The effect of DBN circuits on the energy consumption of a program

Chrizzell Jay Sanchez
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— The purpose of this report is to take a programmed that students created in MIPS, which take two inputs in the form of words and searches for the words within a paragraph and outputs how many times they are found, and simulate how implementing the different Deep Belief Network Circuits would affect the instruction count of the code. First the program is introduced, with an outline of how it works, along with possible inputs to test the program. After, the concept of DBN's are introduced, with brief descriptions of a few possible circuits. Then, the DBN circuits are implemented into the program, and the energy consumption is looked compared between four of the possible circuits. Finally, the results are cross referenced, and a decision is made which one consumes the least energy.**

***Keywords*—Deep Belief Network (BN), Probabilistic Interpolation Recorder (PIR), Deep Convolutional Generative Adversarial Network (DCGAN), Deep Learning.**

I. Introduction

A. *Project Design*

The purpose of this code is to take an input paragraph, which was hardcoded in this case, and search for two specific words that are designated by the user. The words that can be searched for can be any combination of capital or lower-case letters, and up to ten letters long. In order to do this, a previous code was modified and repurposed. The previous code would look to find how many times a certain letter was found residing within a paragraph. This was changed in order to find whole words rather than just letters. First, the paragraph would be taken, and the user would be prompted to input two words to find within the paragraph. Then, the program would load the first letter of the first input word and the first letter of the paragraph and compare the two. If the letters matched, the next letter of the word and paragraph would be loaded, until the word was found. If the word matches, then the counter for that word would go up by 1, and it would continue through the end of the paragraph. However, if the letters do not match, it will go to the next letter in the paragraph but compare it to the first letter in the input word until it matched. After finishing the paragraph, the program would print the results and repeat the process for the second input word.

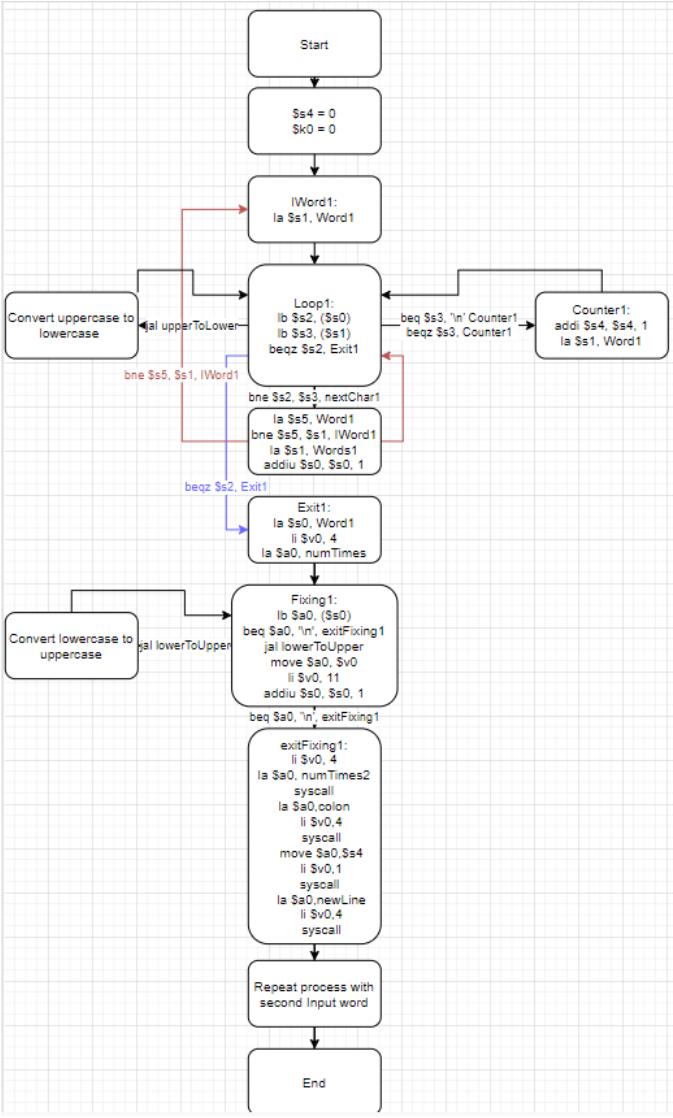

Fig.1: Flowchart of the assembly program.

B. *Test Cases*

In order to test that the code was properly working, the inputs that the code took were varied and stress tested. The first two words that were attempted to be found were the words "UCF" and "KnIgHt." These were chosen because this would test whether the program could still function even if the words are not properly formatted, with having random letters capitalized.

The next set of words chosen were "grad" and "erg". These words were chosen because they are parts of words that are contained in the paragraph but are not whole words. This shows that the program is not looking for words entirely but comparing letter by letter.

