# Inference and Evaluation of Deep Convolutional Neural Networks on Microchip's Hardware Accelerator Vectorblox

EDHPC – 5th Oct 2023


Matteo Dadà


Luca Zulberti


Pietro Nannipieri


Luca Fanucci


Silvia Moranti

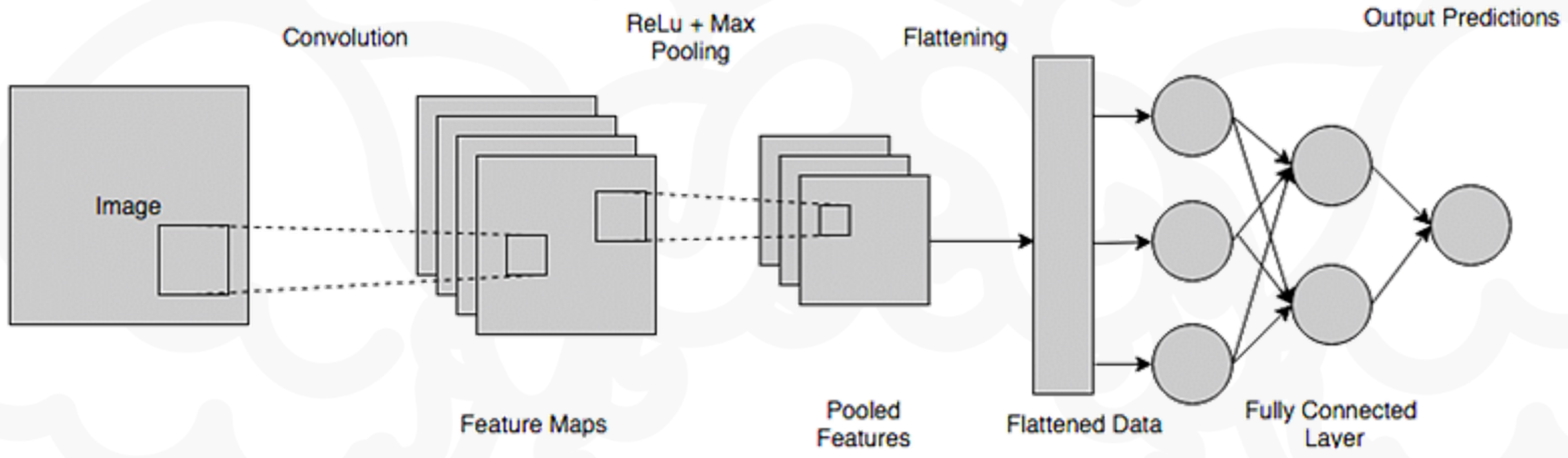
# Outline

1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

# Outline

1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

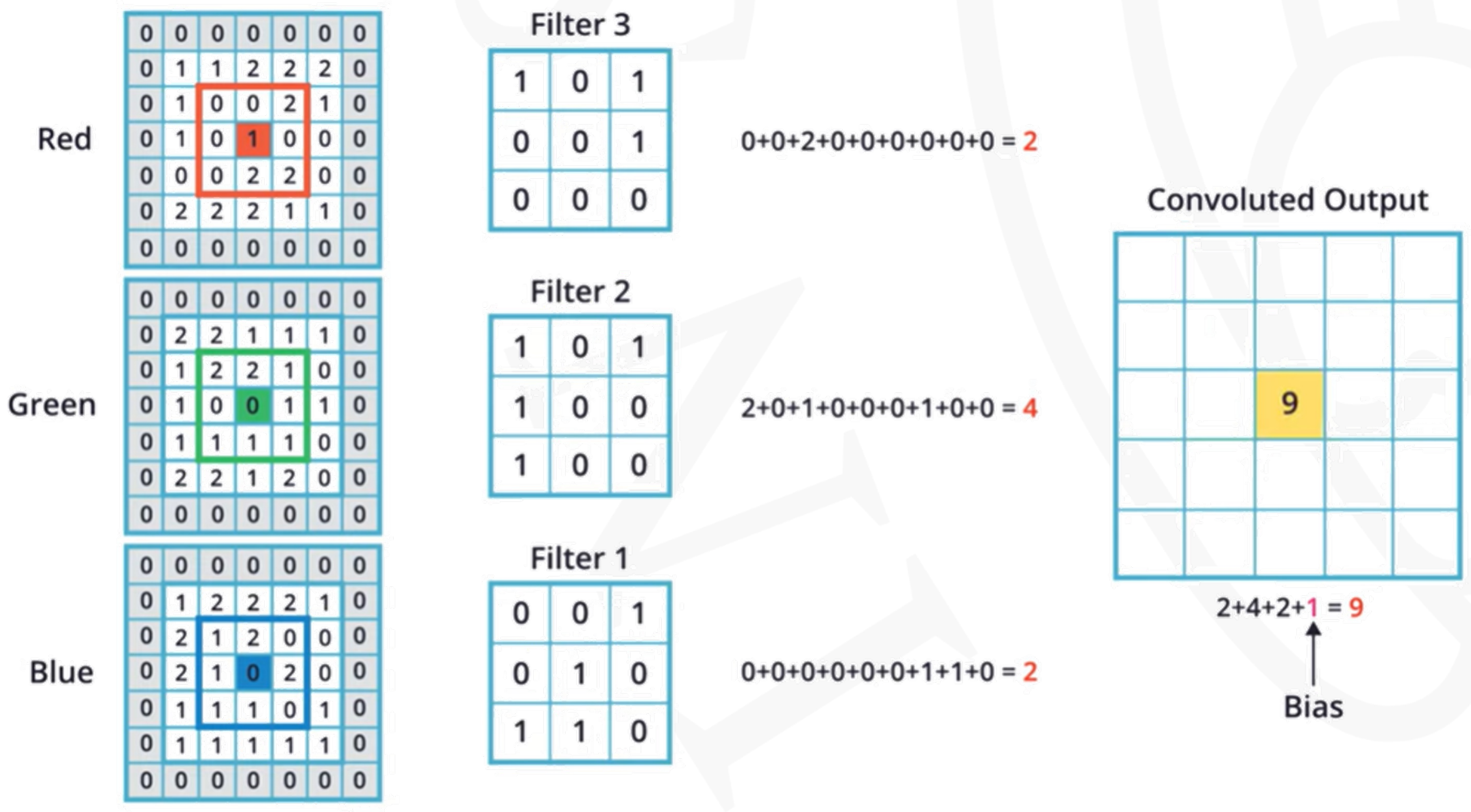
# Convolutional neural networks (CNNs): 3 main types of layers

