**Resource Consumption between Convolutional, Artificial and Spiking Neural networks**

Nirav Reddi
Graduate Student
1328, Market st, Tacoma
98402
+1 253 426 7338
nreddi98@uw.edu

## ABSTRACT

In this paper, a study of convolutional , artificial networks and spiking neural networks is performed through an experiment. Through experimentation,a study is made about differences between them , their applications and then profiled , to study them on the metrics of resources they consume and accuracy that can be achieved through using them. Neural networks were developed on common MNIST dataset and profiled using Nvidia Nsight Compute.

### CCS Concepts

• **Computing Methodologies→Artificial Intelligence→Computer Vision→Computer Vision Task→Neural Networks.**

### Keywords

Convolutional Neural networks, Artificial Neural Networks , Pytorch,SnnTorch, CUDA

# 1. INTRODUCTION

The paper compares the performance of Artificial Neural Networks , Convolutional Neural Networks and Spiking Neural networks .The paper and the project started with the intention of comparing spiking neural networks and their efficiency in NeuroMorphic Computing.

At the start of the paper , there wasn't any provided proof that spiking neural networks are more efficient than ANNs and CNNs. There was an article about IBMs breakthrough in this field with proof that SNNs are efficient . This is the first time Neural Networks are being compared based on the same accelerators , hardware and software. During the draft of the paper and project , authors are unaware of the IBMs paper or results .

The neural networks, being compared mostly run on similarly shaped neural networks , which make the neural network performances comparable .Although it needs to be acknowledged that , there are a lot of other things at play when a neural network runs, but for the comparison purpose it is, assumed that all of the cases were the same . The paper compares and concludes that Spiking Neural networks do run more efficiently consuming lower power and clock rate .

# 2. Background

Neuromorphic simulators are references to the existing brain simulators which existed till last year, without any GPU accelerators . There's a published paper on SuperNeuro[1] , which is the first Simulator

to exist that uses GPU acceleration . Taking this paper as the basis , this experiment further use SnnTorch[2], which is used to create Spiking neural networks, based on top of Pytorch[3] .This makes the comparison more practical and similar . As both the packages use similar libraries and Nvidia Cuda accelerators under the hood. With ANN's ,CNN's and SNN's all using the same libraries , and Cuda toolkits. It would be a good case to compare these models and their hardware utilization .

While IBM's paper did, announce that they achieved low resource utilisations for neural networks on the NorthStar platform. It didn't have any accessible papers to have a reference and, thus having a bias free experiment for this particular experimentation. SuperNeuro[1] paper, was way towards , Neuroscience end of the spectrum, this comparison between resource utilization of CNN's , SNN's and ANN's is a useful one in the context of neuromorphic computing.

# 3. Environment

## 3.1 Pytorch[3]

Pytorch and Tensorflow have become industry standards for Machine Learning and Deep Learning . Over the time both of them have evolved drastically. Tensorflow has completely shifted onto cloud and Google collab IDE. Any attempt to use tensor flow locally on the PC has become painful to resolve the library dependencies and cross library usage.Because of this reason after various attempts to run neural networks in tensor flow. Pytroch was selected for its robust usage and GPU acceleration compatibility with the NVIDIA GPUs. This was also because on the other end of the spectrum, Spiking neural Networks , currently only worked on NVIDIA GPUs and utilized similar NVIDIA cuda acceleration for their execution. This made comparison between these libraries and neural networks more reasonable .

## 3.2 snnTorch[2]

While the background started with using a superneuro simulator[1], on going around the search for running spiking neural networks. The Open Neuromorphic[5] community had snnTorch[2] library which used GPU acceleration and had semantics similar to Pytorch[3]. Hence for the use case of comparing neural networks snnTorch[2] is a very good place to start , if we are using Pytorch on the other end for ANNs and CNNs , snnTorch[2] is a good place to start.

