INFN
BOLOGNA

IN SUPREMÆ DIGNITATIS 1343

UNIVERSITÀ DI PISA

ALMA MATER STUDIORUM A.D. 1088

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fast accelerating approaches for Deep Learning in quantitative imaging

Mattia Ricchi, Fabrizio Alfonsi, Camilla Marella, Marco Barbieri, Alessandra Retico, Leonardo Brizi, Alessandro Gabrielli, Claudia Testa

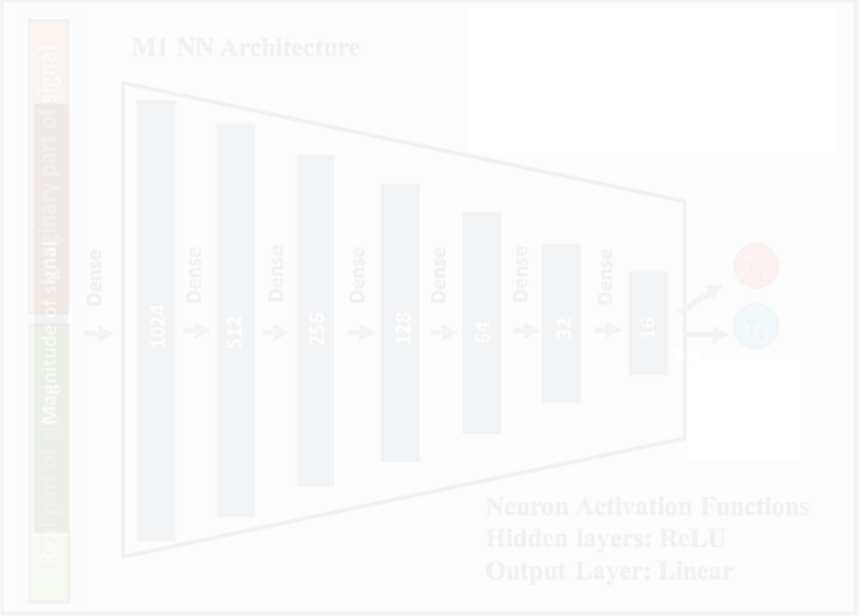
National Institute of Nuclear Physics (INFN), Division of Bologna, Italy
Department of Computer Sciences, University of Pisa, Italy
Department of Physics and Astronomy, University of Bologna, Italy

Magnetic Resonance Fingerprinting (MRF)

Magnetic Resonance Fingerprinting (MRF) is a fast **quantitative** MRI method that permits the simultaneous non-invasive quantification of multiple important properties of a material or tissue [Ma, D., Gulani, V., Seiberlich, N. et al. Magnetic resonance fingerprinting. Nature (2013)].

The MRF maps are **quantitative**: the value of the pixels is the real value of T_1 and T_2 of the tissues expressed in ms

Neural Network map reconstruction

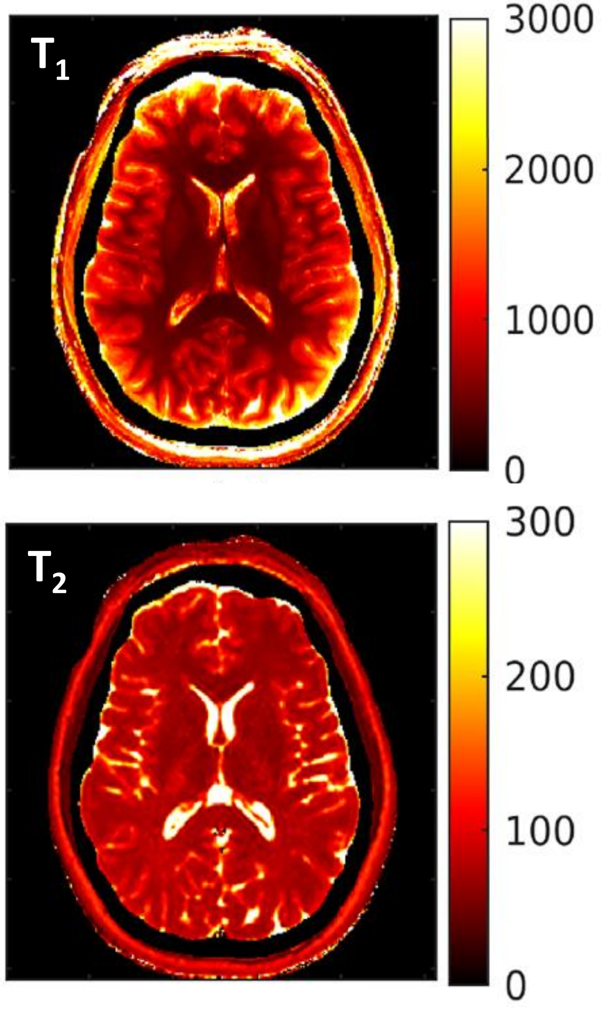


Why accelerating the NN training?

MRF sequence is not standardised, several factors may influence the results:

- Research centre / hospital
- Scanner: vendor, magnetic field strength
- Sequence and its parameters
- Parameters to be retrieved

Every time the sequence is changed, the network must be trained again

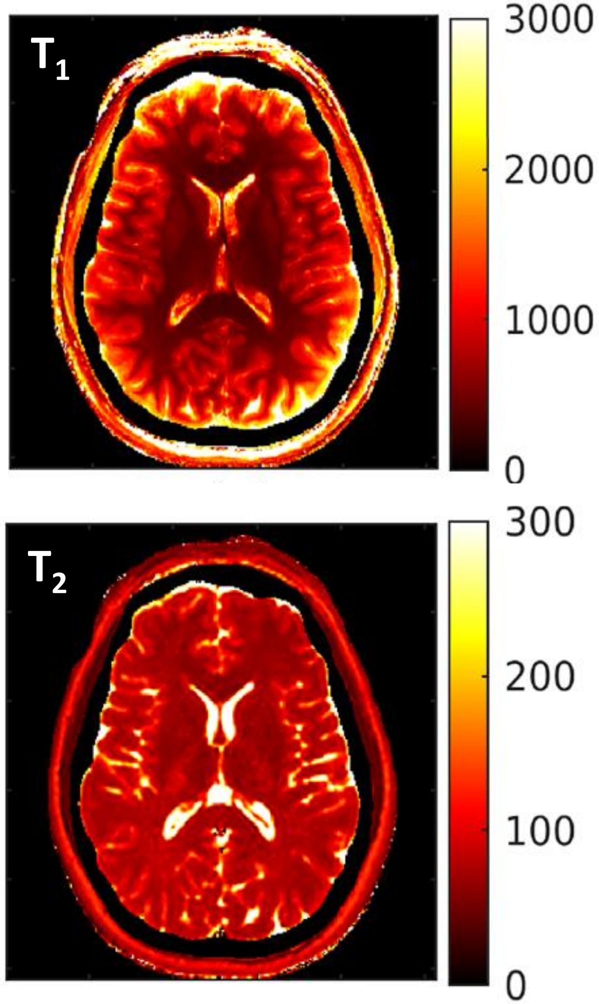


Barbieri, Marco et al. *A deep learning approach for magnetic resonance fingerprinting: Scaling capabilities and good training practices investigated by simulations.* (2021).

Magnetic Resonance Fingerprinting (MRF)

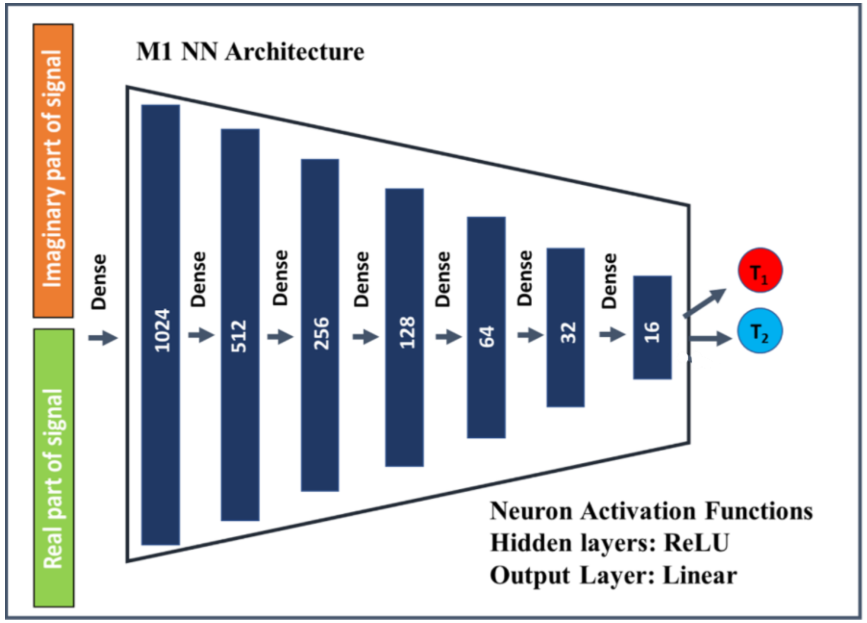
Magnetic Resonance Fingerprinting (MRF) is a fast **quantitative** MRI method that permits the simultaneous non-invasive quantification of multiple important properties of a material or tissue [Ma, D., Gulani, V., Seiberlich, N. et al. Magnetic resonance fingerprinting. Nature (2013)].

The MRF maps are **quantitative**: the value of the pixels is the real value of T_1 and T_2 of the tissues expressed in ms



Barbieri, Marco et al. *A deep learning approach for magnetic resonance fingerprinting: Scaling capabilities and good training practices investigated by simulations.* (2021).

Neural Network map reconstruction



Why accelerating the NN training?

