Imperial College London

Efficient Deployment of CNNs under Resource Constraints

Christos-Savvas Bouganis

christos-savvas.bouganis@imperial.ac.uk

Our vision

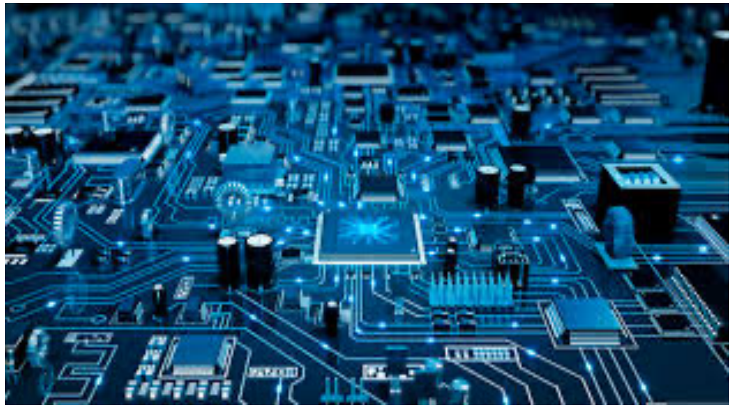
To research and develop intelligent autonomous systems

“see”

+

“understand”

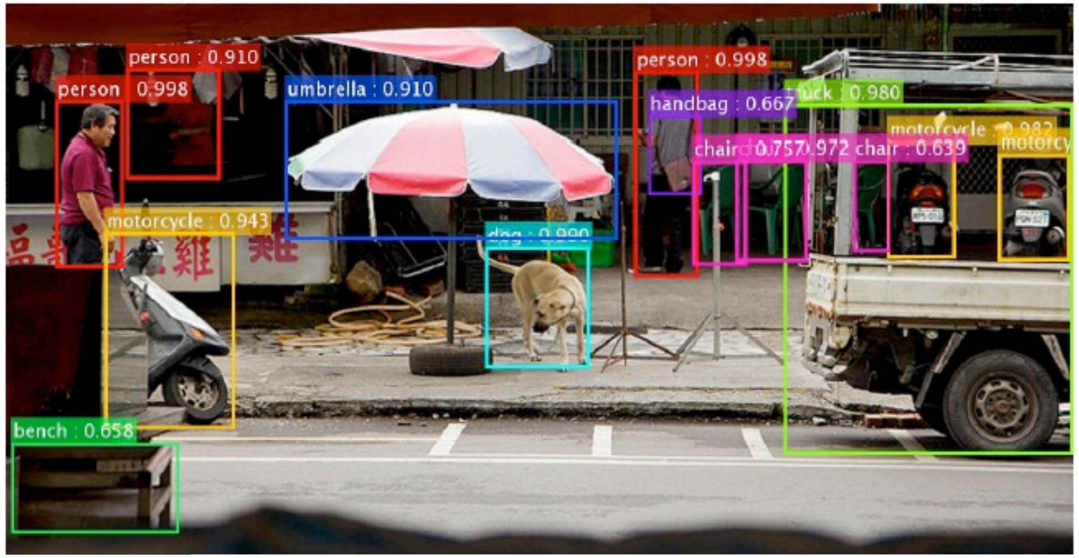
+

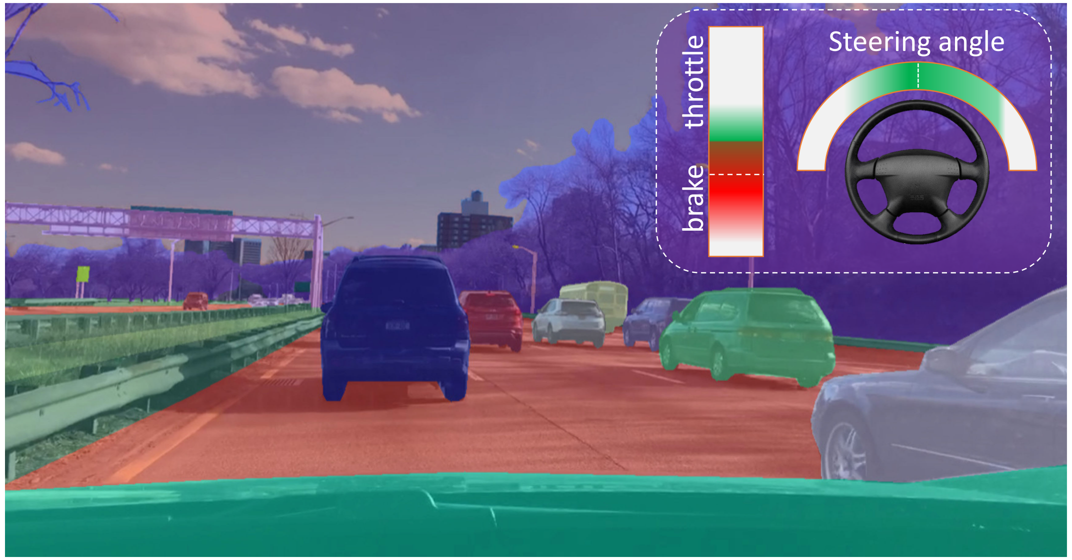


“process”

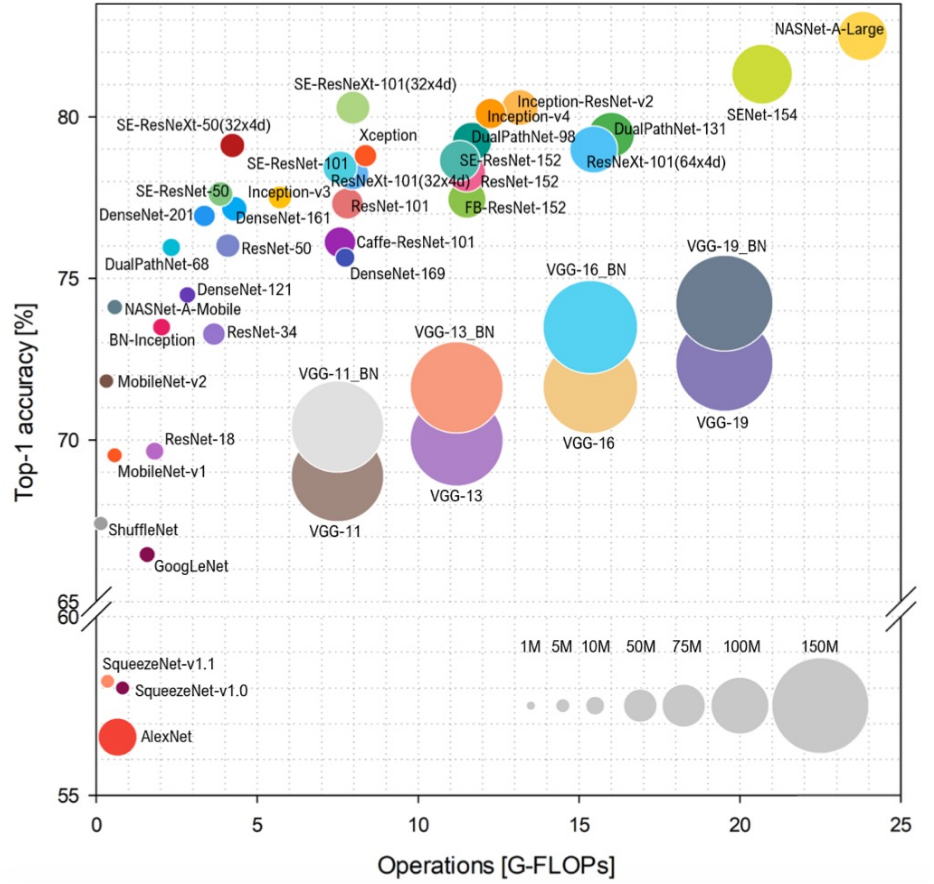
Challenges: *high performance, low power, limited resources*