## 3.3 NVIDIA Nsight Compute[4]

Nvidia has profiling software that is mostly optimized for gaming . But NVIDIA Nsight Compute,as it also has the ability to profile metrics for Cuda API, along with hardware acceleration . It generates all the graphs in terms of percentages . While keeping the CPU and GPU frequency , as 1Hz.

## 3.4 Dataset[6]

MINST dataset was chosen because it was the only dataset that had proven accuracy comparable to those of ANN's and CNN's. Attempts were made to do it on the CIFHAR dataset, but doing it on such a varied

dataset on spiking neural networks , with additional parameters like num_steps and beta . It is difficult to make it comparable to ANNs and CNNs.

Accuracies are being mentioned,beforehand while comparing the datasets. The goal of this paper is to compare the hardware utilization , instead of accuracy .

| **Neural network** | **Accuracy** |
|---|---|
| SNN | 80 |
| ANN | 90 |
| CNN | 97 |

**Table 1 : accuracies achieved on the three neural networks , on the same dataset with same transformation**

# 4. Neural Networks

Neural networks in themselves can be constructed in various forms and methodologies . There can be all possible forms of activation functions, optimizers and neural structure with any number of layers and neurons in each one of them . But , for a fair comparison it was tried to keep them similar on both ends of the spectrum of ANNs and SNNs.

CNNs aren't an exact copy of the ANNs and SNNs . An attempt was made to have a comparison of CNN+SNN with CNN. But that particular model did generate a report but was not processed by the Nvidia Nsight Compute. But it could still benchmark CNNs , but only ANNs and SNNs can have a fair comparison as they have the same structure .

## 4.1 Artificial neural network

The artificial neural network has the structure :

```
self.block_1 = nn.Linear(28,1000)
self.block_2 = nn.Linear(1000,1000)
self.classifier = nn.Sequential(
        nn.Flatten(),
        nn.Linear(in_features=1000*28,
                out_features=10)
        )
```

It's a very basic structure that generates a 90% accuracy . It works with a simple gray scale image, with dimension 28*28.

## 4.2 Spiking Neural Network

Now , taking the ANN as a reference , generate a spiking neural network with similar neural network shape and size. But this time populated by Spiking layer in between as mentioned below :

```
self.fc1 = nn.Linear(num_inputs,
num_hidden)
self.lif1 = snn.Leaky(beta=beta)
self.fc2 = nn.Linear(num_hidden,
num_outputs)
self.lif2 = snn.Leaky(beta=beta)
```

Spiking neural networks have additional parameters to be populated as you have observed. These parameters as are :

```
# Network Architecture
num_inputs = 28*28
num_hidden = 1000
num_outputs = 10
```

```
# Temporal Dynamics
num_steps = 25
beta = 0.95
```

What these layers are and why they have those values , is beyond the scope of this paper . but you can reference the Open Neuromorphic computer [5] and snnTorch[2] documentation . But on the upper lever it's the same structure as 28->1000->1000->10, that the ANN has.Hence these two can be compared, as these have all the parameters that they have in common , same values and shapes of the parameters.

## 4.3 Convolutional Neural Network

Although it's irrelevant to the scope of comparison that is being made through this experiment. Having a convolutional neural network, that is widely used in the industry and most of the models , sets a good benchmark to draw or have a relative comparison . Hence, convolutional neural network with this shape for comparison is used :

```
ciphar10_model(
  (block_1): Sequential(
        (0): Conv2d(1, 10, kernel_size=(3,
3), stride=(1, 1), padding=(1, 1))
        (1): ReLU()
        (2): Conv2d(10, 100, kernel_size=(3,
3), stride=(1, 1), padding=(1, 1))
        (3): ReLU()
        (4): MaxPool2d(kernel_size=2,
stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (block_2): Sequential(
        (0): Conv2d(100, 100,
kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1))
        (1): ReLU()
        (2): Conv2d(100, 10, kernel_size=(3,
3), stride=(1, 1), padding=(1, 1))
        (3): ReLU()
        (4): MaxPool2d(kernel_size=2,
stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (classifier): Sequential(
        (0): Flatten(start_dim=1,
end_dim=-1)
        (1): Linear(in_features=490,
out_features=10, bias=True)
  )
)
```

This is a decently complicated model that generates 97% accuracy for the MINST dataset[6] .This theoretically should consume the most amount of resources from the GPU , because it's computationally hungry and performs accordingly for the hardware demands .