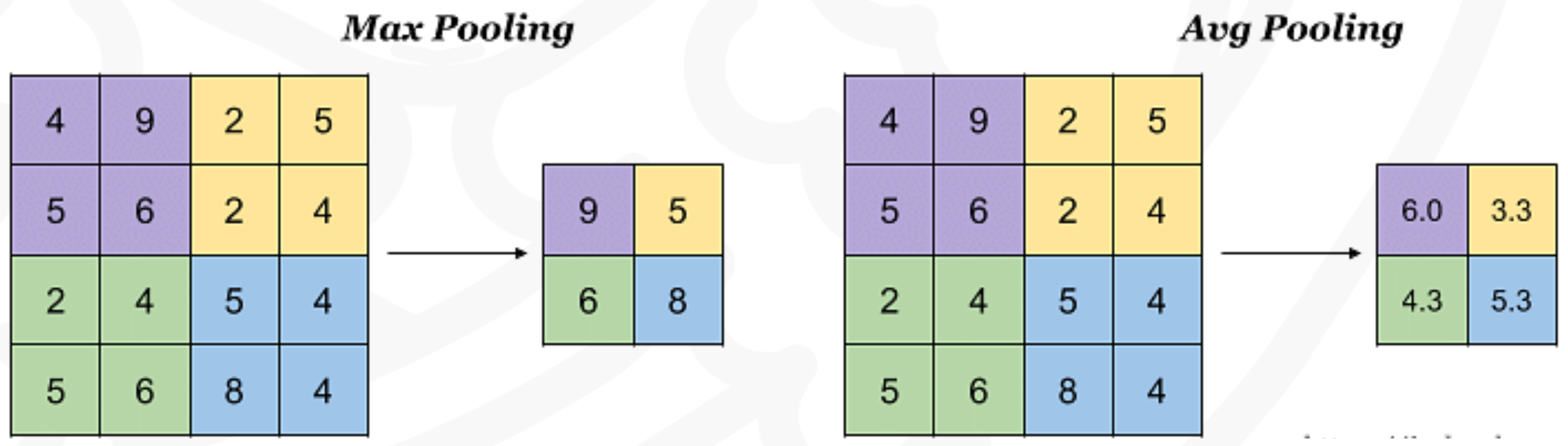


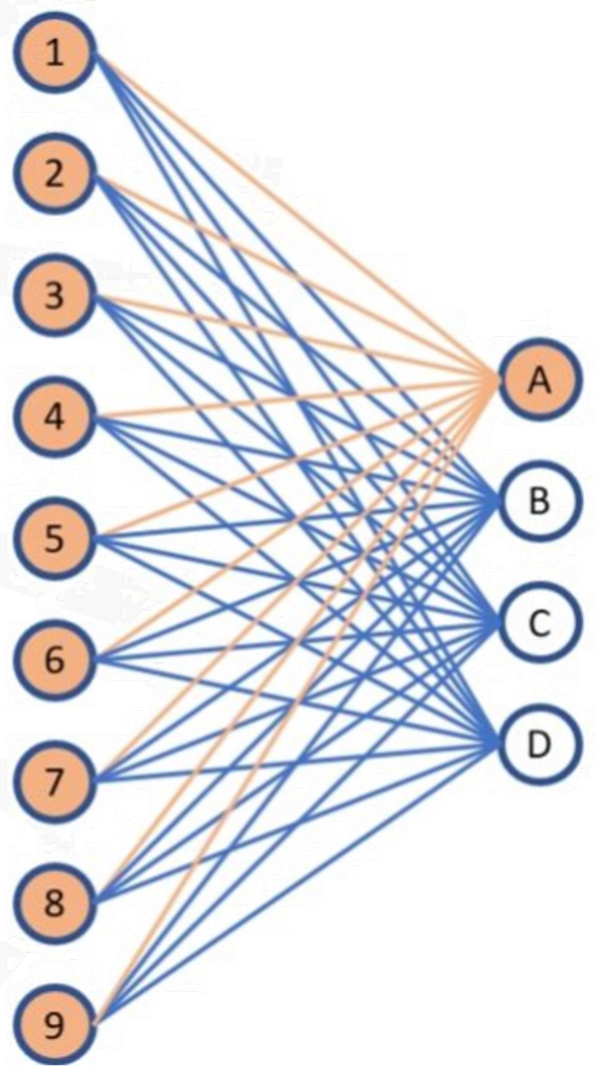

1) Convolutional layer

2) Pooling layer

3) Dense layer (final classifier)

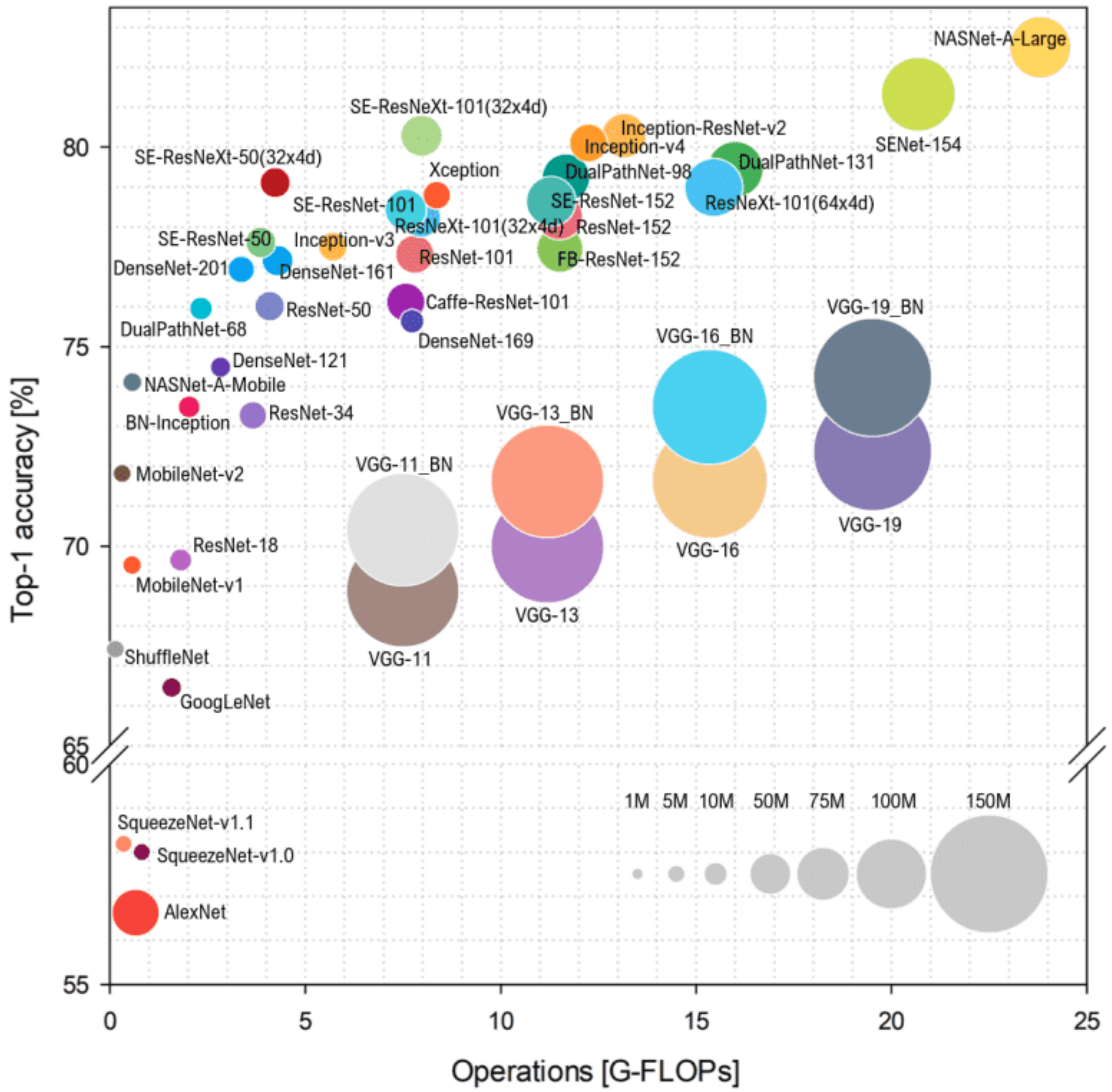


# Introduction

Increasingly complex and high-performance networks needed to achieve ever greater accuracy:




Source: Best deep CNN architectures and their principles: from AlexNet to EfficientNet

## Why choose FPGAs to accelerate (C)NNs?

### Good trade-off between performance and flexibility:

- area and power optimization with respect to GPUs
- more flexible solution than ASICs, thanks to their reprogrammability
- radiation-tolerant FPGAs are suitable for AI use in space applications

### However, the same drawback of ASICs:

- bigger design effort and a more complicated design flow compared to general-purpose solutions

**Need for toolflows to automate CNN mapping on FPGAs**

## (C)NN-FPGA Toolflows metrics

<table>
<tr><td rowspan="1">Input Interface</td><td colspan="6">Tensorflow, Pytorch, Caffe, MxNet, DarkNet…</td></tr>
<tr><td>Supported Layer List</td><td colspan="3">Dense, Convolutional, Pooling, Reshape...</td><td colspan="3">ResNet, Inception…</td></tr>
<tr><td>Hardware Architecture</td><td colspan="6">Streaming Architecture / Single Computation Engine</td></tr>
<tr><td>Portability</td><td colspan="2">Different Vendors and Families</td><td colspan="2">Different Setups (SoCs, host FPGA-servers, standalone FPGA devices…)</td><td colspan="2">Different Sizes</td></tr>
<tr><td>Arithmetic Precision</td><td colspan="3">Fixed Point (FXP) / Floating Point (FP)</td><td colspan="3">Dynamic / Uniform</td></tr>
<tr><td>Design Space Exploration (DSE)</td><td colspan="6">User Driven / Not User Driven</td></tr>
</table>

| Toolflow | Portability | Arithmetic Precision | Interface | HW architecture | DSE | Supported Layers |
|---|---|---|---|---|---|---|
| **VectorBlox SDK** | Microchip's PolarFire SoC and FPGAs | Dynamic FXP | TensorFlow, Caffe, MxNet, PyTorch, DarkNet… | CoreVectorBlox | User-driven | CNN, Dense, Res., Incep |

## Portability

One of the primary constraints inherent in vendor-specific toolflows like VectorBlox lies in portability, as it exclusively supports Microchip's PolarFire and PolarFire SoC FPGAs.

Hardware used:
PolarFire SoC FPGA IcicleKit, currently only PolarFire FPGA is rad-hard.

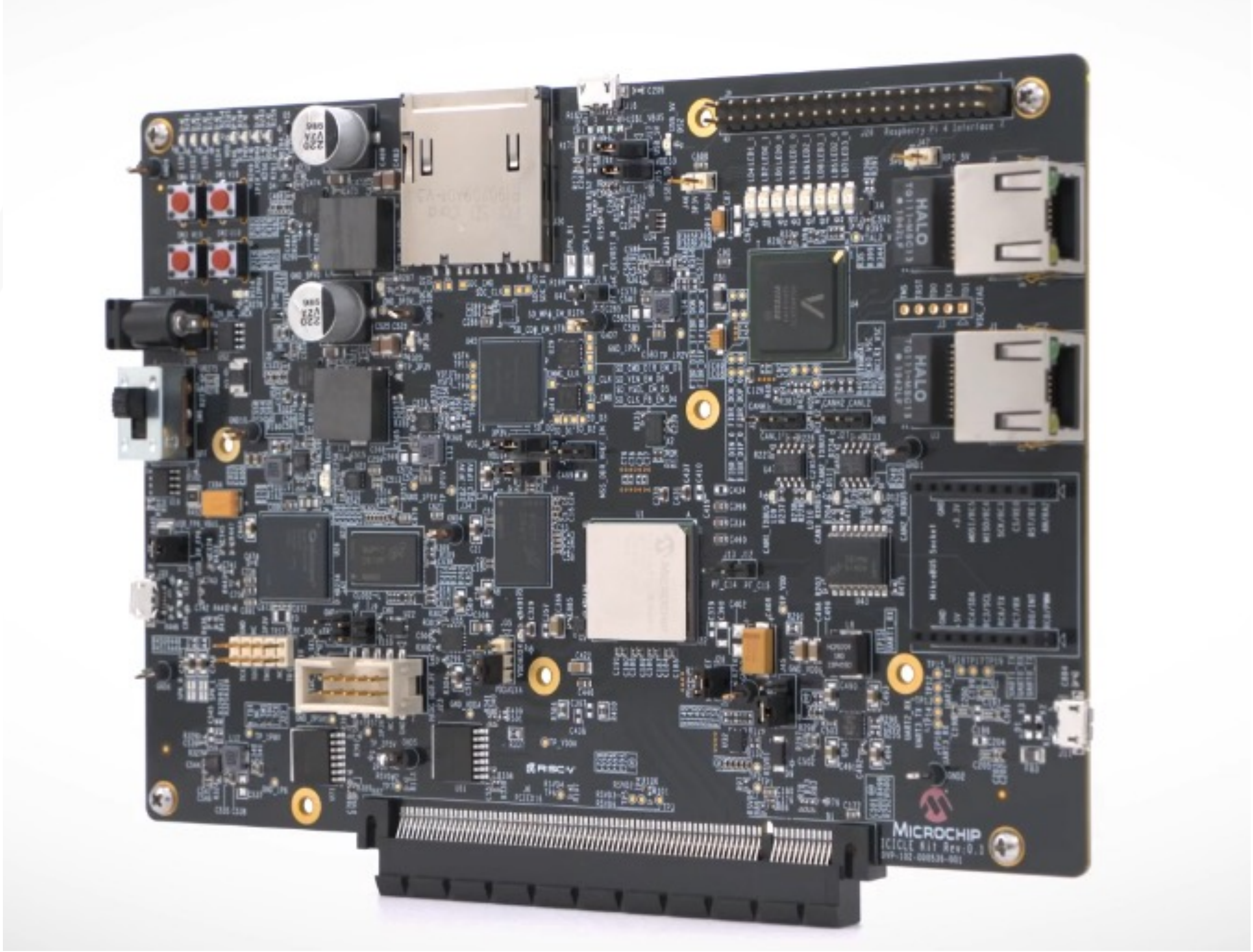
# Objectives of this work

This work aims to characterize the use of PolarFire to accelerate CNNs on satellite applications.

**The use of PolarFire for this purpose should be considered attractive for several reasons:**

- PolarFire FPGAs are low power at mid-range densities with relevant security and reliability features.
- PolarFire FPGAs are an interesting solution in terms of cost.