MRF sequence is not standardised, several factors may influence the results:

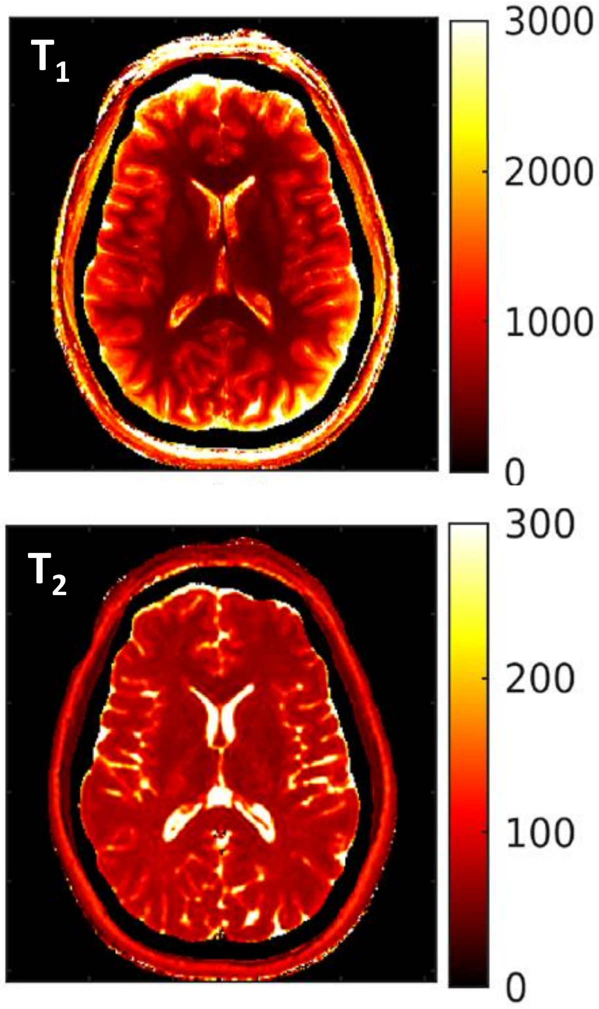
- Research centre / hospital
- Scanner: vendor, magnetic field strength
- Sequence and its parameters
- Parameters to be retrieved

Every time the sequence is changed, the network must be trained again

Magnetic Resonance Fingerprinting (MRF)

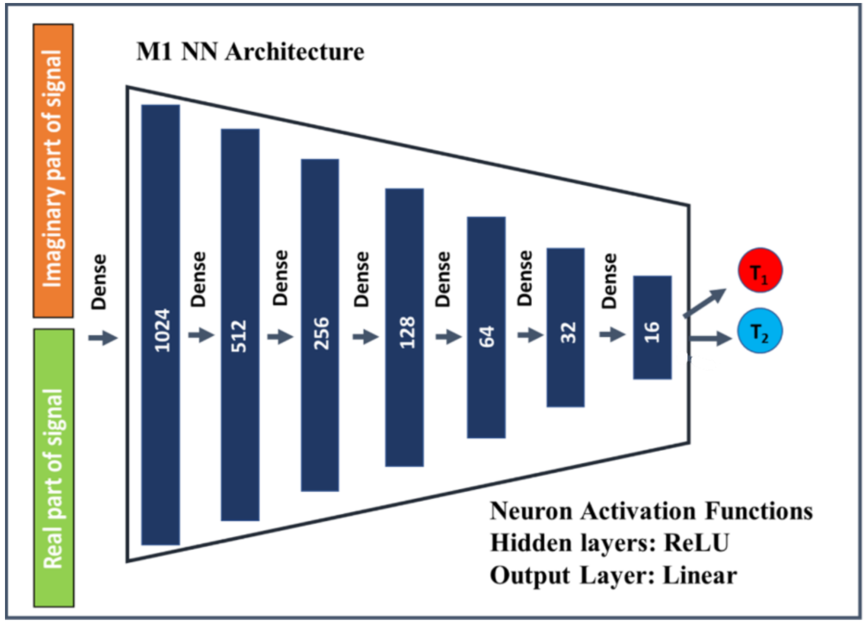
Magnetic Resonance Fingerprinting (MRF) is a fast **quantitative** MRI method that permits the simultaneous non-invasive quantification of multiple important properties of a material or tissue [Ma, D., Gulani, V., Seiberlich, N. et al. Magnetic resonance fingerprinting. Nature (2013)].

The MRF maps are **quantitative**: the value of the pixels is the real value of T_1 and T_2 of the tissues expressed in ms



Barbieri, Marco et al. *A deep learning approach for magnetic resonance fingerprinting: Scaling capabilities and good training practices investigated by simulations.* (2021).

Neural Network map reconstruction



Why accelerating the NN training?

MRF sequence is not standardised, several factors may influence the results:

- Research centre / hospital
- Scanner: vendor, magnetic field strength
- Sequence and its parameters
- Parameters to be retrieved

Every time the sequence is changed, the network must be trained again

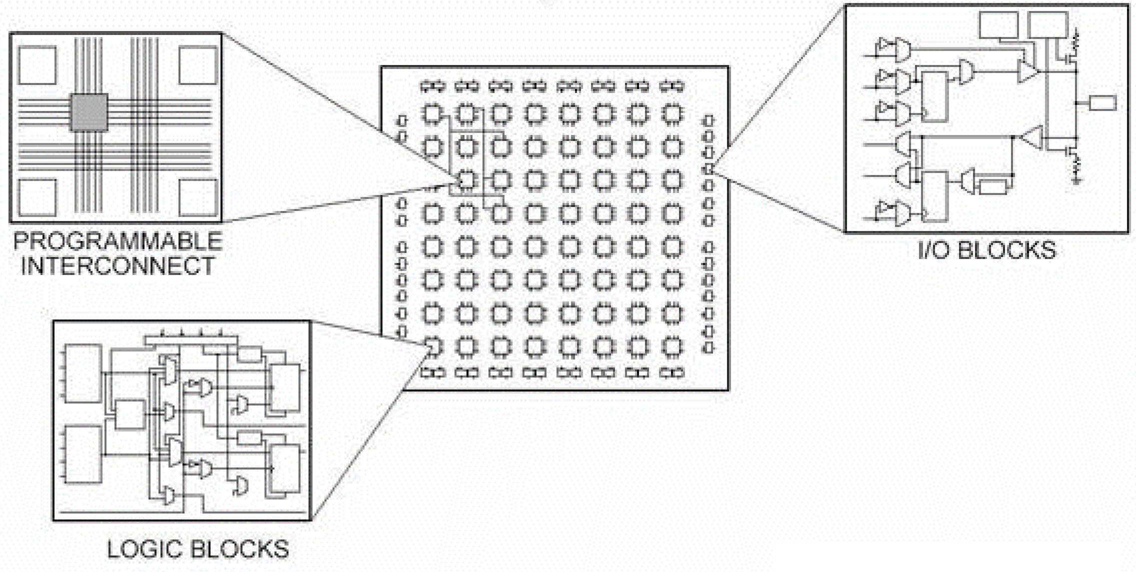
Field Programmable Gate Array (FPGA)

Configurable integrated circuit that can be **repeatedly programmed** after manufacturing.

Consist of **configurable logic blocks**, **programmable interconnects**, and flexible I/O, allowing for *custom hardware solutions*.

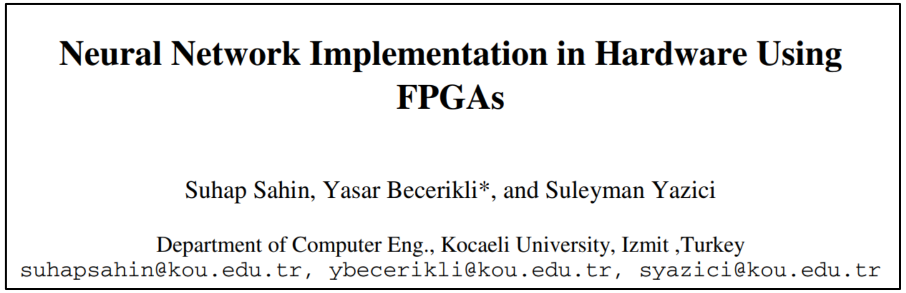
Key Advantages:

- **Parallel Processing**: FPGAs can handle multiple operations simultaneously, ideal for high-performance computing and real-time processing.
- **Low and Fixed Latency**: Direct hardware implementation reduces delay compared to software running on a CPU or GPU.
- **Custom Logic**: Designers can implement specific algorithms directly in hardware, allowing for optimized performance.



Neural Networks on FPGA

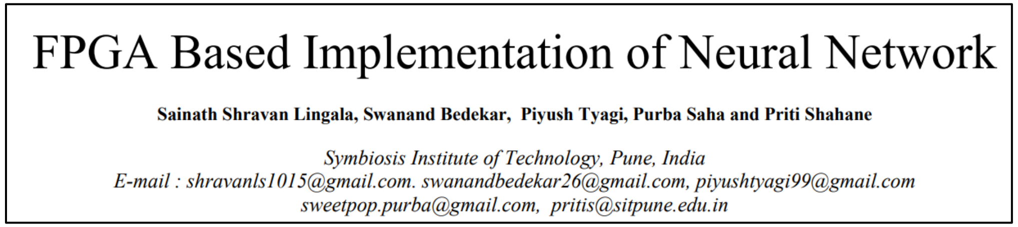
While GPUs have traditionally powered neural networks, FPGAs are gaining traction for their **customizable** hardware, better **parallel processing**, low and fixed latency and **lower power consumption** making then suitable for **real time applications**.



Neural Network Implementation in Hardware Using FPGAs

Suhap Sahin, Yasar Becerikli*, and Suleyman Yazici

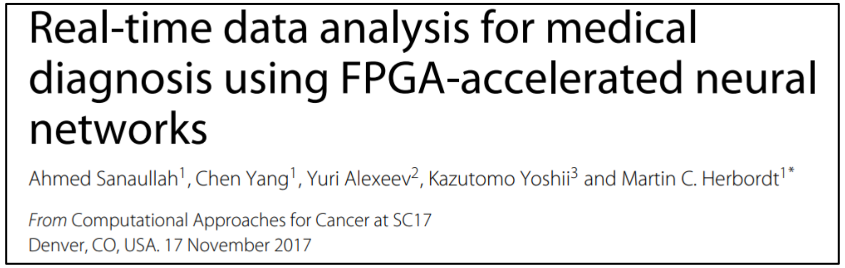
Department of Computer Eng., Kocaeli University, Izmit ,Turkey
suhapsahin@kou.edu.tr, ybecerikli@kou.edu.tr, syazici@kou.edu.tr



FPGA Based Implementation of Neural Network

Sainath Shravan Lingala, Swanand Bedekar, Piyush Tyagi, Purba Saha and Priti Shahane

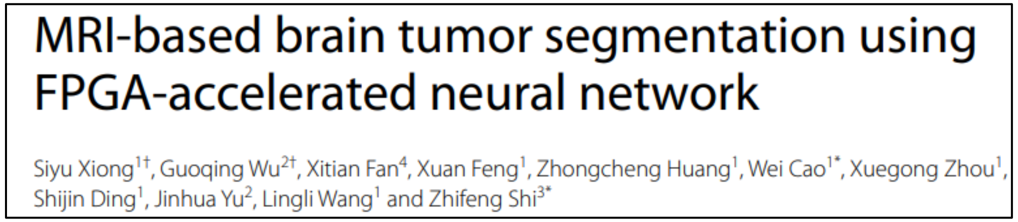
Symbiosis Institute of Technology, Pune, India
E-mail : shravanls1015@gmail.com. swanandbedekar26@gmail.com, piyushtyagi99@gmail.com
sweetpop.purba@gmail.com, pritis@sitpune.edu.in



Real-time data analysis for medical diagnosis using FPGA-accelerated neural networks

