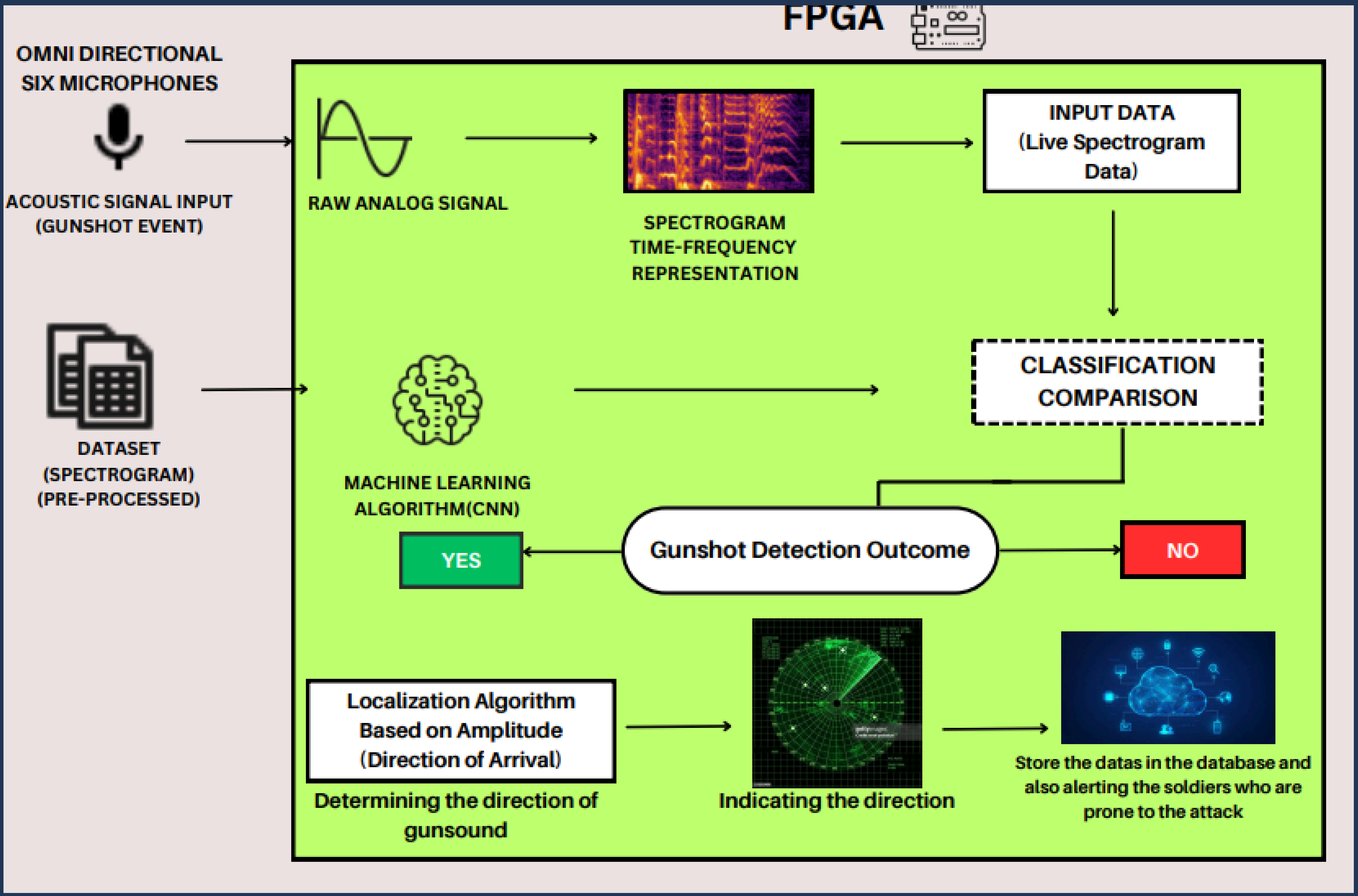
TEAM ID- EMD-06

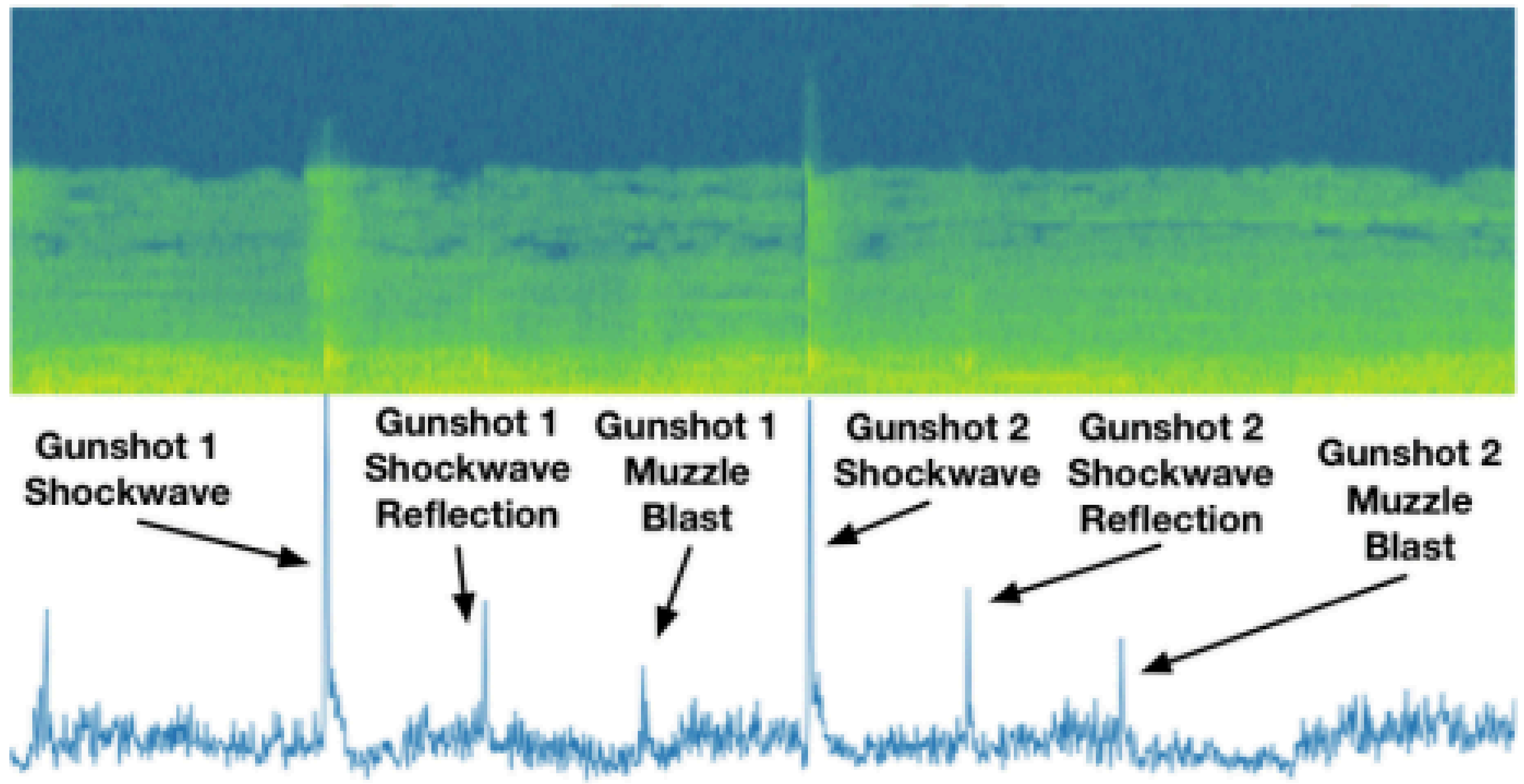
FPGA-Based CNN Accelerator for Real-Time Gunshot Detection Using Systolic Arrays and Adaptive Processing

Introduction

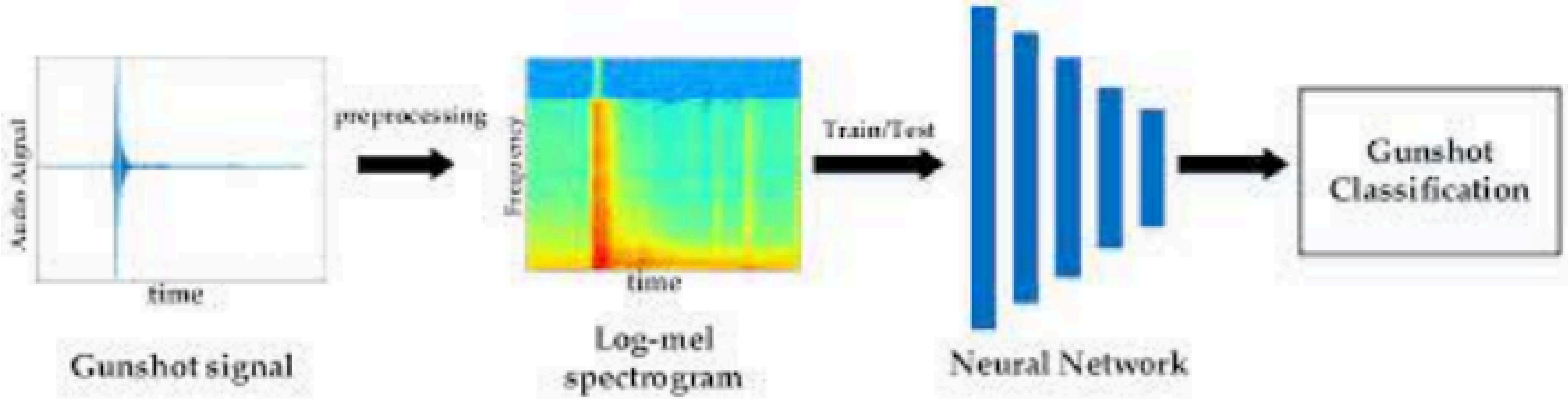
Our Project aims in developing a custom hardware accelerator that boosts the performance of CNN model to classify the gunsounds. Initially our machine receives audio signals from various microphones and converts those audio signals into spectrograms. The spectrogram is fed to a CNN model that classifies the gunshot sound, which then fed into a localization algorithm that determines the amplitude and determines the direction of arrival of Gunshot.

We impose different methods to compute the pixels from spectrogram compared with traditional methods for computation which enhances the performance of CPU+Accelerator.