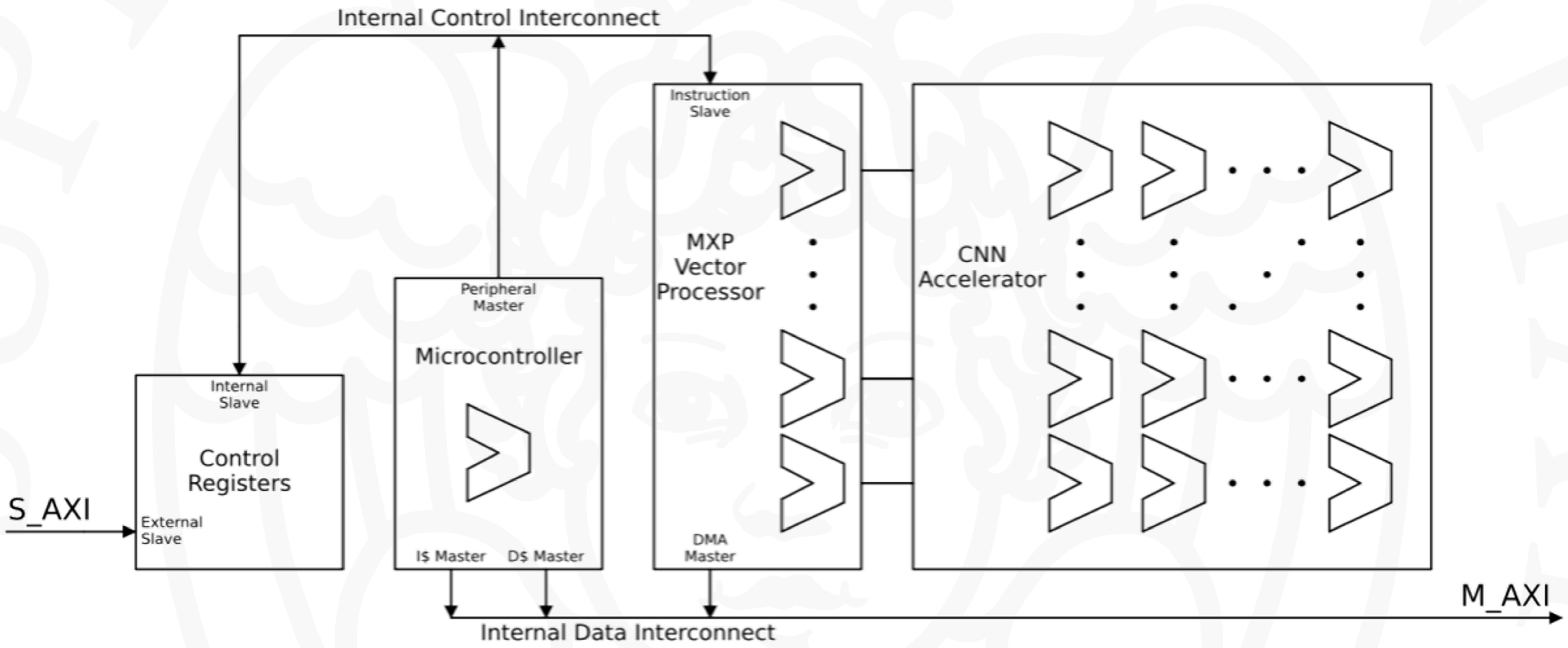
Hardware used:
PolarFire SoC FPGA IcicleKit.



# Outline

1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

## HW Architecture: CoreVectorBlox IP




Primary blocks of CoreVectorBlox:

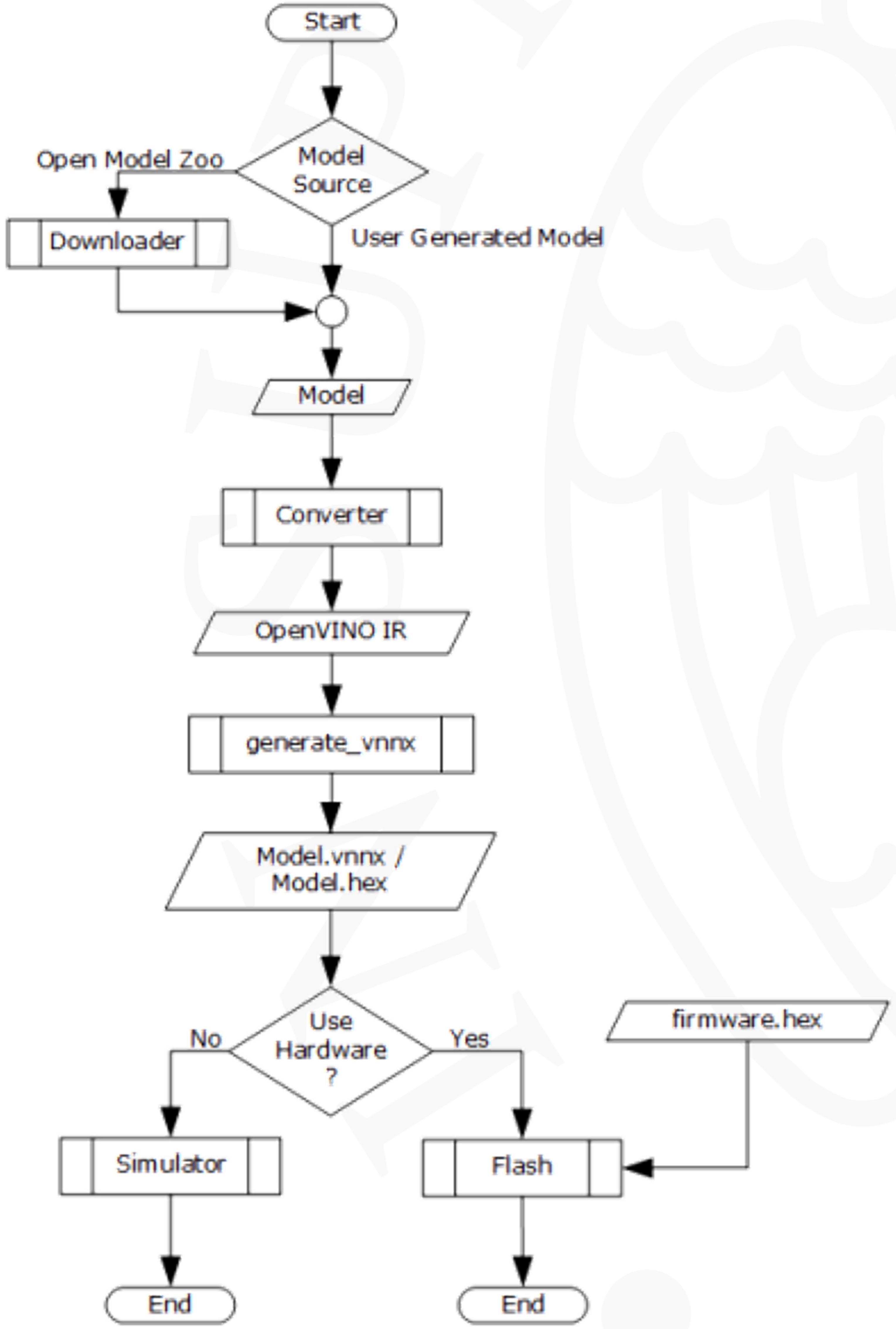
- Control Registers
- Microcontroller
- MXP Vector Processor
- CNN Accelerator

### Design Space Exploration (DSE)

Design Space Exploration (DSE) is completely **up to the user**. Three size configurations (V250, V500, V1000) are available depending on the desired level of performance. **V500** was chosen.