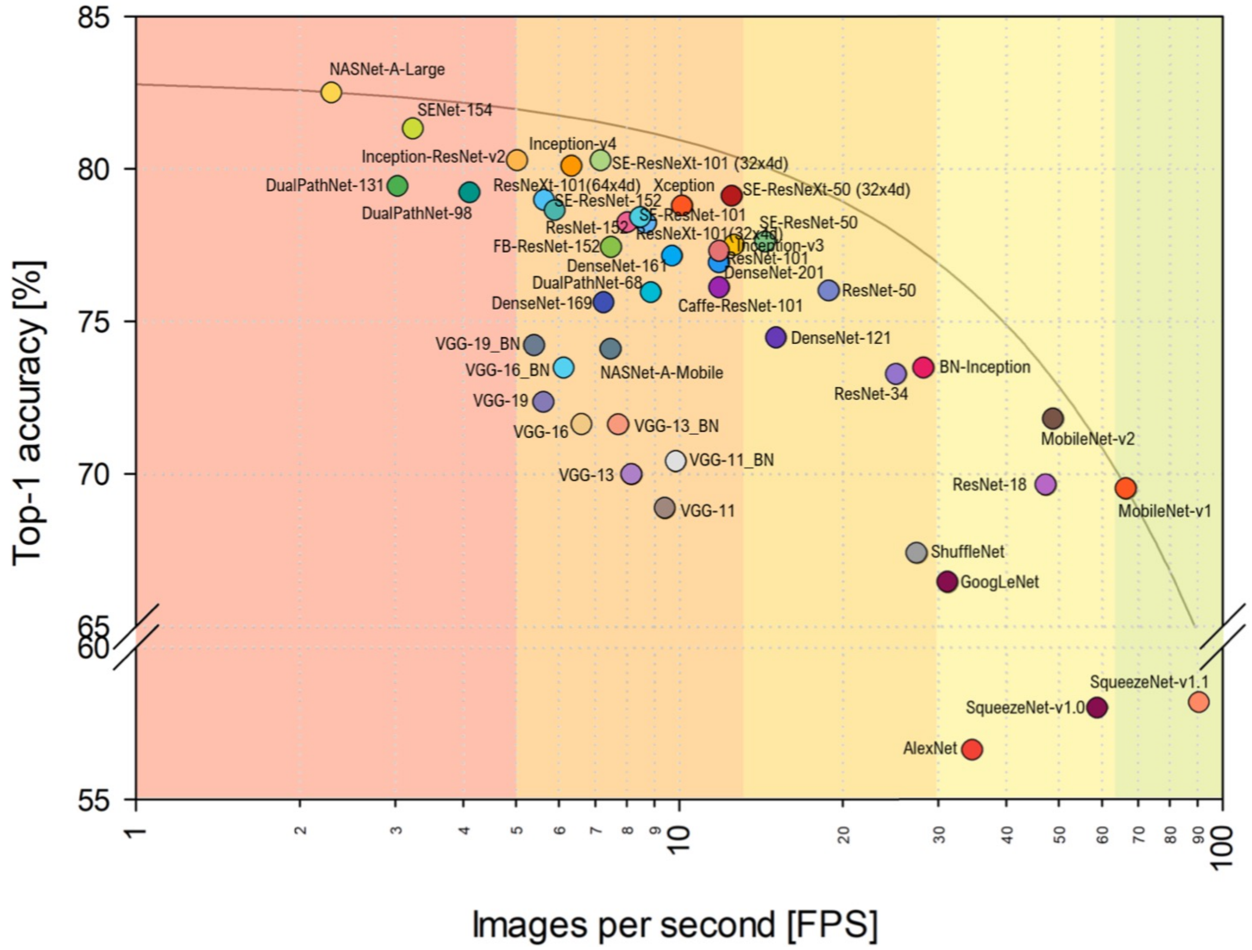
Many Machine Learning success stories





Evolution of ML classification models

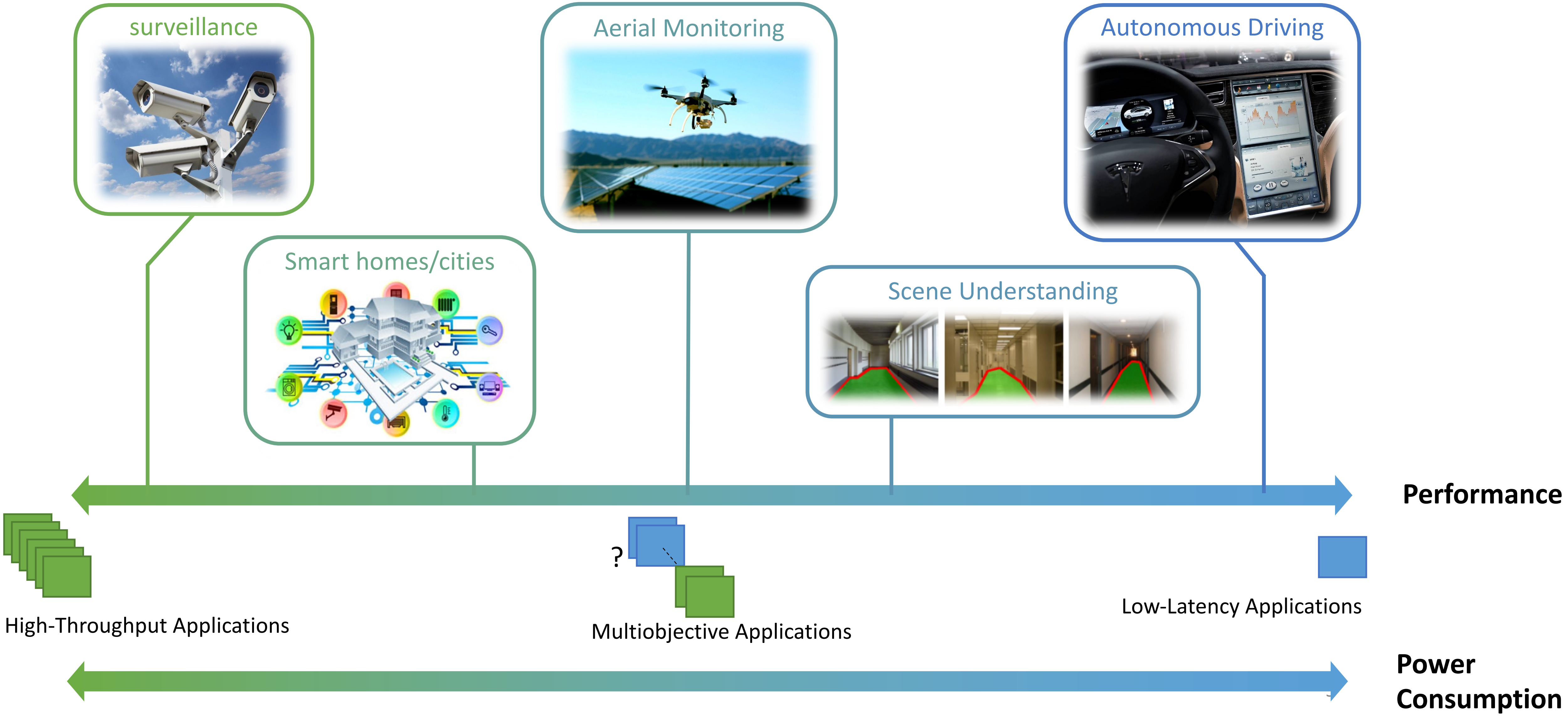




Observation: A fast evolving Pareto front that requires tools

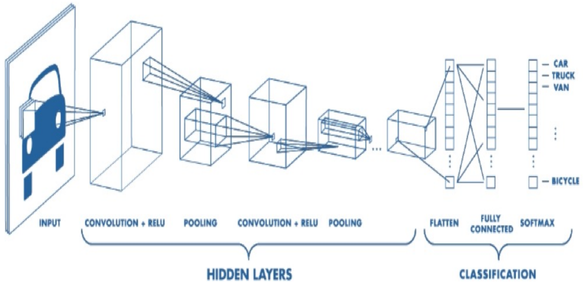
S. Bianco, R. Cadene, L. Celona and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," in *IEEE Access*, vol. 6, pp. 64270-64277, 2018, doi: 10.1109/ACCESS.2018.2877890.

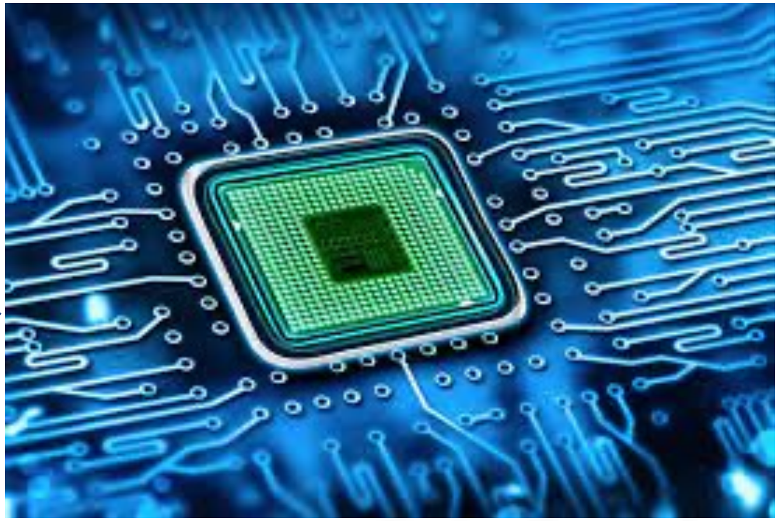
DNNs in the Embedded Space – Variability in Performance Requirements



What do we need

torch TensorFlow Caffe



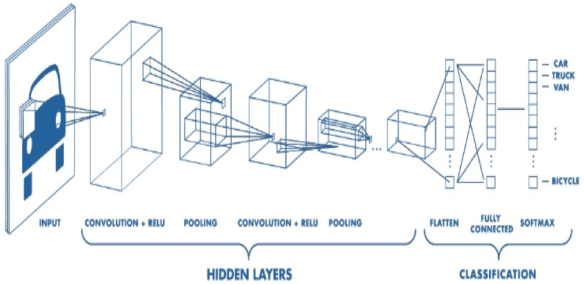


Objectives
- Throughput
- Latency
- Resources
- Power

Challenging?