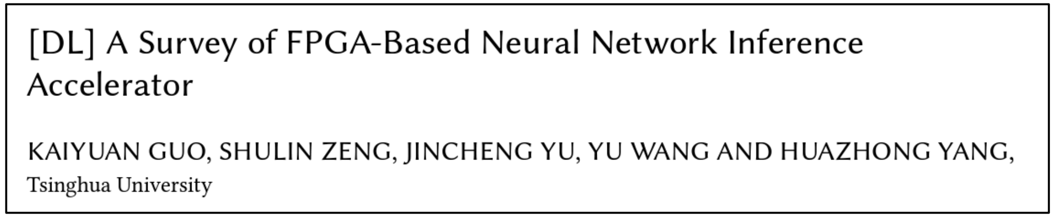
Ahmed Sanaullah[1], Chen Yang[1], Yuri Alexeev[2], Kazutomo Yoshii[3] and Martin C. Herbordt[1*]

From Computational Approaches for Cancer at SC17
Denver, CO, USA. 17 November 2017



MRI-based brain tumor segmentation using FPGA-accelerated neural network

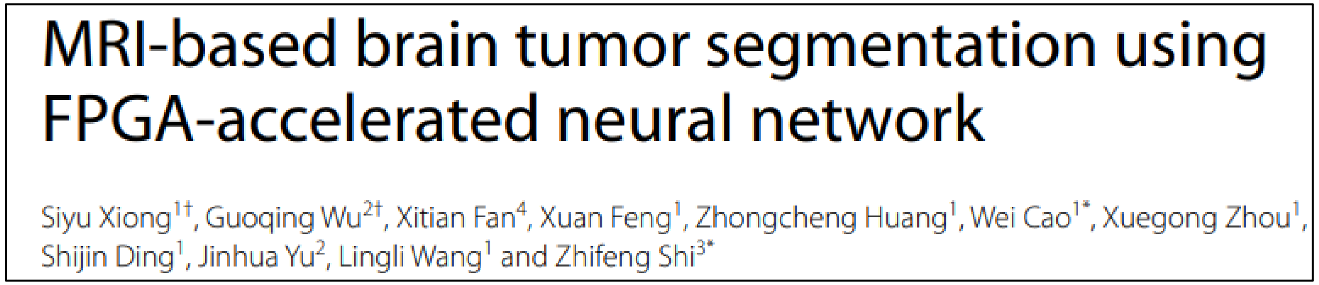
Siyu Xiong[1†], Guoqing Wu[2†], Xitian Fan[4], Xuan Feng[1], Zhongcheng Huang[1], Wei Cao[1*], Xuegong Zhou[1], Shijin Ding[1], Jinhua Yu[2], Lingli Wang[1] and Zhifeng Shi[3*]



[DL] A Survey of FPGA-Based Neural Network Inference Accelerator

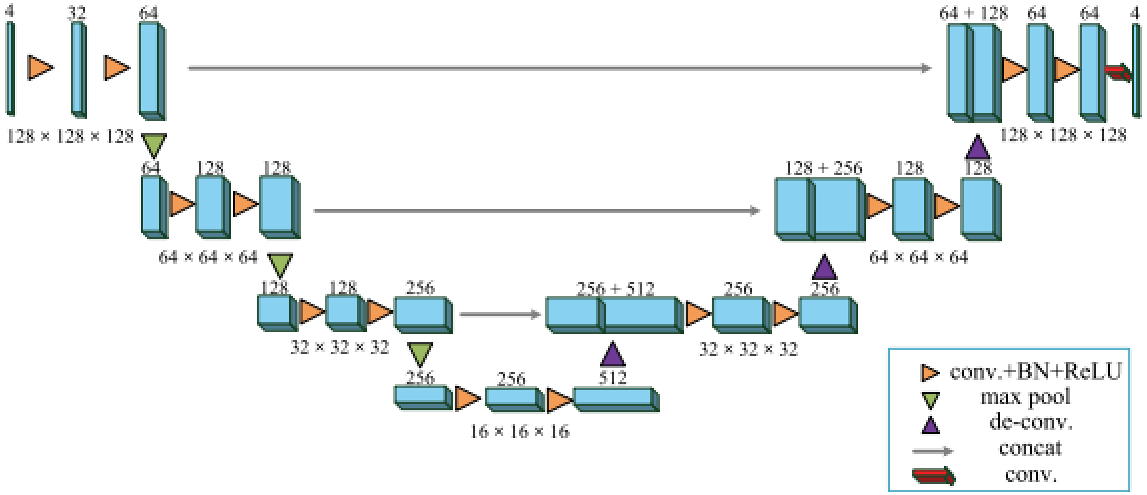
KAIYUAN GUO, SHULIN ZENG, JINCHENG YU, YU WANG AND HUAZHONG YANG,
Tsinghua University

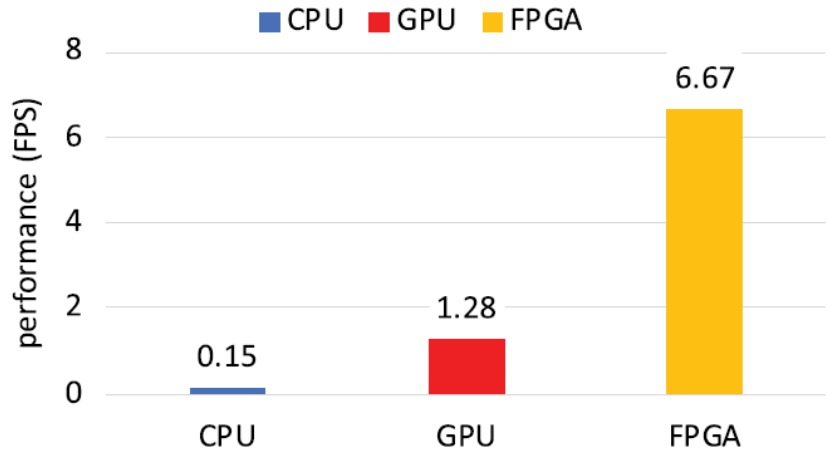
Neural Networks on FPGA

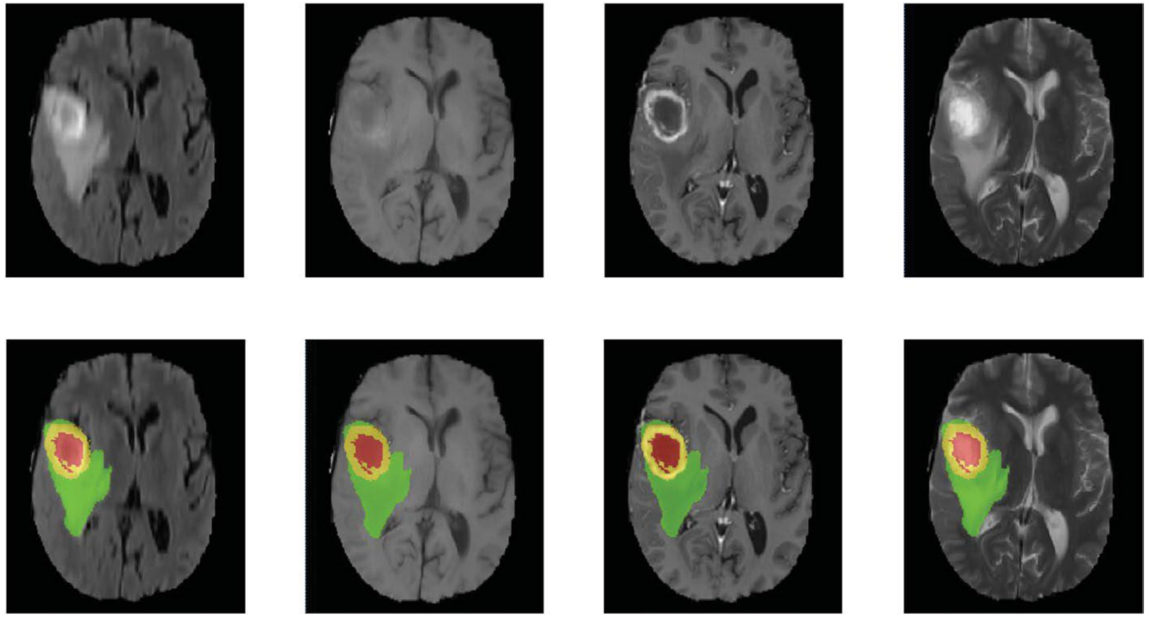


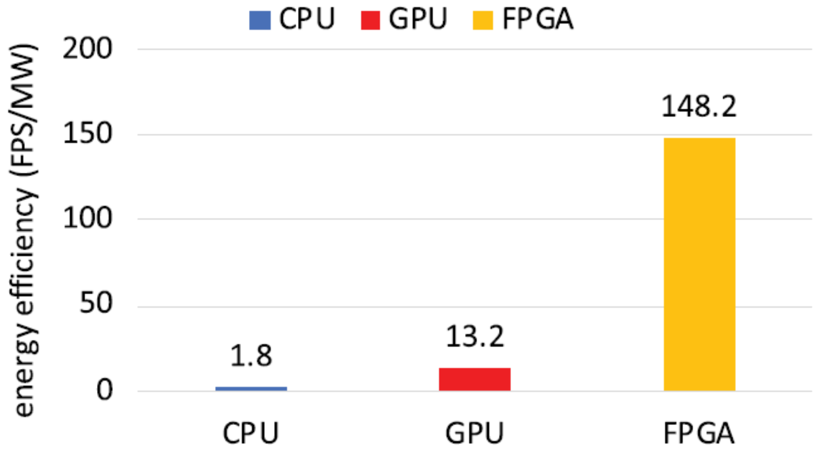
MRI-based brain tumor segmentation using FPGA-accelerated neural network

Siyu Xiong[1†], Guoqing Wu[2†], Xitian Fan[4], Xuan Feng[1], Zhongcheng Huang[1], Wei Cao[1*], Xuegong Zhou[1], Shijin Ding[1], Jinhua Yu[2], Lingli Wang[1] and Zhifeng Shi[3*]



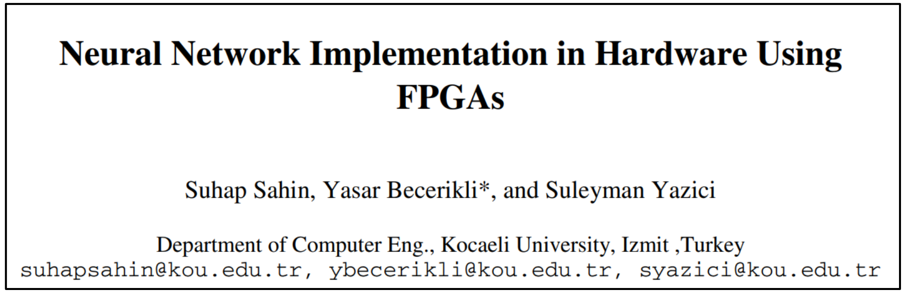






Neural Networks on FPGA

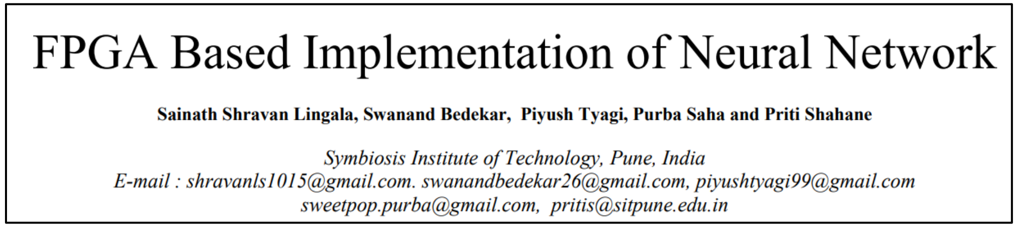
While GPUs have traditionally powered neural networks, FPGAs are gaining traction for their **customizable** hardware, better **parallel processing**, low and fixed latency and **lower power consumption** making then suitable for **real time applications**.