In the next section , it's discussed about how these 3 neural networks consumed resources on the GPU. The reasons and a proof , that spiking neural networks consume a lot less or resources

# 5. Results

Through this experiment, 4 major resource consumptions by the neural networks are memory transfer, GPU clock frequency,Memory utilization and Cuda API utilization profiled[4] and compared. These all are the metrics that determine the amount of GPU being utilized and resources

being consumed by the neural network to run .

# 5.1 CNN

These proved the most possible accuracy on our Dataset MINST[6], so this one also consumes most resources and enters blocked state multiple times while executing.

### 5.1.1 Memory transfer

The green candles on the graph are . no of times over the time , memory transfers took place between CPU and GPU.

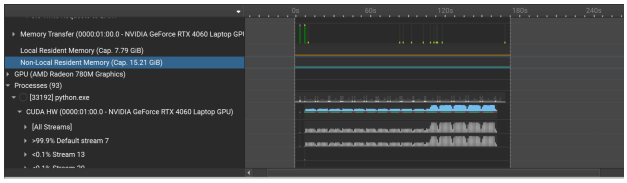


**Figure 1: memory transfer of CNN, indicated with green candles**

### 5.1.2 GPU clock frequency

The line graph with the green color shows the rise and fall in GPU clock frequency over the time as the execution takes place.

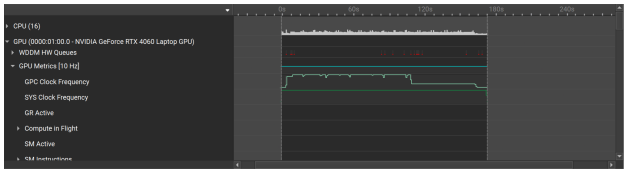


**Figure 2: GPU clock frequency of CNN, indicated with light green line graph**

### 5.1.3 Memory utilization

Notice the gray colored column graph which represents the percentage of the memory being utilized in the GPU . Which spikes to almost 100% for the execution of CNN.

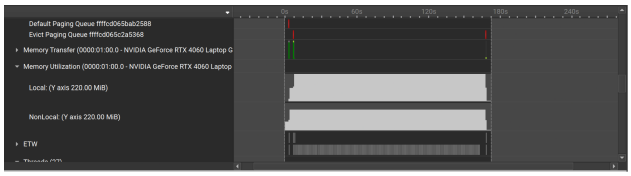


**Figure 3: Memory Utilization of CNN, indicated with gray filled line graph**

### 5.1.4 CUDA API

Cuda is the nVidia's api or library to accelerate the parallel programming on the GPU , to extract faster performance on the GPU . The gray columns on the graphs show how many times that API is used to accelerate the execution in terms of percentages.Also , the time stamps where there is no gray column is the one where execution or GPU entered a blocked state.

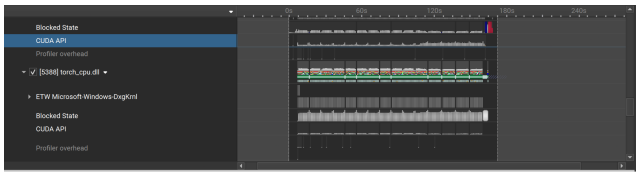


**Figure 4: CUDA API of CNN, indicated with gray column graph**

# 5.2 ANN

These consume way lesser resources , when compared to the ones that are consumed by the CNN's on a similar scale.

### 5.2.1 Memory Transfer

The memory transfer is better than CNNs , where CNN required transfer of memory frequently over the time , these Linear

layers transferred at the start and end on single go .