# CNNs to FPGA toolflows: metrics and comparison

## VectorBlox SDK Flow Diagram




## Input Interface

- Accepts models in many different frameworks (Tensorflow, PyTorch, Caffe…)

## Model Compression

- Model Optimization (removing dropout, batch normalization…)
- Quantization (from FP32 to INT8):
  - Dynamic Fixed-Point (FXP) Quantization (Different bitwidths and scale factors across different layers)
- Calibration
- Runtime Generation (BLOB generation)

## Toolflow Comparison

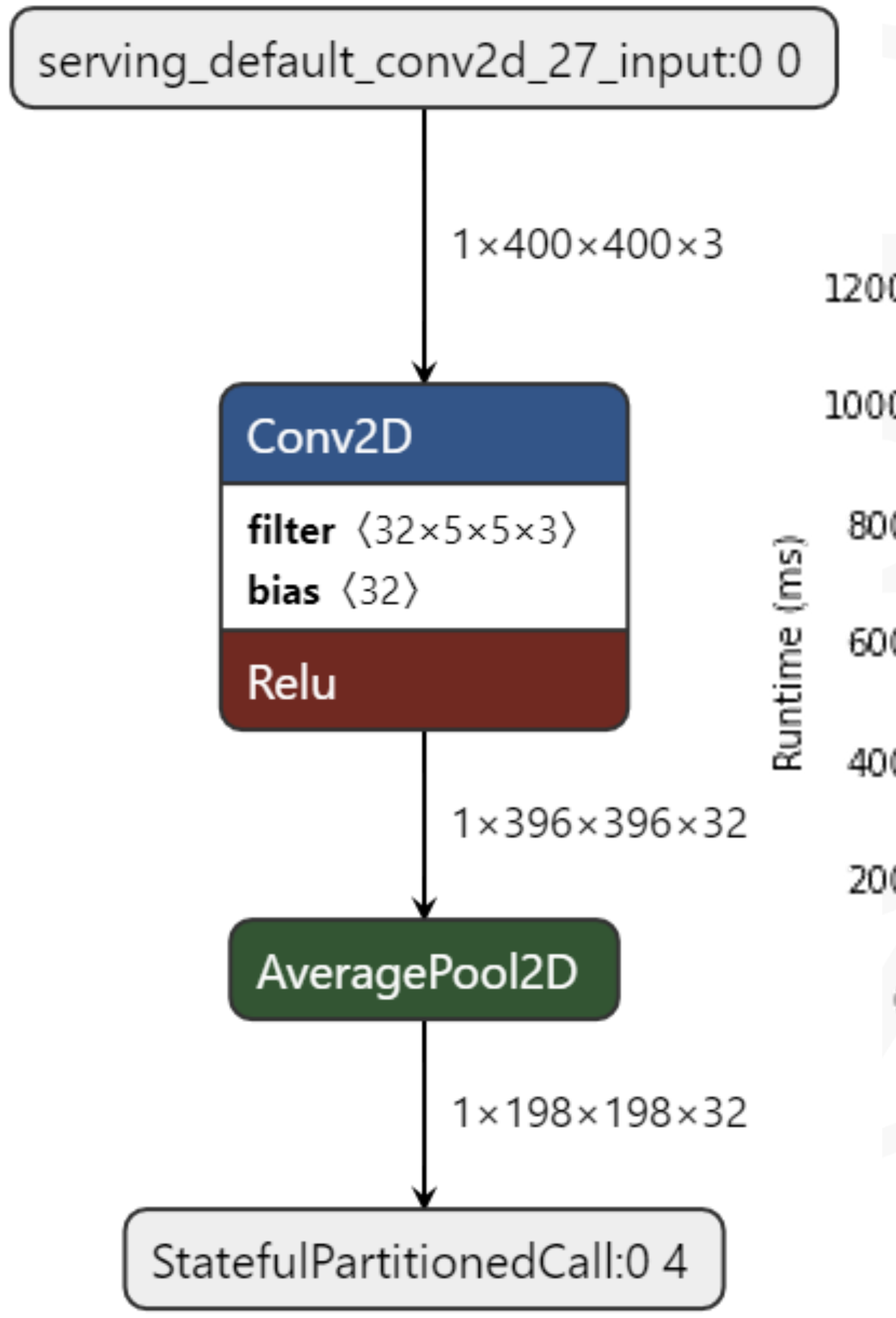
| Toolflow | Portability | Arithmetic Precision | Interface | HW architecture | DSE | Supported Layers |
|---|---|---|---|---|---|---|
| **VectorBlox SDK** | Microchip's PolarFire SoC and FPGAs | Dynamic FXP | TensorFlow, Caffe, MxNet, PyTorch and DarkNet | CoreVectorBlox | User-driven | CNN, Dense, Res., Incep |
| **Vitis AI** | Xilinx (AMD) SoC & Versal/Alveo cards | Dynamic FXP | PyTorch, TensorFlow, and ONNX | CPU+SCE | User-driven | CNN, Dense, Res., Incep., RNN |
| **Matlab DL HDL Toolbox** | Xilinx/Intel SoC | Dynamic FXP (uniform bitwidth/ different scaling factors) | PyTorch, TensorFlow, and ONNX | Deep Learning Processor | User-driven | CNN, Dense, Res., Incep., LSTM |
| **fpgaConvNet** | Xilinx SoC | Uniform FXP & FP | Caffe & Torch | Reconfigurable Streaming | Not User-driven | CNN, Res., Incep., Dense |
| **FP-DNN** | Intel Standalone | Uniform FXP & FP | TensorFlow | CPU+SCE | Not User-driven | CNN, Dense, Res., RNN |
| **Snowflake** | Xilinx SoC | Uniform FXP | Torch | CPU+SCE | Not User-driven | CNN, Res., Incep |