Neural Network Implementation in Hardware Using FPGAs

Suhap Sahin, Yasar Becerikli*, and Suleyman Yazici

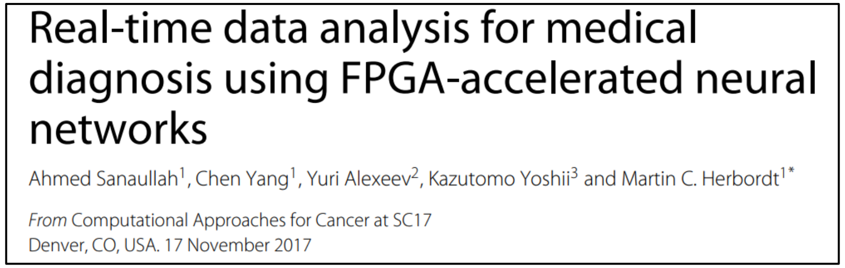
Department of Computer Eng., Kocaeli University, Izmit ,Turkey
suhapsahin@kou.edu.tr, ybecerikli@kou.edu.tr, syazici@kou.edu.tr



FPGA Based Implementation of Neural Network

Sainath Shravan Lingala, Swanand Bedekar, Piyush Tyagi, Purba Saha and Priti Shahane

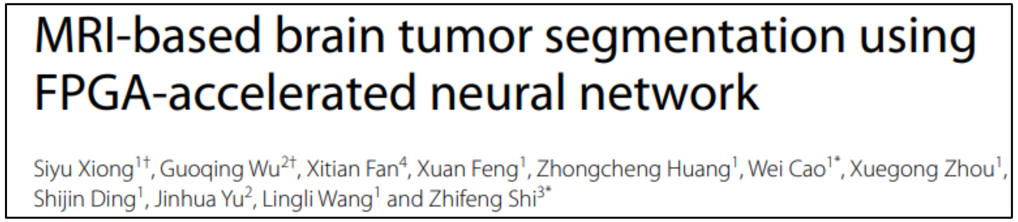
Symbiosis Institute of Technology, Pune, India
E-mail : shravanls1015@gmail.com. swanandbedekar26@gmail.com, piyushtyagi99@gmail.com
sweetpop.purba@gmail.com, pritis@sitpune.edu.in



Real-time data analysis for medical diagnosis using FPGA-accelerated neural networks

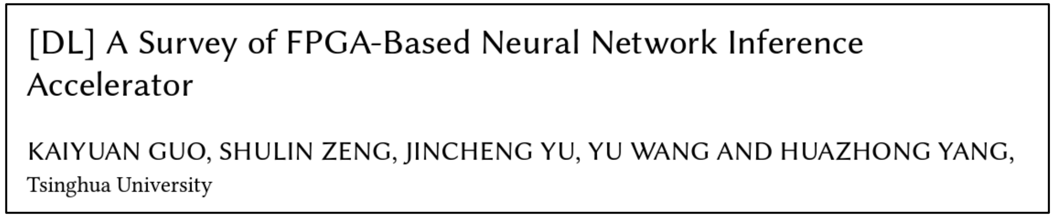
Ahmed Sanaullah[1], Chen Yang[1], Yuri Alexeev[2], Kazutomo Yoshii[3] and Martin C. Herbordt[1*]

From Computational Approaches for Cancer at SC17
Denver, CO, USA. 17 November 2017



MRI-based brain tumor segmentation using FPGA-accelerated neural network

Siyu Xiong[1†], Guoqing Wu[2†], Xitian Fan[4], Xuan Feng[1], Zhongcheng Huang[1], Wei Cao[1*], Xuegong Zhou[1], Shijin Ding[1], Jinhua Yu[2], Lingli Wang[1] and Zhifeng Shi[3*]



[DL] A Survey of FPGA-Based Neural Network Inference Accelerator

KAIYUAN GUO, SHULIN ZENG, JINCHENG YU, YU WANG AND HUAZHONG YANG, Tsinghua University

All of these works are just about **inference** on FPGA. What about **training**?

Why training a Neural Network on FPGA?

Performance and speed

Efficient parallel processing & custom architecture optimization

Energy efficiency

Lower power consumption compared to CPU and GPU

Flexibility

Reconfigurability for different neural network architectures and algorithms

Latency

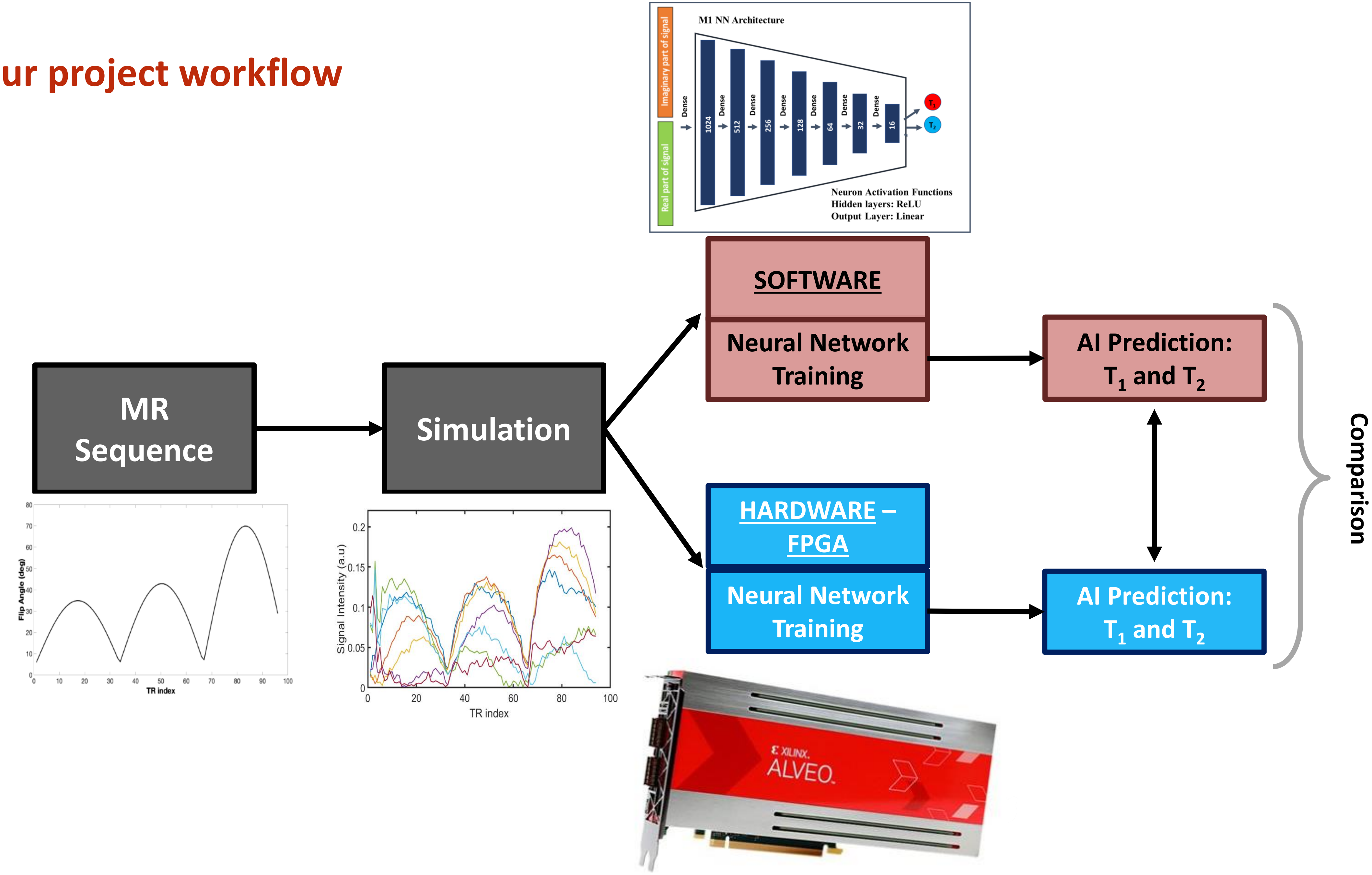
Lower and fixed latency due to parallel processing and closer memory access

High throughput

The parallel processing capabilities of FPGAs can handle high data throughput

These characteristics make FPGAs particularly well-suited for **real time applications** of Neural Networks that require **extensive and repeated training**, also with very large datasets.

Our project workflow



MR Sequence

Simulation

SOFTWARE

Neural Network Training

AI Prediction: T_1 and T_2

HARDWARE – FPGA

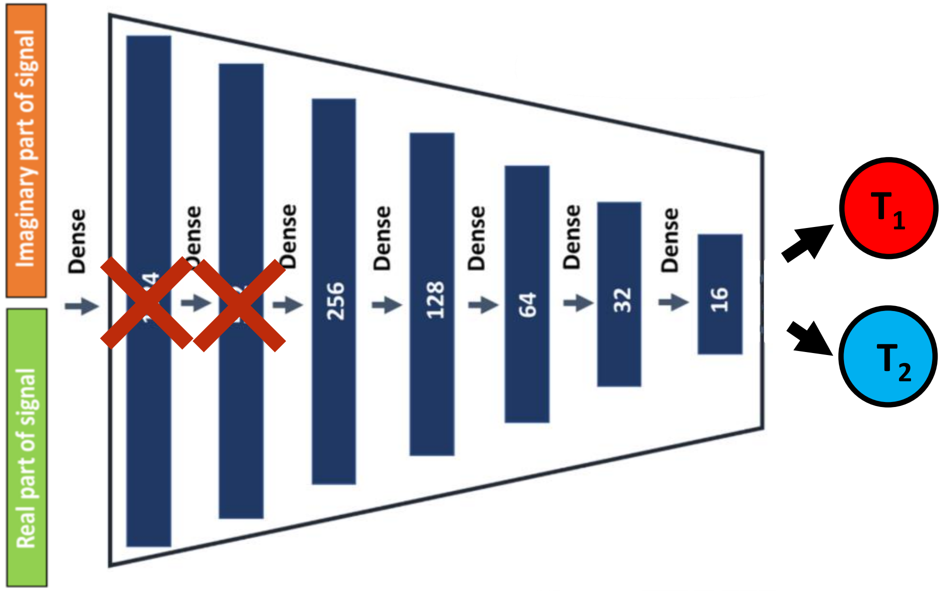
Neural Network Training

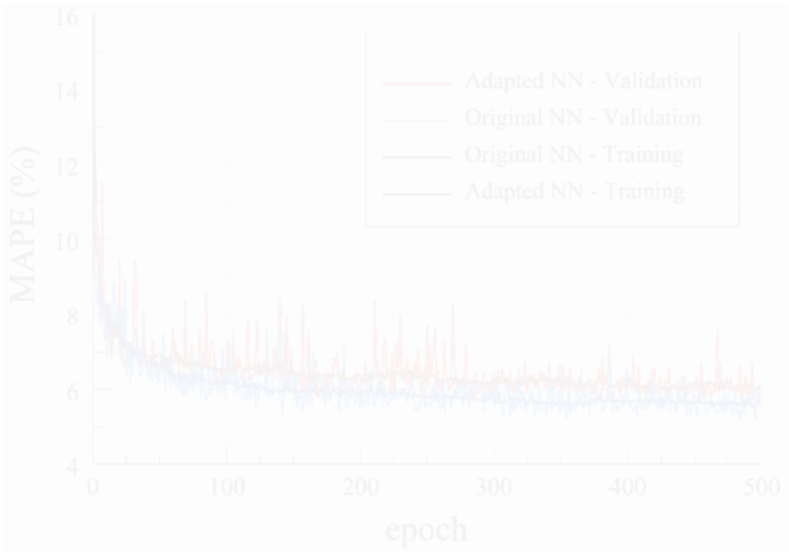
AI Prediction: T_1 and T_2

Comparison

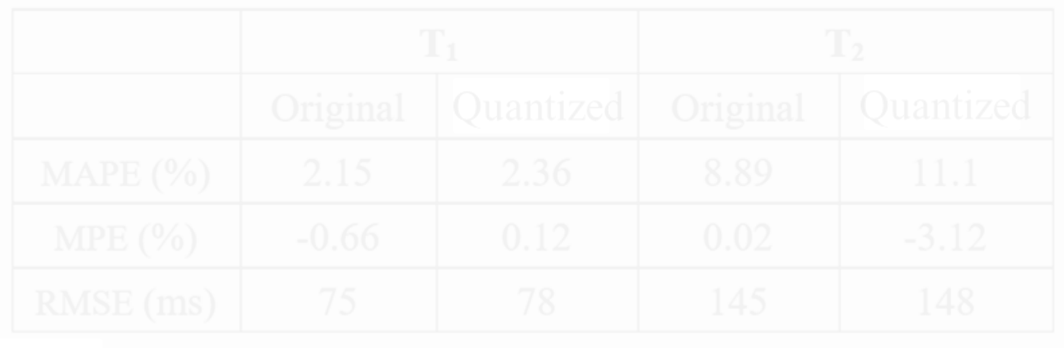
Neural Network Quantization Aware Training

1. Reduce network dimensions to meet available FPGA resources





2. Perform Quantization Aware Training so that the network operates with integers only

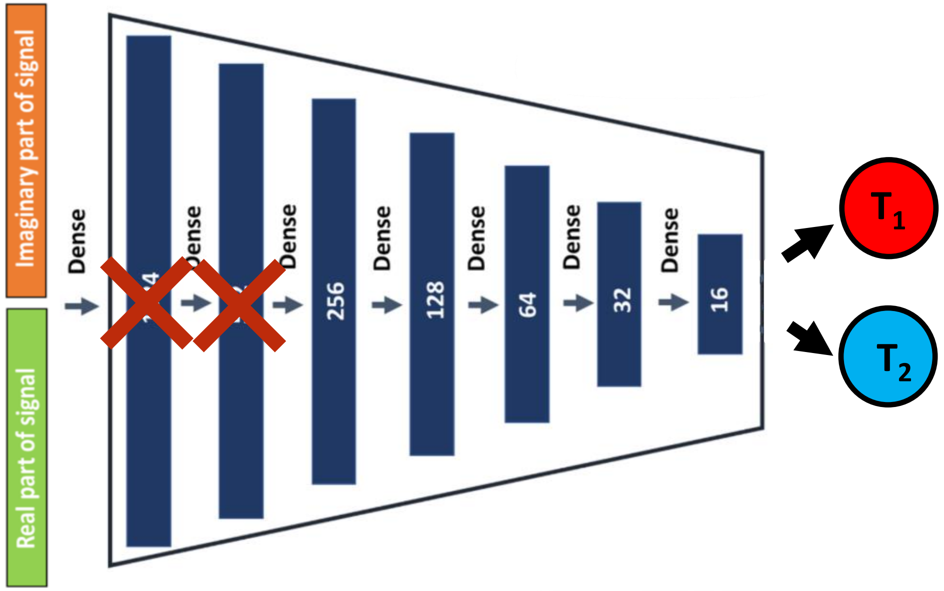


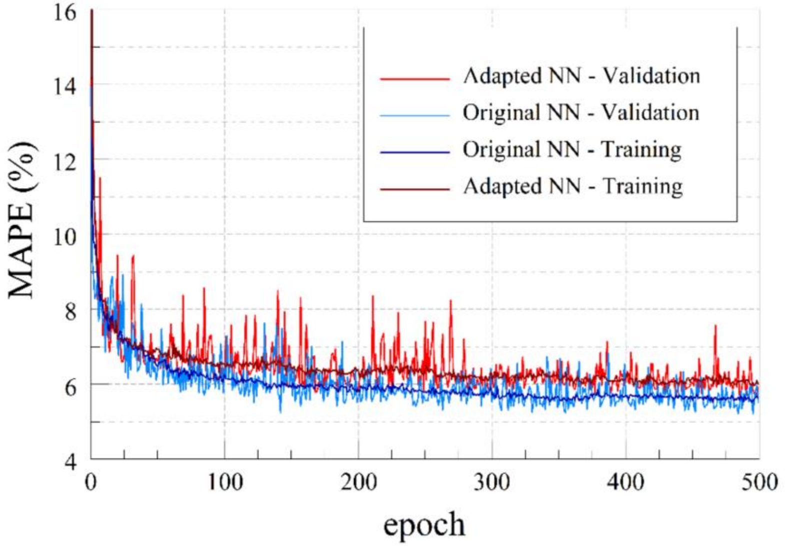
	T_1		T_2	
	Original	Quantized	Original	Quantized
MAPE (%)	2.15	2.36	8.89	11.1
MPE (%)	-0.66	0.12	0.02	-3.12
RMSE (ms)	75	78	145	148

The quantized model uses lower precision without affecting the neural network performance.

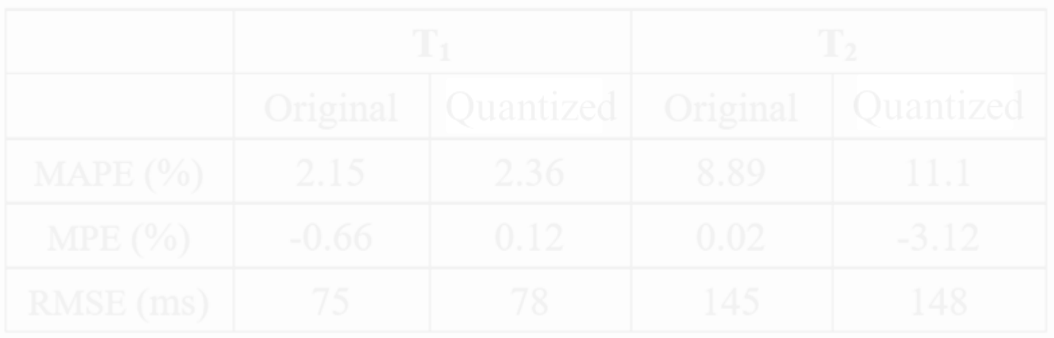
Neural Network Quantization Aware Training

1. Reduce network dimensions to meet available FPGA resources





2. Perform Quantization Aware Training so that the network operates with integers only

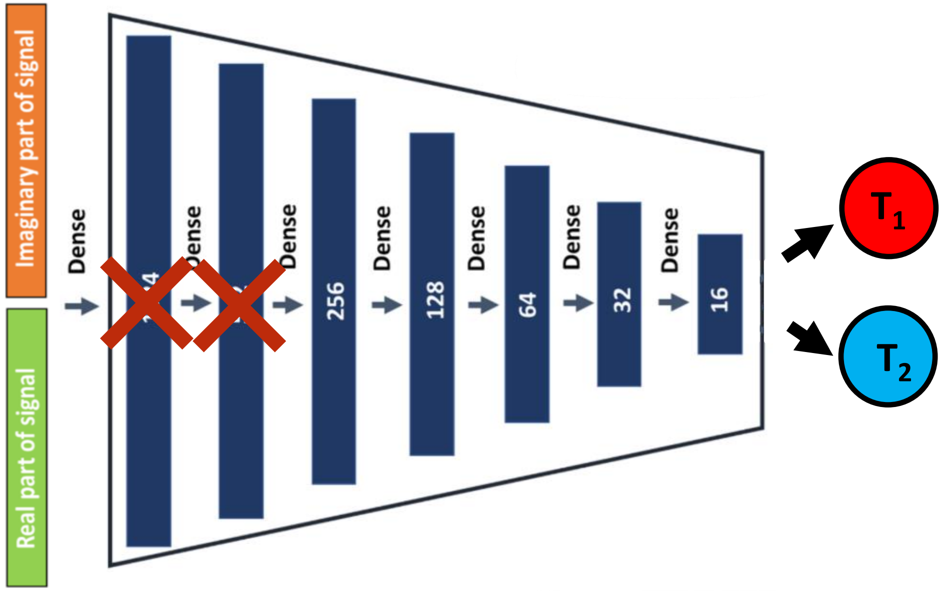


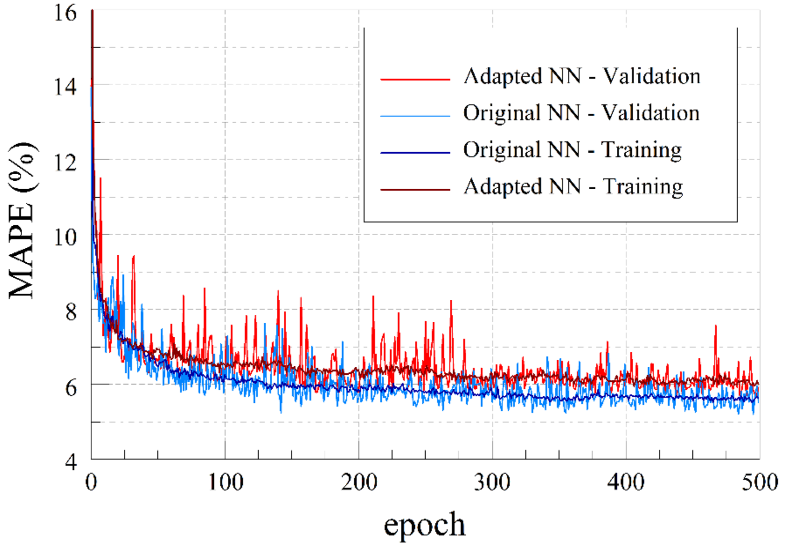
	T_1		T_2	
	Original	Quantized	Original	Quantized
MAPE (%)	2.15	2.36	8.89	11.1
MPE (%)	-0.66	0.12	0.02	-3.12
RMSE (ms)	75	78	145	148

The quantized model uses lower precision without affecting the neural network performance.

Neural Network Quantization Aware Training

1. Reduce network dimensions to meet available FPGA resources





2. Perform Quantization Aware Training so that the network operates with integers only

	T_1		T_2	
	Original	Quantized	Original	Quantized
MAPE (%)	2.15	2.36	8.89	11.1
MPE (%)	-0.66	0.12	0.02	-3.12
RMSE (ms)	75	78	145	148

The quantized model uses lower precision without affecting the neural network performance.

Neural Network VHDL implementation

Low-level approach, writing every firmware component in VHDL

Node working principle

Implemented in VHDL the function $\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ that represents the behaviour of a single node.

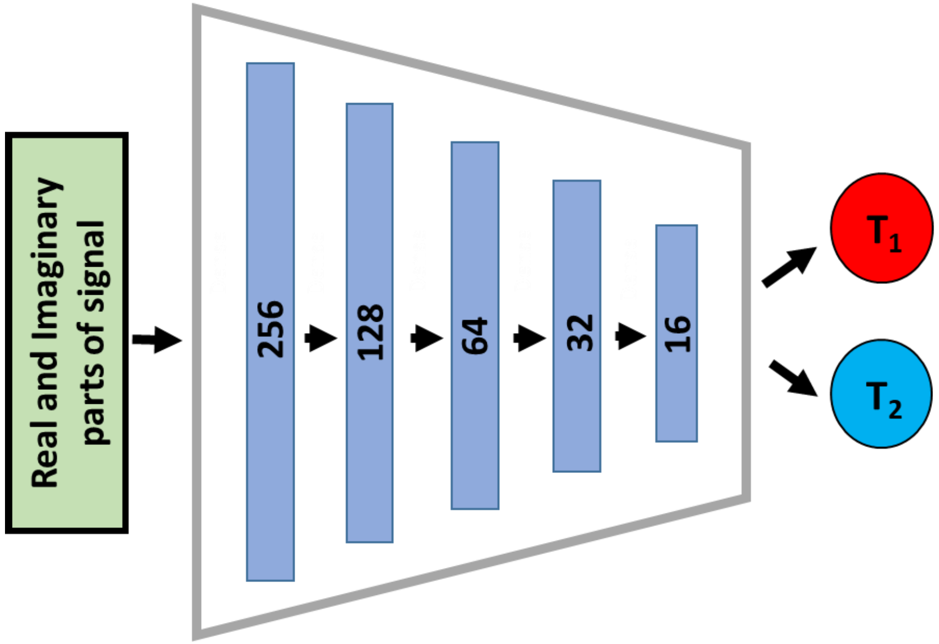
Backpropagation process

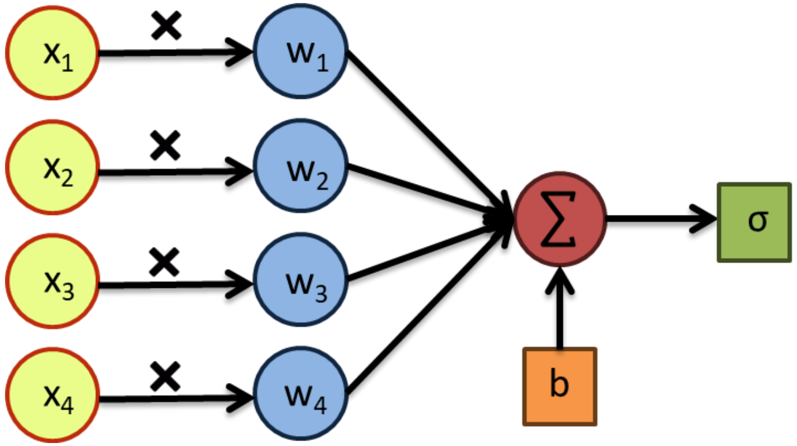
Implemented in VHDL the following formulae for backpropagation

$$\delta^L = \nabla\mathcal{L} \odot \delta'(z^L) \qquad \frac{\partial\mathcal{L}}{\partial b^l} = \delta^l$$

$$\delta^l = \left(\left(\boldsymbol{w}^{l+1}\right)^T \delta^{l+1}\right) \odot \delta'(z^l) \qquad \frac{\partial\mathcal{L}}{\partial \boldsymbol{w}^l} = y^{l-1}\delta^l$$

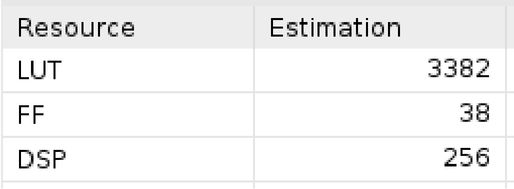
These **two blocks** will cover all the NN operations by **serial iterations**. The whole NN operations cannot be implemented in FPGA concurrently because of resource limitation.





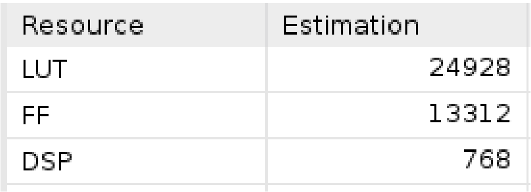
FPGA Neural Network synthesis and resource estimation

Node resources



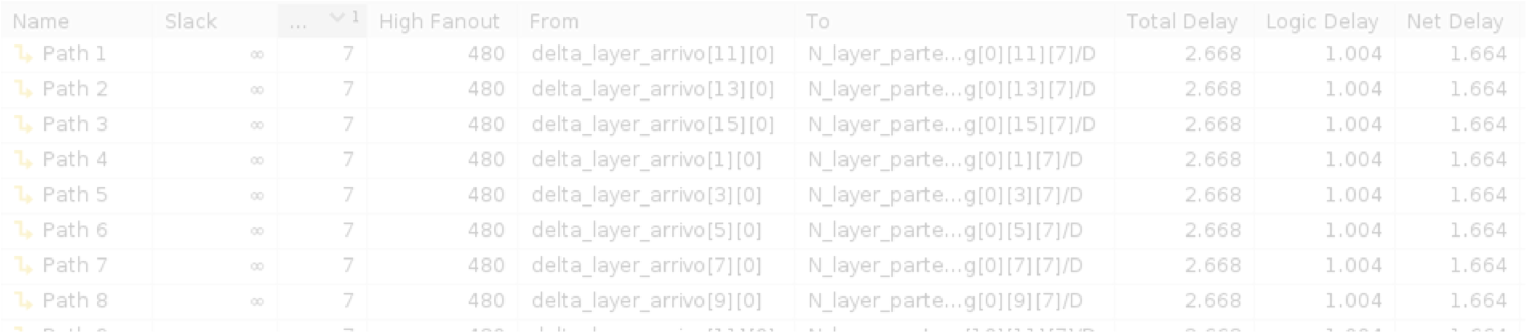
Resource	Estimation
LUT	3382
FF	38
DSP	256

Backpropagation resources



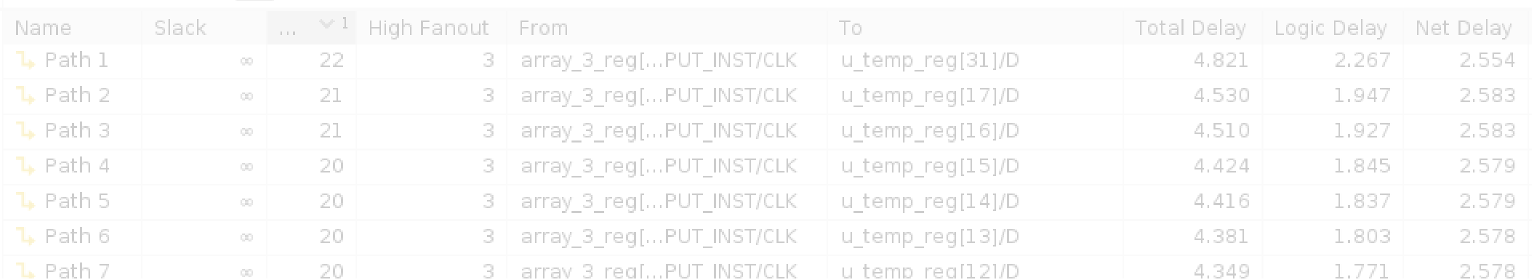
Resource	Estimation
LUT	24928
FF	13312
DSP	768

Backpropagation synthesis



Name	Slack	... ^1	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	7	480	delta_layer_arrivo[11][0]	N_layer_parte...g[0][11][7]/D	2.668	1.004	1.664
Path 2	∞	7	480	delta_layer_arrivo[13][0]	N_layer_parte...g[0][13][7]/D	2.668	1.004	1.664
Path 3	∞	7	480	delta_layer_arrivo[15][0]	N_layer_parte...g[0][15][7]/D	2.668	1.004	1.664
Path 4	∞	7	480	delta_layer_arrivo[1][0]	N_layer_parte...g[0][1][7]/D	2.668	1.004	1.664
Path 5	∞	7	480	delta_layer_arrivo[3][0]	N_layer_parte...g[0][3][7]/D	2.668	1.004	1.664
Path 6	∞	7	480	delta_layer_arrivo[5][0]	N_layer_parte...g[0][5][7]/D	2.668	1.004	1.664
Path 7	∞	7	480	delta_layer_arrivo[7][0]	N_layer_parte...g[0][7][7]/D	2.668	1.004	1.664
Path 8	∞	7	480	delta_layer_arrivo[9][0]	N_layer_parte...g[0][9][7]/D	2.668	1.004	1.664

Node synthesis



Name	Slack	... ^1	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	22	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[31]/D	4.821	2.267	2.554
Path 2	∞	21	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[17]/D	4.530	1.947	2.583
Path 3	∞	21	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[16]/D	4.510	1.927	2.583
Path 4	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[15]/D	4.424	1.845	2.579
Path 5	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[14]/D	4.416	1.837	2.579
Path 6	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[13]/D	4.381	1.803	2.578
Path 7	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[12]/D	4.349	1.771	2.578

Accelerator FPGA based card: **Alveo U250**
(1.7M LUTs, 3.4M Flip-Flops, 12k DSPs, 2.6k BRAMs)

Resources used

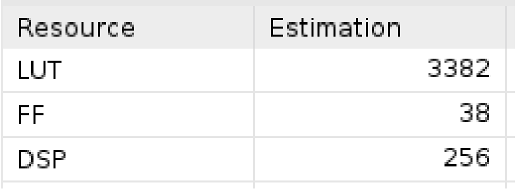
Percentage of the available resources that are used, including the internal memory.

