Imperial College London

fpgaConvNet: A Toolflow for Mapping Convolutional Neural Networks on Embedded FPGAs

Dr. Christos-Savvas Bouganis

Marionet UK Many-core Research Network
11th of September, Bristol University, UK

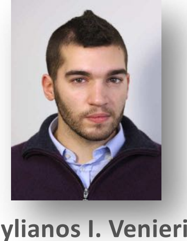
Intelligent Digital Systems Lab
Dept. of Electrical and Electronic Engineering

www.imperial.ac.uk/idsl

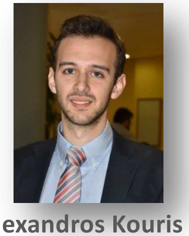
Imperial College London

intelligent Digital Systems Lab

The team



Stylianos I. Venieris
Machine Learning



Alexandros Kouris
Machine Learning,
Robotics

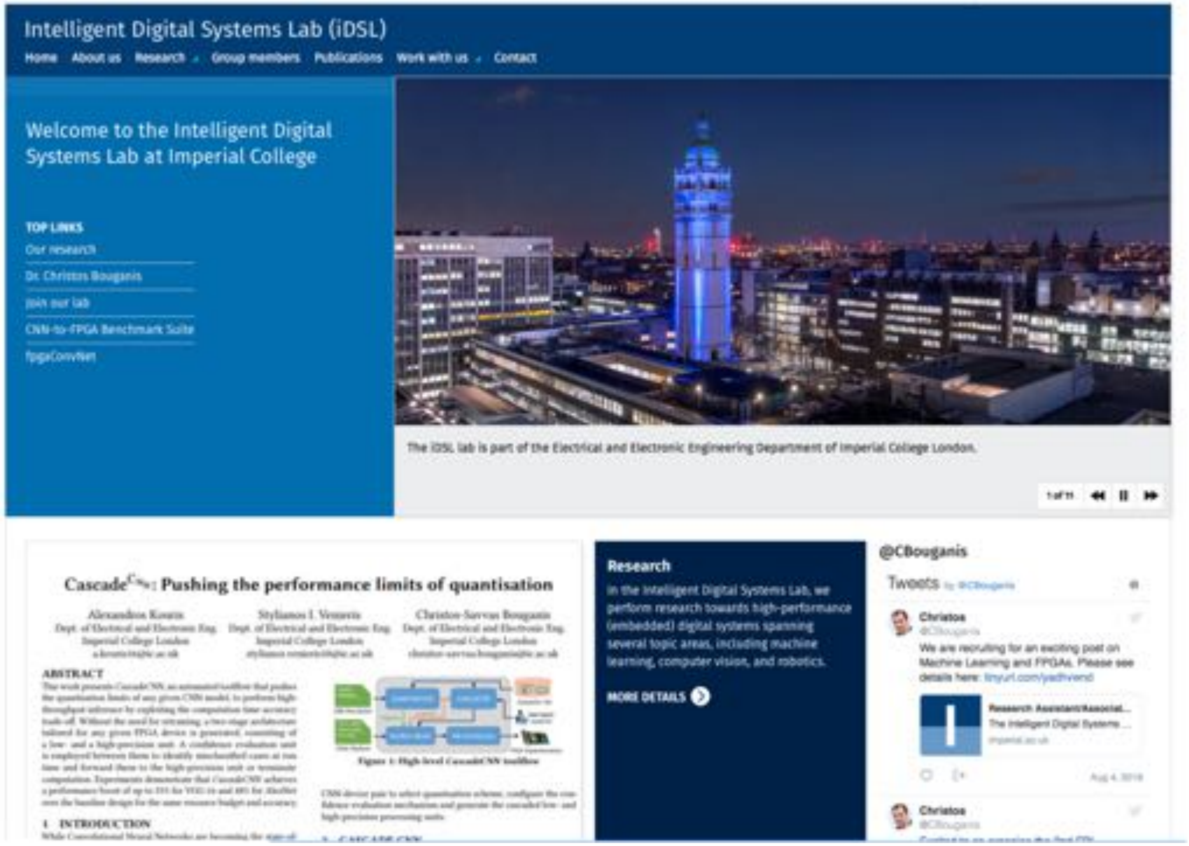
Konstantinos Boikos
Computer Vision,
SLAM

Manolis Vasileiadis
Computer Vision

Mudhar Bin Rabieah
Machine Learning

Nur Ahmadi
Brain-Machine Interface

Christos-Savvas Bouganis
iDSL Lab Director
Imperial College London



Intelligent Digital Systems Lab (iDSL)

Welcome to the Intelligent Digital Systems Lab at Imperial College

TOP LINKS

- Our research
- Dr. Christos Bouganis
- Join our lab
- CNN-to-FPGA Benchmark Suite
- fpgaConvNet

The iDSL lab is part of the Electrical and Electronic Engineering Department of Imperial College London.

Cascade^{CNN}: Pushing the performance limits of quantisation

Alexandros Kouris, Dept. of Electrical and Electronic Eng., Imperial College London
Stylianos I. Venieris, Dept. of Electrical and Electronic Eng., Imperial College London
Christos-Savvas Bouganis, Dept. of Electrical and Electronic Eng., Imperial College London

ABSTRACT

This work presents CascadeCNN, an automated toolflow that pushes the quantisation limits of any given CNN model, to perform high-throughput inference by exploiting the computation time-accuracy trade-off. Without the need for retraining, a two-stage architecture tailored for any given FPGA device is generated, consisting of a low- and a high-precision unit. A confidence evaluation unit is employed between them to identify misclassified cases at run time and forward them to the high-precision unit or terminate computation. Experiments demonstrate that CascadeCNN achieves a performance boost of up to 55% for VGG-16 and 48% for AlexNet over the baseline design for the same resource budget and accuracy.

Figure 1: High-level CascadeCNN toolflow

CNN-device pair to select quantisation scheme, configure the confidence evaluation mechanism and generate the cascaded low- and high-precision processing units.

1 INTRODUCTION

While Convolutional Neural Networks are becoming the state-of-

2 CASCADE CNN

Research

In the Intelligent Digital Systems Lab, we perform research towards high-performance (embedded) digital systems spanning several topic areas, including machine learning, computer vision, and robotics.

MORE DETAILS

@CBouganis

Tweets by @CBouganis

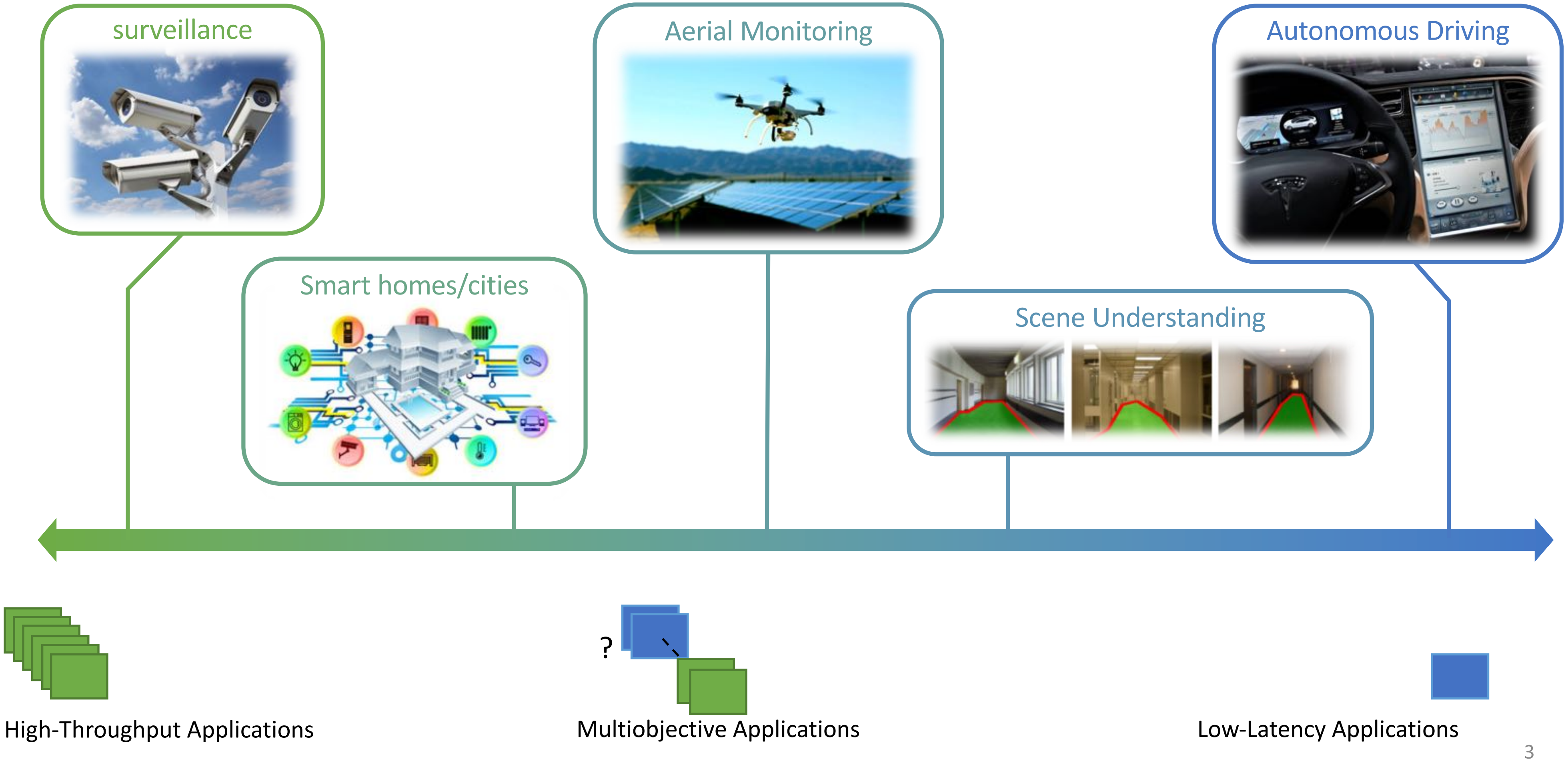
Christos
@CBouganis
We are recruiting for an exciting post on Machine Learning and FPGAs. Please see details here: tinyurl.com/yadhvend

Research Assistant/Associat...
The Intelligent Digital Systems ...
imperial.ac.uk

Aug 4, 2018

Christos
@CBouganis

DNNs in the Embedded Space – Variability in Performance Requirements



surveillance

Smart homes/cities

Aerial Monitoring

Scene Understanding

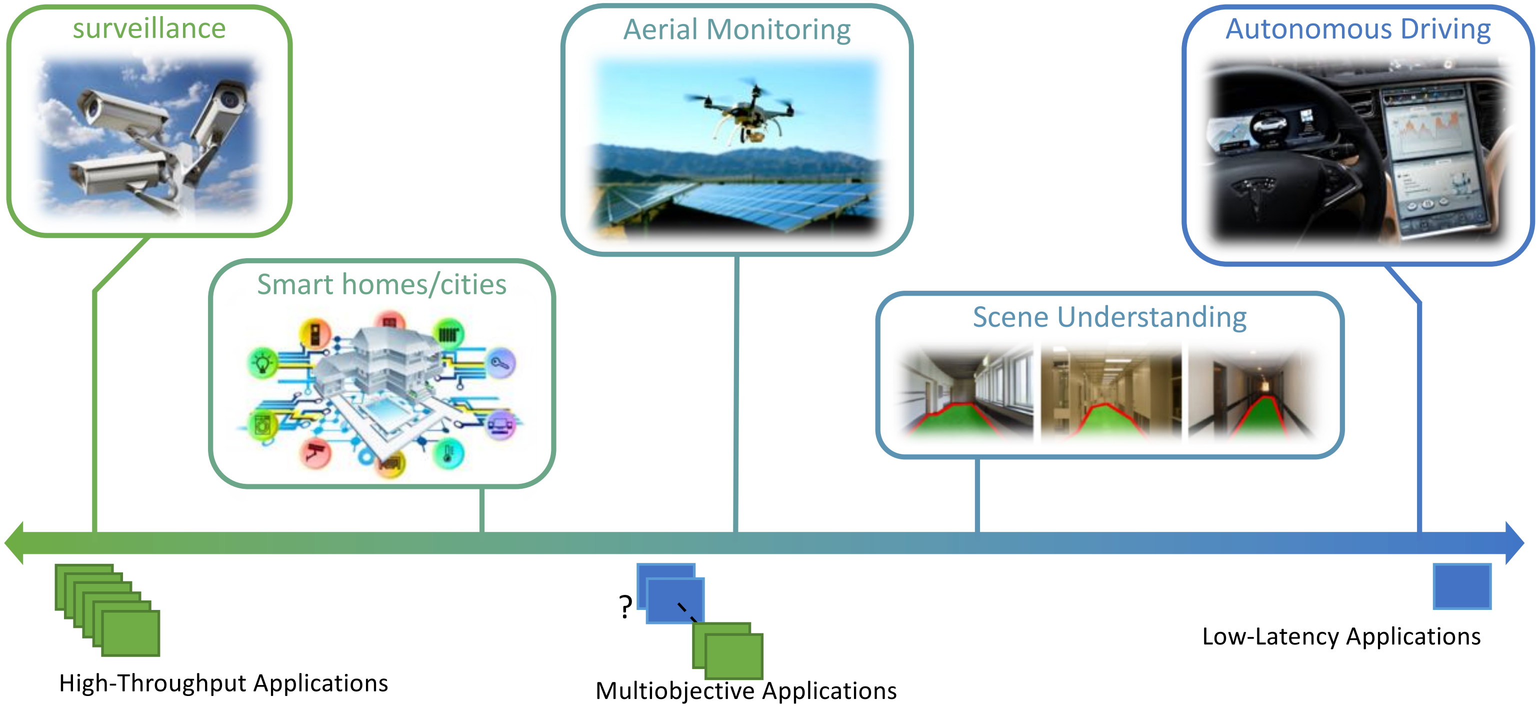
Autonomous Driving

High-Throughput Applications

Multiobjective Applications

Low-Latency Applications

DNNs in the Embedded Space – Variability in Performance Requirements

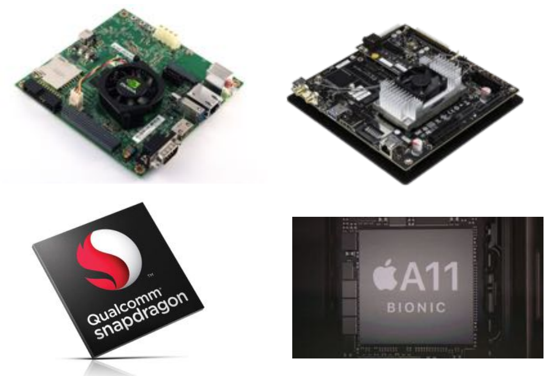


Power constraints

- Absolute power consumption
- Performance-per-Watt

Conventional and Unconventional Embedded Platforms for Neural Networks

GPUs – Tegra K1, X1 and X2
DSPs – Qualcomm Hexagon,
Apple Neural Engine, …



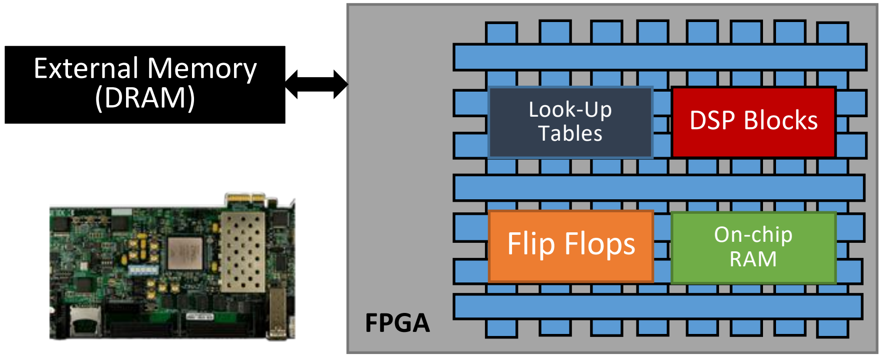
✓ High throughput
✗ Low latency
✗ Low power
✓ Tools

customisation

TPU
Myriad X
GraphCore

FPGAs

- Custom datapath
- Custom memory subsystem
- Programmable interconnections
- Reconfigurability