What do we need

torch TensorFlow Caffe



FPGA

Objectives
- Throughput
- Latency
- Resources
- Power

Challenging? It depends

Customisation leads to efficiency and performance

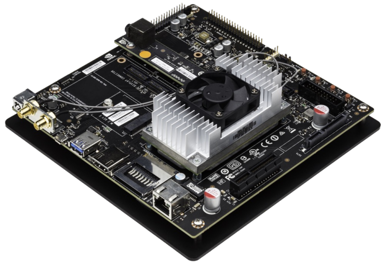
Customisation

Generic

Application Specific

DSPs
Qualcomm Hexagon,
Apple Neural Engine,

GPUs
Tegra K1, X1 and X2

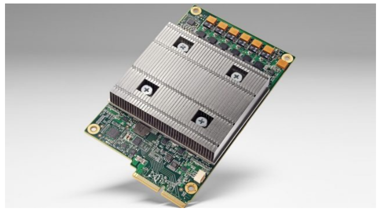


FPGAs

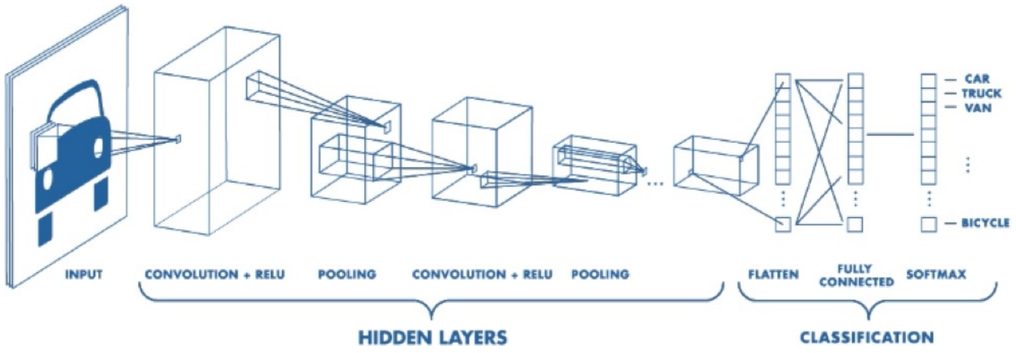
Custom datapath

Custom memory subsystem

ASICs
TPU



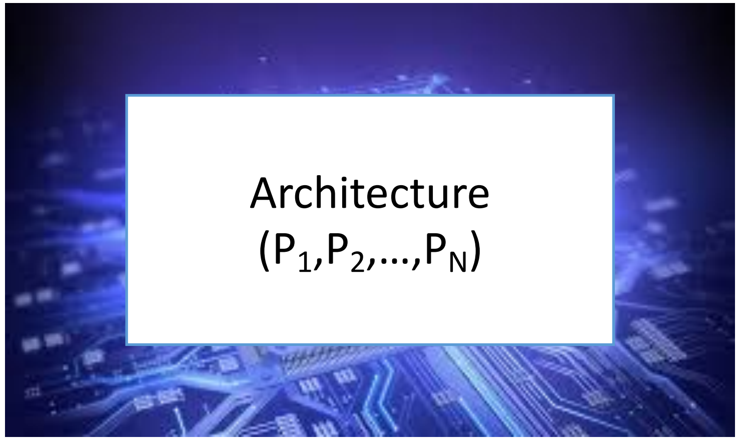
The Challenge of the Mapping Problem



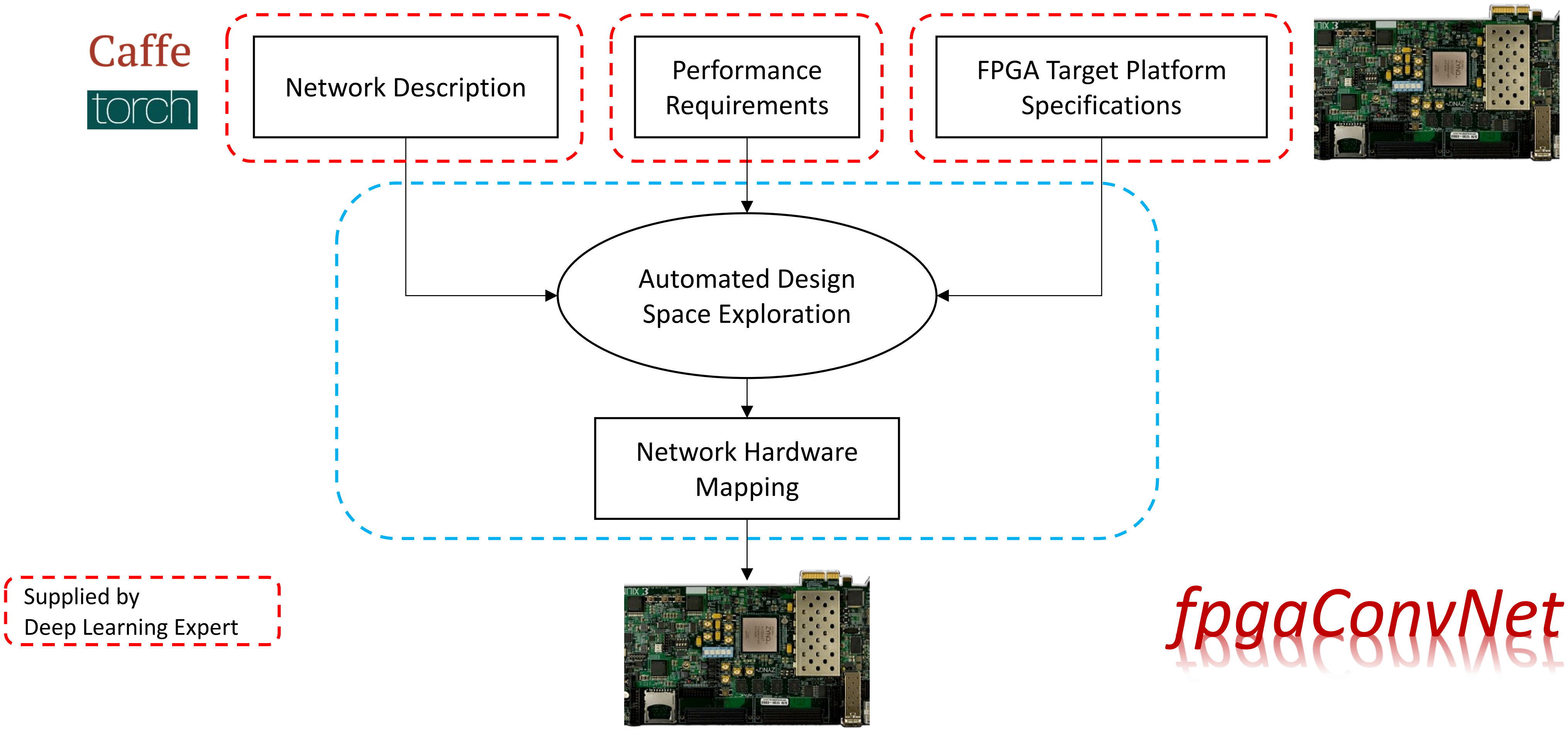
Parameters	Value
LC	2M
BRAMS (36kbits)	1,880
DSPs	3,360

Specifications
- Latency
- Throughput
- Power consumption

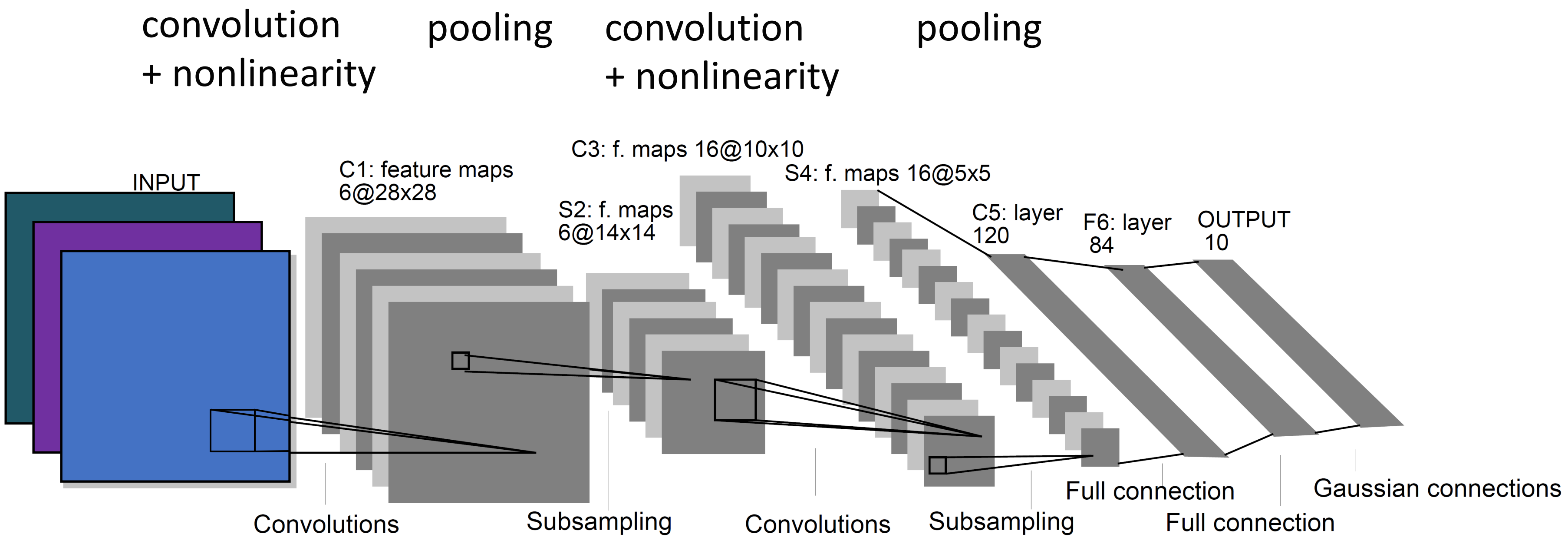
Challenges:
- Diversity of operations in modern NN
- Diversity and resources of modern FPGAs
- Competition (or need for performance) => Highly customised architecture
- Large number of parameters in the target architecture => DSE



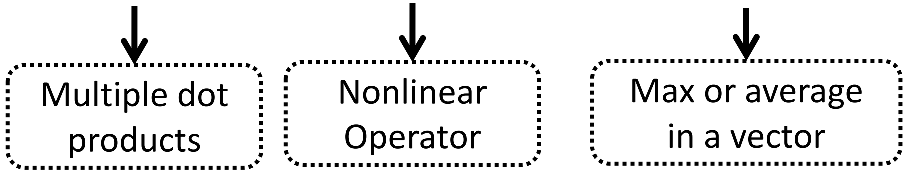
fpgaConvNet: Mapping CNNs to FPGAs



Under the hood: Convolutional Neural Networks (ConvNets)

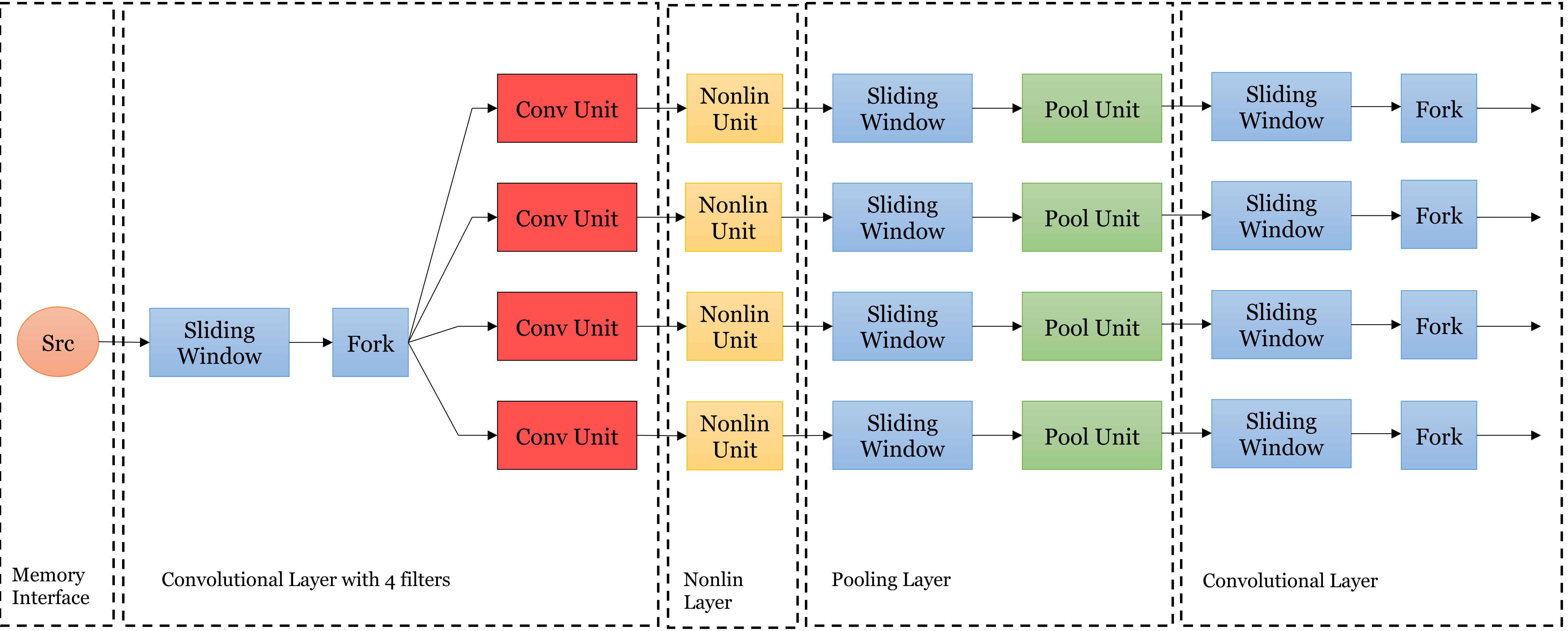


- ConvNet Inference
  - Tailored to images and data with spatial patterns
  - Built as a sequence of layers (Convolutional, Nonlinearity and Pooling Layer)
  - Feedforward operation
  - <u>Inherently streaming</u>



Imperial College London

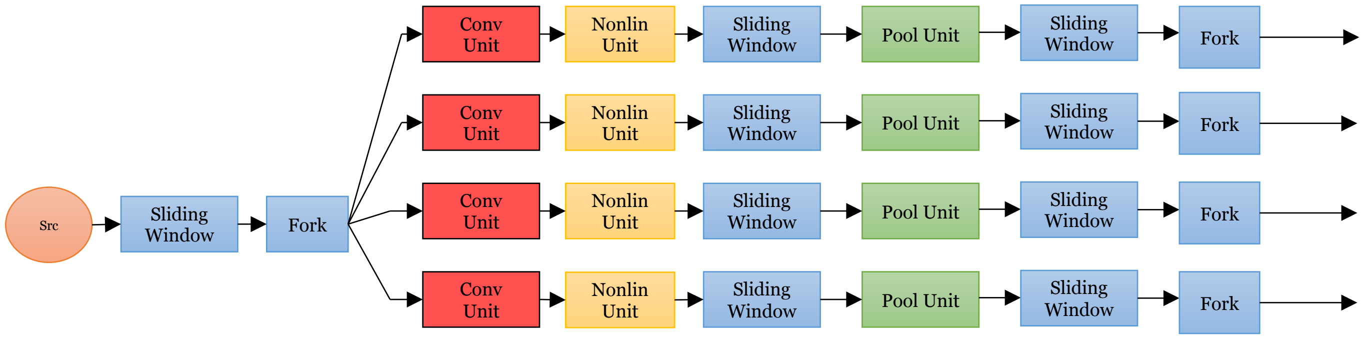
fpgaConvNet – Streaming Architecture for CNNs