The next test case was "grant." And "grant". These were chosen because the word grant is found several times within the paragraph, but only once does it have a period after it. Thus, if the program properly passes, it showcases that even non letters are accounted for.

```
Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight,...):
KnIgHt
# Number of times the word 'UCF' was found in the input statement: 2
# Number of times the word 'KNIGHT' was found in the input statement: 6
-- program is finished running --
```

```
Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight,...):
erg
# Number of times the word 'GRAD' was found in the input statement: 3
# Number of times the word 'ERG' was found in the input statement: 1
-- program is finished running --
```

```
Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight,...):
grant
# Number of times the word 'GRANT.' was found in the input statement: 1
# Number of times the word 'GRANT' was found in the input statement: 7
-- program is finished running --
```

Table I: Energy consumption for a branch instruction in the designs provided in [1-4].

Design	Energy Consumption For Each Branch Instruction
[1]	0.2 pJ
[2]	$0.03e^{+5}$ pJ
[3]	$6e^{+5}$ pJ
[4]	$5e^{+5}$ pJ

II. DBN Circuit

A DBN Circuit is a circuit in which the concept of Deep Learning can be applied. It is a model that consists of several layers of logic that are connected to each other to determine a probability based upon its inputs. Each layer builds upon the previous layer to decide an output. By training a DBN, classifications can be decided after several hundreds or even thousands of inputs. It is an algorithm that can be used in several real-life applications. A DBN is made up of several Restricted Boltzmann Machines connected. One possible circuit is the probabilistic interpolation recorder (PIR) circuit. The PIR circuit takes inputs and outputs them digitally. The benefit of a PIR circuit is that it lowers energy consumption and it reduces the need for many physical devices such as resistors and capacitors and instead replaces them with transistors. A possible circuit to train DBN's is known as a Deep Convolutional Generative Adversarial Network (DCGAN). This circuit is a combination of two different models, one being the discriminator model, while the other is a generator model. The generator model will create takes inputs in the form of images, and slightly alter them. The discriminator model will then take the images fed from the generator model among unaltered images and attempt to determine whether an image was altered or not.

While DCGAN is a software implementation, there are also possible hardware architectures that can be used in DBN's. Using hardware to implement a DBN circuit is helpful because it can offer speedups several times faster than an optimized software program. One implementation is that of several cores in series. There is also a route for a DBN to attempt character recognition. This can be used by implementing FPGAs and using a software approach rather than a hardware approach.

III. Results and Discussion

Overall, when the code implemented searches for "knights" and "knightS" it has 13826 ALU instructions, 7092 jump instructions, 12391 branch instructions, 2860 memory, and 11083 others. In order to find the difference between the different circuits, we substitute the energy difference for the branch instructions for the circuits and find the total overall energy. The ALU instructions take 27652 pJ, the Jump instructions take 28368 pJ, the Memory instructions take 286000 pJ, and the other instructions take 55415 pJ. Table II

shows the Energy Consumption between the four different circuits when they are applied.

Table II: Total Energy consumption for the assembly program using designs provided in [1-4].

Design	Total Energy Consumption
[1]	$39.99e^4$ pJ
[2]	$37.57e^6$ pJ
[3]	$7.43e^9$pJ
[4]	$6.2e^9$pJ

IV. Conclusion

The DBN Circuit that consumes the least amount of energy is the first design, which is the PIR circuit. This circuit has 2 orders of magnitude less energy consumption than the next smallest circuit. This could be because the main use of a PIR circuit is to minimize energy consumption. From this project, I further enhanced my understanding of arrays and how to manipulate them, learned how to take technical papers and make them easier to understand, learned what a DBN is and how they can be implemented, learned how to write and format a research paper, and learned about image recognition technology.

References

[1] H. Pourmeidani, S. Sheikhfaal, R. Zand, and R. F. DeMara, "Probabilistic Interpolation Recoder for Energy-Error-Product Efficient DBNs with p-bit Devices," IEEE Transactions on Emerging Topics in Computing, 2020.

[2] A. Roohi, S. Sheikhfaal, S. Angizi, D. Fan, and R. F. DeMara, "ApGAN: Approximate GAN for Robust Low Energy Learning from Imprecise Components," IEEE Transaction on Computer, vol. 69, no. 3, 2020.

[3] D. Le Ly and P. Chow, "High-performance reconfigurable hardware architecture for restricted boltzmann machines," IEEE Transactions on Neural Networks, vol. 21, no. 11, pp. 1780–1792, 2010.

[4] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "Vlsi implementation of deep neural network using integral stochastic computing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2688–2699, 2017.

[5] K. Sanni, G. Garreau, J. L. Molin and A. G. Andreou, "FPGA implementation of a Deep Belief Network architecture for character recognition using stochastic computation," *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, MD, 2015, pp. 1-5.