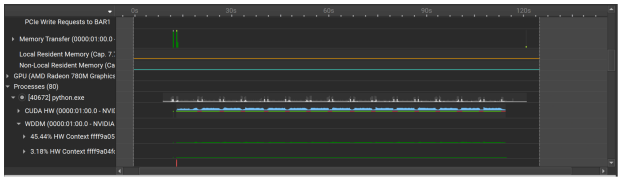

**Figure 5: memory transfer of ANN, indicated with green candles**

### 5.2.2 GPU clock Frequency

The rise in GPU frequency is similar to that of CNNs. It is minutely less , but it's consistent throughout the whole execution , unlike CNNs .

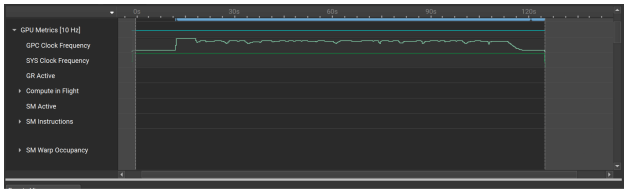

**Figure 6: GPU clock frequency of ANN, indicated with light green line graph**

### 5.2.3 Memory Utilization

The memory utilization of the linear layers of the neural network is half of that of the convolutional layers. This is a significant reduction as seen in the graphs.

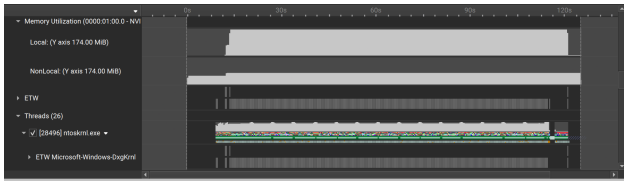

**Figure 7: Memory Utilization of ANN, indicated with gray filled line graph**

### 5.2.4 CUDA API

Cuda Api is utilized for the execution of the linear layered Neural Network. Although the use of the Cuda Api is substantially less than that utilized by the Convolutional neural network.

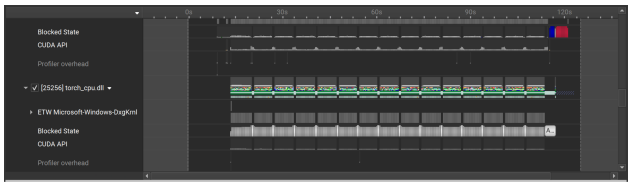

**Figure 8: CUDA API of ANN, indicated with gray column graph**

## 5.3 SNN

Spiking neural networks substantially reduced resources as compared to CNNs and ANNs as mentioned in the graphs below .

### 5.3.1 Memory Transfer

The memory utilization is similar to ANN, where it happens only at the start and the end of the execution on a single go.

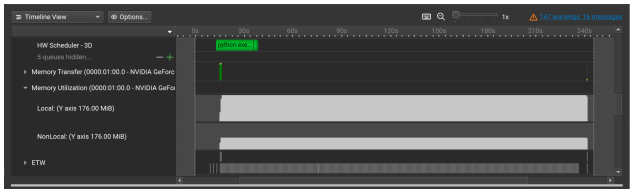

**Figure 9: memory transfer of SNN, indicated with green candles**

### 5.3.2 GPU clock frequency

The GPU clock frequency is the least for this execution . It hardly spikes when compared to ANNs and CNNs.

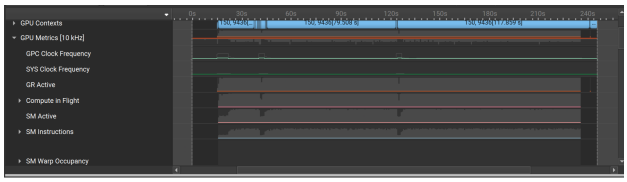


**Figure 10: GPU clock frequency of SNN, indicated with light green line graph**

### 5.3.3 Memory utilization

The memory utilization of the SNN is similar to that of ANNs . Although visually SNNs seem to be lower, these are negligible differences .