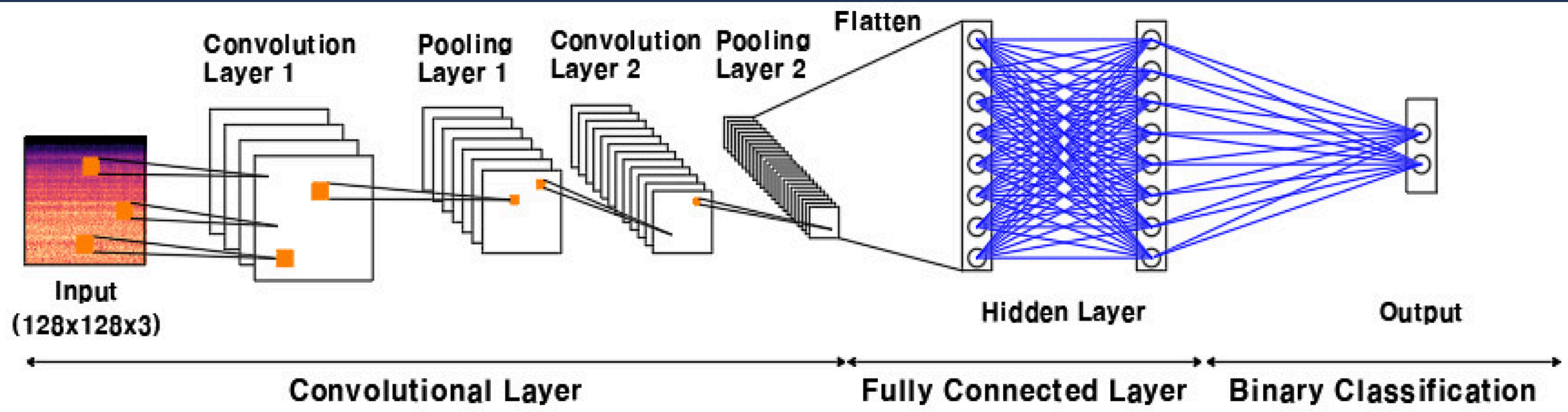

FPGA
OMNI DIRECTIONAL
SIX MICROPHONES
ACOUSTIC SIGNAL INPUT
(GUNSHOT EVENT)
RAW ANALOG SIGNAL
SPECTROGRAM
TIME-FREQUENCY
REPRESENTATION
INPUT DATA
(Live Spectrogram
Data)
DATASET
(SPECTROGRAM)
(PRE-PROCESSED)
MACHINE LEARNING
ALGORITHM(CNN)
CLASSIFICATION
COMPARISON
Gunshot Detection Outcome
YES
NO
Localization Algorithm
Based on Amplitude
(Direction of Arrival)
Determining the direction of
gunsound
Indicating the direction
Store the datas in the database and
also alerting the soldiers who are
prone to the attack



Gunshot 1 Shockwave
Gunshot 1 Shockwave Reflection
Gunshot 1 Muzzle Blast
Gunshot 2 Shockwave
Gunshot 2 Shockwave Reflection
Gunshot 2 Muzzle Blast



Audio Signal
time
Gunshot signal
preprocessing
Frequency
time
Log-mel spectrogram
Train/Test
Neural Network
Gunshot Classification


Convolution
Layer 1
Pooling
Layer 1
Convolution
Layer 2
Pooling
Layer 2
Flatten
Input
(128x128x3)
Hidden Layer
Output
Convolutional Layer
Fully Connected Layer
Binary Classification

CONVENTIONAL MATRIX MULTIPLICATION:

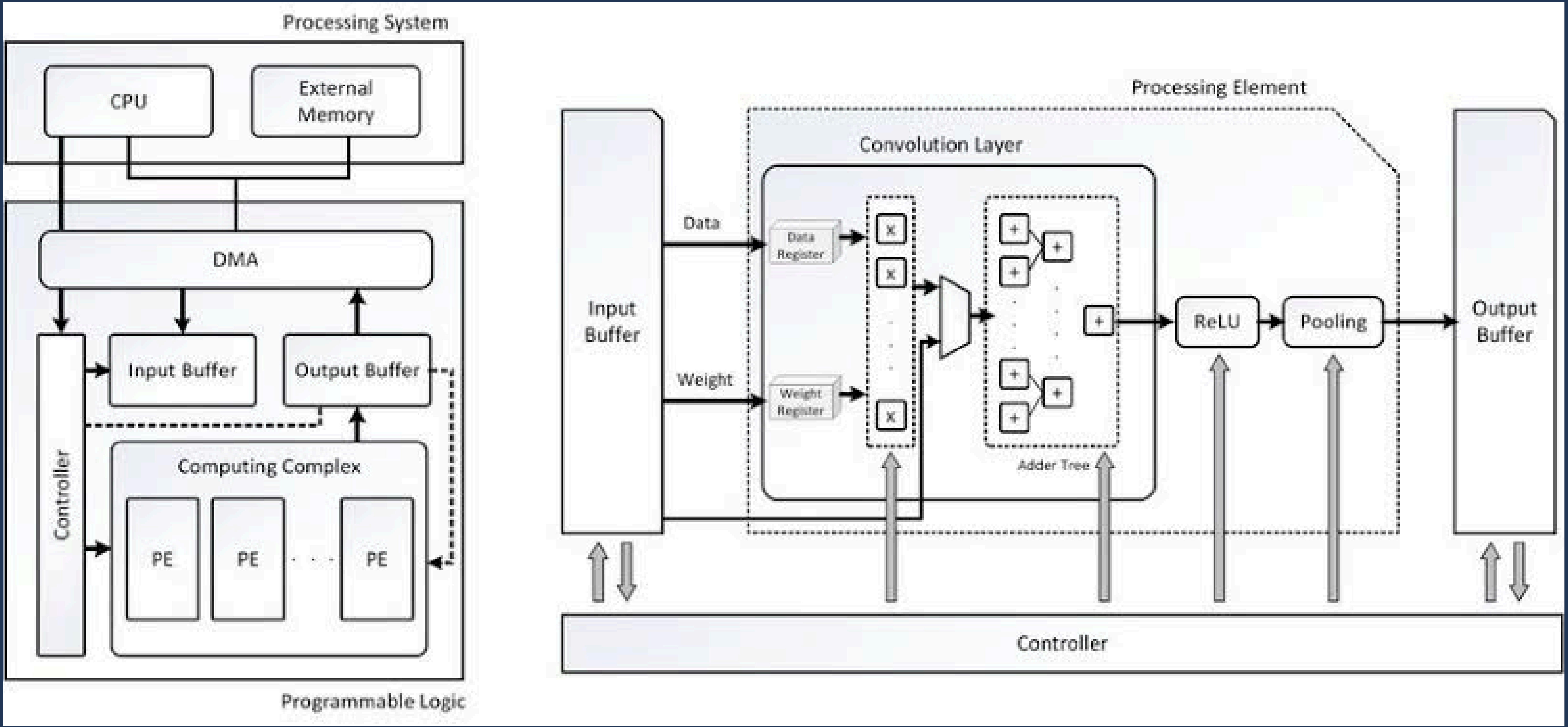
0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344

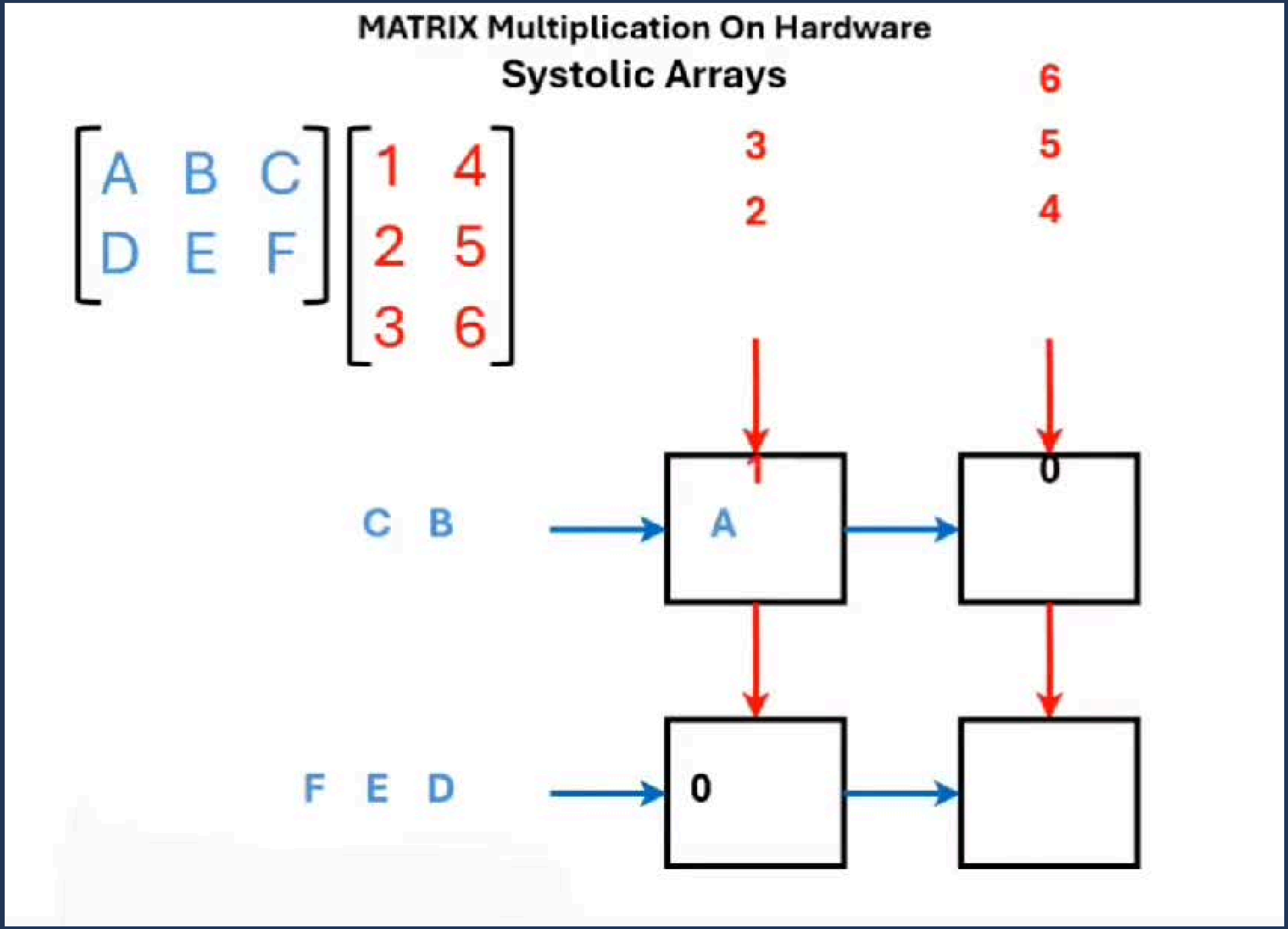
Hardware Architecture that is already in place