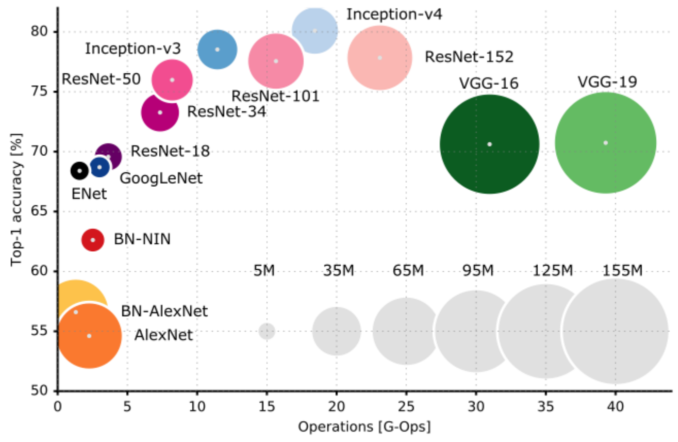


Imperial College London

fpgaConvNet – Streaming Architecture for CNNs

CNN Hardware SDF Graph

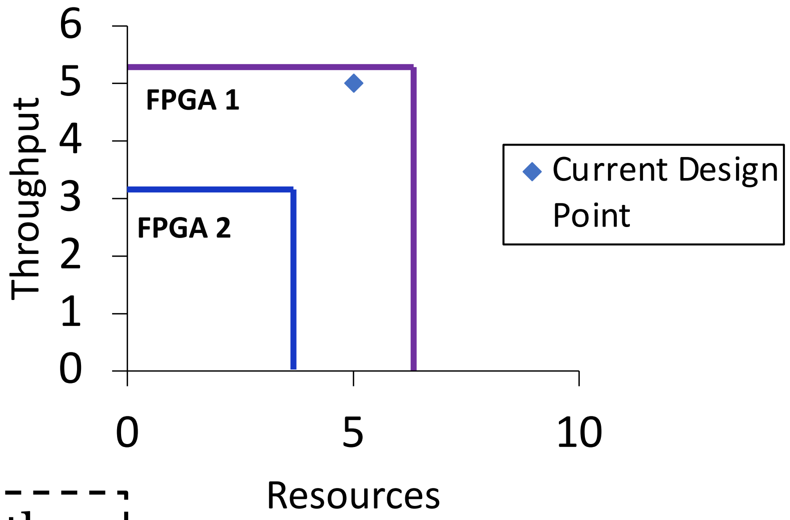




Complex Model➔ Bottlenecks:

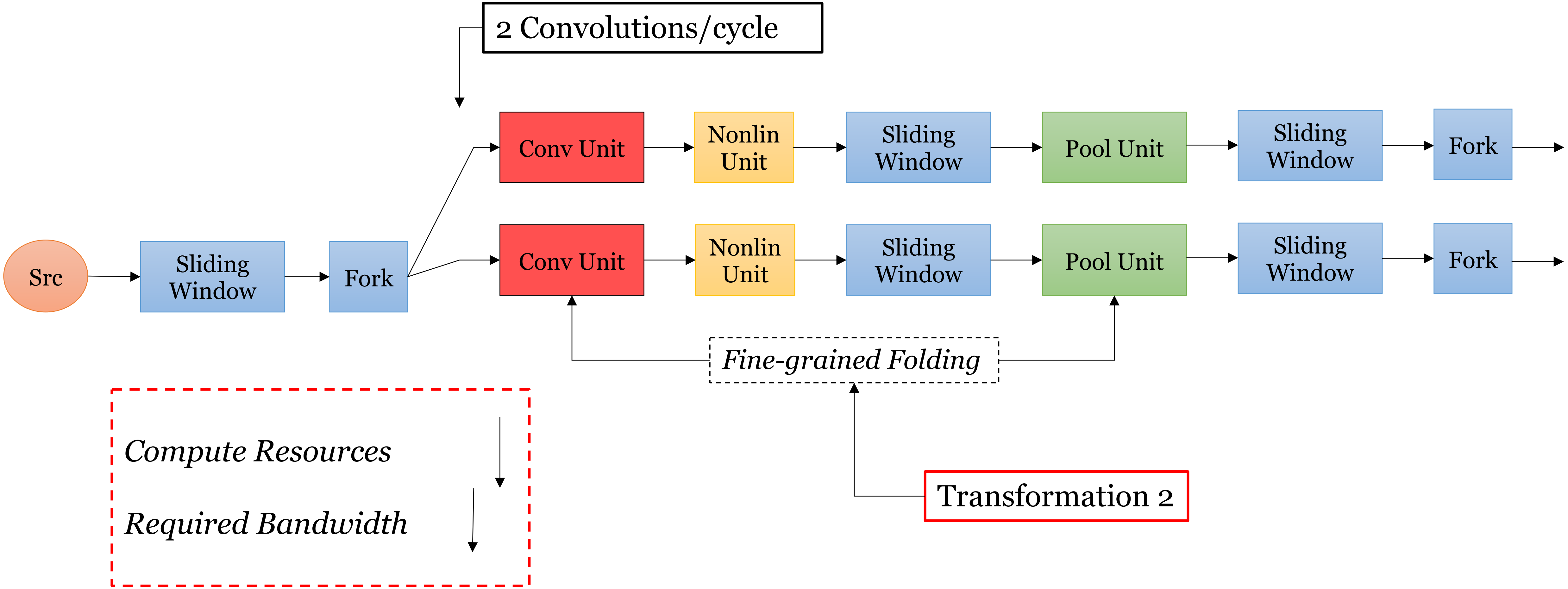
- Limited *compute resources*
- Limited *on-chip memory capacity* for model parameters
- Limited *off-chip memory bandwidth*

Design Space

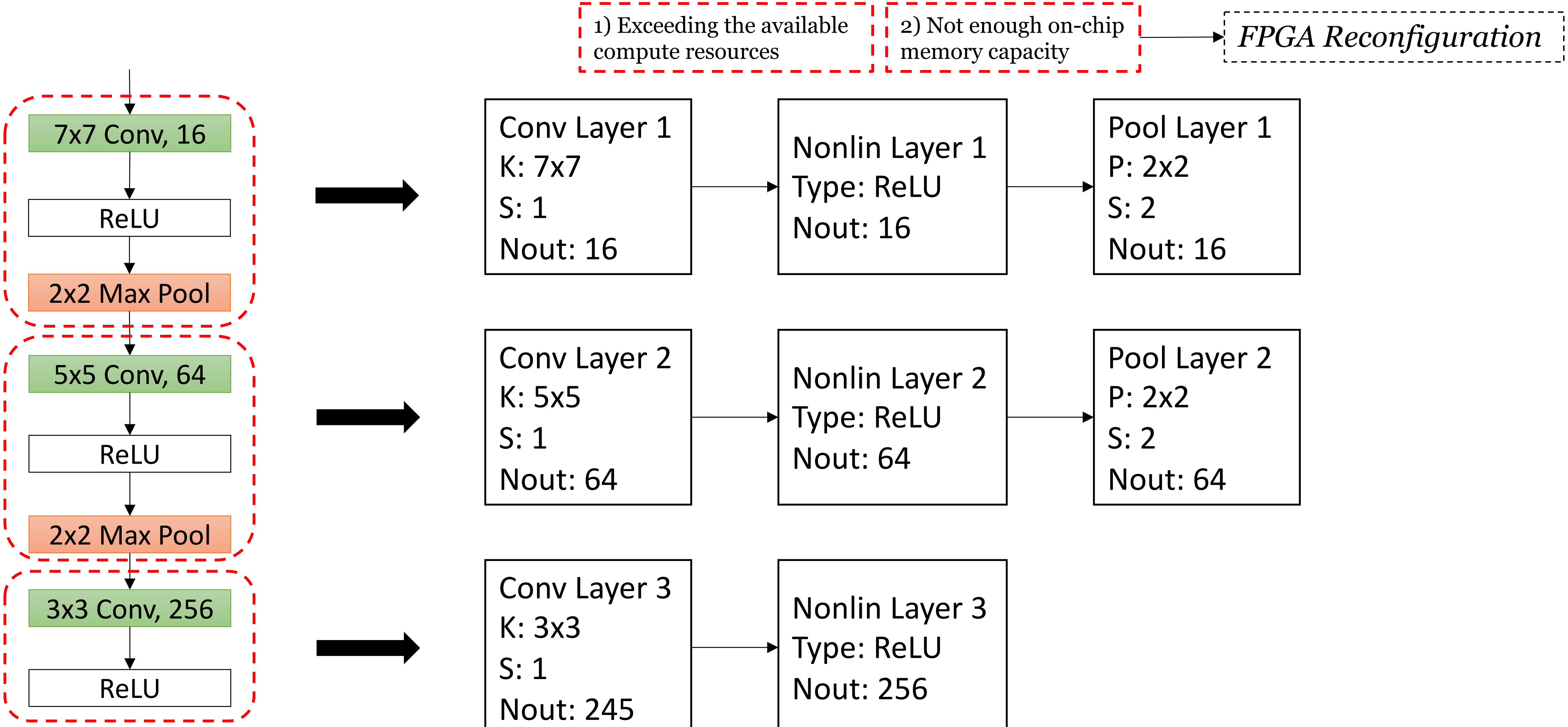


Define a set of **graph transformations** to traverse the design space in **fast** and **principled** way

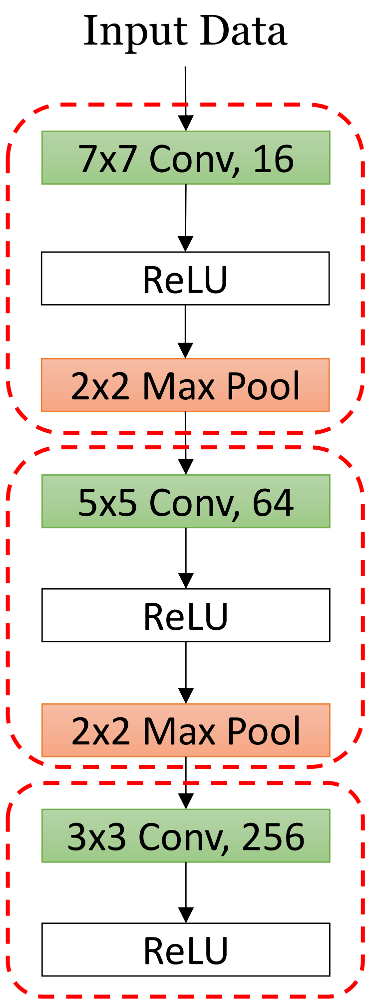
Transformations 1 & 2: Coarse- and fine-grained Folding