# Outline

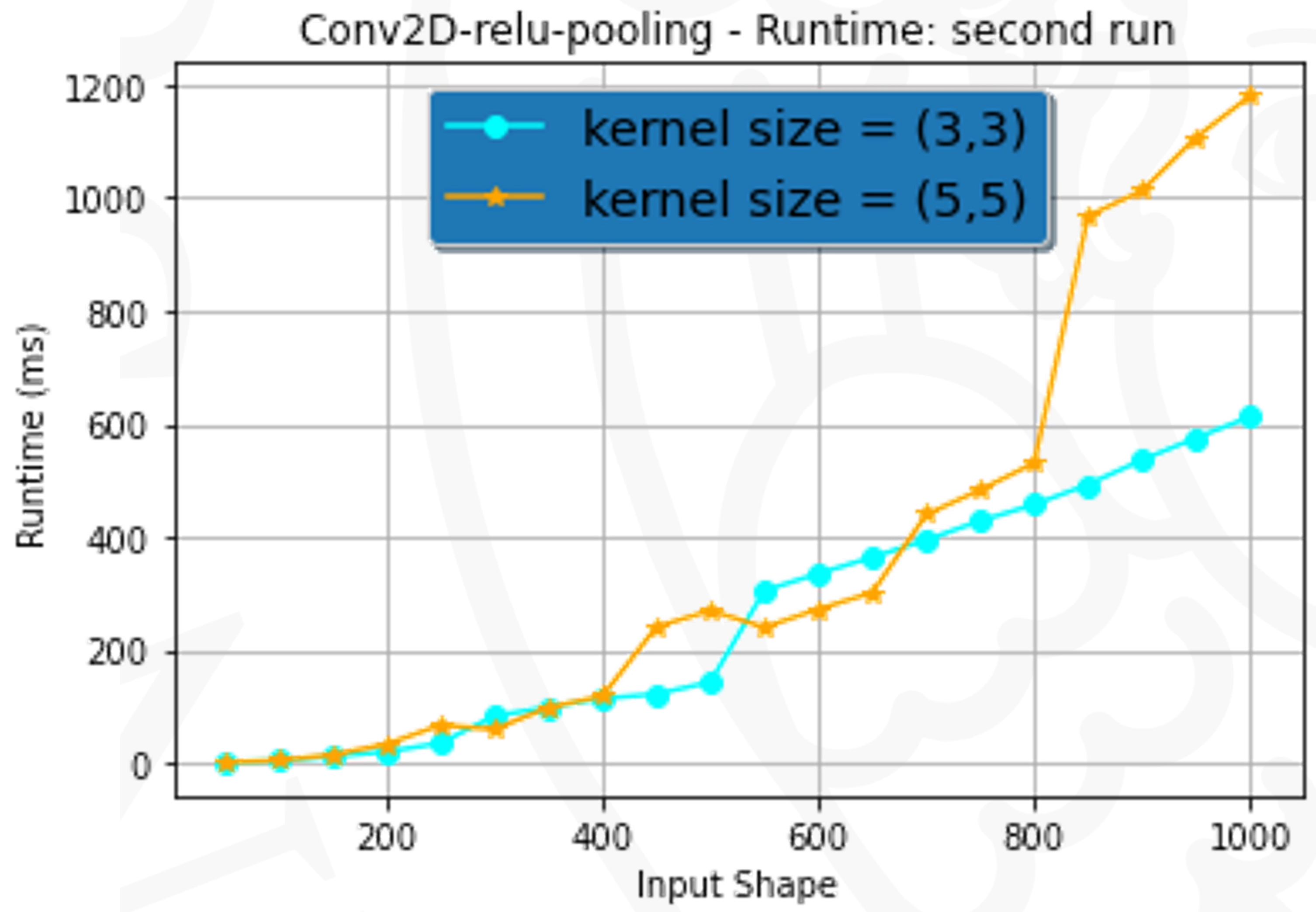
1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

# Evaluation of some of the best-known patterns

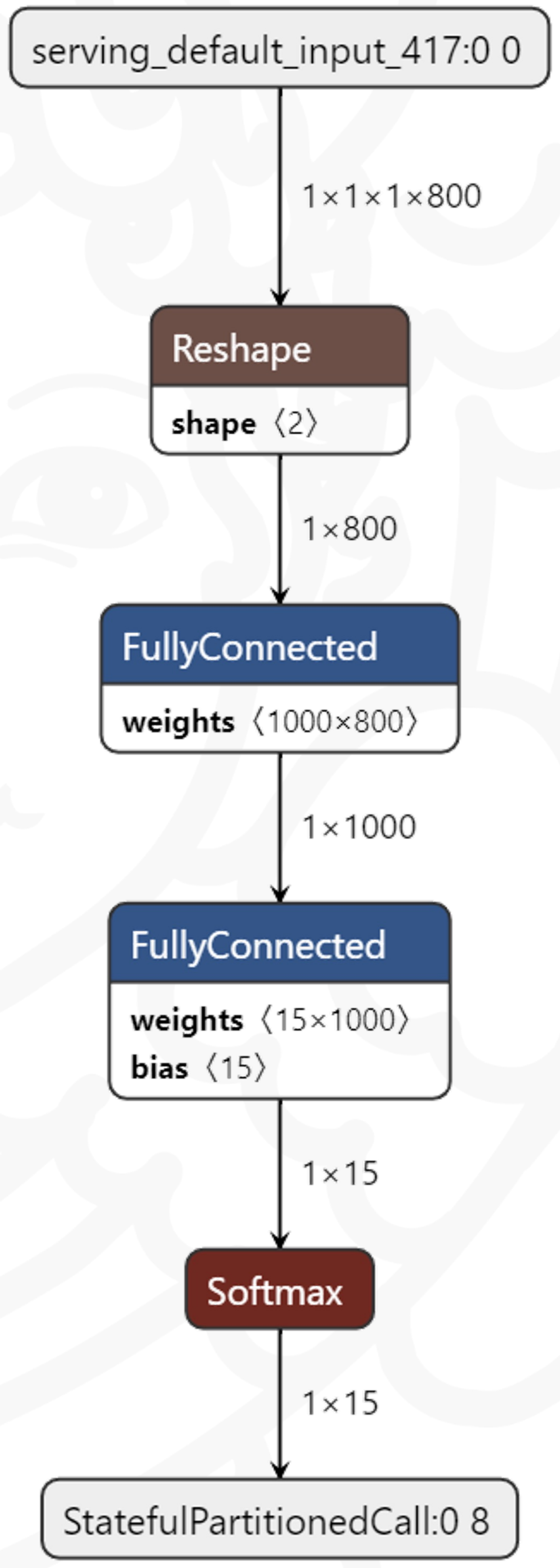
## a) Convolutional Block (Conv2D-Relu-AveragePool2D)



