural network was not included in this experiment. As mentioned before, many embedded system applications prefer INNs because the ability to perform neural network operations without an FPU can dramatically reduce cost. Thus, in order to use the ARM Cortex A9 without an FPU, only the INN was used.

First, the software INN was run directly on the A9 with a subset of the testing data to gather timing information. The feed forward algorithm was run for many iterations. Using the reporting tools available in Carbon Design and its cycle-accurate timing, the average number of cycles required for a single feed forward operation was found to be 8811 cycles, with a standard deviation of 126 cycles. Next, the Verilog INN was tested in ModelSim and as a Carbon Design component. This implementation was extremely fast because all of the computations at a single layer could be performed simultaneously with dedicated hardware. The feed forward operation took only a single cycle in these tests and did not exceed that at any time. Figure 4 shows the timing diagram generated by ModelSim. The test data is first read into the INN chip and then once it has received an entire sample, it performs the classification. The relevant region is outlined and shows how quickly the classification is achieved after all the inputs are received.

6. DISCUSSION

The software implementation of the INN was very slow compared to the hardware version. When working with smaller sets of data, the limitations of software are not always that apparent. Even from the results gathered here, 9000 cycles would take fewer than 5 microseconds on a modern 2 Ghz processor. While that may be imperceivable by itself, as the amount of data grows, those 5 microseconds add up and become very significant, and researchers working with big datasets frequently do experience the limitations of software that become apparent when that happens.

Conversely, the hardware INN chip was extremely fast. Having dedicated circuitry for all of the calculations made the 5 microsecond operation occur in nanoseconds. This is an enormous time saving and could scale up very well to working on large datasets. In order to make faster systems for working with big data, hardware implementations of machine learning classifiers are going to become increasingly important. Cost and power are also significant concerns in co-design, and they influence how much dedicated hardware will be included in any system. As shown in this work, neural networks at least may be converted into very low-cost systems by removing the FPU requirement and using integer-based computations. Expending effort in making other classifiers integer-computable would likely be worthwhile and save cost in the future.

The classification results on the healthcare data also showed that some relevant information exists that could be used to predict care costs. The accuracy was certainly not as high as it could be, and there is much room for improving the model. Regardless, it performed well above a random classifier and demonstrates that more information can be collected from healthcare data than just information for disease detection or predisposition. Such results exemplify why big data is becoming a very important discussion; big data techniques could be used to detect, predict, and understand so much new information beyond what is visible on the surface. A different data model or classifier may perform better on this dataset, but the more research focused on analyzing large datasets in unique ways, the more new information will be gathered.

7. CONCLUSION

This work focuses on big data from two perspectives. First, it uses big data as an analytics tool to attempt to learn new information. In this case, the question was if healthcare data could be used to predict more than traditional "health" information; could it be used to look at the economics of healthcare? A data model was generated that attempted to capture the most relevant features that indicated expected healthcare expenditures. While the results showed that the model and classifier could be improved, they did demonstrate that there is important information to be gathered beyond just health data by performing well above a random-chance classifier.

Second, big data was viewed from an optimization perspective. While many machine learning algorithms and tools have been heavily optimized to work as quickly as possible in software, there are still limitations. Those speed limitations become very apparent as the amount of data grows, so a hardware neural network was synthesized and prototyped in a virtual environment to show its benefits. The hardware network could perform a massive number of operations in parallel and increased performance dramatically. These results would argue for the usage of dedicated machine learning chips in the field of big data, since they can scale to large amounts of data much more easily than software. While there is an additional cost for the hardware, it can be minimized by reducing requirements like the FPU, and over time, the cost impact could be further reduced by the savings in speed and power.

8. REFERENCES

[1] N. Wilson, "Interpreting Google flu trends data for pandemic H1N1 influenza: the New Zealand experience," European Communicable Disease Bulletin 14.44 (2008): 429-433.

[2] K. James, et al., "Fuzzy models and algorithms for pattern recognition and image processing," Vol. 4. Springer, 2005.

[3] D. Mahmoodi, et al., "FPGA Simulation of Linear and Nonlinear Support Vector Machine," Journal of Software Engineering and Applications, 2011, 4, 320 - 328.

[4] R. Reyna-Rojas, et al., "Object Recognition System-on-Chip Using the Support Vector Machines," EURASIP Journal on Applied Signal Processing, 2005:7, 993 - 1004.

[5] S. Eberhardt, et al., "Design of Parallel Hardware Neural Network Systems from Custom Analog VLSI 'Building Block' Chips," Neural Networks, 1989. IJCNN., pp 183 - 190.

[6] S. Jung and S. Kim, "Hardware Implementation of a Real-Time Neural Network Controller With a DSP and an FPGA for Nonlinear Systems," IEEE Transactions on Industrial Electronics, February, 2007, VOL. 54, NO.1.

[7] S. Sardar, et al., "A hardware/software co-design model for face recognition using cognimem neural network chip," Image Information Processing (ICIIP), 2011 International Conference on. IEEE, 2011.

[8] Y. Sun and A. C. Cheng, "Machine learning on-a-chip: A high-performance low-power reusable neuron architecture for artificial networks in ECG classifications," Computer in Biology and Medicine, 2012 Jul; 42(7):751-7.

[9] E. Won, A hardware implementation of artificial neural networks using field programmable gate

arrays, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 581 (3) (2007) 816–820.

[10] V. Vanhoucke, et al., “Improving the speed of neural networks on CPUs,” pp. 1–8.

[11] T. Behan, “Integer Neural Networks On Embedded Systems,” Recent Advances in Technologies, Maurizio A Strangio (Ed.), ISBN: 978-953-307-017-9.

[12] V. P. Plagianakos and M. N. Vrahatis, “Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights.”

[13] B. Shang and D. Goldman, “Does Age or Life Expectancy Better Predict Health Care Expenditures?,” Health Econ. 17:487-501 (2008).

[14] K. Michaud, et al., “Direct Medical Costs and Their Predictors in Patients With Rheumatoid Arthritis,” Arthritis & Rheumatism, Vol. 48, No. 10, October 2003, pp 2750-2762.

[15] D. E. Arterburn, et al., "Impact of morbid obesity on medical expenditures in adults." International Journal of Obesity 29.3 (2005): 334-339.

[16] G. M. Owens, “Gender Differences in Health Care Expenditures, Resource Utilization, and Quality of Care,” J Manag Care Pharm. 2008; 14(3)(suppl S): S2-S6.

[17] C. C. Wee, et al., “Health Care Expenditures Associated With Overweight and Obesity Among US Adults: Importance of Age and Race,” American Journal of Public Health, Jan. 2005, Vol 95, No.1.