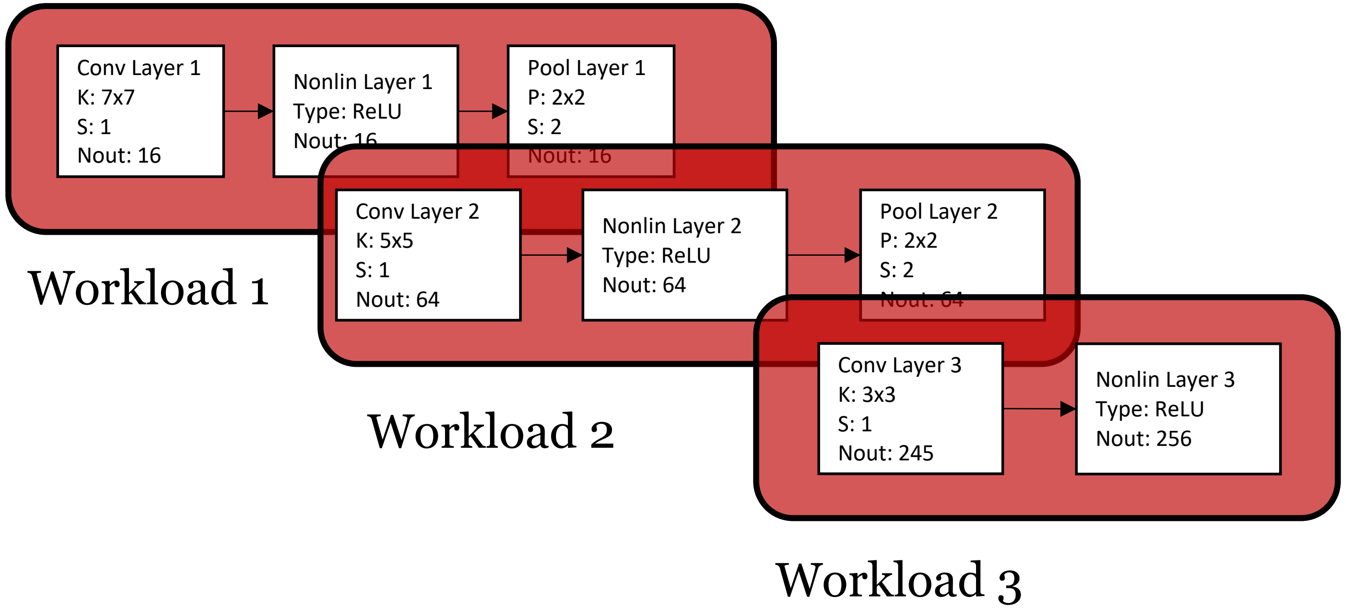


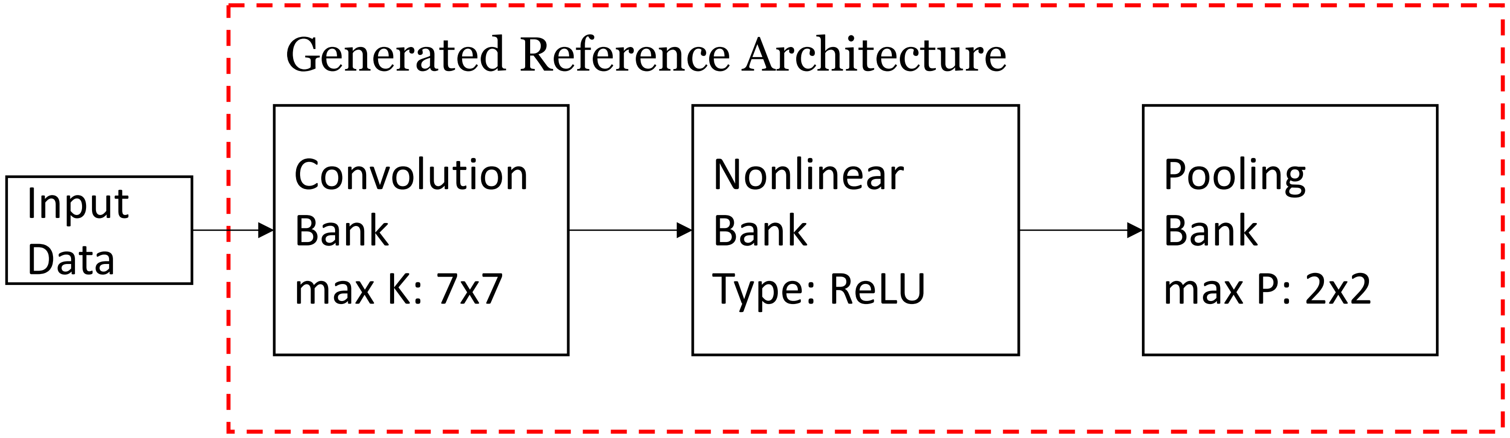
Transformation 3: Graph Partitioning with Reconfiguration



Transformation 4: Weights Reloading

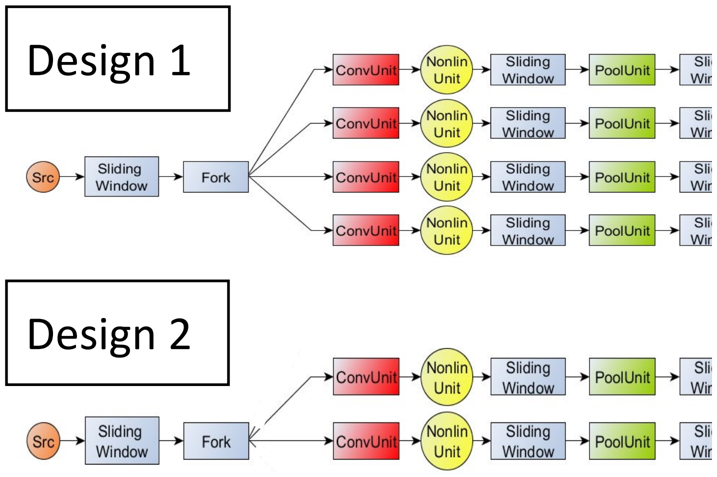






fpgaConvNet – Design Space Exploration and Optimisation

- Synchronous Dataflow Modelling
  - Capture hardware mappings as matrices
  - Transformations as *algebraic operations*
  - Analytical *performance model*
  - Cast design space exploration as a mathematical optimisation problem



Hardware Stages

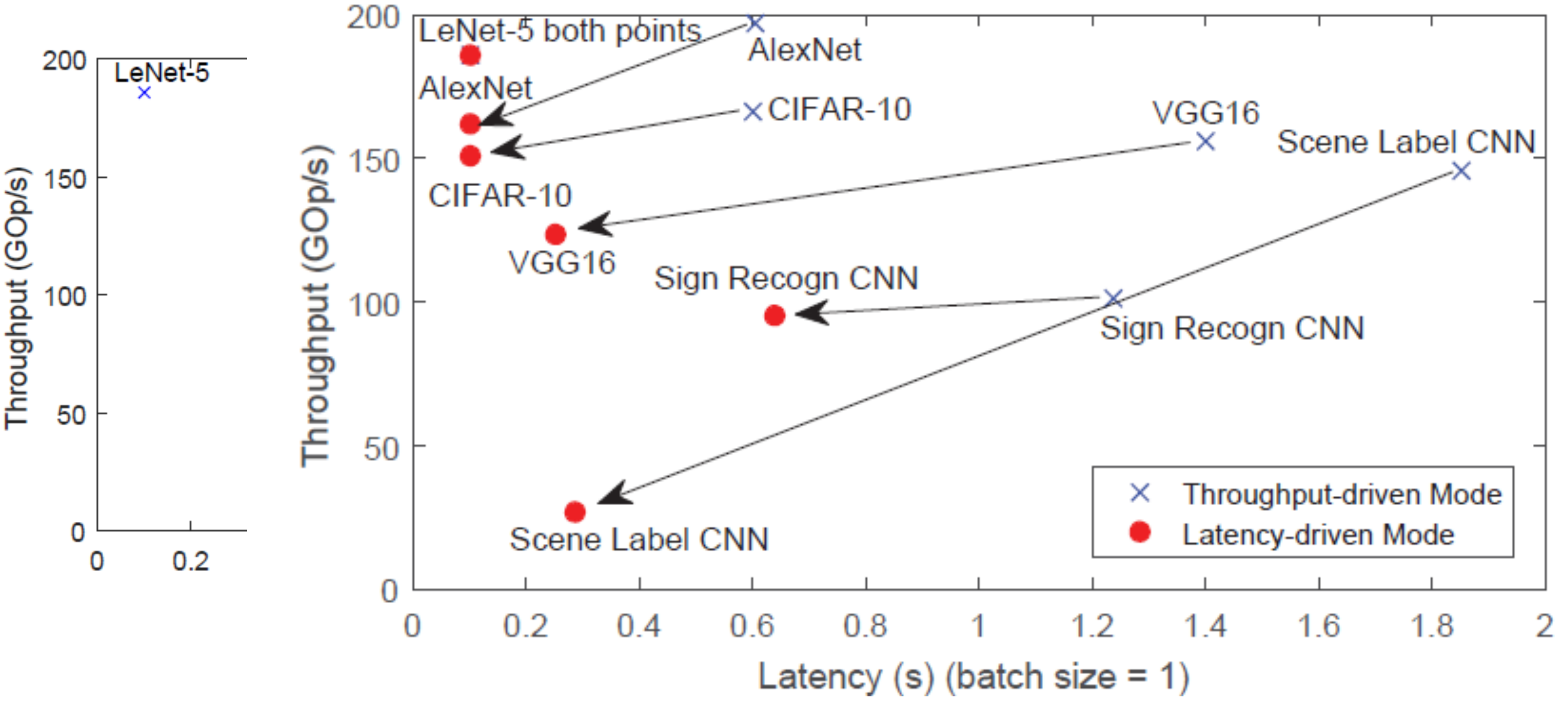
Interconnections

$$\Gamma_1 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & K^2 & -K^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4K^2 & -4K^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & -4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4P^2 & -4P^2 \end{bmatrix}$$

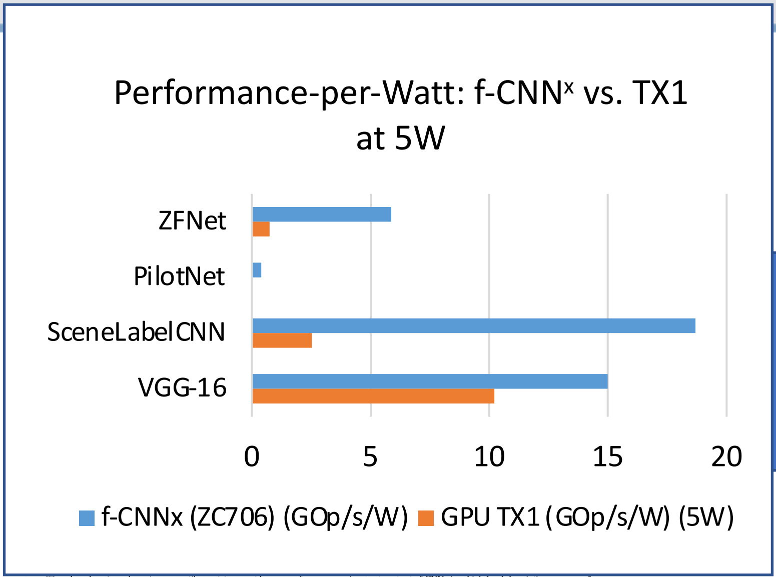
$$\Gamma_2 = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & K^2 & -K^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2K^2 & -2K^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2P^2 & -2P^2 \end{bmatrix}$$

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

Meeting the performance requirements



Extensions



based systems have to cope with servicing a wide range of concurrent CNN applications, from bioinformatics to visual search [5], with stringent response-time demands. In such scenarios, a dedicated model is trained for each particular task, leading to the parallel execution of several CNNs on the same target platform. Moreover, the latency-sensitive nature of modern applications prohibits the use of batch processing. As a result, in both emerging embedded and cloud applications there is a requirement for the latency-driven mapping of multiple CNNs on the computing platform of the target system.

Currently, the conventional computing infrastructure of complex autonomous systems and data centres comprises CPUs and GPUs, which are able to provide high processing

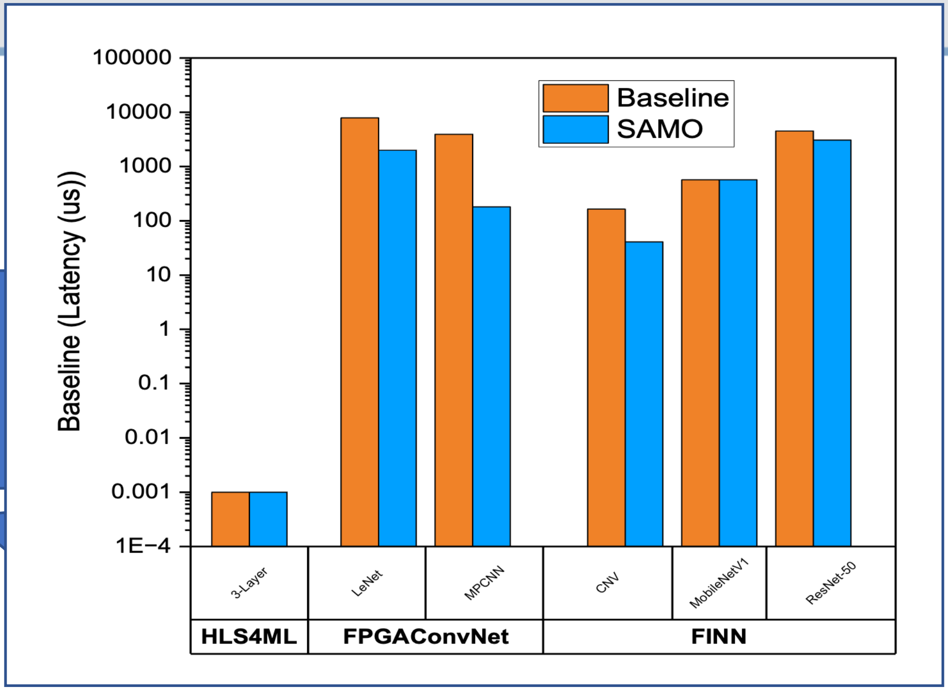
input a target set of CNNs in a high-level description, performing fast design space exploration and generating a synthesisable Vivado HLS hardware design.

To the best of our knowledge, this work addresses for the first time in the literature the mapping of multiple CNNs.

II. MULTIPLE CNNS ON RECONFIGURABLE LOGIC

A. Background on Multi-CNN Systems

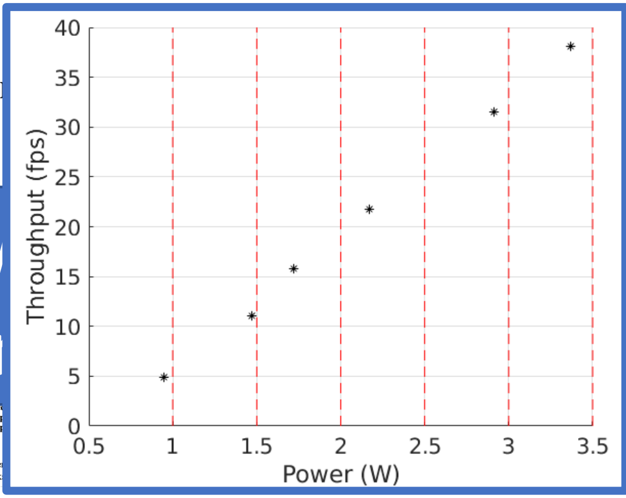
Multi-CNN systems employ a number of models, with each one trained for a different task. In the embedded space, drones and self-driving cars run a variety of concurrent tasks, such as navigation and obstacle avoidance [9]. In the cloud, services are increasingly heterogeneous, with diverse workloads



fpgaConvNet

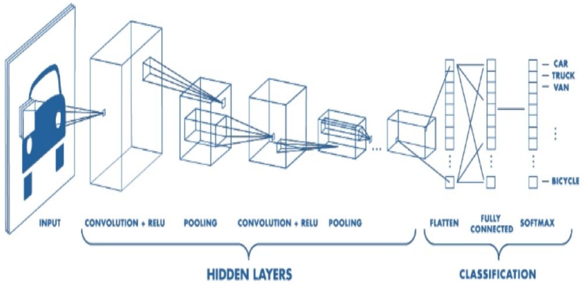
3D CNNs

Power-av
mappi



To approximate or not

torch TensorFlow Caffe



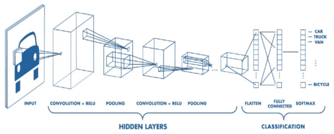
FPGA

Objectives
- Throughput
- Latency
- Resources
- Power

Faithful mapping

To approximate or not

torch TensorFlow Caffe



FPGA

Objectives
- Throughput
- Latency
- Resources
- Power

Faithful mapping

Introduce approximations: What can you gain?

Approximations in DNN

Weight quantisation

Pruning

Topology search

Retraining

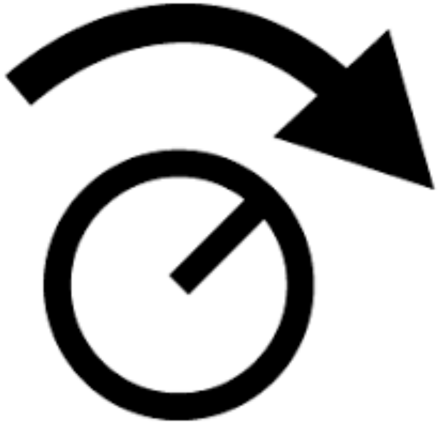
Hardware aware

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

“SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”,
Iandola, Forrest N; Han, Song; Moskewicz, Matthew W; Ashraf, Khalid; Dally, William J; Keutzer, Kurt (2016).

Problem setting - Assumptions

Training Data



Few

Many

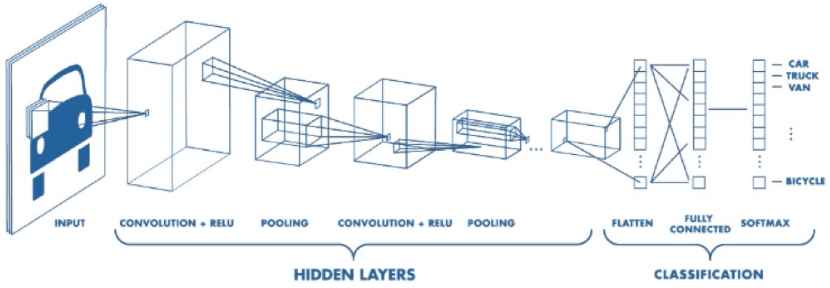
generic

specific

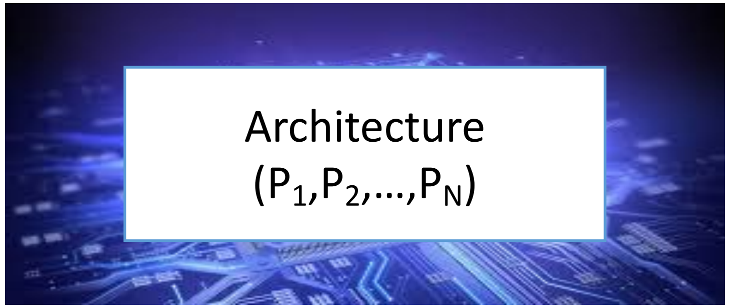
Assumption 1:

No training data is available
Validation data is available

Assumption 2:



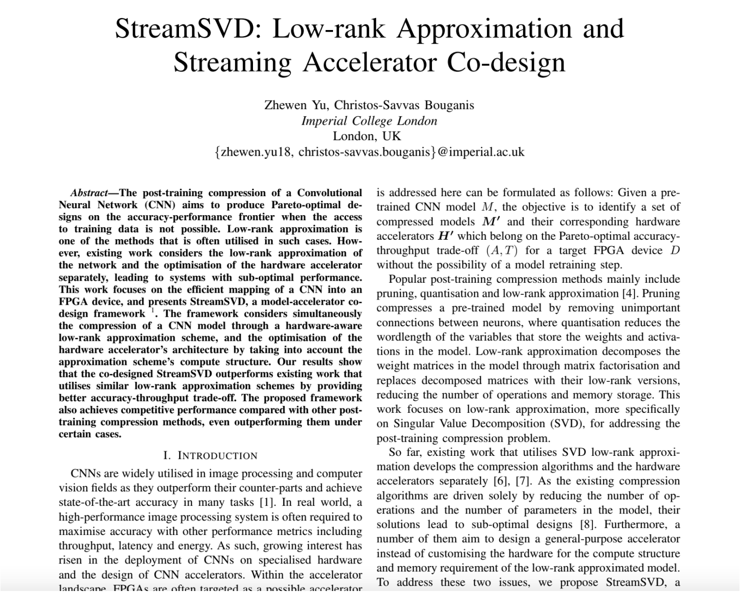
Assumption 3:



SteamSVD: System Overview

Idea

Explore redundancy **across** kernels of the same layer



StreamSVD: Low-rank Approximation and Streaming Accelerator Co-design

Zhewen Yu, Christos-Savvas Bouganis
Imperial College London
London, UK
{zhewen.yu18, christos-savvas.bouganis}@imperial.ac.uk

***Abstract*—The post-training compression of a Convolutional Neural Network (CNN) aims to produce Pareto-optimal designs on the accuracy-performance frontier when the access to training data is not possible. Low-rank approximation is one of the methods that is often utilised in such cases. However, existing work considers the low-rank approximation of the network and the optimisation of the hardware accelerator separately, leading to systems with sub-optimal performance. This work focuses on the efficient mapping of a CNN into an FPGA device, and presents StreamSVD, a model-accelerator co-design framework [1]. The framework considers simultaneously the compression of a CNN model through a hardware-aware low-rank approximation scheme, and the optimisation of the hardware accelerator's architecture by taking into account the approximation scheme's compute structure. Our results show that the co-designed StreamSVD outperforms existing work that utilises similar low-rank approximation schemes by providing better accuracy-throughput trade-off. The proposed framework also achieves competitive performance compared with other post-training compression methods, even outperforming them under certain cases.**

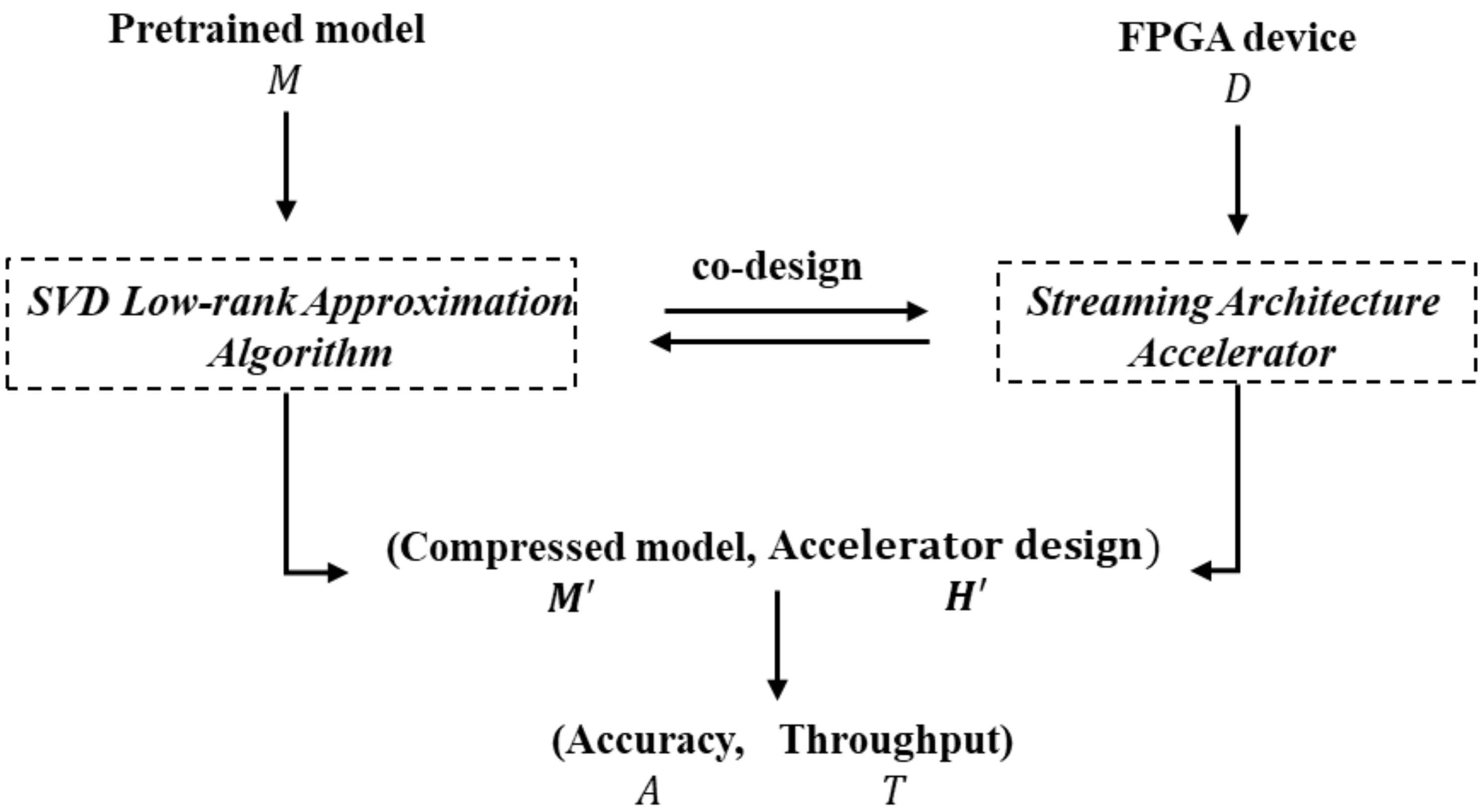
I. INTRODUCTION

CNNs are widely utilised in image processing and computer vision fields as they outperform their counter-parts and achieve state-of-the-art accuracy in many tasks [1]. In real world, a high-performance image processing system is often required to maximise accuracy with other performance metrics including throughput, latency and energy. As such, growing interest has risen in the deployment of CNNs on specialised hardware and the design of CNN accelerators. Within the accelerator landscape, FPGAs are often targeted as a possible accelerator

is addressed here can be formulated as follows: Given a pre-trained CNN model M, the objective is to identify a set of compressed models $\boldsymbol{M'}$ and their corresponding hardware accelerators $\boldsymbol{H'}$ which belong on the Pareto-optimal accuracy-throughput trade-off (A, T) for a target FPGA device D without the possibility of a model retraining step.

Popular post-training compression methods mainly include pruning, quantisation and low-rank approximation [4]. Pruning compresses a pre-trained model by removing unimportant connections between neurons, where quantisation reduces the wordlength of the variables that store the weights and activations in the model. Low-rank approximation decomposes the weight matrices in the model through matrix factorisation and replaces decomposed matrices with their low-rank versions, reducing the number of operations and memory storage. This work focuses on low-rank approximation, more specifically on Singular Value Decomposition (SVD), for addressing the post-training compression problem.

So far, existing work that utilises SVD low-rank approximation develops the compression algorithms and the hardware accelerators separately [6], [7]. As the existing compression algorithms are driven solely by reducing the number of operations and the number of parameters in the model, their solutions lead to sub-optimal designs [8]. Furthermore, a number of them aim to design a general-purpose accelerator instead of customising the hardware for the compute structure and memory requirement of the low-rank approximated model. To address these two issues, we propose StreamSVD, a



SVD Low-rank Approximation Algorithm

For a convolutional layer with C input channels, F output channels, $K \times K$ kernel size

Weights $M^{F\times C\times K\times K}$

Unfold

$W^{m\times n}$

SVD

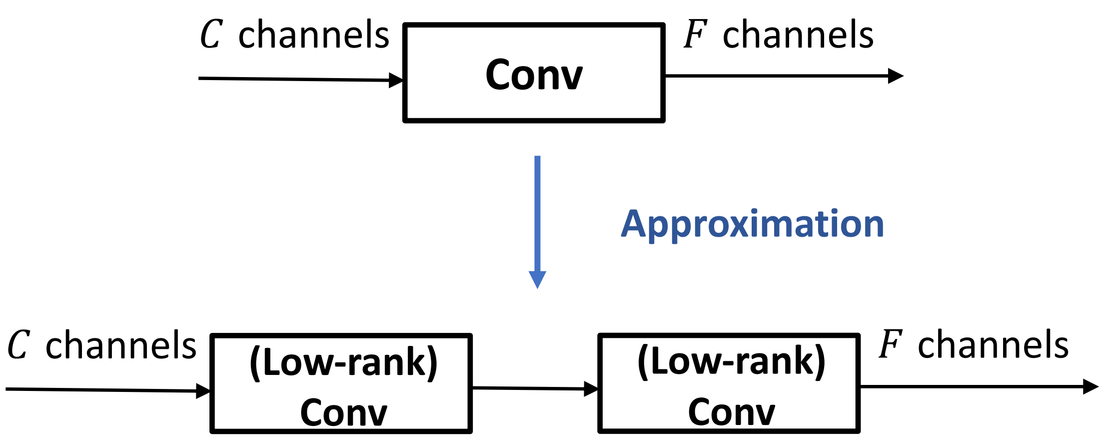
$$W^{m\times n} = U\Sigma V^T = (U\Sigma^{\frac{1}{2}})(\Sigma^{\frac{1}{2}}V^T) = W_2 W_1$$

Reduce the rank to R

$$(\widehat{W}_2)^{m\times R}(\widehat{W}_1)^{R\times n}$$

Quantisation

$$\widehat{W}_2{}'\widehat{W}_1{}'$$



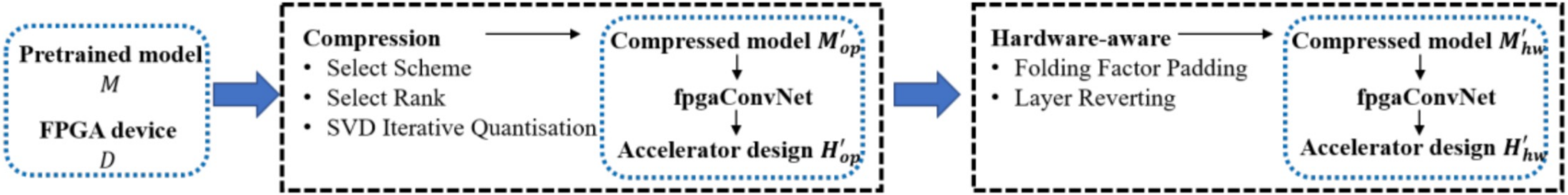
scheme (S)	original weight matrix	unfolded matrix	two low-rank layers
s_0 [35]	$M^{F\times C\times K\times K}$	F groups of $W^{KK\times C}$	$Conv(1\times 1, C, RF, 1)$ -> $Conv(K\times K, RF, F, F)$
s_1 [28]		$W^{F\times CKK}$	$Conv(K\times K, C, R, 1)$ -> $Conv(1\times 1, R, F, 1)$
s_2 [29]		$W^{FK\times CK}$	$Conv(1\times K, C, R, 1)$ -> $Conv(K\times 1, R, F, 1)$
s_3 [31]		C groups of $W^{F\times KK}$	$Conv(K\times K, C, CR, C)$ -> $Conv(1\times 1, CR, F, 1)$

Optimisation

Design space:

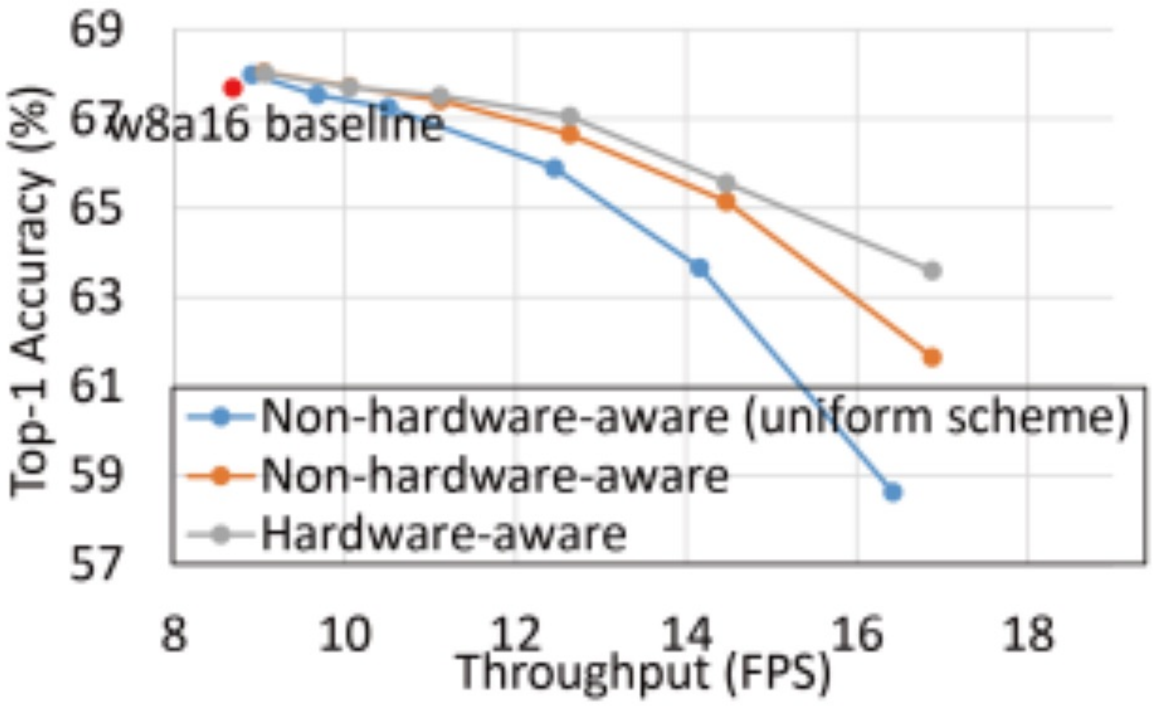
- Select decomposition scheme per layer
- Select rank R per layer: controls approximation
- Each layer is tuned to the most appropriate scheme
- Relative importance of each layer is derived from the Taylor pruning criterion

$$I_f = \sum_{w \in f} (w \frac{\partial L}{\partial w})^2$$

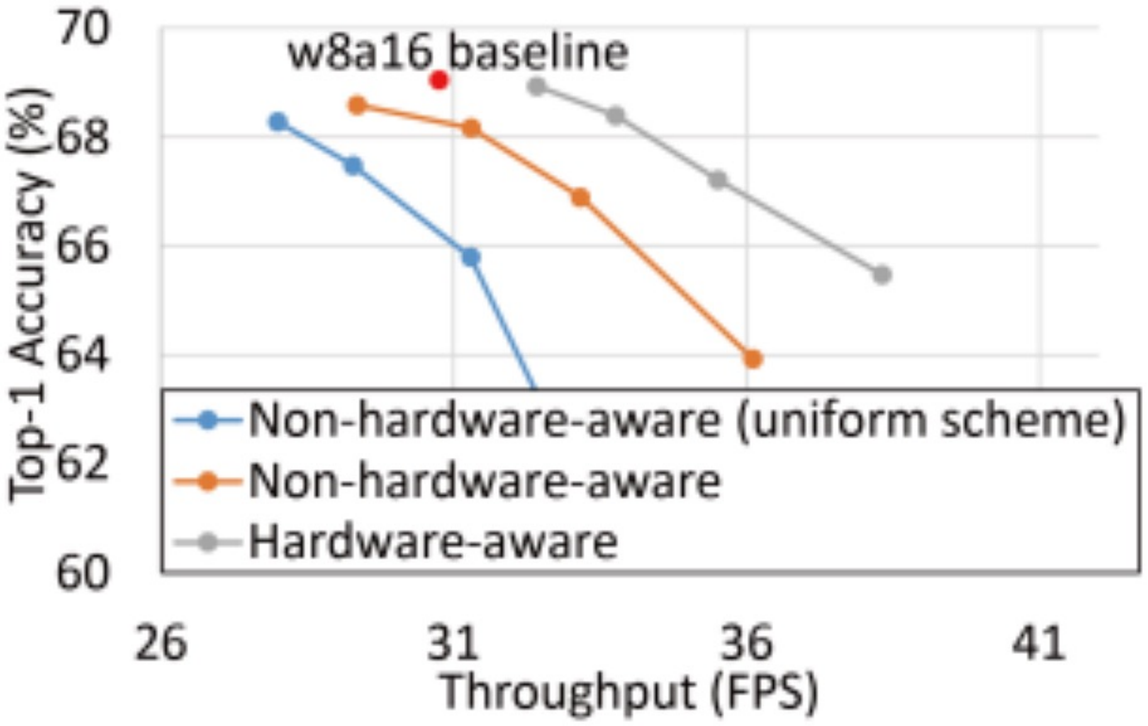


Hardware aware and per layer optimisations

VGG-11 BN



ResNet-18



Results

TABLE III: Comparison with other FPGA accelerators

	Model	Compression method	Post -training	Accuracy (%)	Relative Throughput Efficiency
[12]	VGG-16	w16a16, pruning	no	-	1.31×
[41]		w16a16, pruning	no	-	1.65×
[42]		f16, low-rank	no	70.46	0.18×
[43]		w8a16-BFP	yes	68.26	1.17×
[18]		w16a16, low-rank	yes	64.64	0.37×
[44]		w16a16, low-rank	yes	-	0.44×
fpgaConvNet [39]		w16a16	yes	-	0.46×
StreamSVD		w8a16, low-rank	yes	70.20	0.72×
StreamSVD		w8a16, low-rank	yes	65.20	1.00×
[45]	ResNet-18	w2a8-BFP, low-rank	no	68.23	0.87×
[44]		w16a16	yes	-	0.12×
StreamSVD		w8a16, low-rank	yes	68.39	1.00×

Our method is competitive with other compression methods

Summary

- Customisation is **key**, but also a **challenge** in the design of DNN systems under resource constraints
- Large opportunities in the ML space for approximations
  - Availability of data (and time)?
- Exposing the hardware capabilities to the algorithm can lead to performance gains
  - Challenging task
  - Rethink current approaches to fully utilise the underlying hardware

customisation

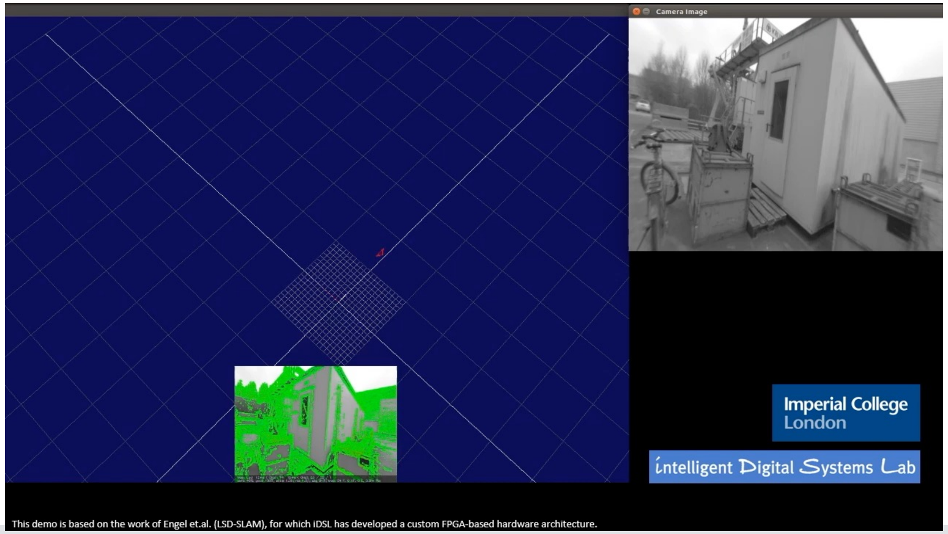
Some of our work

Autonomous Navigation

Hunan Pose Estimation

Traffic Detection

Localisation and Mapping



Questions

SVD-NAS

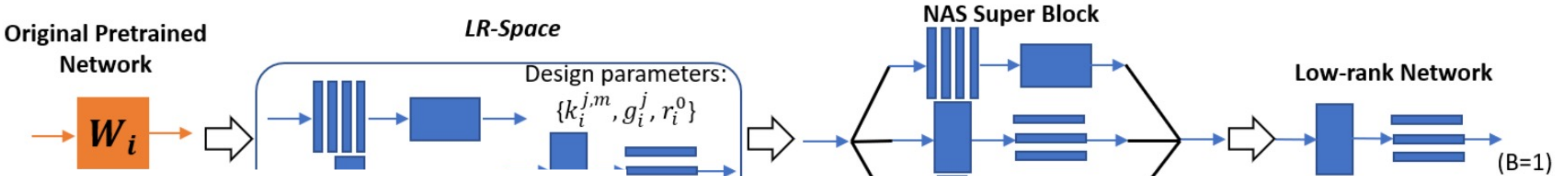


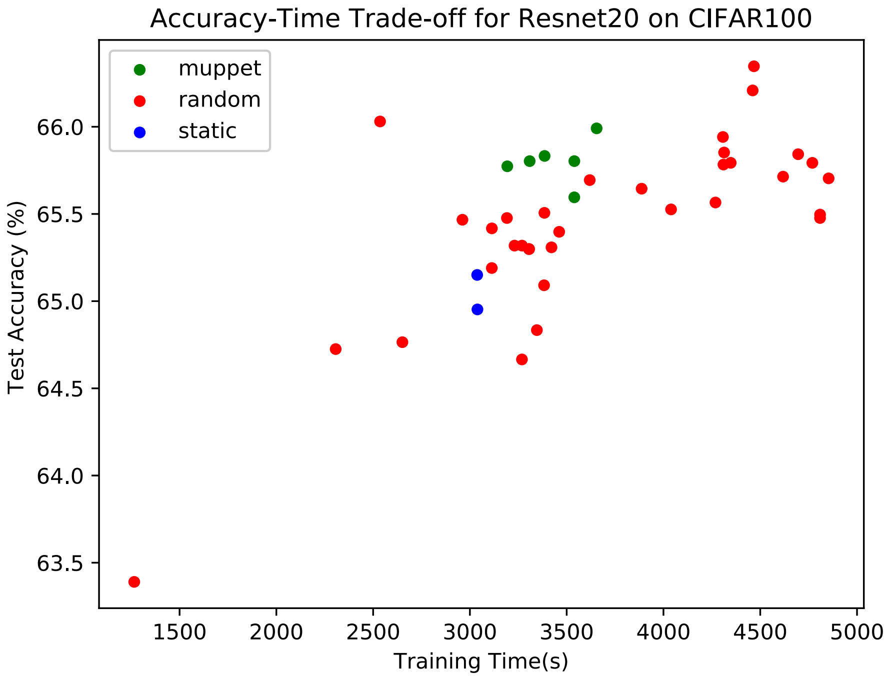
Table 1: Post-training results of low-rank approximation. * no fine-tuning. ** fine-tuning with 25k synthetic images

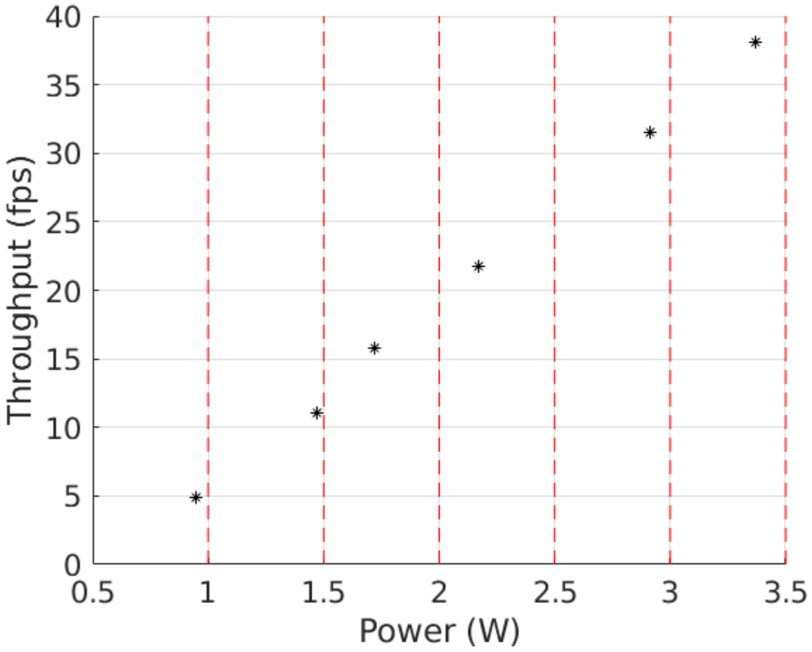
Model	Method	Δ FLOPs (%)	Δ Params (%)	Δ Top-1 (pp)	Δ Top-5 (pp)
ResNet-18	**SVD-NAS**	**-58.60**	**-68.05**	−13.35* **−5.85****	−9.14* **−3.34** **
	ALDS [18]	-42.31	-65.14	-18.70	-13.38
	LR-S2 [8]	-56.49	-57.91	-38.13	-33.93
	F-Group[21]	-42.31	-10.66	-69.34	-87.63
MobileNetV2	**SVD-NAS**	**-12.54**	-9.00	−15.09* **−9.99** **	−7.79* **−6.11** **
	ALDS [18]	-2.62	**-37.61**	-16.95	-10.91
	LR-S2 [8]	-3.81	-6.24	-17.46	-10.34
EfficientNet-B0	**SVD-NAS**	**-22.17**	**-16.41**	−10.11* **−7.67** **	−5.49* **−4.06** **
	ALDS [18]	-7.65	-10.02	-16.88	-9.96
	LR-S2 [8]	-18.73	-14.56	-22.08	-14.15

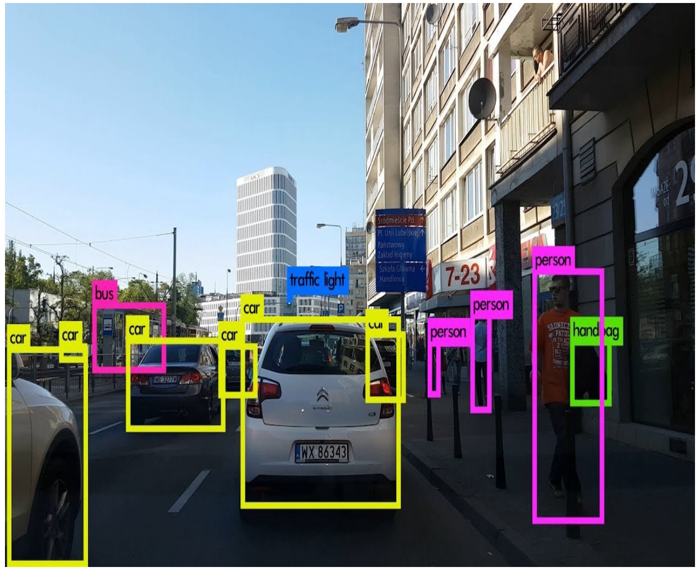
Table 4: Latency-driven search results on Pixel 4

Model	Objective	Δ Top-1 (pp)	Δ FLOPs (%)	Δ Latency (%)	Latency (ms)
ResNet-18	FLOPs	-5.83	**-59.17**	-44.52	76.70
	Latency	-5.67	-54.78	**-49.46**	69.87
MobileNetV2	FLOPs	-9.99	**-12.54**	-1.03	30.66
	Latency	-8.22	-9.55	**-4.75**	29.51
EfficientNet-B0	FLOPs	-9.45	**-22.85**	-1.92	67.08
	Latency	-10.49	-21.39	**-6.46**	63.97

What we are looking into...







DNN Training MuPPET

DNN Training MOCHA

Object Detection to FPGA mapping

Homomorphic Encryption ML loads

On-device adaptation