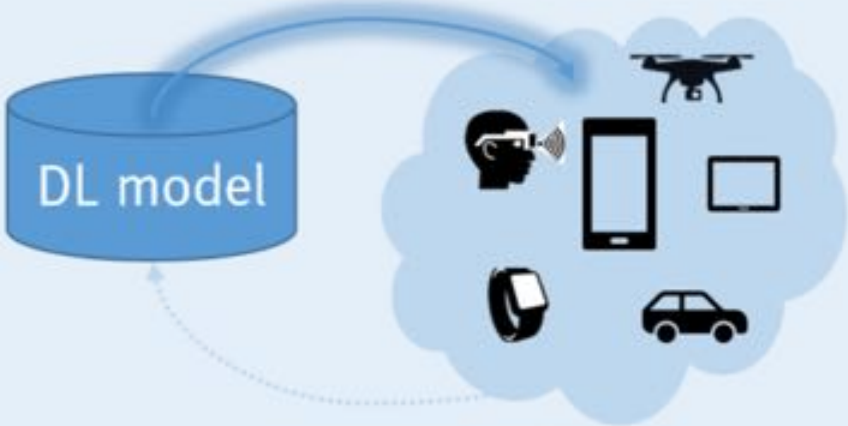
✓ High throughput
✓ Low latency
✓ Low power
✗ Tools

Challenge: Huge design space
Our Approach: Automated toolflows

Research Areas / Challenges

- Mapping Automation
- Multiple CNN Mapping

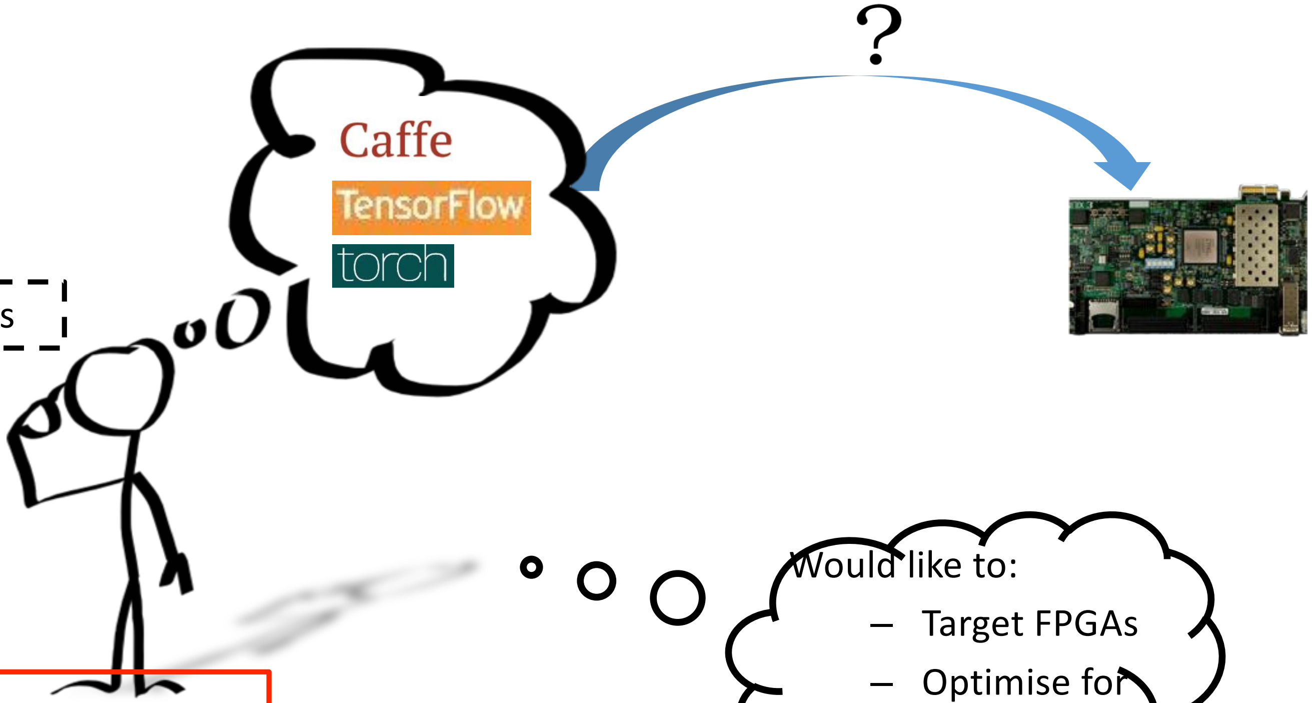
Challenge #1: Mapping Automation



Challenge #1: Mapping Automation

Little knowledge about FPGAs
Ease of deployment
“Good” designs

Deep Learning Developers



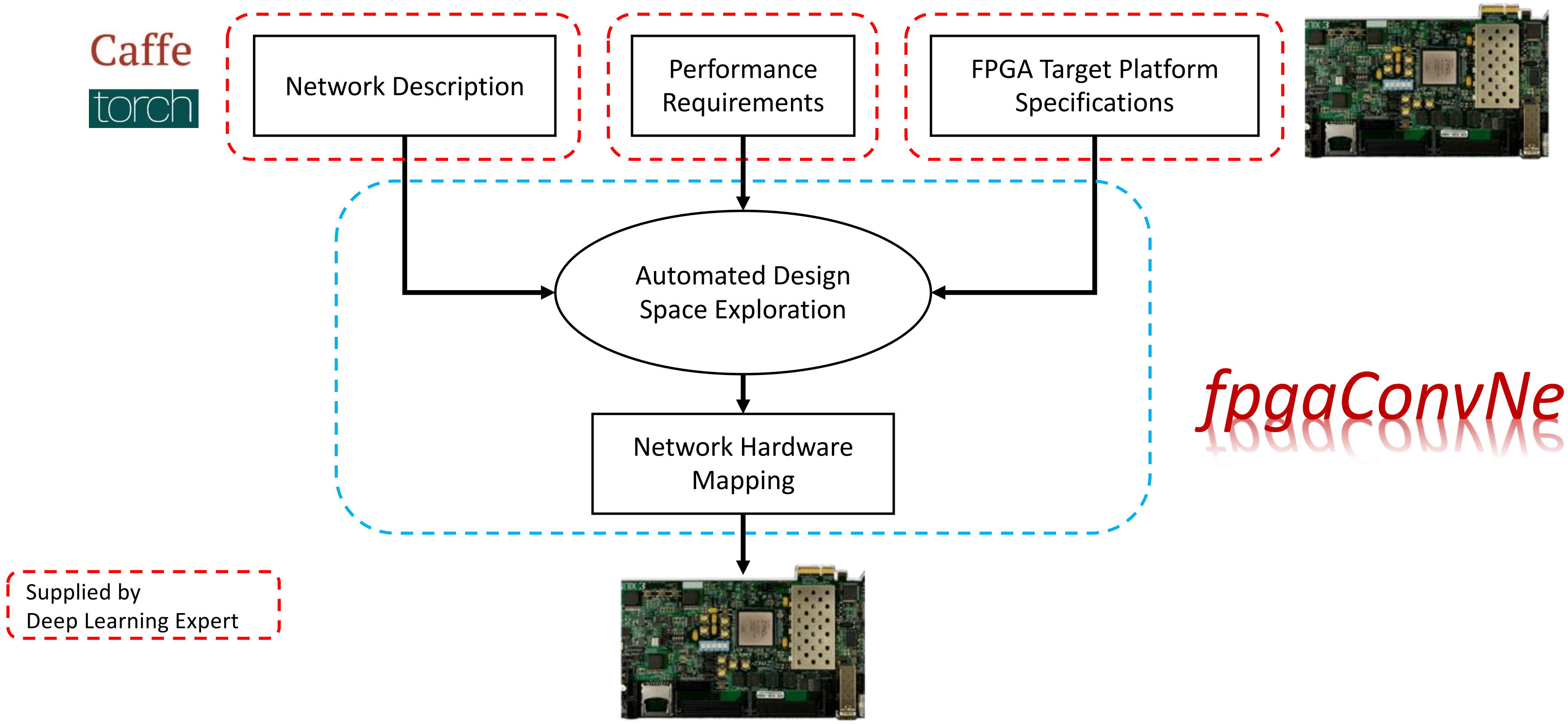
Would like to:

- Target FPGAs
- Optimise for high performance

Challenges:

- Learn to design hardware
- High-dimensional design space
- Diverse application-level needs
- High utilization of the FPGA resources
- Design automation (+> change of requirements)

Challenge #1: Automated CNN-to-FPGA Toolflow



Caffe

torch

Network Description

Performance Requirements

FPGA Target Platform Specifications

Automated Design Space Exploration

Network Hardware Mapping

fpgaConvNet

Supplied by
Deep Learning Expert

fpgaConvNet – CNN Modelling Framework

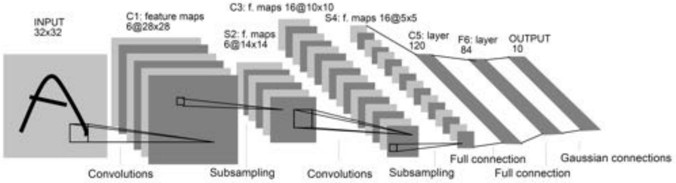
Key Characteristics

- Differentiation factors:
  - Streaming architecture
  - Hardware design tailored to the target CNN
  - No limit on #weights, or size of CNN

- Synchronous Dataflow Modelling for CNNs
  - CNN as a data-driven graph
  - Workload is represented as a matrix
  - Each layer mapped to a tunable set of hardware building blocks

- Design space exploration based on **transformations**
  - Coarse-grained folding
  - Fine-grained folding
  - Graph partitioning with reconfiguration
  - Weight Reloading

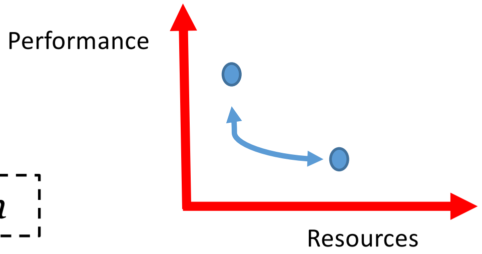
Streaming



Analytical Power

Max Throughput or Min Latency

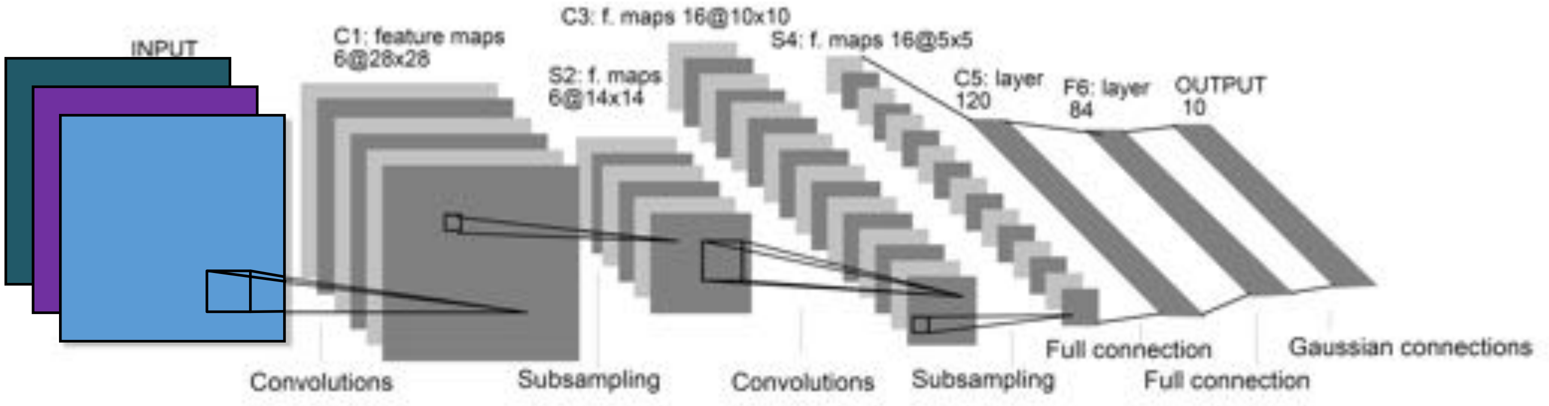
$$t_{total}(B, N_P, \mathbf{\Gamma}) = \sum_{i=1}^{N_P} t_i(B, \mathbf{\Gamma}_i) + (N_P - 1) \cdot t_{reconfig.}$$



Customisation

Under the hood: Convolutional Neural Networks (ConvNets)

convolution + nonlinearity | pooling | convolution + nonlinearity | pooling



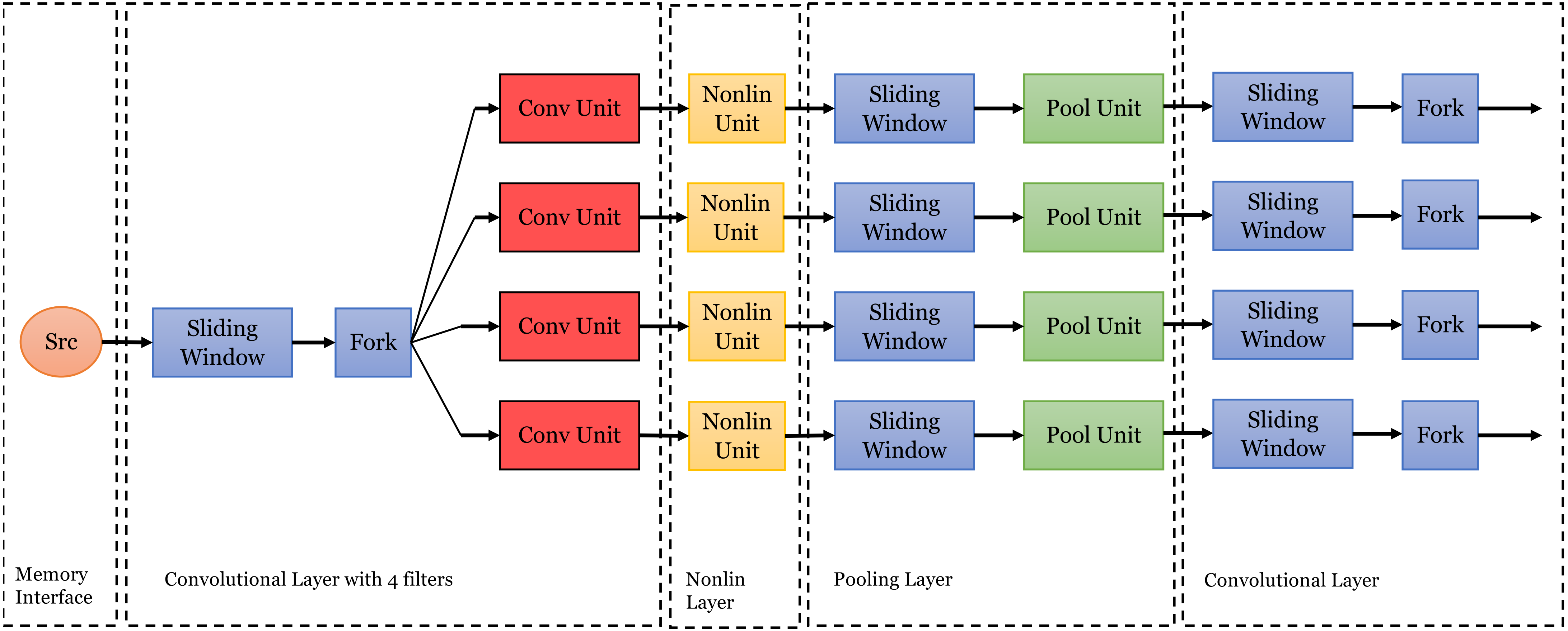
- ConvNet Inference
  - Tailored to images and data with spatial patterns
  - Built as a sequence of layers (Convolutional, Nonlinearity and Pooling Layer)
  - Feedforward operation
  - Inherently streaming

Multiple dot products

Nonlinear Operator

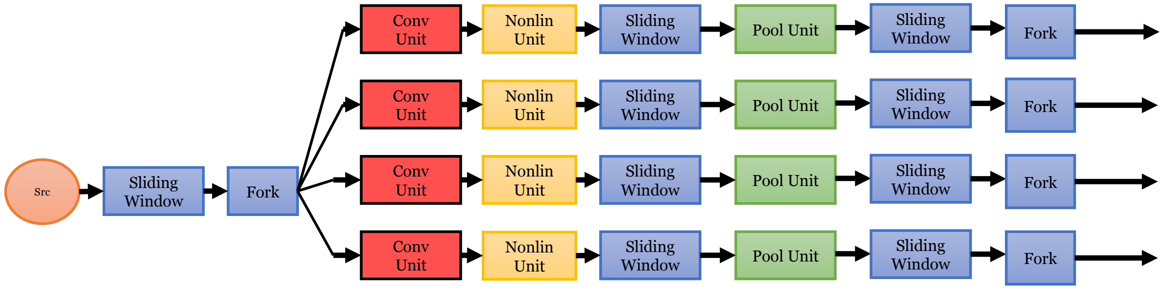
Max or average in a vector

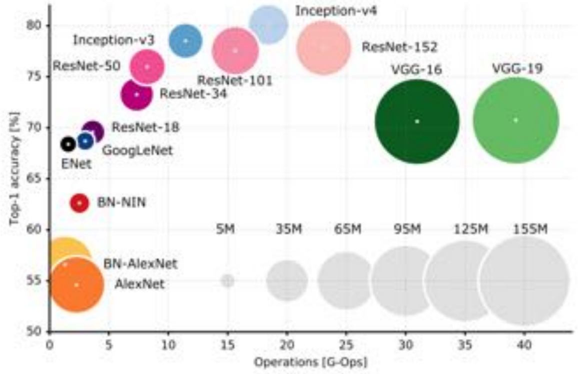
fpgaConvNet – Streaming Architecture for CNNs



fpgaConvNet – Streaming Architecture for CNNs

CNN Hardware SDF Graph

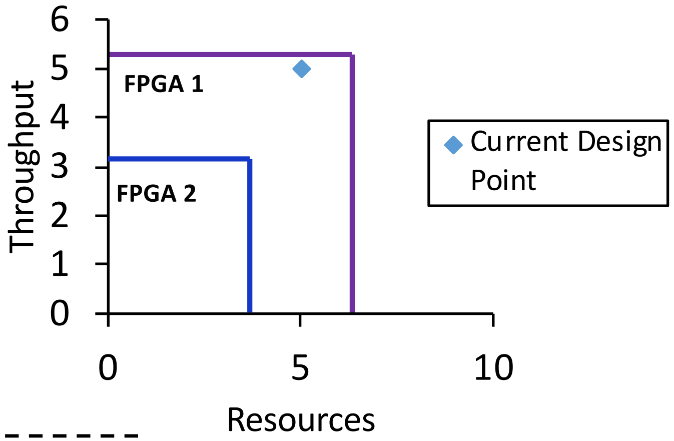




Complex Model➔ Bottlenecks:

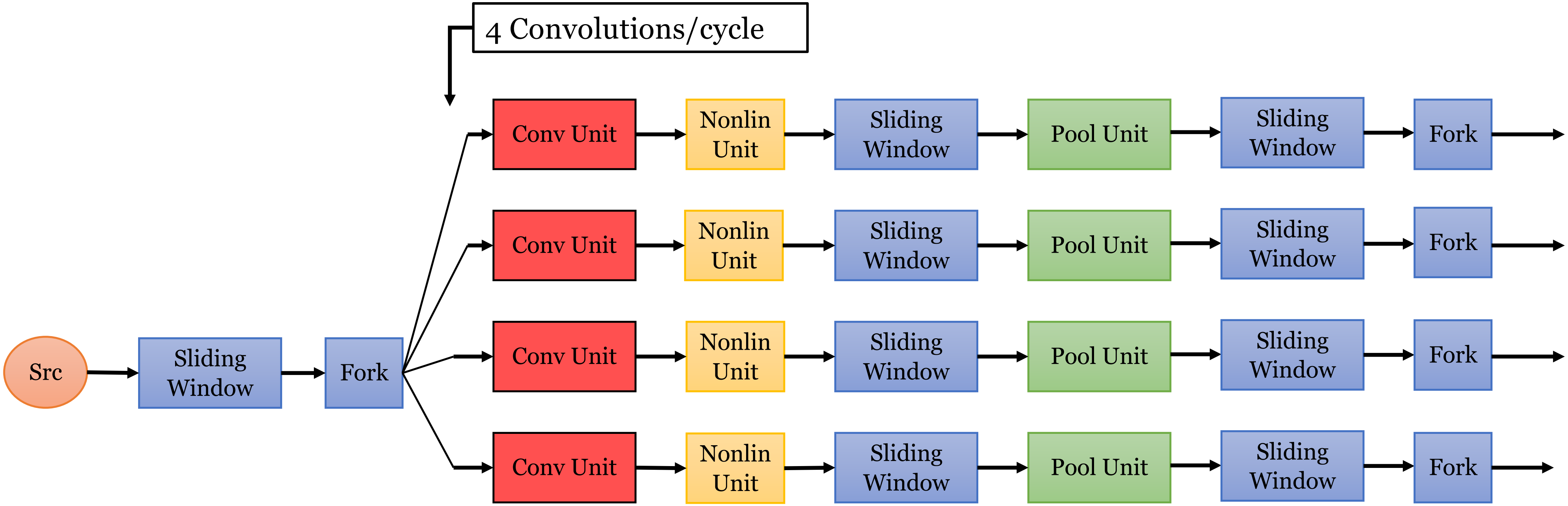
- Limited *compute resources*
- Limited *on-chip memory capacity* for model parameters
- Limited *off-chip memory bandwidth*

Design Space



Define a set of graph transformations to traverse the design space in **fast** and **principled** way

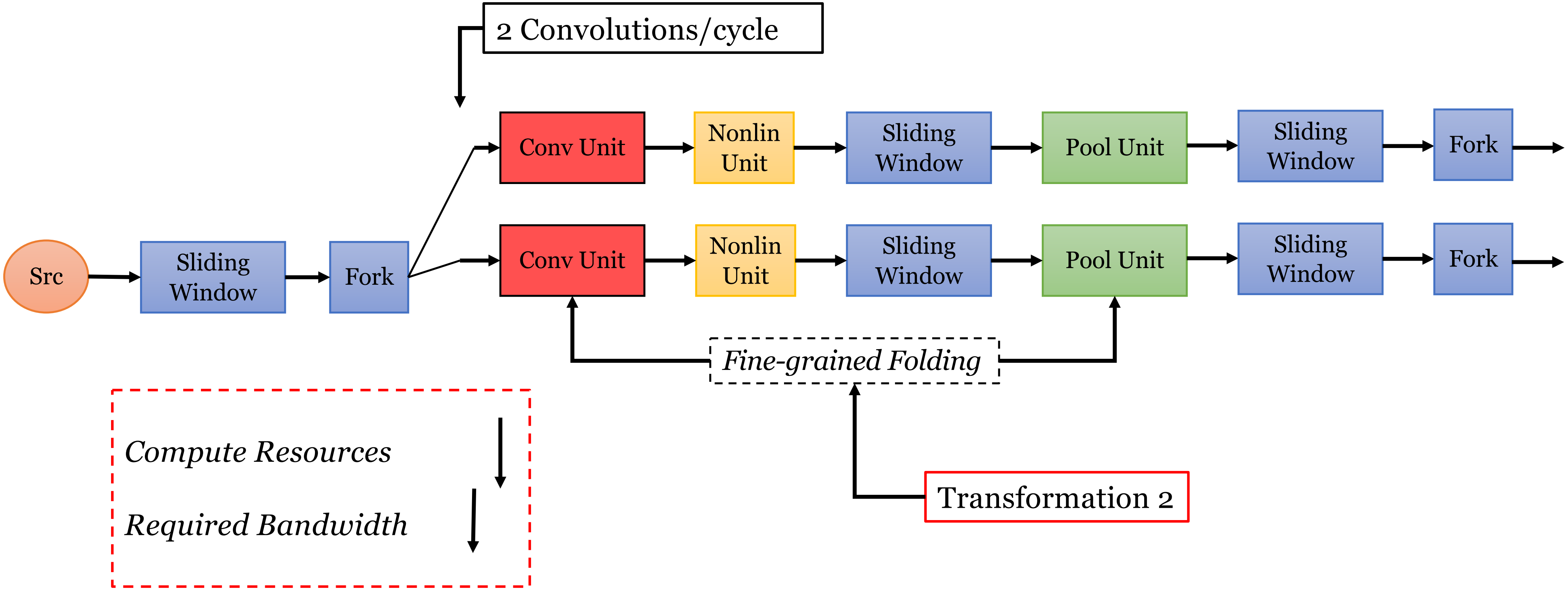
Transformation 1: Coarse-grained Folding



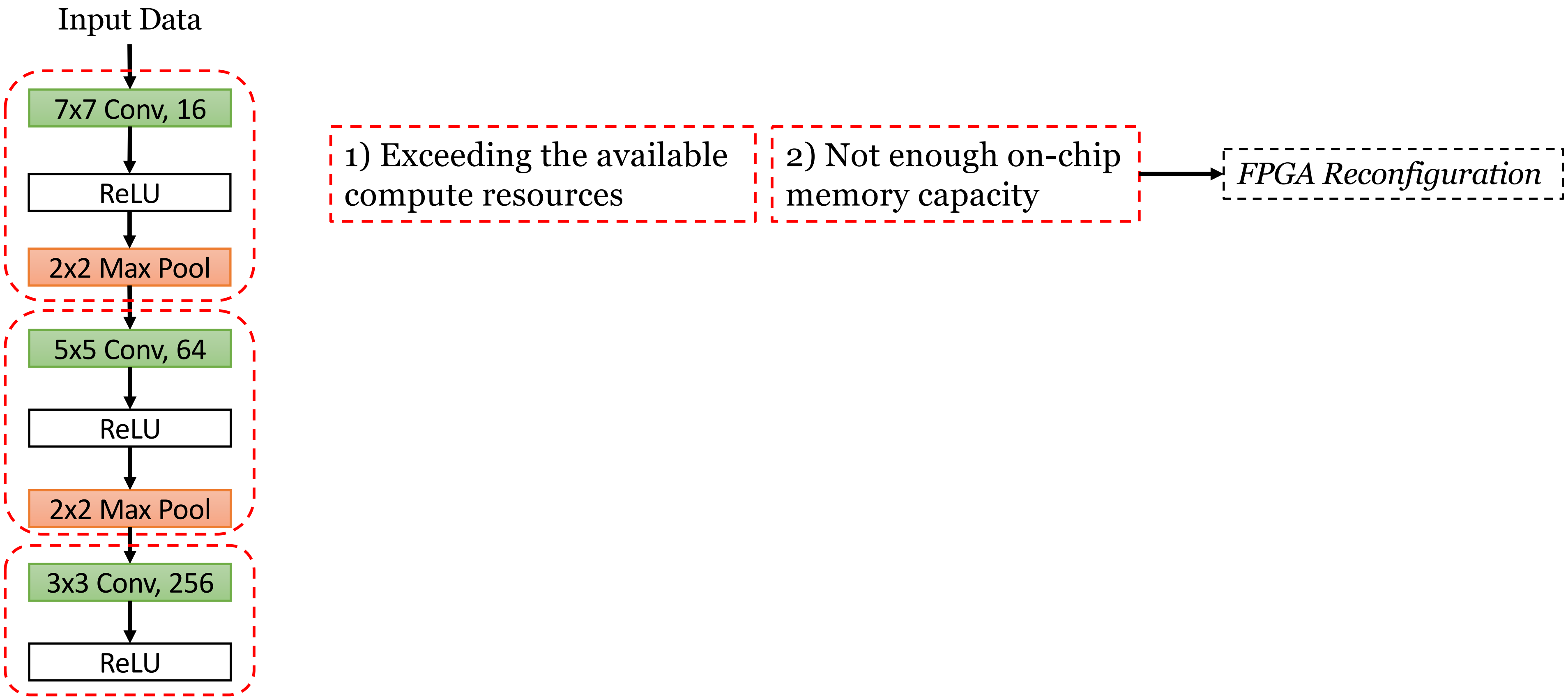
1) Exceeding the available compute resources

2) Not enough off-chip memory bandwidth

Transformation 1: Coarse-grained Folding



Transformation 3: Graph Partitioning with Reconfiguration



Input Data

7x7 Conv, 16

ReLU

2x2 Max Pool

5x5 Conv, 64

ReLU

2x2 Max Pool

3x3 Conv, 256

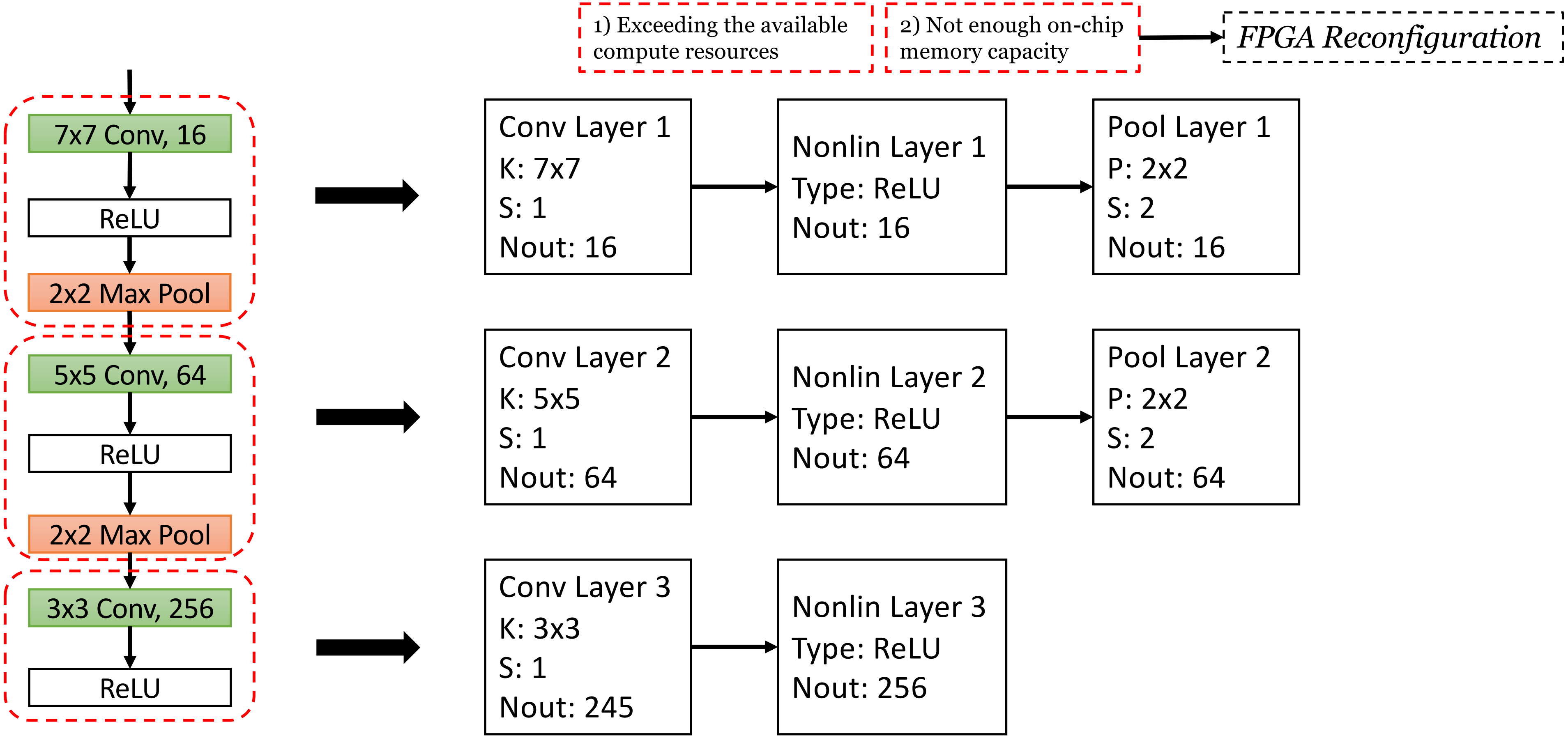
ReLU

1) Exceeding the available compute resources

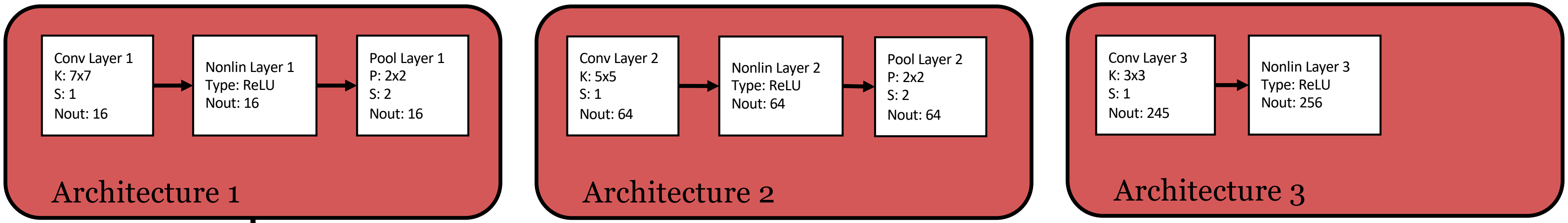
2) Not enough on-chip memory capacity

FPGA Reconfiguration

Transformation 3: Graph Partitioning with Reconfiguration

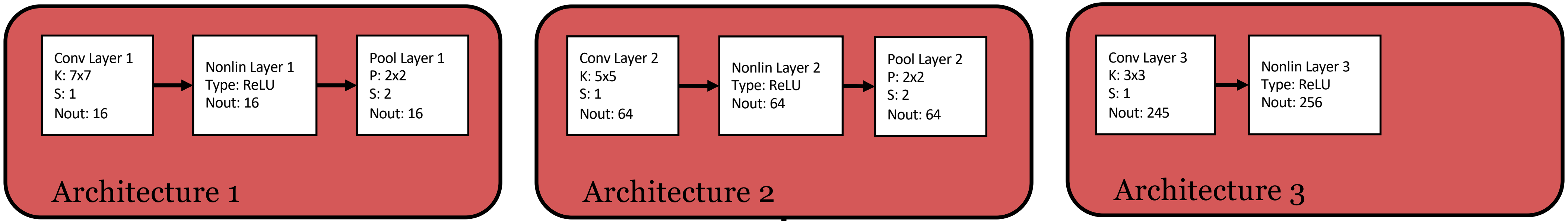


Transformation 3: Graph Partitioning with Reconfiguration

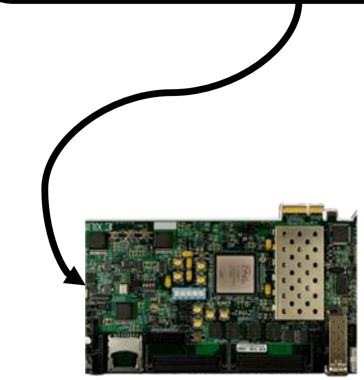


- Reconfigure FPGA
- Run network over batch
- Write-back to off-chip memory

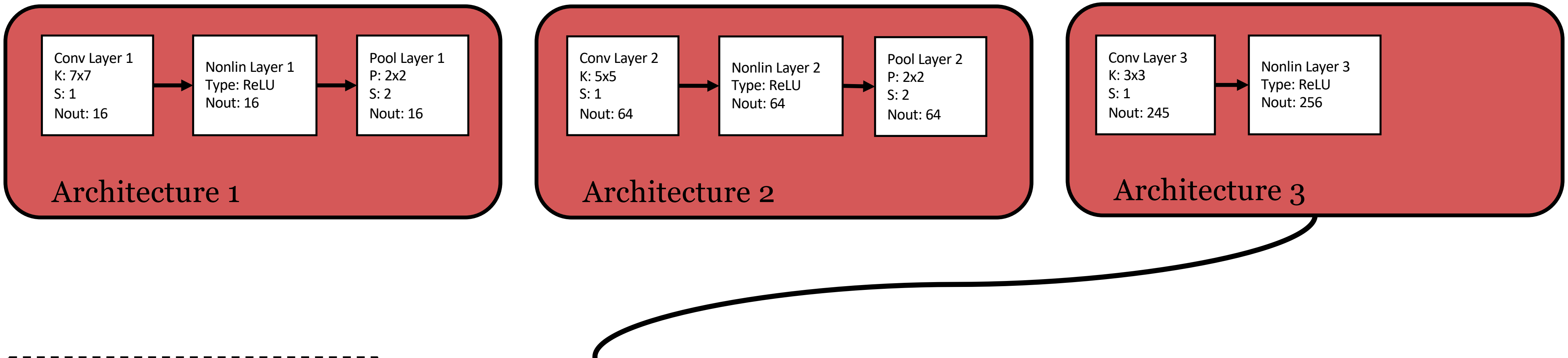
Transformation 3: Graph Partitioning with Reconfiguration



- Reconfigure FPGA
- Run network over batch
- Write-back to off-chip memory

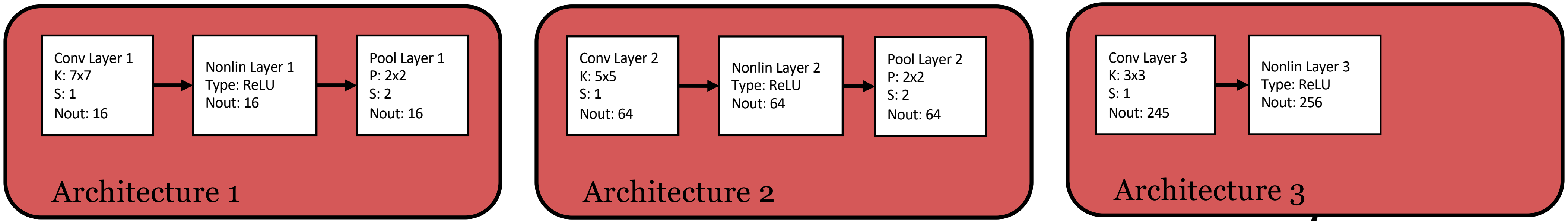


Transformation 3: Graph Partitioning with Reconfiguration



- Reconfigure FPGA
- Run network over batch
- Write-back to off-chip memory

Transformation 3: Graph Partitioning with Reconfiguration



Conv Layer 1
K: 7x7
S: 1
Nout: 16

Nonlin Layer 1
Type: ReLU
Nout: 16

Pool Layer 1
P: 2x2
S: 2
Nout: 16

Architecture 1

Conv Layer 2
K: 5x5
S: 1
Nout: 64

Nonlin Layer 2
Type: ReLU
Nout: 64

Pool Layer 2
P: 2x2
S: 2
Nout: 64

Architecture 2

Conv Layer 3
K: 3x3
S: 1
Nout: 245

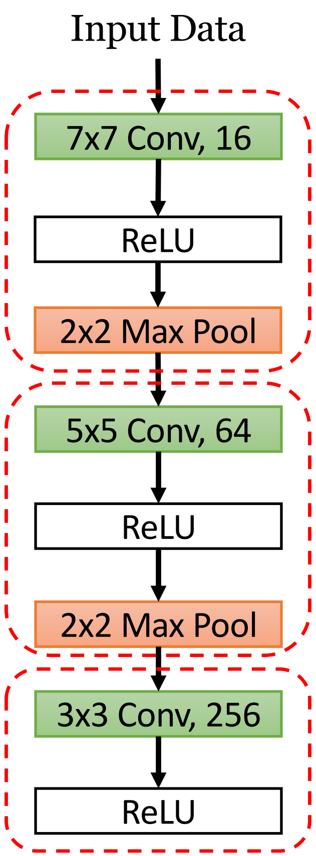
Nonlin Layer 3
Type: ReLU
Nout: 256

Architecture 3

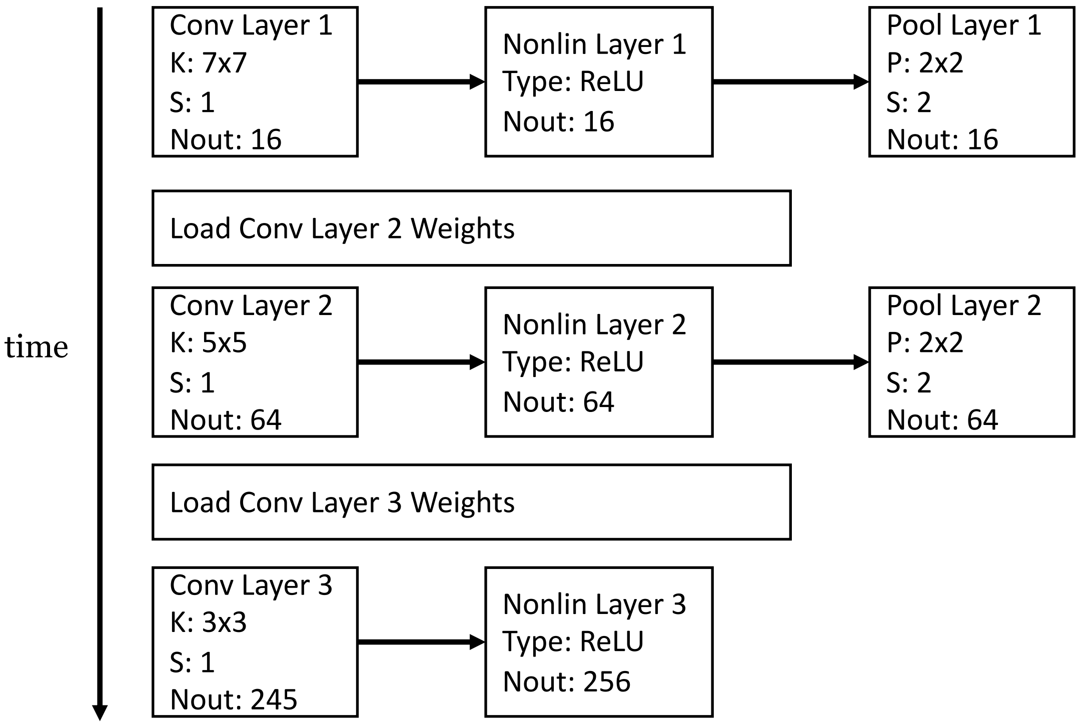
- Reconfigure FPGA
- Run network over batch
- Write-back to off-chip memory

- Batch processing amortises reconfiguration cost → high throughput
- Latency-sensitive applications?

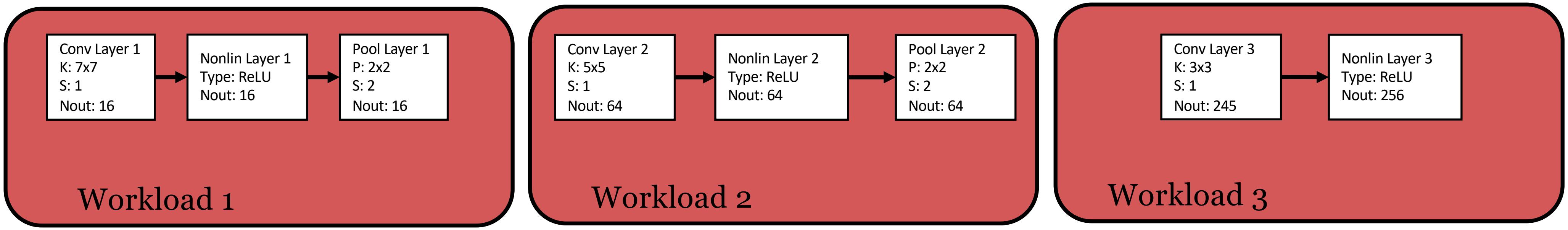
Transformation 4: Weights Reloading



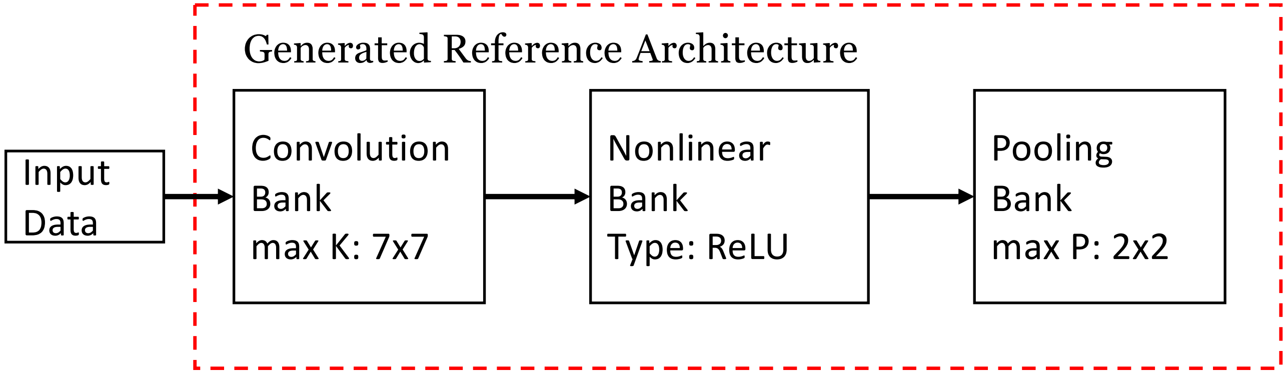
Run-time vs bitstream-level reconfiguration to explore the latency-throughput trade-off



Transformation 4: Weights Reloading



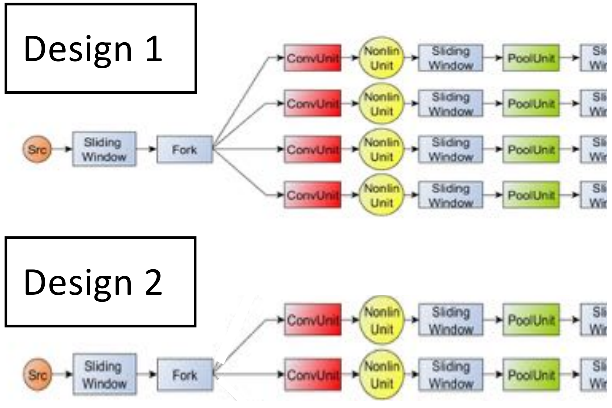
- Reload weights from off-chip memory and reconfigure datapath
- Run network over batch
- Write-back to off-chip memory



fpgaConvNet – Design Space Exploration and Optimisation

- SDF-based Framework
  - Capture hardware mappings as matrices
  - Transformations as *algebraic operations*
  - Any local transformation *propagates* through the network
  - *Static Scheduling*
  - Analytical *performance model*
  - Cast design space exploration as a multiobjective optimization problem

$$t_{total}(B, N_P, \mathbf{\Gamma}) = \sum_{i=1}^{N_P} t_i(B, \mathbf{\Gamma}_i) + (N_P - 1) \cdot t_{reconfig.}$$



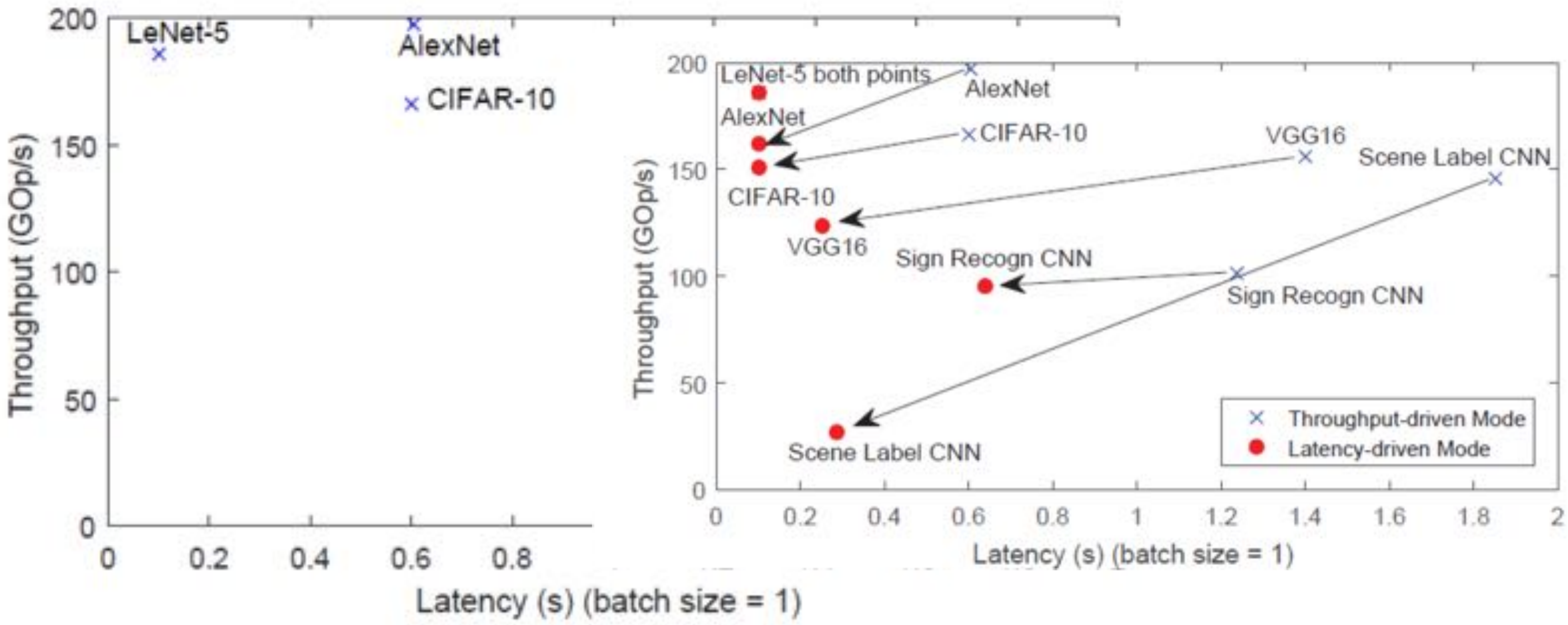
Hardware Stages

Interconnections

$$\mathbf{\Gamma}_1 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & K^2 & -K^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4K^2 & -4K^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4P^2 & -4P^2 \end{bmatrix}$$

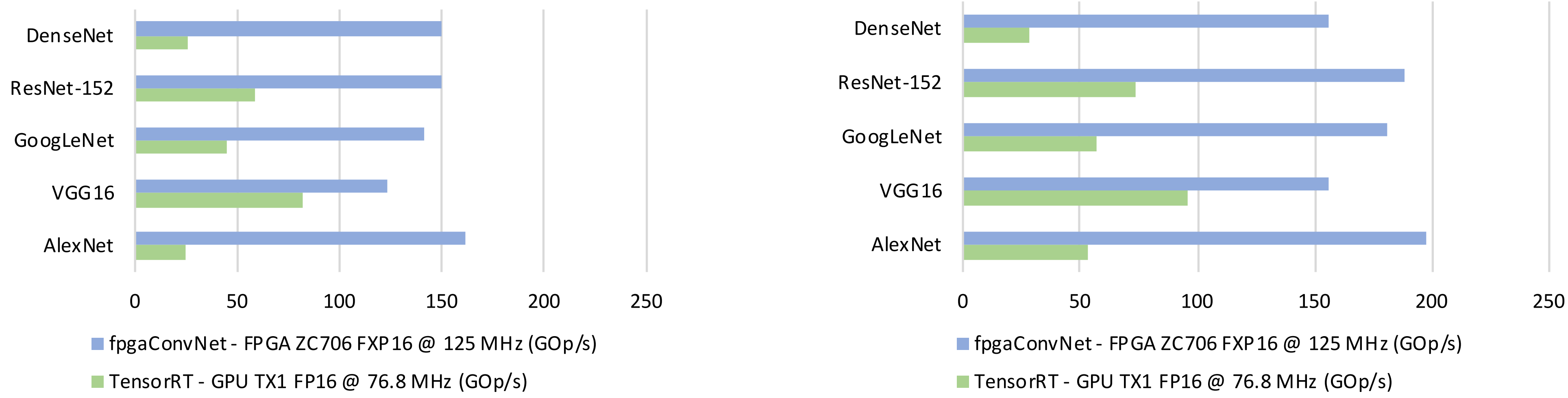
$$\mathbf{\Gamma}_2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & K^2 & -K^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{2K^2} & -2K^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2P^2 & -2P^2 \end{bmatrix}$$

Meeting the performance requirements



Comparison with Embedded GPUs: Same absolute power constraints (5W)

fpgaConvNet vs Embedded GPU (GOp/s) for the same absolute power constraints (5W)

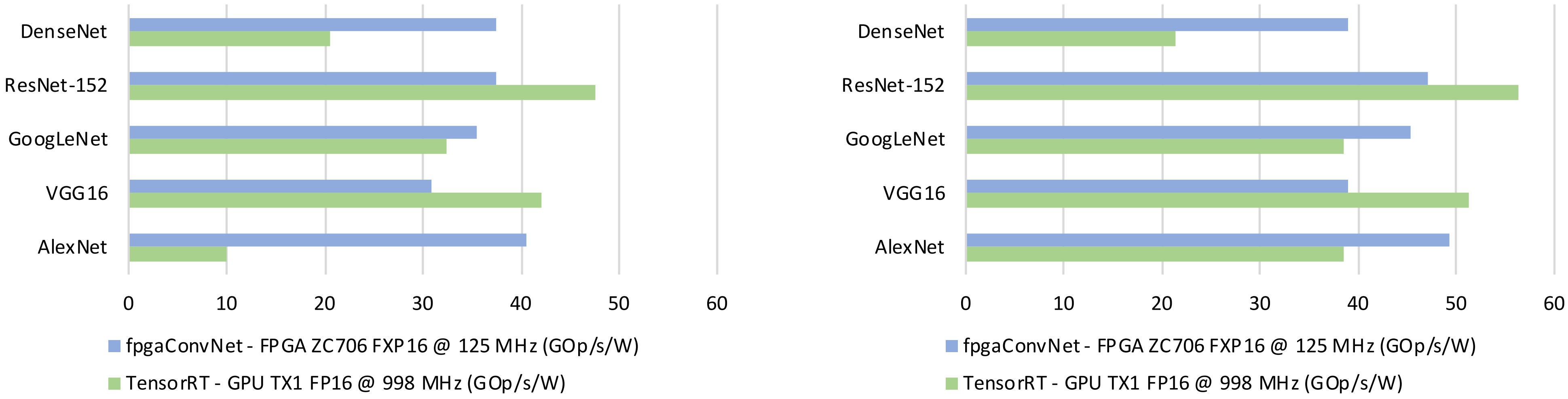


- Latency-driven scenario → batch size of 1
- Up to 6.65× speedup with an average of 3.95× (3.43× geo. mean)

- Throughput-driven scenario → favourable batch size
- Up to 5.53× speedup with an average of 3.32× (3.07× geo. mean)

Comparison with Embedded GPUs: Performance-per-Watt

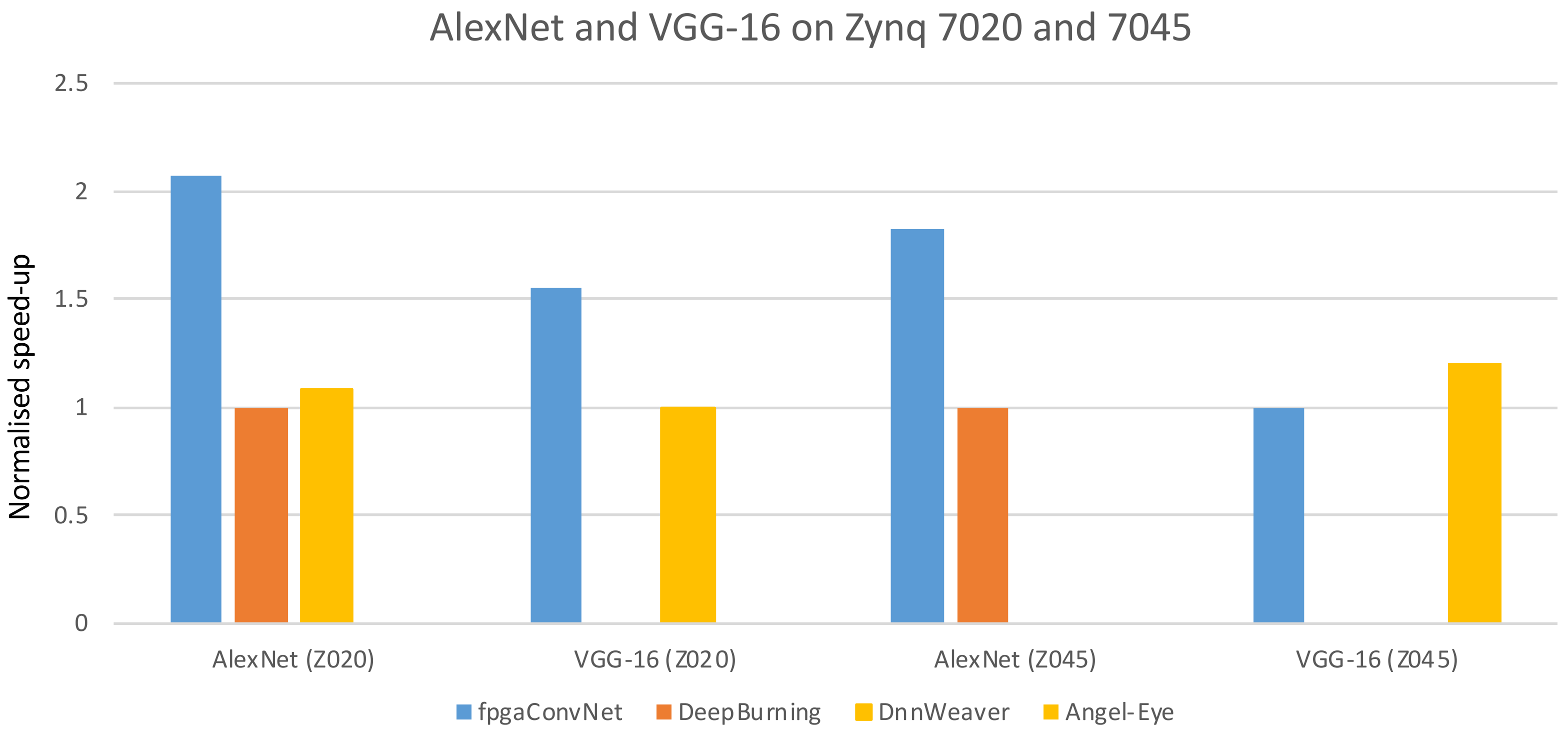
fpgaConvNet vs Embedded GPU (GOp/s/W)



- Latency-driven scenario → batch size of 1
- Average of 1.70× (1.36× geo. mean) in GOp/s/W

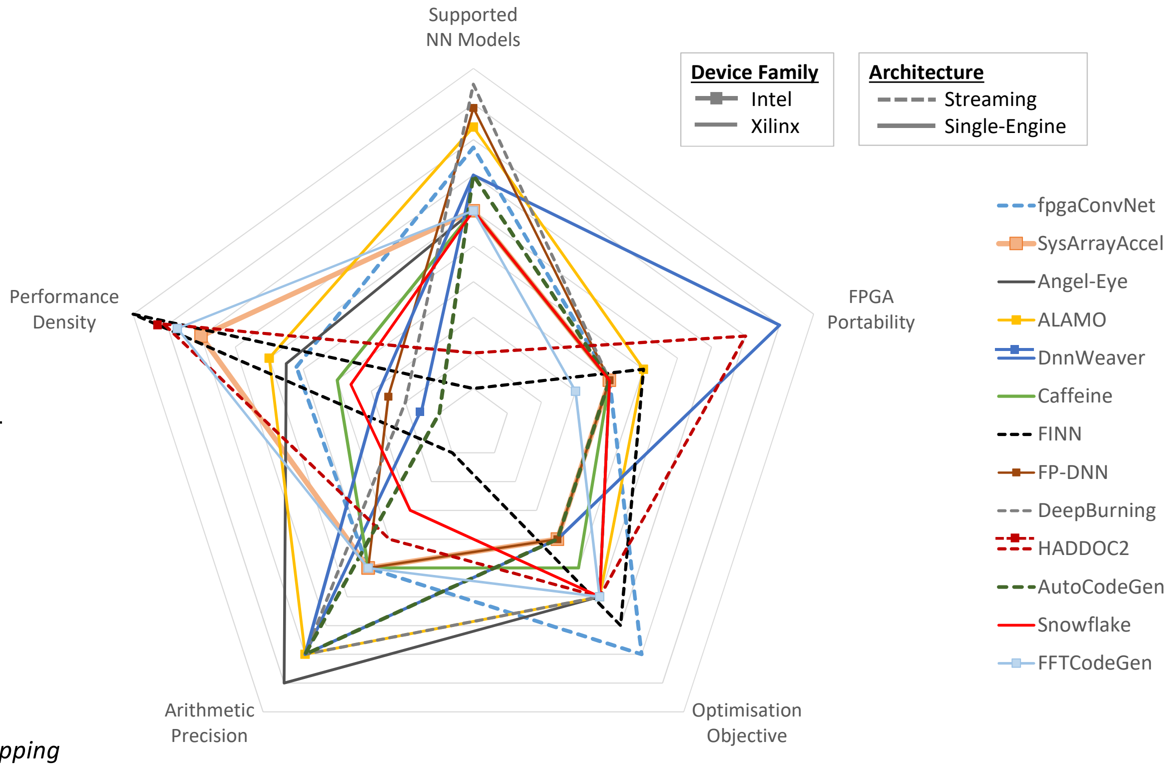
- Throughput-driven scenario → favourable batch size
- Average of 1.17× (1.12× geo. mean) in GOp/s/W

Results: Comparison with existed FPGA frameworks



Other approaches

Toolflow Name	Interface	Year
fpgaConvNet [86][87][88][85]	Caffe & Torch	May 2016
DeepBurning [90]	Caffe	June 2016
Angel-Eye [68][23][24]	Caffe	July 2016
ALAMO [58][56][57][55][59]	Caffe	August 2016
HADDOC2 [1][2]	Caffe	September 2016
DNNWEAVER [75][76]	Caffe	October 2016
Caffeine [98]	Caffe	November 2016
AutoCodeGen [54]	Proprietary Input Format	December 2016
FINN [84][19]	Theano	February 2017
FP-DNN [22]	TensorFlow	May 2017
Snowflake [21][10]	Torch	May 2017
SysArrayAccel [91]	C Program	June 2017
FFTCodeGen [100][97][96][95]	Proprietary Input Format	December 2017

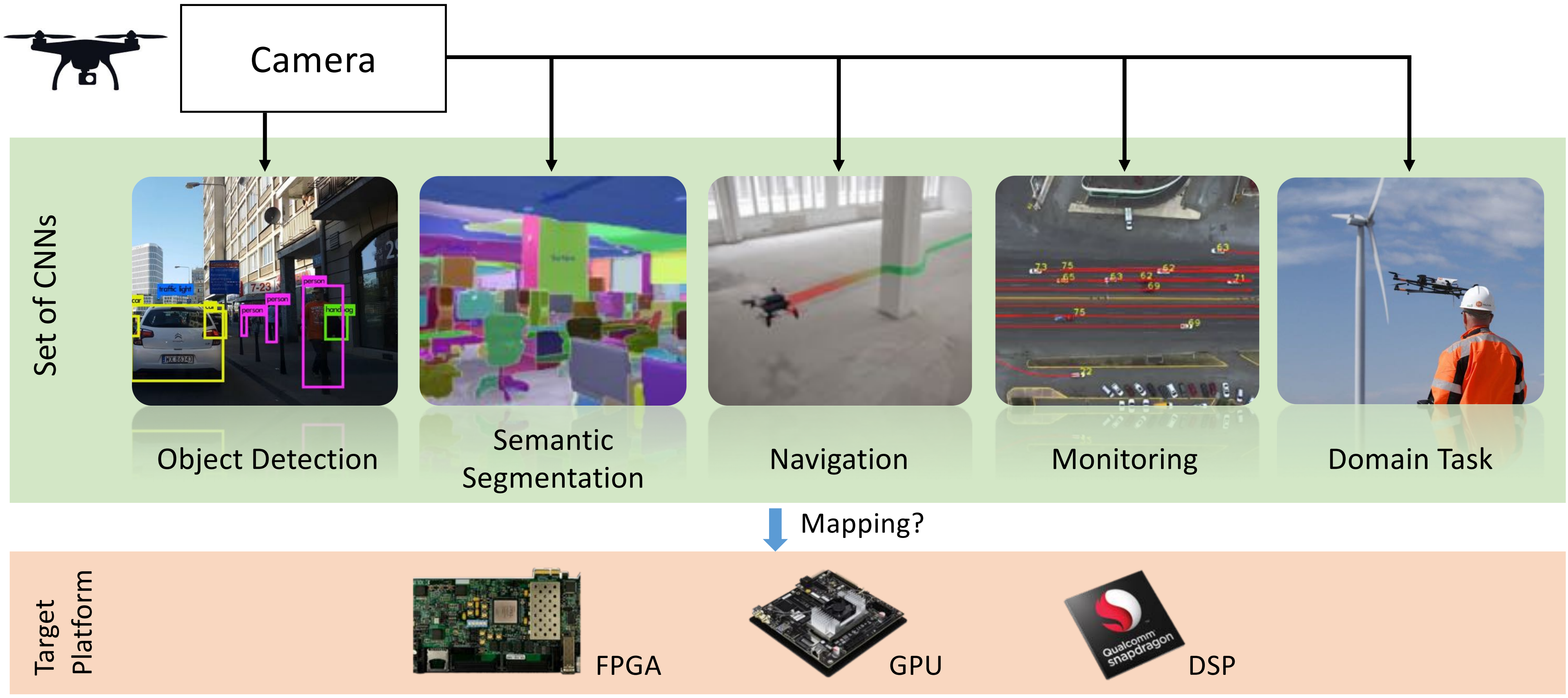


Stylianos I. Venieris, Alexandros Kouris and Christos-Savvas Bouganis, "*Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions*", ACM Computing Surveys, 2018

Imperial College
London

Challenge #2: Multi-CNN Systems

Challenge #2: Multi-CNN Systems – Autonomous Drones



Camera

Set of CNNs

Object Detection

Semantic Segmentation

Navigation

Monitoring

Domain Task

Mapping?

Target Platform

FPGA

GPU

DSP

The Problem setting and Challenges

Given a number of CNNs:

CNN_1, CNN_2, …, CNN_N

find a mapping to an FPGA device that meets user requirements such as Latency and Throughout per CNN

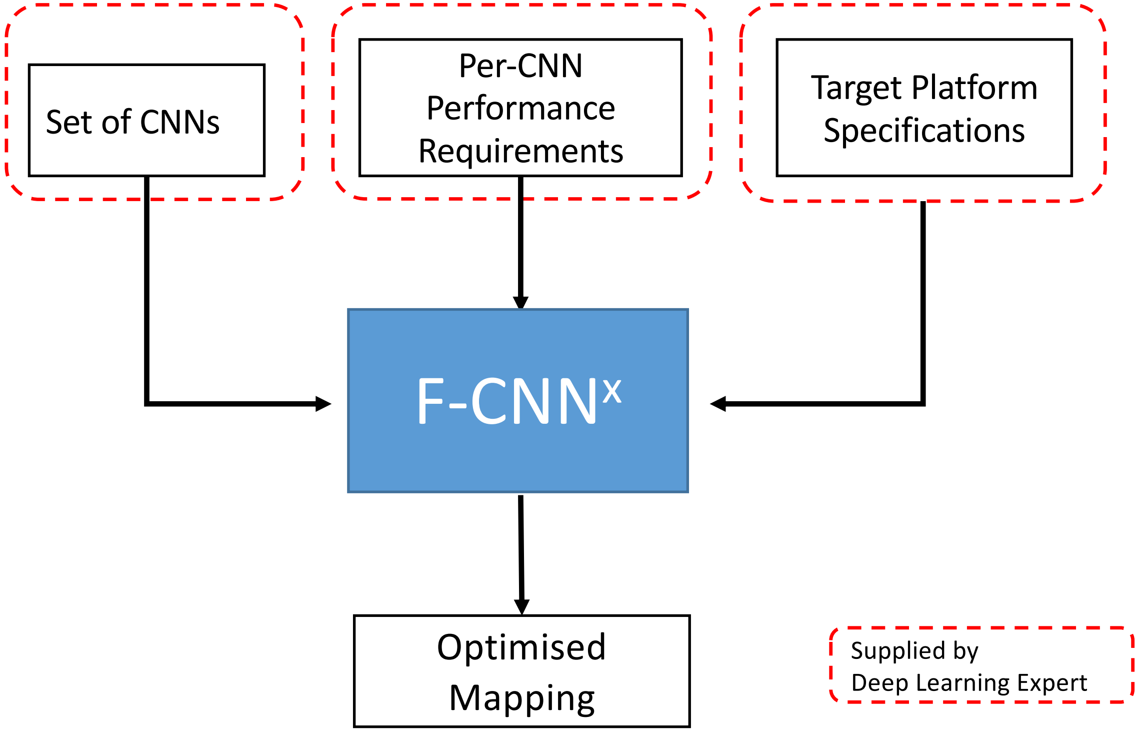
(Extra) Challenges:

- Resource allocation per CNN
- Memory Bandwidth allocation per CNN
- Scalability

Challenge #2: Multi-DNN System

Challenges:

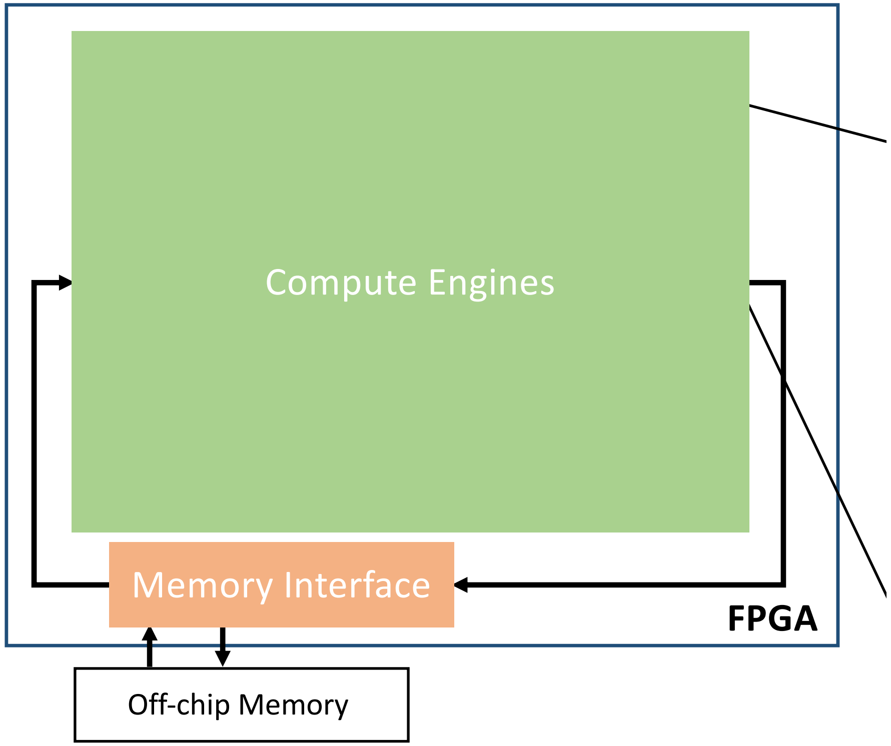
- Resource allocation among CNNs
- Design automation
- Models with different performance constraints, e.g. required throughput and latency
- Competing for the same pool of resources
- High-dimensional design space


Set of CNNs
Per-CNN Performance Requirements
Target Platform Specifications
F-CNN^x
Optimised Mapping
Supplied by Deep Learning Expert

Multi-CNN Hardware Architecture

Key characteristics

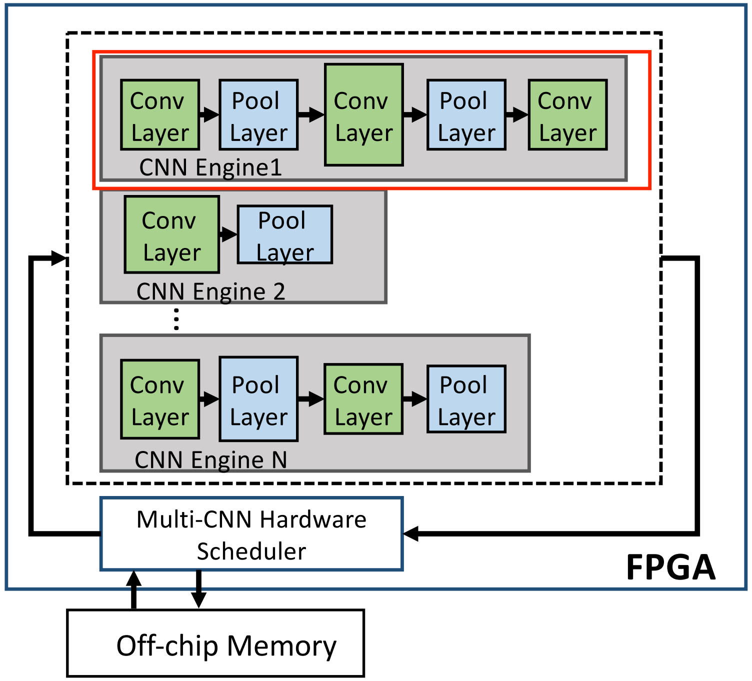
- One hardware engine per CNN – highly customisable
- Hardware scheduler to control memory access schedule



Multi-CNN Hardware Architecture

Key characteristics

- One hardware engine per CNN – highly customisable
- Hardware scheduler to control memory access schedule

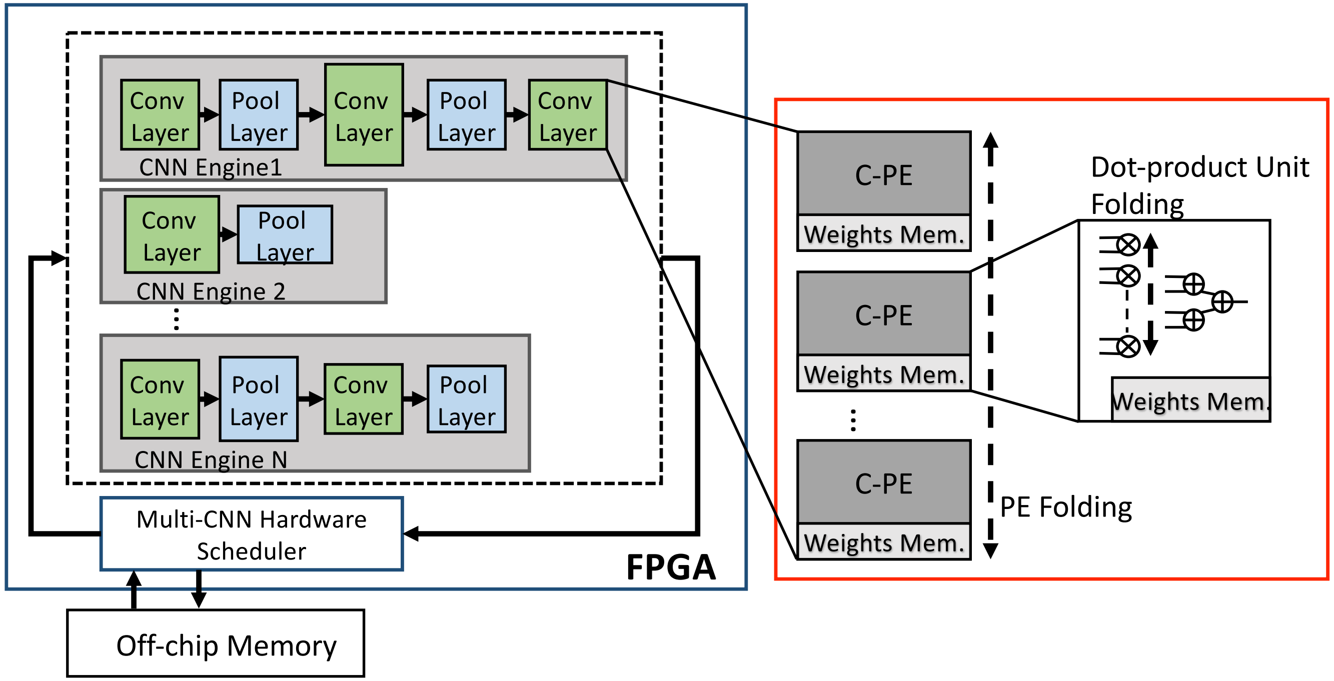


Parameter	Symbol
Pipeline structure	Γ_i

Multi-CNN Hardware Architecture

Key characteristics

- One hardware engine per CNN – highly customisable
- Hardware scheduler to control memory access schedule

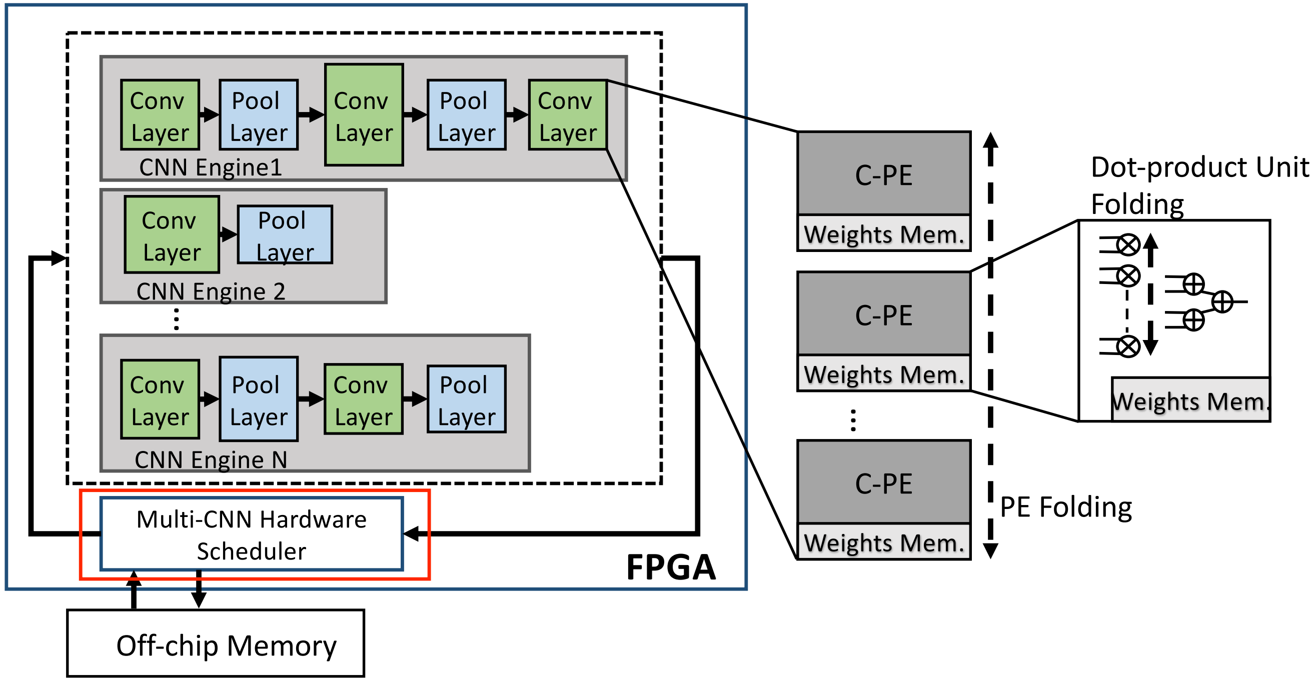


Parameter	Symbol
Pipeline structure	Γ_i
No. of PEs in each stage	$N_{\mathrm{PE},i,j}$
No of MAC operators within each PE	$N_{\mathrm{op},i,j}$

Multi-CNN Hardware Architecture

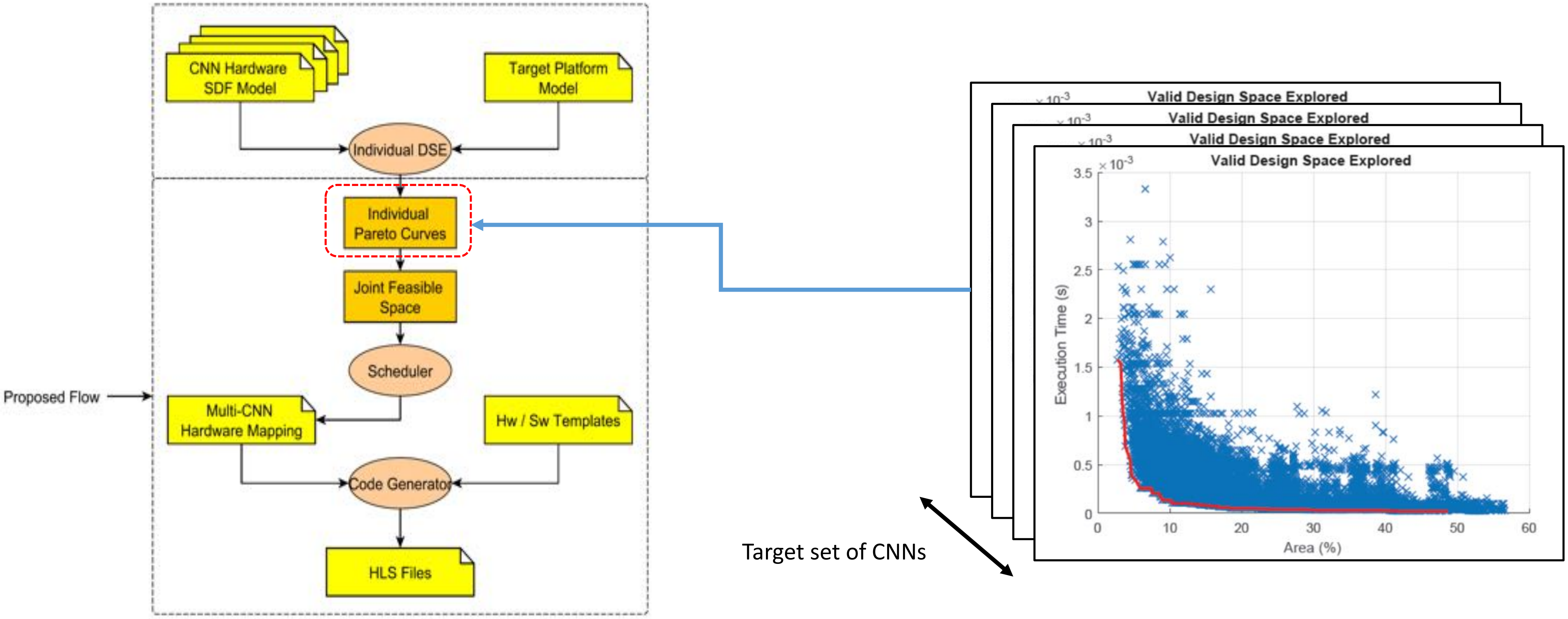
Key characteristics

- One hardware engine per CNN – highly customisable
- Hardware scheduler to control memory access schedule

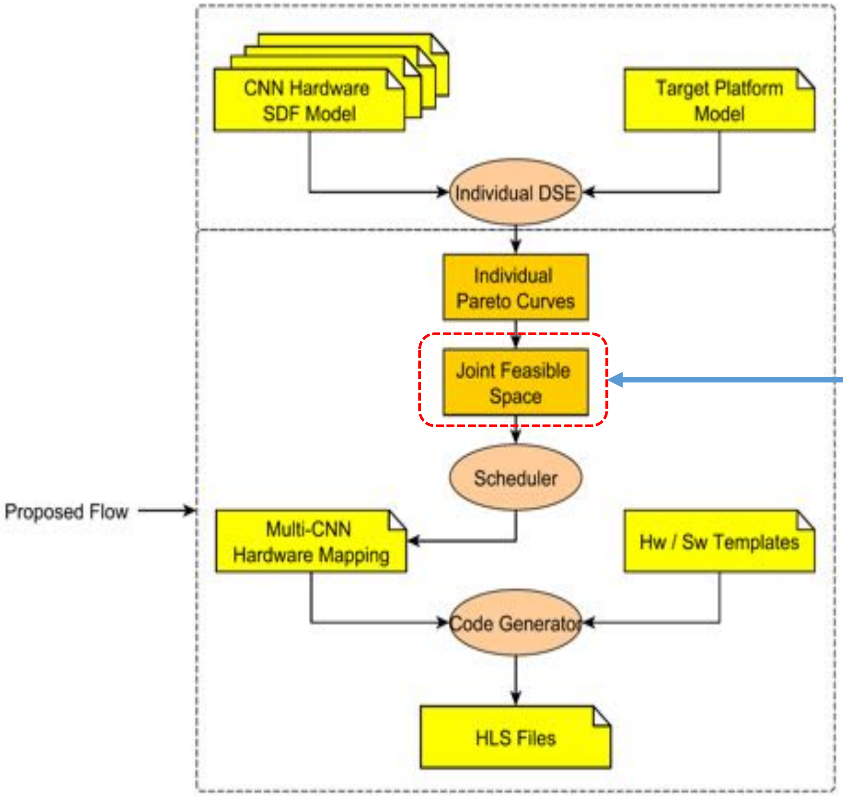


Parameter	Symbol
Pipeline structure	Γ_i
No. of PEs in each stage	$N_{\text{PE},i,j}$
No of MAC operators within each PE	$N_{\text{op},i,j}$
Schedule	S

Proposed Design Space Exploration Method

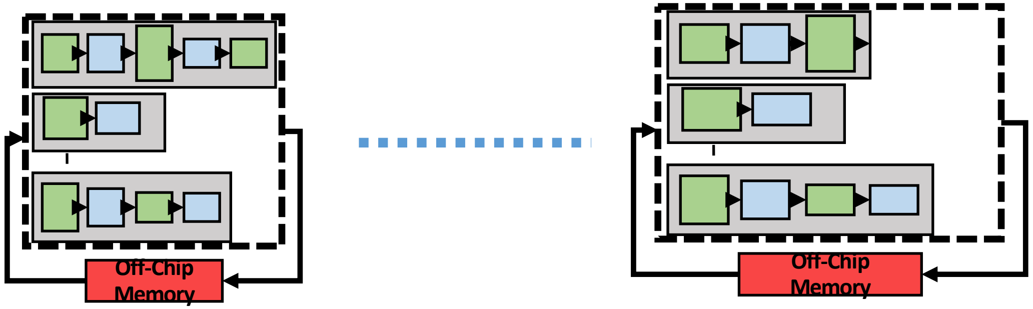


Proposed Design Space Exploration Method

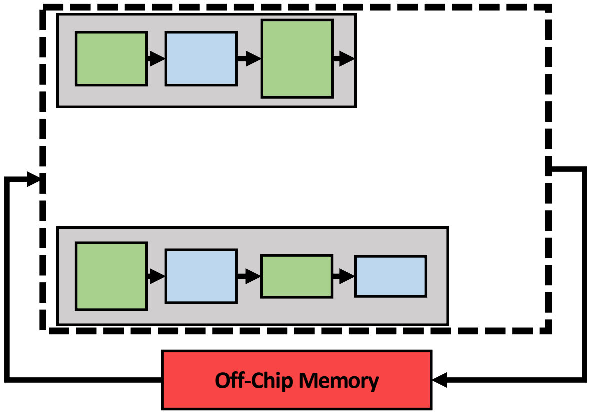


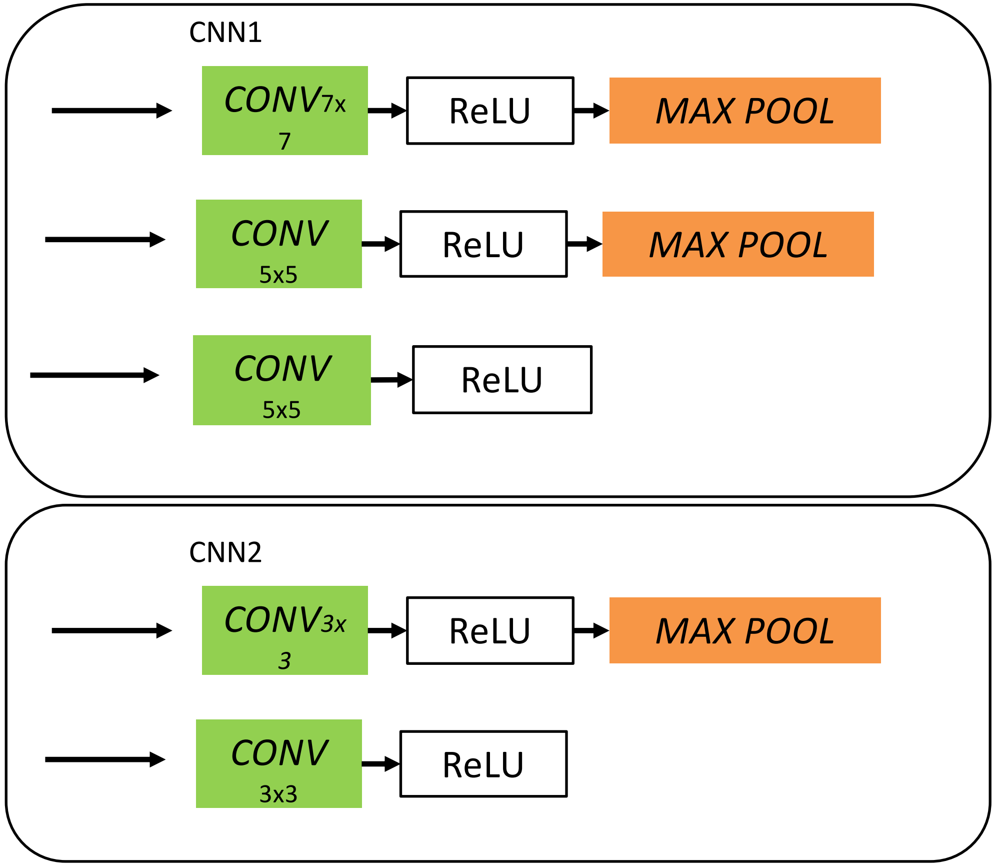
On-chip Resource Feasibility

$$\sum_{i=1}^{N} \boldsymbol{rsc}(\sigma_i) \leq \boldsymbol{rsc}_{Avail.}$$



An example



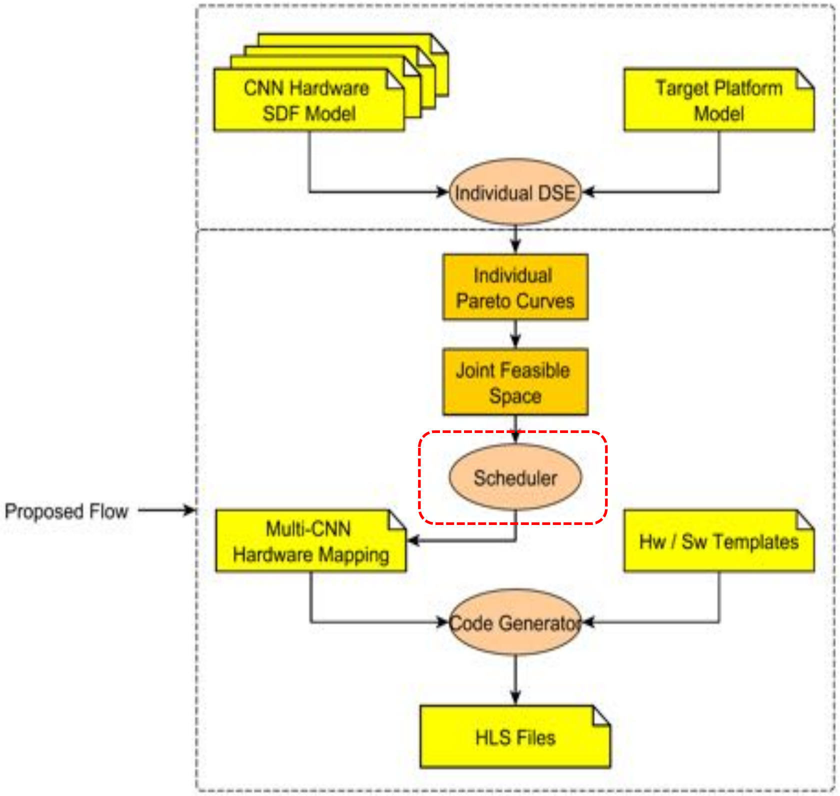


For each CNN

- A set of subgraphs
- Bandwidth requirements

Possible memory contention

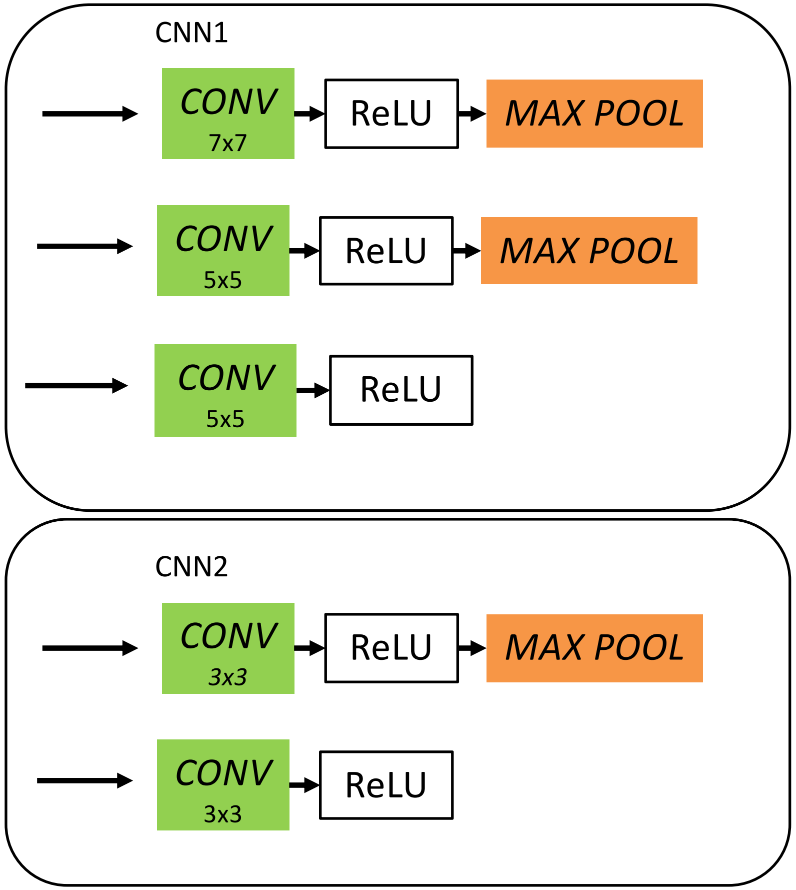
Proposed Design Space Exploration Method



- Memory contention
  - Problem 1: Performance model =! Actual performance (scheduler)
  - Problem 2: Not full utilization of the memory bandwidth

- CNN inference over a stream of inputs
  - Cast to a **cyclic scheduling problem**
  - Search for a periodic solution
- Optimal ILP scheduler has very high runtimes for large-sized problems
- We propose a heuristic Resource Constrained List Scheduler (RCLS).

Slow-down Scheduler

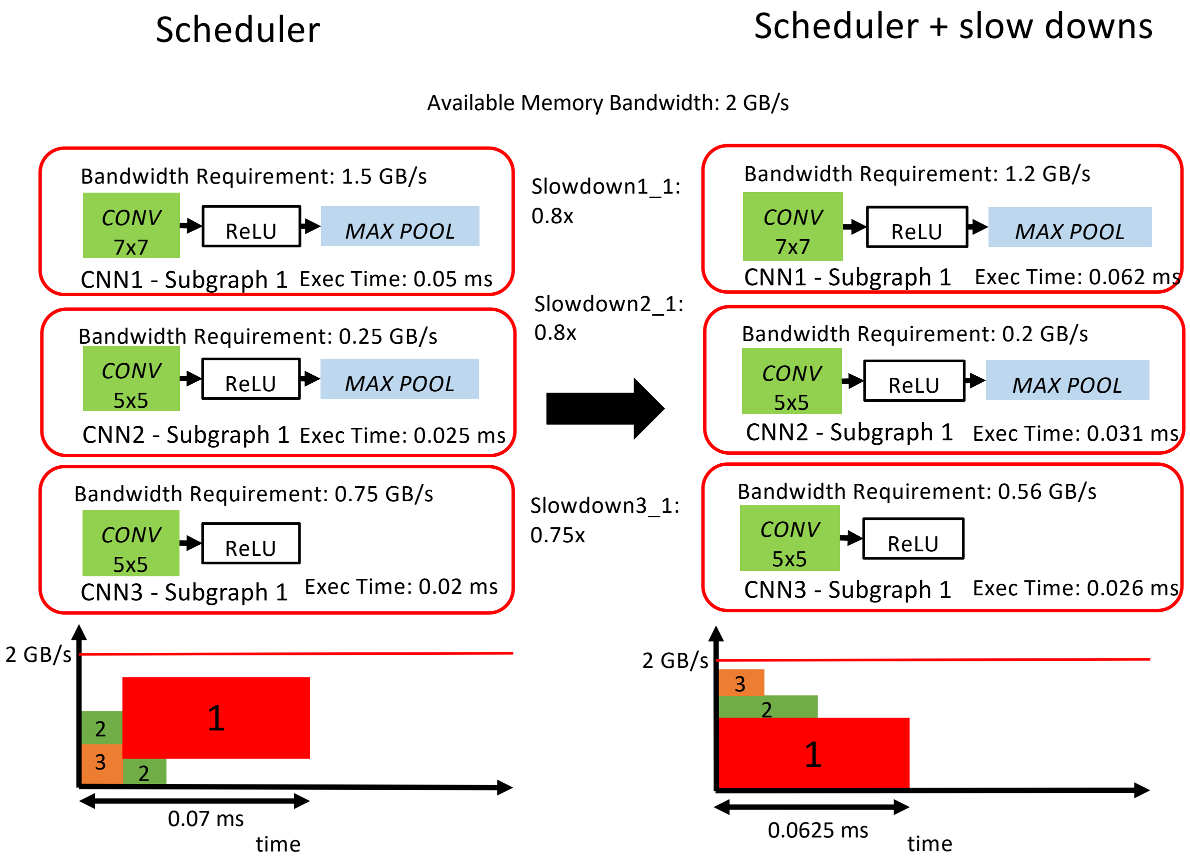


- Increase the latency and decrease the bandwidth proportionally
- One slow-down factor per subgraph

$$L'(s_{i,j}) = \frac{1}{sl_{i,j}} \times L(s_{i,j})$$ Latency Increase

$$b'(s_{i,j}) = sl_{i,j} \times b(s_{i,j})$$ Bandwidth Decrease

The effect of slow-downs



Scheduler

Scheduler + slow downs

Available Memory Bandwidth: 2 GB/s

Bandwidth Requirement: 1.5 GB/s

CONV 7x7 → ReLU → MAX POOL

CNN1 - Subgraph 1 Exec Time: 0.05 ms

Bandwidth Requirement: 0.25 GB/s

CONV 5x5 → ReLU → MAX POOL

CNN2 - Subgraph 1 Exec Time: 0.025 ms

Bandwidth Requirement: 0.75 GB/s

CONV 5x5 → ReLU

CNN3 - Subgraph 1 Exec Time: 0.02 ms

Slowdown1_1: 0.8x

Slowdown2_1: 0.8x

Slowdown3_1: 0.75x

Bandwidth Requirement: 1.2 GB/s

CONV 7x7 → ReLU → MAX POOL

CNN1 - Subgraph 1 Exec Time: 0.062 ms

Bandwidth Requirement: 0.2 GB/s

CONV 5x5 → ReLU → MAX POOL

CNN2 - Subgraph 1 Exec Time: 0.031 ms

Bandwidth Requirement: 0.56 GB/s

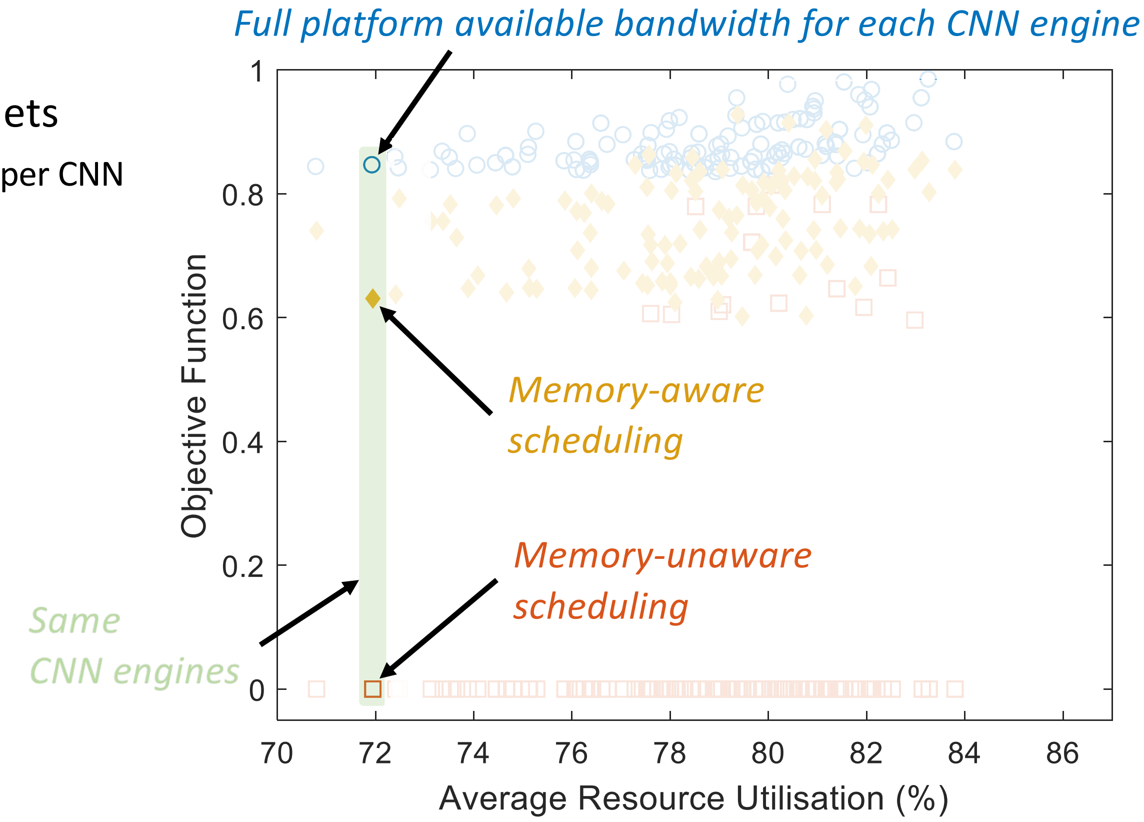
CONV 5x5 → ReLU

CNN3 - Subgraph 1 Exec Time: 0.026 ms

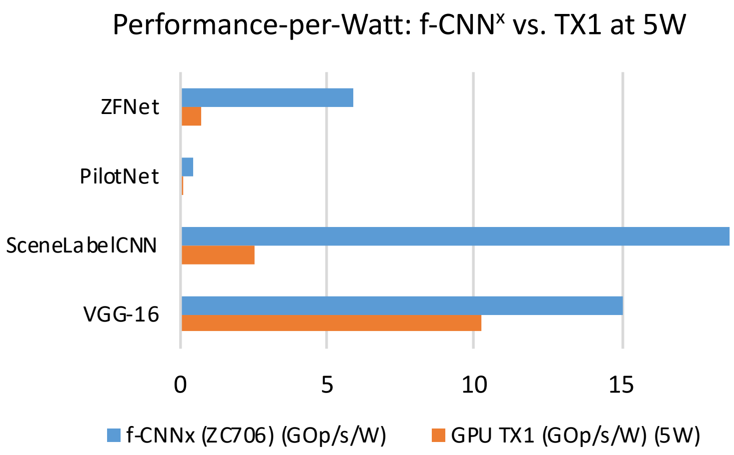
Effect of the Proposed DSE

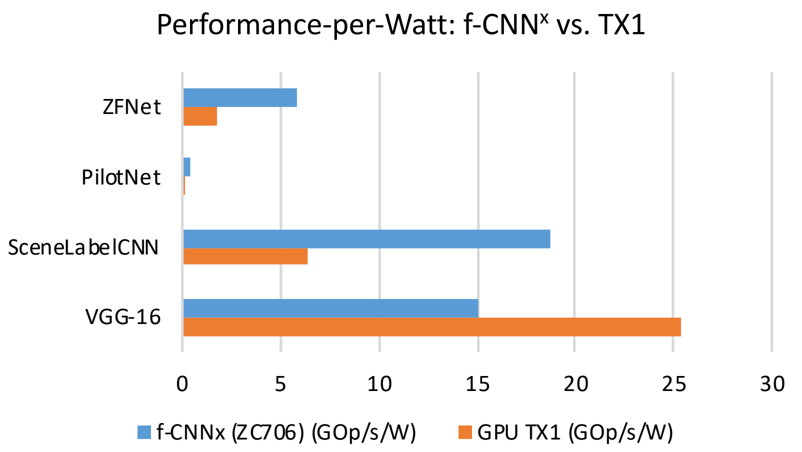
- 3-CNN benchmark on ZC706

- Explored joint design points appear in triplets
  - Blue → peak platform-supported performance per CNN
  - Red → contention-unaware design
  - Yellow → memory-aware design



Comparison with Embedded GPUs





- Latency-driven scenario → batch size of 1
- Up to 19.09× speedup with an average of 6.85× (geo. mean)

- Latency-driven scenario → batch size of 1
- Up to 9.61× speedup with an average of 2.76× (geo. mean)

Conclusions

- Performance (efficiency) comes from customisation

- ML applications:
  - Fast moving area => new computational blocks appear frequently
  - Diverse application areas (ADAS, drones, Video analytics)

- To improve hardware's efficiency

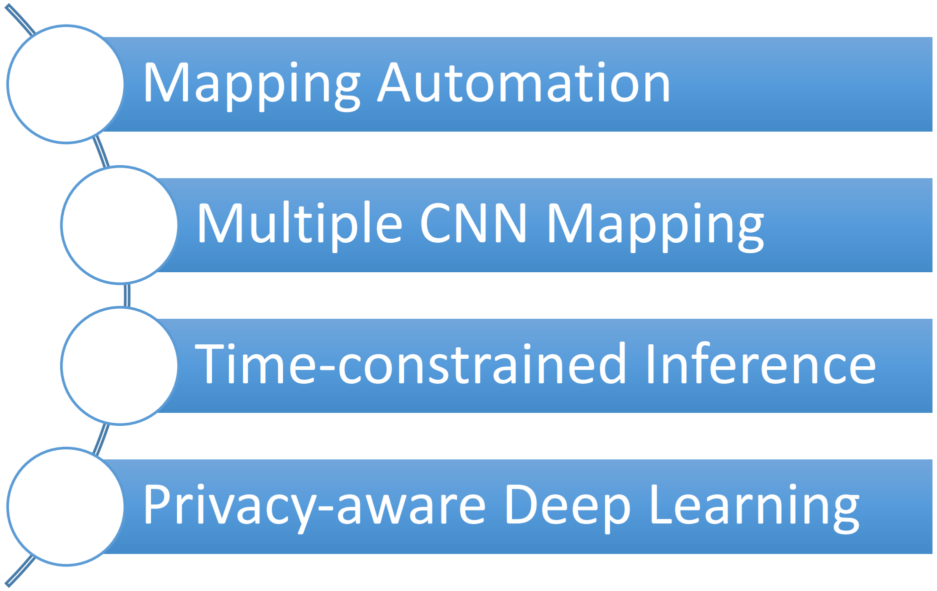
  => highly customisable architecture

  => large design space

- Need for Tools

Summary

Research topics



- Mapping Automation
- Multiple CNN Mapping
- Time-constrained Inference
- Privacy-aware Deep Learning

A. Kouris and C-S Bouganis, "Learning to Fly by MySelf: A Self-Supervised CNN-based Approach for Autonomous Navigation",   IROS, 2018

Publications

- Alexandros Kouris, Stylianos I. Venieris, and Christos-Savvas Bouganis. 2018. ***CascadeCNN: Pushing the performance limits of quantisation.*** *In SysML.*
- Alexandros Kouris, Stylianos I. Venieris, and Christos-Savvas Bouganis. 2018. ***CascadeCNN: Pushing the Performance Limits of Quantisation in Convolutional Neural Networks***. *In 2018 28th International Conference on Field Programmable Logic and Applications (FPL).*
- C. Kyrkou, G. Plastiras, T. Theocharides, S. I. Venieris, and C. S. Bouganis. 2018. ***DroNet: Efficient Convolutional Neural Network Detector for Real-Time UAV Applications.*** *In 2018 Design, Automation Test in Europe Conference Exhibition (DATE). 967–972.*
- Michalis Rizakis, Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. 2018. ***Approximate FPGA-based LSTMs under Computation Time Constraints.*** *In Applied Reconfigurable Computing - 14th International Symposium, ARC 2018, Santorini, Greece, May 2 - 4, 2018, 3–15.*
- Stylianos I. Venieris and Christos-Savvas Bouganis. 2016. ***fpgaConvNet: A Framework for Mapping Convolutional Neural Networks on FPGAs.*** *In 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). 40–47.*
- Stylianos I. Venieris and Christos-Savvas Bouganis. 2017. ***fpgaConvNet: A Toolflow for Mapping Diverse Convolutional Neural Networks on Embedded FPGAs.*** *In NIPS 2017 Workshop on Machine Learning on the Phone and other Consumer Devices.*
- Stylianos I. Venieris and Christos-Savvas Bouganis. 2017. ***fpgaConvNet: Automated Mapping of Convolutional Neural Networks on FPGAs*** (Abstract Only). *In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 291–292.*
- S. I. Venieris and C. S. Bouganis. 2017. ***Latency-Driven Design for FPGA-based Convolutional Neural Networks.*** *In 2017 27th International Conference on Field Programmable Logic and Applications (FPL).*
- S. I. Venieris and C. S. Bouganis. 2018. ***f-CNNx: A Toolflow for Mapping Multiple Convolutional Neural Networks on FPGAs.*** *In 2018 28th International Conference on Field Programmable Logic and Applications (FPL).*
- Stylianos I. Venieris, Alexandros Kouris, and Christos-Savvas Bouganis. 2018. ***Toolflows for Mapping Convolutional Neural Networks on FPGAs: A Survey and Future Directions***. *In ACM Computing Surveys 51, 3, Article 56 (June 2018), 39 pages.*