Proposed Idea

- SYSTOLIC ARRAY MATRIX MULTIPLICATION
- ZERO SPARSE COMPUTATION
- ADAPTIVE NOISE FILTER - KERNAL SIZE SELECTION
- PIPELINE OPTIMIZATION- MINIMIZING LATENCY ACROSS LAYERS

SYSTOLIC ARRAY MULTIPLICATION:

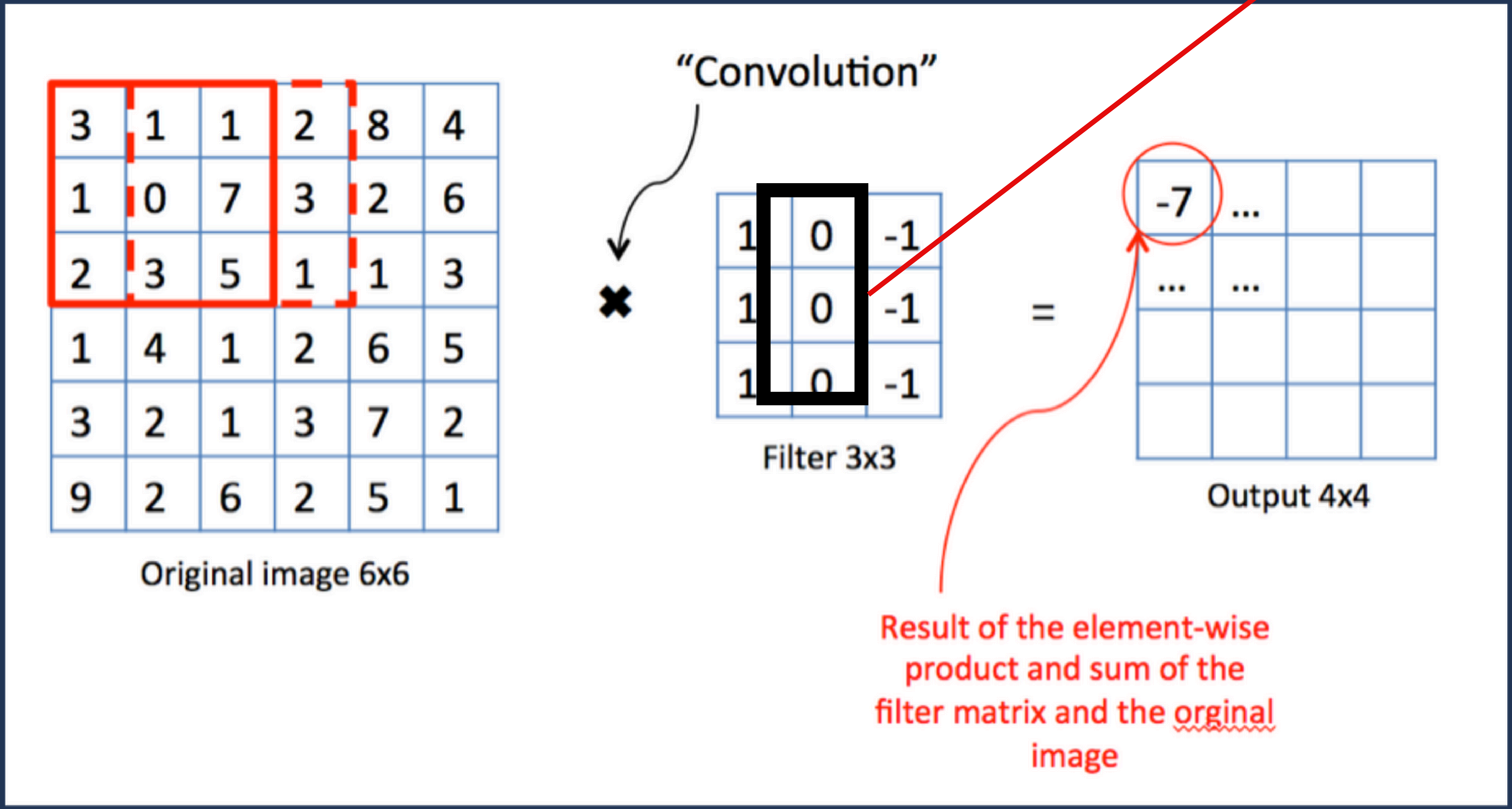


ADAPTIVE KERNAL SIZE SELECTION:

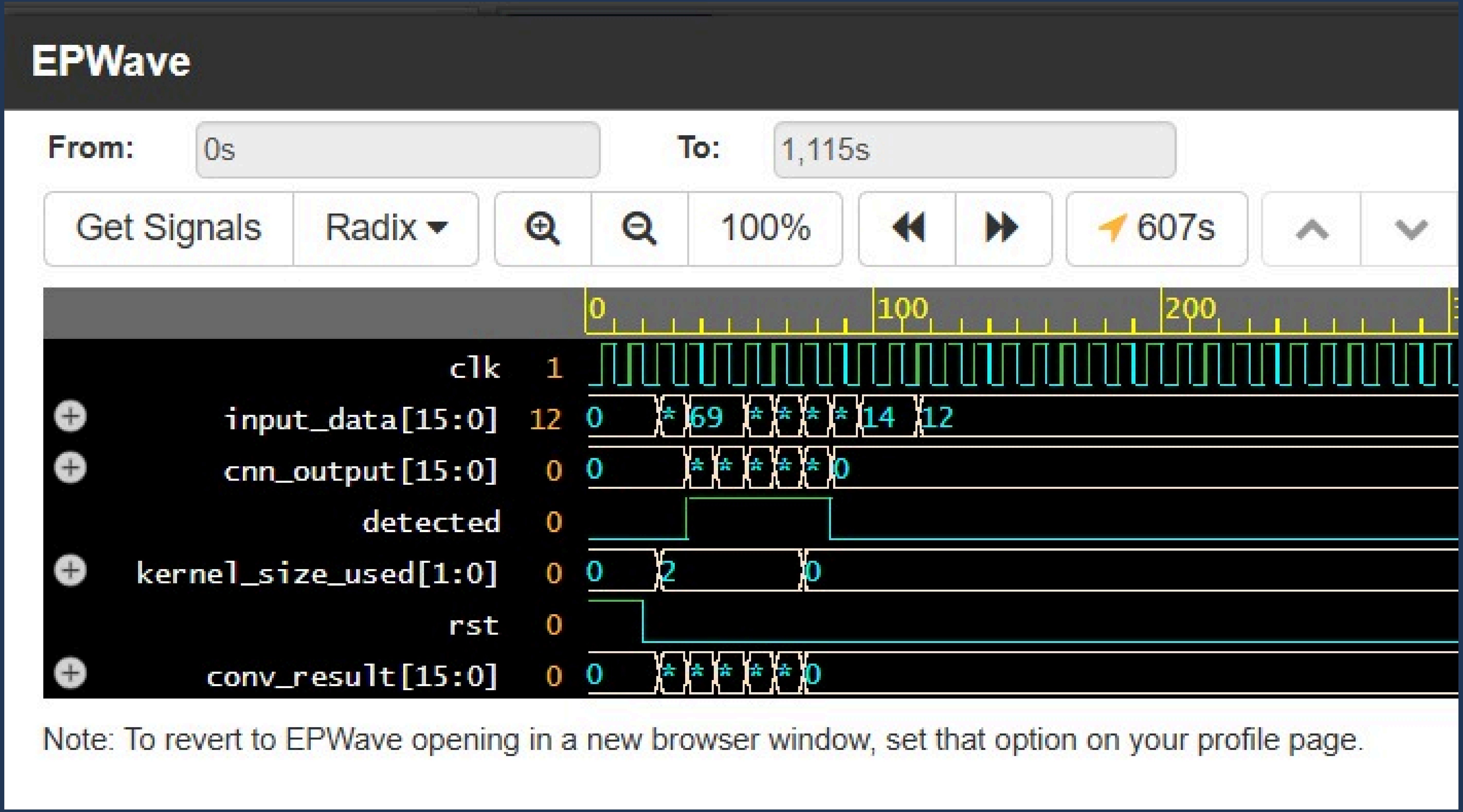
Noise Level	Kernel Size	Why This Choice	Impact
Low Noise (<50 dB)	**3×3**	Small kernels preserve fine details in clean audio	Maximizes speed & sparsity (40% ops skipped)
Medium Noise (50-150 dB)	**5×5**	Balances detail preservation and noise filtering	Captures broader patterns while suppressing interference
High Noise (>150 dB)	**7×7**	Large receptive field needed to overcome noise	Robust detection but higher power (sparsity disabled)

SPARSE COMPUTATION

for zero values in the filter, the multiplication is skipped.



PROOF OF CONCEPT:



Log Share

```
VCD info: dumpfile wave.vcd opened for output.
Time=25 | input_data=76 | cnn_output=0 | kernel_size_used=0 | detected=0
Time=35 | input_data=76 | cnn_output=0 | kernel_size_used=2 | detected=0
Time=45 | input_data=69 | cnn_output=76 | kernel_size_used=2 | detected=1
Time=55 | input_data=69 | cnn_output=69 | kernel_size_used=2 | detected=1
Time=65 | input_data=65 | cnn_output=145 | kernel_size_used=2 | detected=1
Time=75 | input_data=65 | cnn_output=148 | kernel_size_used=2 | detected=1
Time=85 | input_data=12 | cnn_output=210 | kernel_size_used=0 | detected=1
Time=95 | input_data=12 | cnn_output=0 | kernel_size_used=0 | detected=0
Time=105 | input_data=14 | cnn_output=0 | kernel_size_used=0 | detected=0
Time=115 | input_data=14 | cnn_output=0 | kernel_size_used=0 | detected=0
testbench.sv:51: $finish called at 1115 (1s)
```

```
VCD info: dumpfile comparison_wave.vcd opened for output.
|     0 |    76 |       0 |          0 |          0 |            0 |
|     1 |    76 |       0 |          0 |          0 |            0 |
===============================================================
| Cycle | Input | ACC Out | ACC Detected | NoACC Out | NoACC Detected |
---------------------------------------------------------------
|     2 |    69 |      76 |          1 |          0 |            0 |
>> Accelerator detected GUNSHOT at Cycle=2
|     3 |    69 |      69 |          1 |         76 |            1 |
>> Non-Accelerator detected GUNSHOT at Cycle=3
|     4 |    65 |     145 |          1 |         76 |            1 |
|     5 |    65 |     148 |          1 |         76 |            1 |
|     6 |    12 |     210 |          1 |        148 |            1 |
|     7 |    12 |       0 |          0 |        148 |            1 |
|     8 |    14 |       0 |          0 |        148 |            1 |
|     9 |    14 |       0 |          0 |        146 |            1 |
========================================================
| Accelerator Detection Latency = 2 Cycles (20 ns)
| Non-Accelerator Detection Latency = 3 Cycles (30 ns)
| Accelerator Speedup = 1.50 x
========================================================
testbench.sv:87: $finish called at 165 (1s)
Done
```

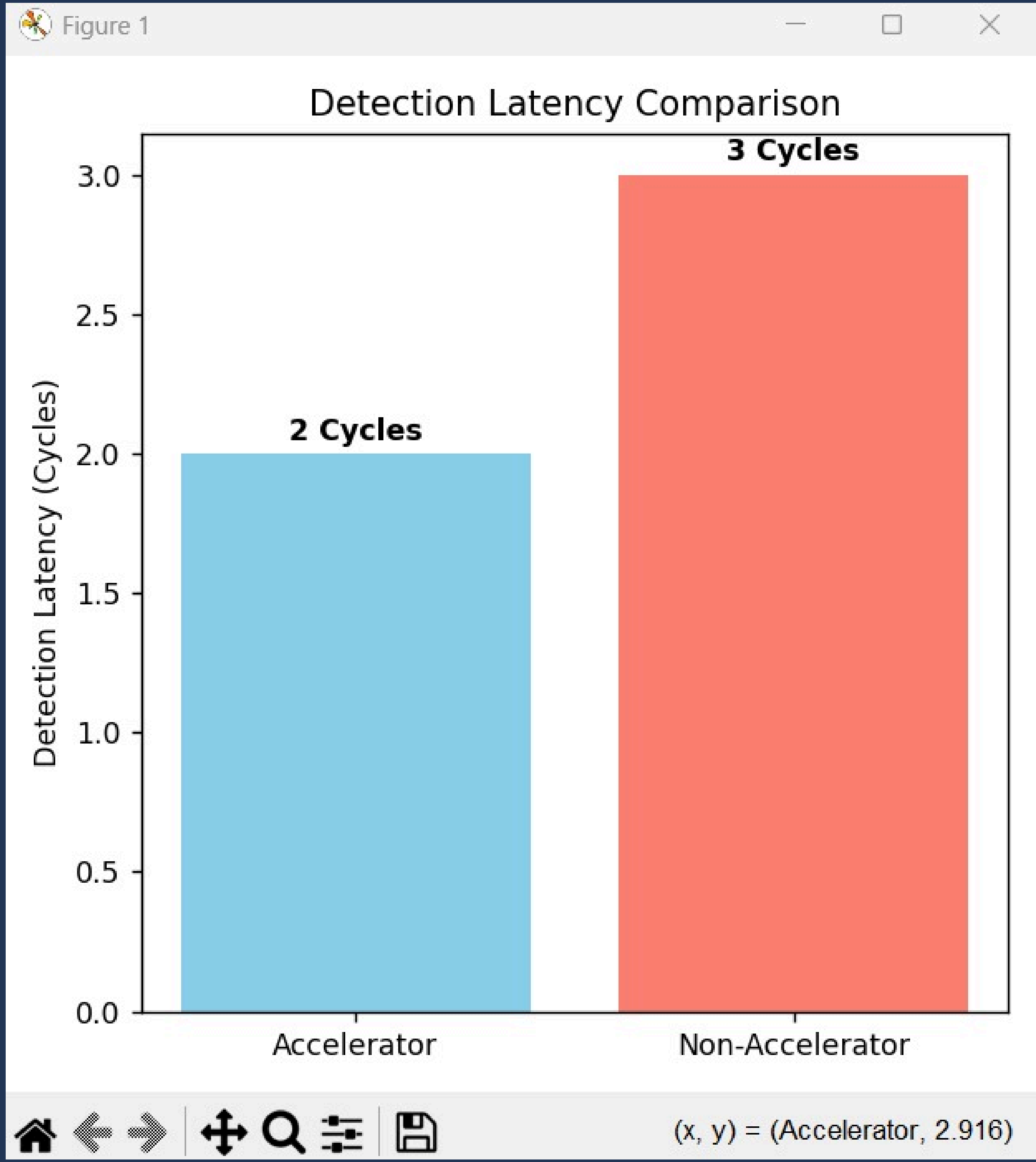


Figure 1
Detection Latency Comparison
3 Cycles
2 Cycles
Detection Latency (Cycles)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Accelerator
Non-Accelerator
(x, y) = (Accelerator, 2.916)