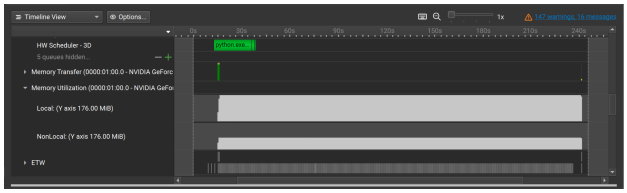


**Figure 11: Memory Utilization of SNN, indicated with gray filled line graph**

### 5.3.4 CUDA API

The Cuda Api utilization by the SNN is the least . And hence the hardware acceleration by it is also least. Notice that SNN, execution also does not enter the block state as often as CNN or ANN.

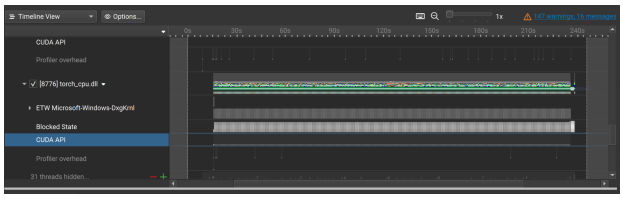


**Figure 12: CUDA API of SNN, indicated with gray column graph**

## 5.4 Comparison

Overall , SNNs run on lower GPU clock frequency and CUDA Api utilization. While consuming over all similar amounts of memory and similar memory transfer .

| Neural network | Accuracy | GPU clock Frequency | CUDA utilization |
|---|---|---|---|
| SNN | 80 | lowest | low |
| ANN | 90 | high | medium |
| CNN | 97 | high | high |

**Table 2 : performance difference of the three neural network based on GPU clock frequency and CUDA utilization**

# 6. Limitations

Although these results seem promising , there are some limitations to what Spiking Neural Networks are doing. These are discussed in this section .

One major difference to be noted is that SNN ran for 3 minutes , whereas ANN ran for 2 minutes . Which means even though SNN with better resource utilization ran for longer.

Apart from this SNN did not give consistent results every time, because multiple attempts to profile SNN were made. And incases where SNN ran for a longer time. The accuracy fell drastically . This might be because of beta and num_steps . SNNs layers must be hit with weights consistently to record the spikes . Whereas this time dependency is not an issue on CNNs and ANNs.

This also calls for cases where hardware resources are not available every time. There is a possibility that the resource unavailability may affect the performance of the spiking neural network.This calls for use cases where SNNs can be deployed on limited or restricted hardware like embedded systems, sensors for microcontrollers.

# 7. Conclusions

The field of spiking neural networks is a very promising one, but also one that is plagued with multiple problems . This leaves researchers with a handful of problems to find solutions to, and doing so can open a possibility of finding an optimal use case and product for Spiking neural networks.

Problems that need immediate solutions are Analysis of brain spiking data and their interpretation and implementation on computer hardware . Spiking Neural Networks need some more mathematical solutions to adapt and implement them in a similar way as ANNs and CNNs , to find more solutions.

# 8. REFERENCES


[1] Prasanna Date, Chathika Gunaratne, Shruti R. Kulkarni, Robert Patton, Mark Coletti, and Thomas Potok. 2023. SuperNeuro: A Fast and Scalable Simulator for Neuromorphic Computing. In Proceedings of the 2023 International Conference on Neuromorphic Systems (ICONS '23). Association for Computing Machinery, New York, NY, USA, Article 40, 1–4. https://doi.org/10.1145/3589737.3606000

[2] SnnTorch Documentation

https://snntorch.readthedocs.io/en/latest/

[3] Pytorch Documentation

https://pytorch.org/resources/

[4] NVIDIA Nsight Compute

https://developer.nvidia.com/nsight-compute

[5]Open Neuromorphic youtube community

https://www.youtube.com/@openneuromorphic

[6]MINST dataset

https://www.kaggle.com/datasets/mannu1170/minst-dataset