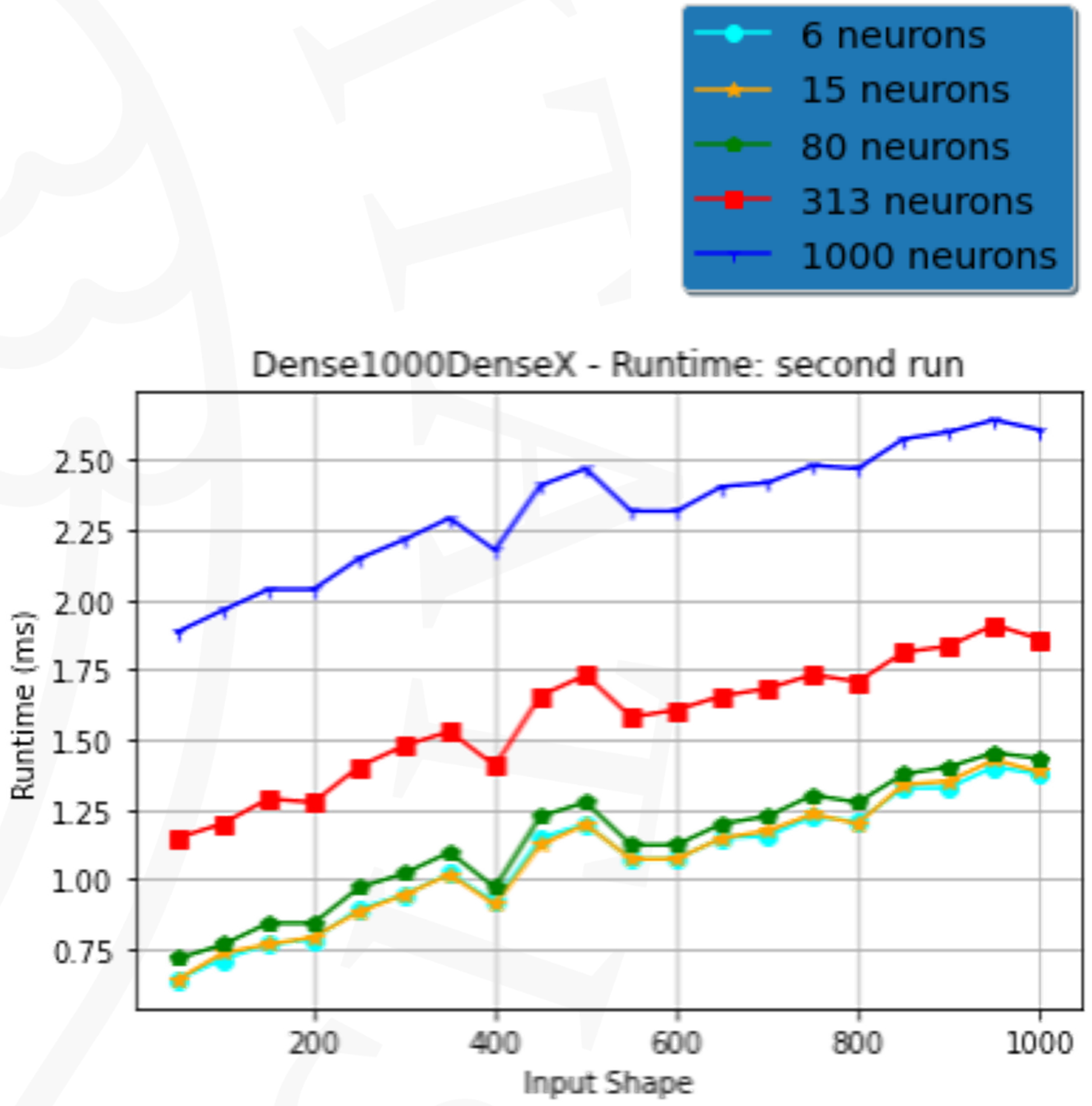

Example model




## b) Dense1000-DenseX-Softmax




Example model



# Outline

1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

## Eurosat dataset (RGB version) from Copernicus Sentinel2

- 64×64 images (RGB)
- 10 classes:
  - AnnualCrop
  - Forest
  - HerbaceousVegetation
  - Highway
  - Industrial
  - Pasture
  - PermanentCrop
  - Residential
  - River
  - SeaLake
- 27000 Images
  - 18900 Training
  - 5400 Validation
  - 2700 Test

# Implementing ML algorithms using Vectorblox

## Resource utilisation:

| Size Configuration | Vector Processor Width | Vector Scratchpad | CNN Accelerator Array Size | Peak CNN Throughput |
|---|---|---|---|---|
| V250 | 128-bit | 64 kB | 16x16 | 79 GOPs |
| V500 | 256-bit | 128 kB | 16x32 | 146 GOPs |
| V1000 | 256-bit | 256 kB | 32x32 | 279 GOPs |

| Configuration | Resource | | | | |
|---|---|---|---|---|---|
| | LUT4 | DFF | Math Blocks | uSRAM | LSRAM |
| V250 | 27244 | 27197 | 90 | 177 | 47 |
| V500 | 46622 | 48546 | 176 | 315 | 84 |
| V1000 | 61847 | 69824 | 304 | 539 | 148 |

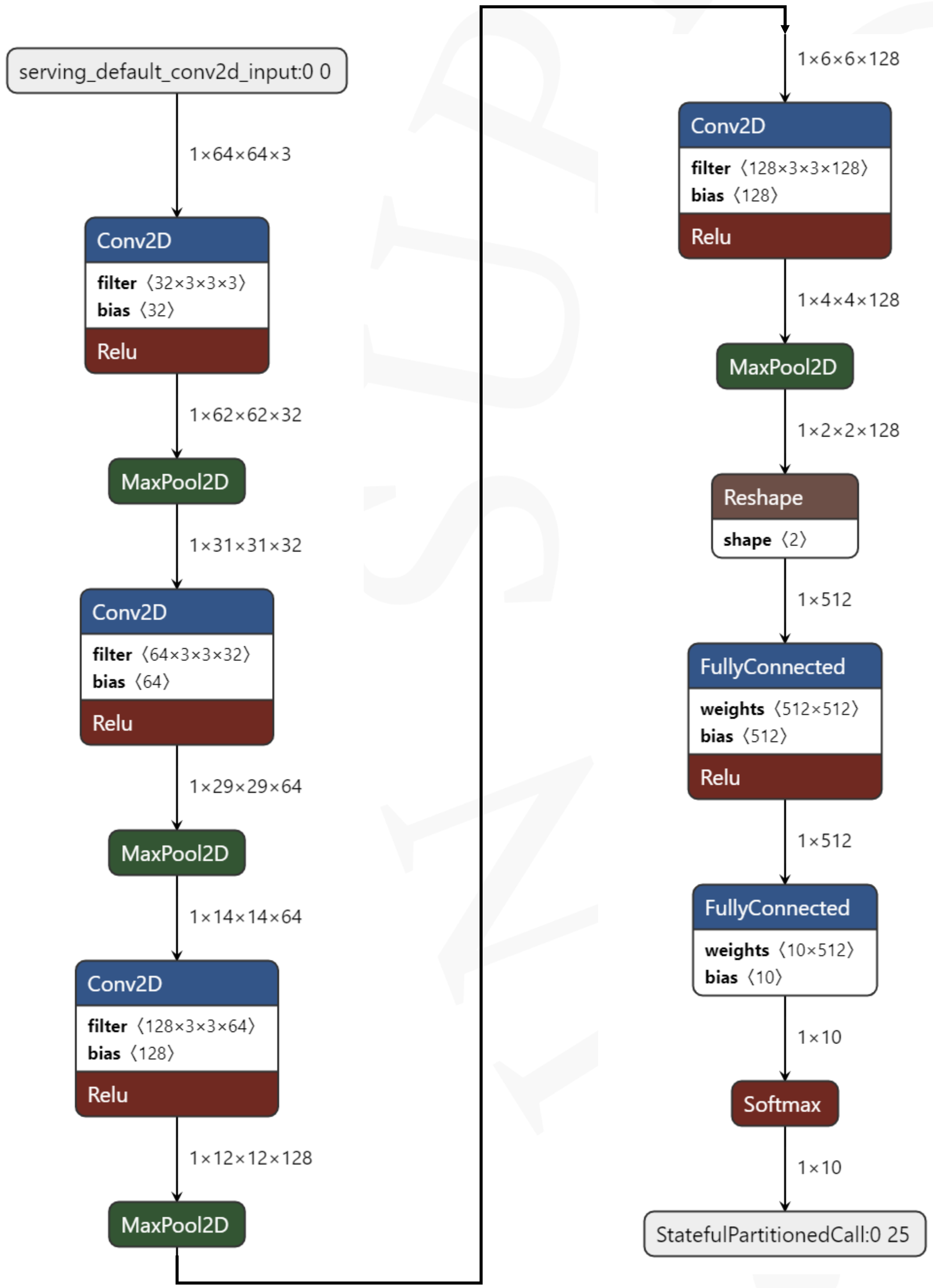
The resources used depend only on the <u>size configuration</u> parameter and not on the network being run. An <u>overlay approach</u> is used, where one instantiation can run different networks without needing to be resynthesized.