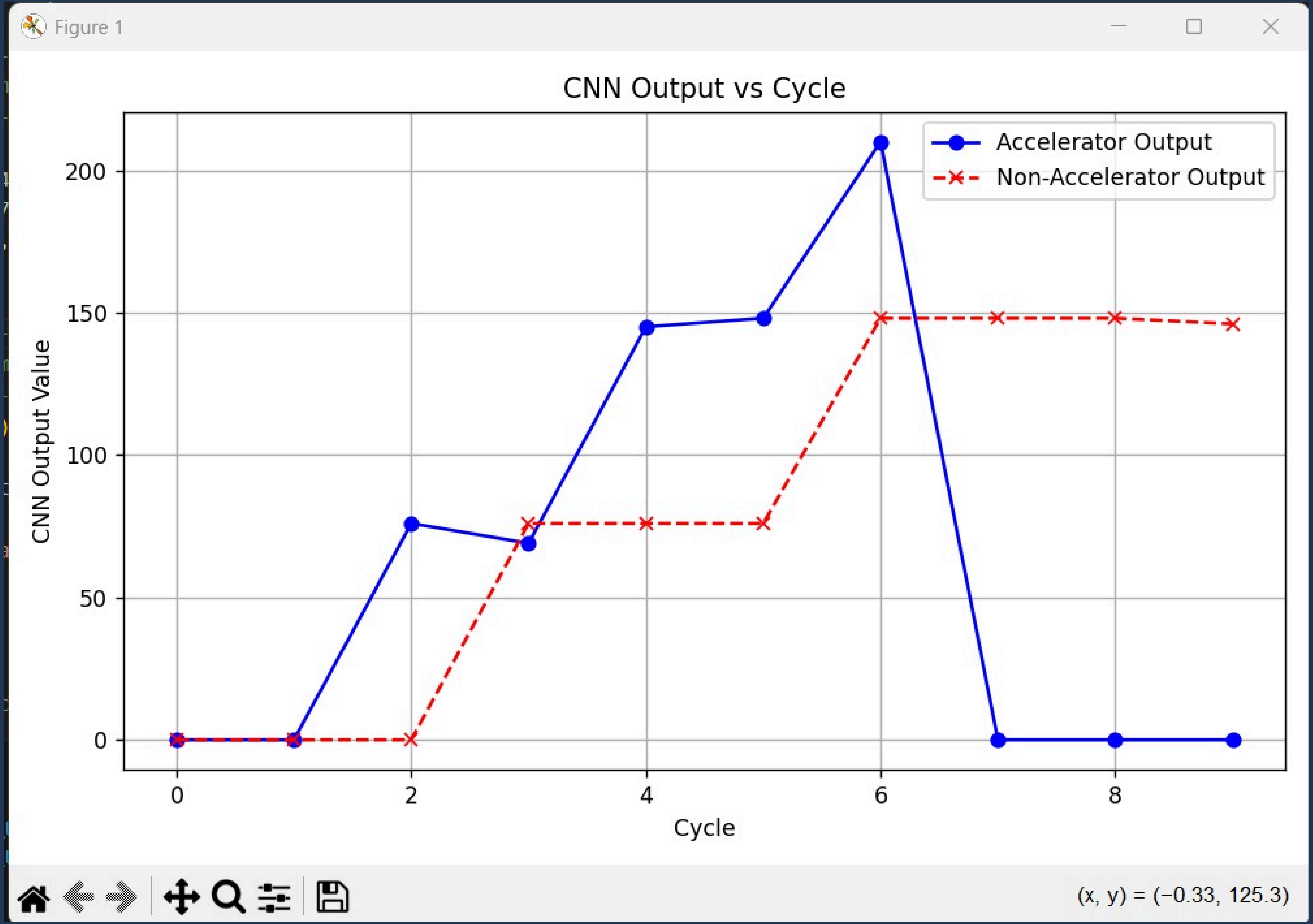


Figure 1
CNN Output vs Cycle
Accelerator Output
Non-Accelerator Output
CNN Output Value
0
50
100
150
200
0
2
4
6
8
Cycle
(x, y) = (−0.33, 125.3)

REFERENCES:

- D. Grespan et al., “Gunshot Detection using Convolutional Neural Networks and Transfer Learning,” in IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2019, doi: 10.1109/MLSP.2019.8918859.
- Y. Chen, T. Luo et al., “Sparse Convolutional Neural Networks on FPGA,” in Field-Programmable Custom Computing Machines (FCCM), 2019, doi: 10.1109/FCCM.2019.00034.
- A. Kalra and S. Deb, “High-Speed Wallace Tree Multiplier Design and Implementation on FPGA,” in International Conference on Computing, Communication and Automation (ICCCA), 2016, doi: 10.1109/CCAA.2016.7813812.
- P. Ramesh et al., “Brent-Kung Based Parallel-Prefix Adder Design for High-Performance Digital Systems,” in International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, doi: 10.1109/ICECA.2018.8474719.

Thank You