LUTs	8%
FFs	4%
DSPs	40%

Based on synthesis results, **clock frequency of 200 MHz** is feasible, but aiming at 250 MHz.

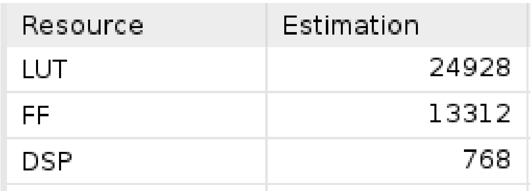
FPGA Neural Network synthesis and resource estimation

Node resources



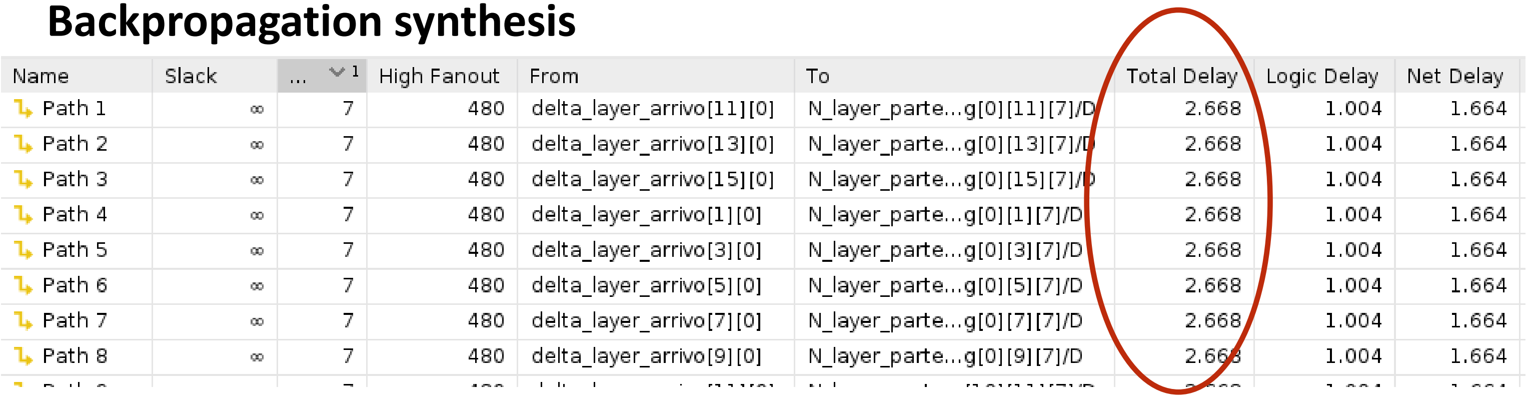
Resource	Estimation
LUT	3382
FF	38
DSP	256

Backpropagation resources



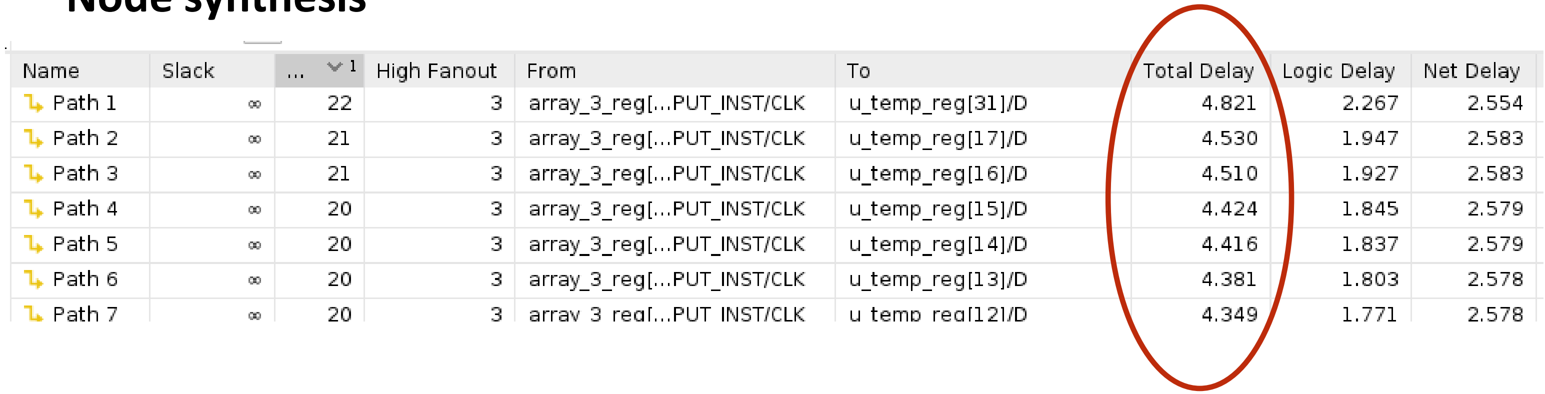
Resource	Estimation
LUT	24928
FF	13312
DSP	768

Backpropagation synthesis



Name	Slack	... 1	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	7	480	delta_layer_arrivo[11][0]	N_layer_parte...g[0][11][7]/D	2.668	1.004	1.664
Path 2	∞	7	480	delta_layer_arrivo[13][0]	N_layer_parte...g[0][13][7]/D	2.668	1.004	1.664
Path 3	∞	7	480	delta_layer_arrivo[15][0]	N_layer_parte...g[0][15][7]/D	2.668	1.004	1.664
Path 4	∞	7	480	delta_layer_arrivo[1][0]	N_layer_parte...g[0][1][7]/D	2.668	1.004	1.664
Path 5	∞	7	480	delta_layer_arrivo[3][0]	N_layer_parte...g[0][3][7]/D	2.668	1.004	1.664
Path 6	∞	7	480	delta_layer_arrivo[5][0]	N_layer_parte...g[0][5][7]/D	2.668	1.004	1.664
Path 7	∞	7	480	delta_layer_arrivo[7][0]	N_layer_parte...g[0][7][7]/D	2.668	1.004	1.664
Path 8	∞	7	480	delta_layer_arrivo[9][0]	N_layer_parte...g[0][9][7]/D	2.668	1.004	1.664

Node synthesis



Name	Slack	... 1	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	22	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[31]/D	4.821	2.267	2.554
Path 2	∞	21	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[17]/D	4.530	1.947	2.583
Path 3	∞	21	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[16]/D	4.510	1.927	2.583
Path 4	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[15]/D	4.424	1.845	2.579
Path 5	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[14]/D	4.416	1.837	2.579
Path 6	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[13]/D	4.381	1.803	2.578
Path 7	∞	20	3	array_3_reg[...PUT_INST/CLK	u_temp_reg[12]/D	4.349	1.771	2.578

Accelerator FPGA based card: **Alveo U250**
(1.7M LUTs, 3.4M Flip-Flops, 12k DSPs, 2.6k BRAMs)

Resources used

Percentage of the available resources that are used, including the internal memory.

LUTs	8%
FFs	4%
DSPs	40%

Based on synthesis results, **clock frequency of 200 MHz** is feasible, but aiming at 250 MHz.

FPGA preliminary simulation

16 nodes implemented both in python and on the FPGA.

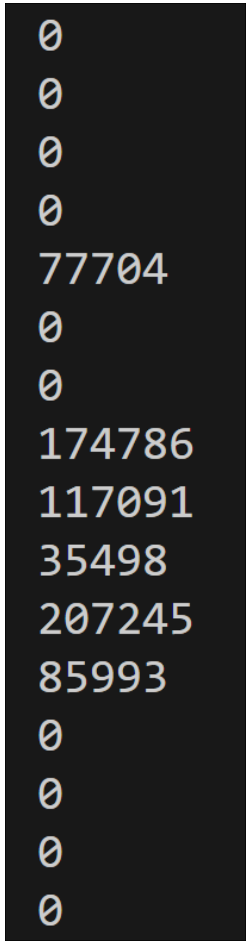
Same input and weights were given to the nodes in python and on the FPGA.

Exactly the same results were obtained.

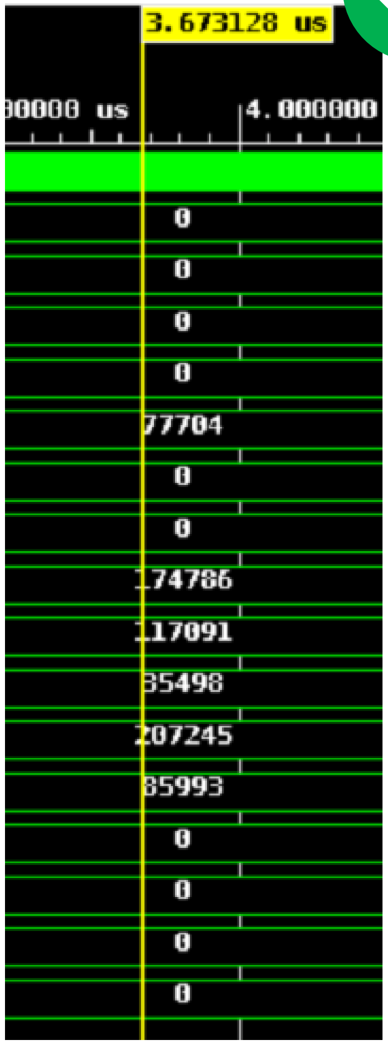
Mathematical syntax has been correctly implemented on the FPGA

Node simulation

Python run



```
0
0
0
0
77704
0
0
174786
117091
35498
207245
85993
0
0
0
0
```



```
0
0
0
0
77704
0
0
174786
117091
35498
207245
85993
0
0
0
0
```

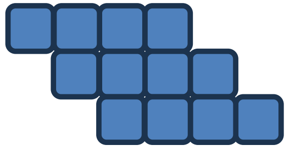
Time estimates

200 MHz targeted **clock frequency**, feasible based on synthesis results

The network is trained with 250 Millions simulated data:

- 250M forward passes, each pass taking 56 clock cycles

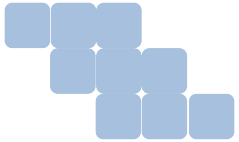
16 nodes
16 nodes
…



🟦 1 clock cycle

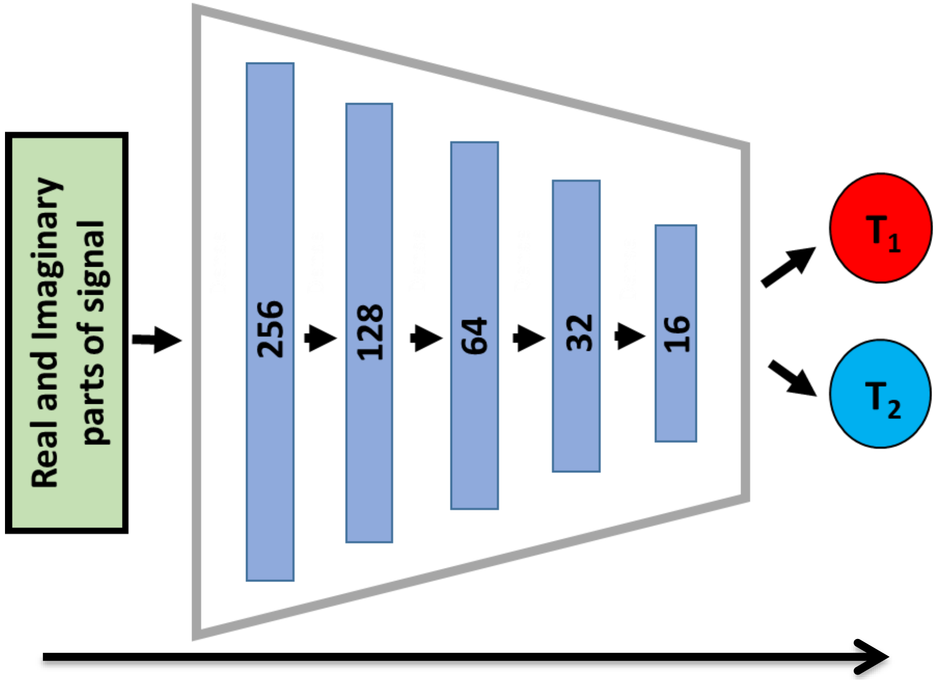
- 250M backpropagations, each pass taking 104 clock cycles

32-16 backprop
32-16 backprop
…

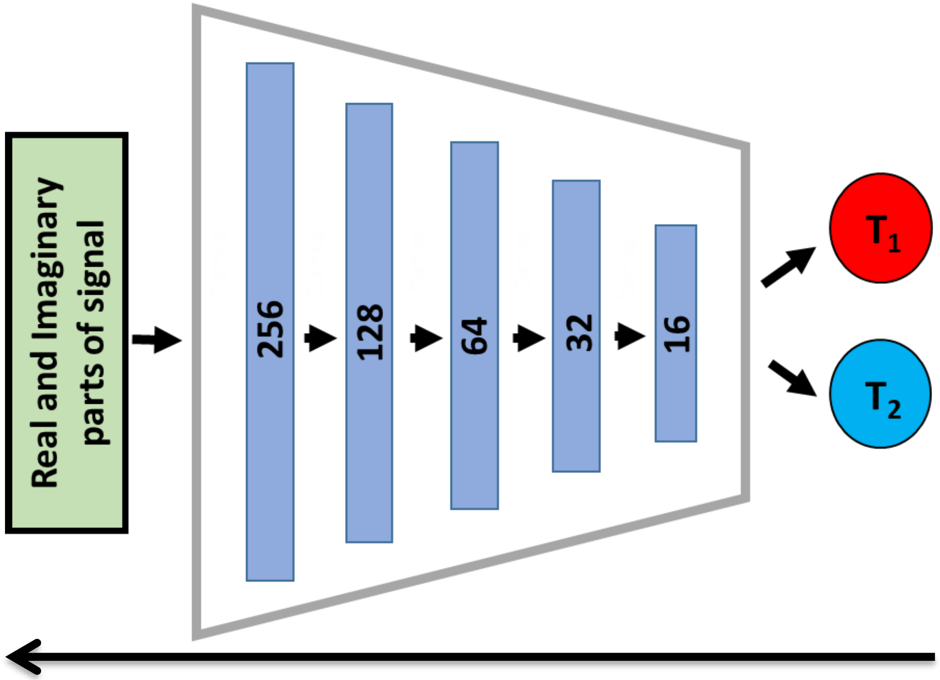


🟦 1 clock cycle

Total latency $\left(\underbrace{5}_{\text{ns}} \cdot \underbrace{(250'000'000 \cdot (56 + 104))}_{\text{Clock cycles}}\right) = 200\ \text{s} + \text{PCIe}$



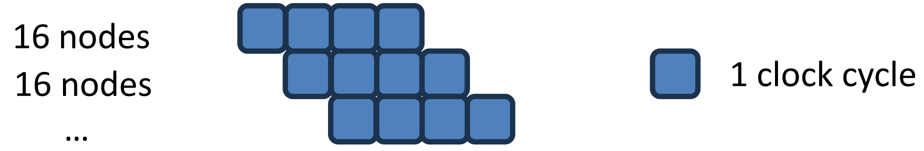
Time estimates



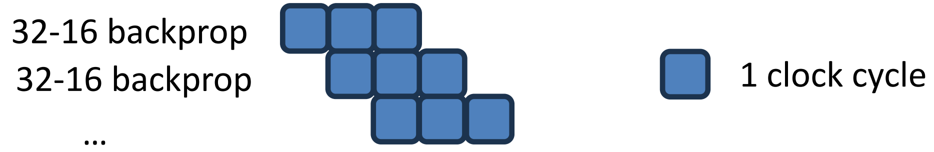
200 MHz targeted **clock frequency**, feasible based on synthesis results

The network is trained with 250 Millions simulated data:

- 250M forward passes, each pass taking 56 clock cycles



- 250M backpropagations, each pass taking 104 clock cycles



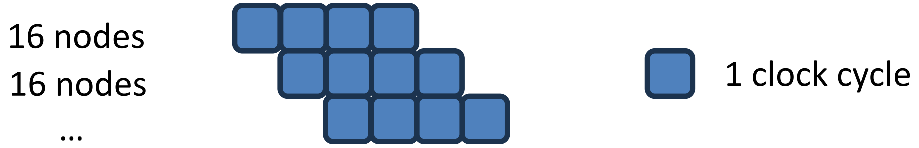
Total latency $\left(\underbrace{5}_{\text{ns}} \cdot \underbrace{(250'000'000 \cdot (56 + 104))}_{\text{Clock cycles}}\right) = 200\text{ s} + \text{PCIe}$

Time estimates

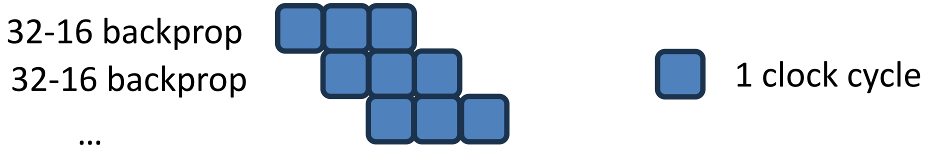
200 MHz targeted **clock frequency**, feasible based on synthesis results

The network is trained with 250 Millions simulated data:

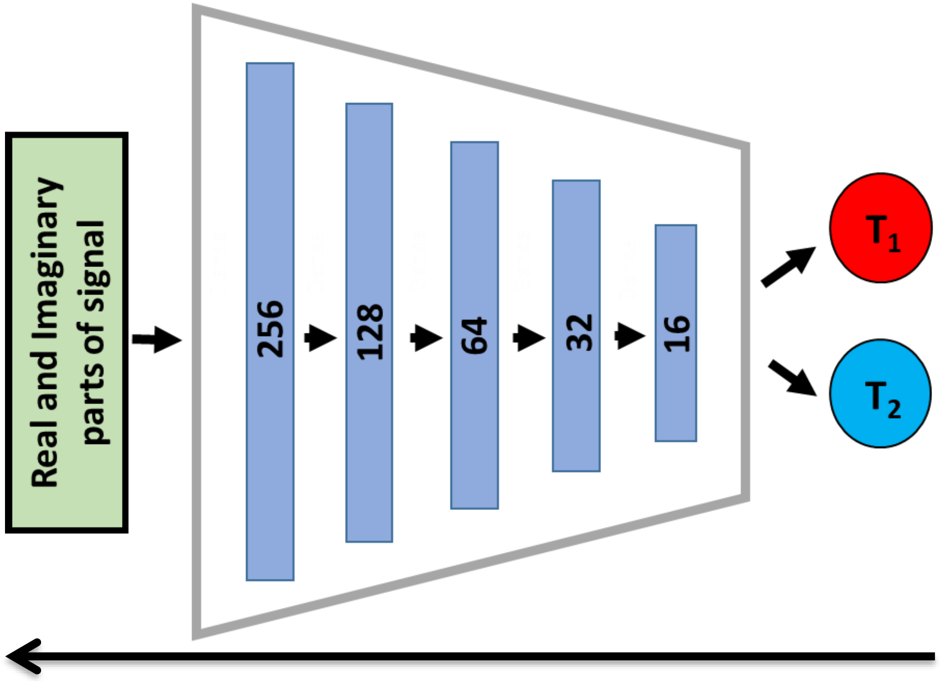
- 250M forward passes, each pass taking 56 clock cycles



- 250M backpropagations, each pass taking 104 clock cycles



Total latency $\left(\underbrace{5}_{\text{ns}}\cdot\underbrace{\left(250'000'000\cdot(56+104)\right)}_{\text{Clock cycles}}\right) = 200\text{ s} + \text{PCIe}$



Total training time on **CPU ~16 hours**

Total training time on **FPGA < 5 minutes**

Conclusions and future work

- MRF is an important and powerful technique that provides quantitative brain maps with a single acquisition
  - Helps to move in the direction of **personalized healthcare**

- Neural Networks are crucial in the reconstruction of MRF quantitative maps
  - Once the neural network is trained it is accurate and fast, but its **training** is really **demanding**

- We are developing a **hardware accelerated** neural network able to reconstruct MRF quantitative maps
  - With a clock frequency of 250 MHz, we are expecting to be able to **train the network in less than 3 minutes**

- Implement optimizing algorithms for
  - Pipelines for additions and multiplications
  - Clock domain management
  - Optimizers for backpropagation

This approach has the potential to enable **real-time brain map reconstruction**

- **Scanners** with integrated NN hardware accelerator for map reconstruction
- Analysis on **mobile devices**

INFN BOLOGNA

UNIVERSITÀ DI PISA

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Thank you for your attention

Mattia Ricchi

mattia.ricchi@phd.unipi.it

National Institute of Nuclear Physics (INFN), Division of Bologna, Italy
Department of Computer Sciences, University of Pisa, Italy
Department of Physics and Astronomy, University of Bologna, Italy