Resource utilisation on MPFS250T:

| Size Configuration: V500 | | | |
|---|---|---|---|
| RESOURCE | V500 | MPFS250T | UTILIZATION % |
| LUT4 | 46622 | 254k | 18,35 % |
| DFF | 48546 | 254k | 19,11 % |
| MATH BLOCKS | 176 | 784 | 22,45 % |

# Analysis of four increasing complexity models

## a) My custom CNN




## b) Transfer learning: Mobilenetv1

| Layer (type) | Output Shape | Param # |
|---|---|---|
| keras_layer (KerasLayer) | (None, 768) | 1832976 |
| dense_2 (Dense) | (None, 10) | 7690 |

## c) Transfer learning: Resnet50v1

| Layer (type) | Output Shape | Param # |
|---|---|---|
| keras_layer_2 (KerasLayer) | (None, 2048) | 23561152 |
| dense_4 (Dense) | (None, 10) | 20490 |

## d) Transfer learning: Inceptionv3

| Layer (type) | Output Shape | Param # |
|---|---|---|
| keras_layer (KerasLayer) | (None, 2048) | 21802784 |
| dense (Dense) | (None, 10) | 20490 |

# Analysis of four increasing complexity models

## COMPARISON:

| | Custom model | Mobilenetv1 | Resnet50v1 | Inceptionv3 |
|---|---|---|---|---|
| Total parameters | 508.618 | 1.840.666 | 23.581.642 | 21.823.274 |
| Trained parameters | 508.618 | 7.690 | 20.490 | 20.490 |
| Untrained parameters | 0 | 1.832.976 | 23.561.152 | 21.802.784 |

| | Custom model | Mobilenetv1 | Resnet50v1 | Inceptionv3 |
|---|---|---|---|---|
| Inference (ms) | 2,054 | 19,489 | 145,578 | 342,643 |
| Data part size | 562,512 (KB) | 2180,352 (KB) | 23,915768 (MB) | 23,385584 (MB) |
| Entire model size | 692,660 (KB) | 7129,516 (KB) | 50,613196 (MB) | 49,260396 (MB) |
| FP32 accuracy % (on TensorFlow) | 90,22 | 95,15 | 95,19 | 94,96 |
| INT8 accuracy % | 88,26 | 87,407 | 88,518 | 87,85 |

# Outline

1. Introduction
2. CNNs to FPGA toolflows: metrics and comparison
3. Evaluation of inference time of some of the best-known patterns, sweeping different parameters
4. Comprehensive analysis and comparison of four increasing complexity models trained on the Eurosat dataset from the ESA mission: Copernicus Sentinel 2
5. *Conclusion*

# Conclusion

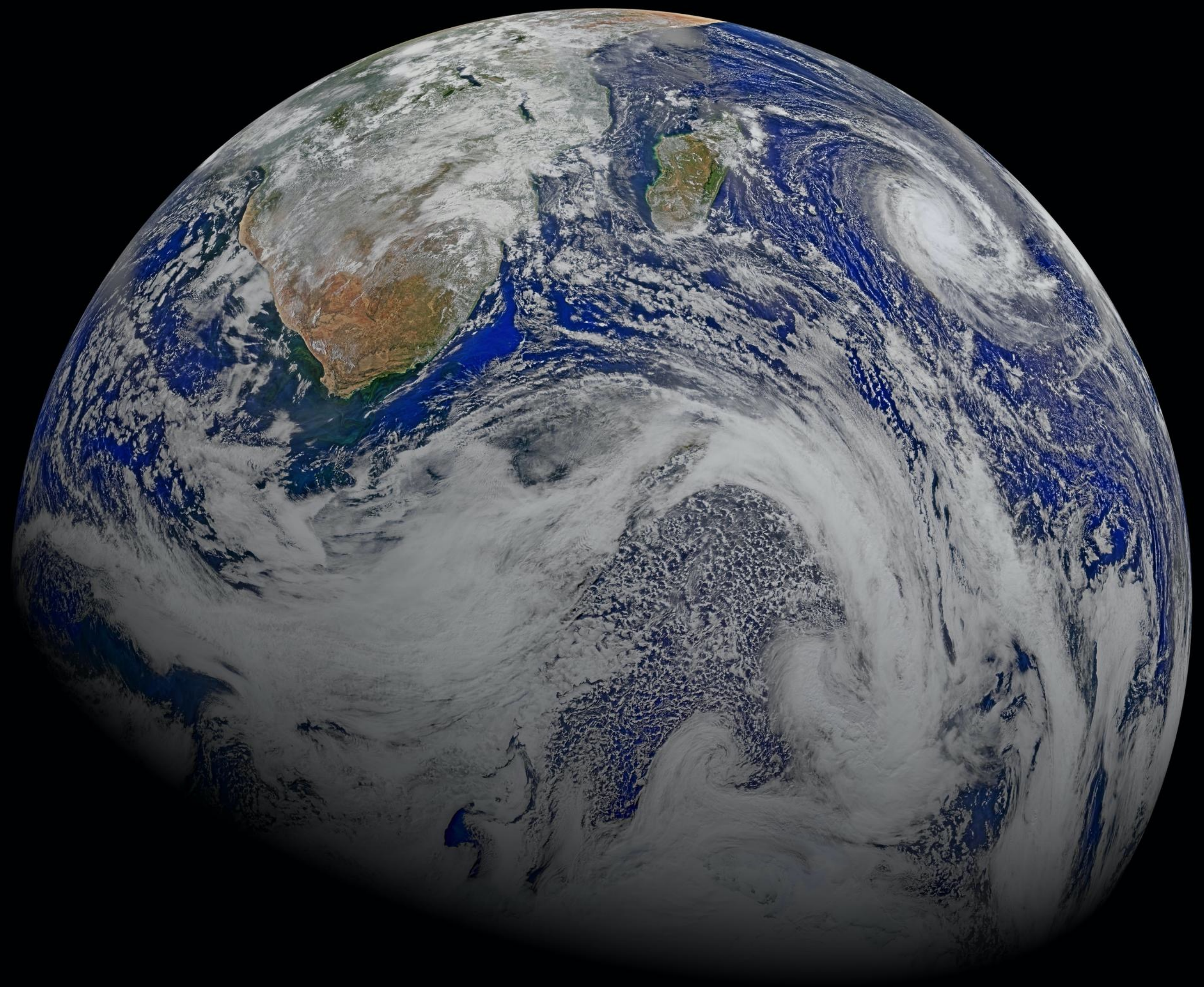
## Impressions on vectorblox:

**Pros**

- Microchip is responsive.
- They are willing to invest in and improve vectorblox, expanding the supported layers and applications for which to use it.
- Potential utilization of VBX in space application thanks to RT PolarFire FPGA.
- Many frameworks supported.
- No need to reprogram FPGA when updating CNN.

**Cons**

- Important layers unsupported (e.g. UpSampling2D).
- RNNs are not supported (therefore FDIR ML models are not supported).
- Inability to perform inference on a CNN expecting time series data input instead of images



# THANKS!

**Pietro Nannipieri**
pietro.nannipieri@unipi.it

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE