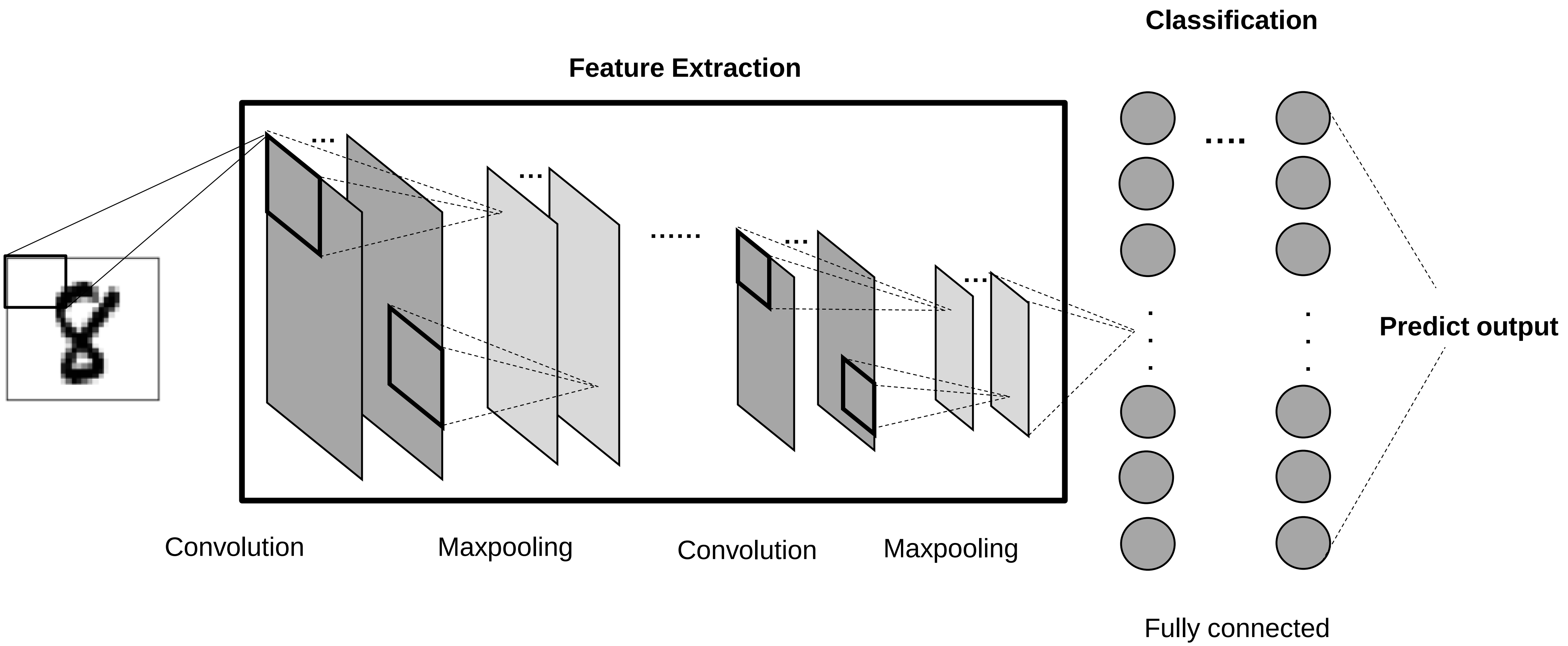
AN HLS-BASED CAPSULE NETWORK ACCELERATION SYSTEM IN AN FPGA

2016124099 박관영
2016124103 박상준
2016124124 사재현
2016124145 양해찬

Table of Contents

Introduction

- Conventional NN models for Image Classification: CNN



Introduction

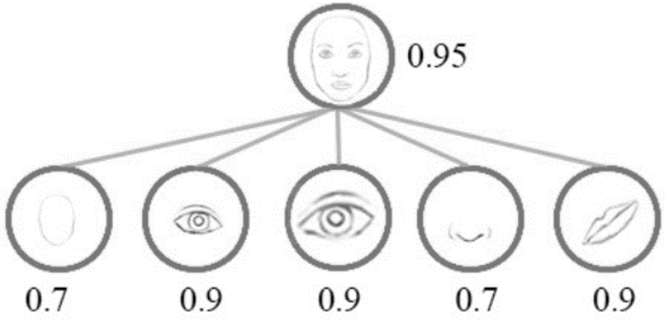
- CNN’s Problems
  - Invariance (throw away the relations between objects)
    - Because of pooling layer (Max pooling)

8	3	5	9
4	7	2	4
6	5	2	1
1	7	3	6

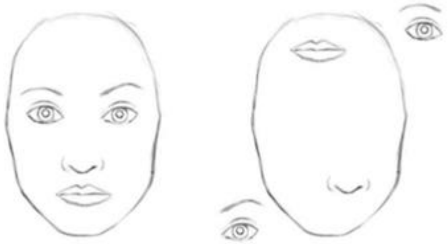
➔

8	9
7	6

Max pooling



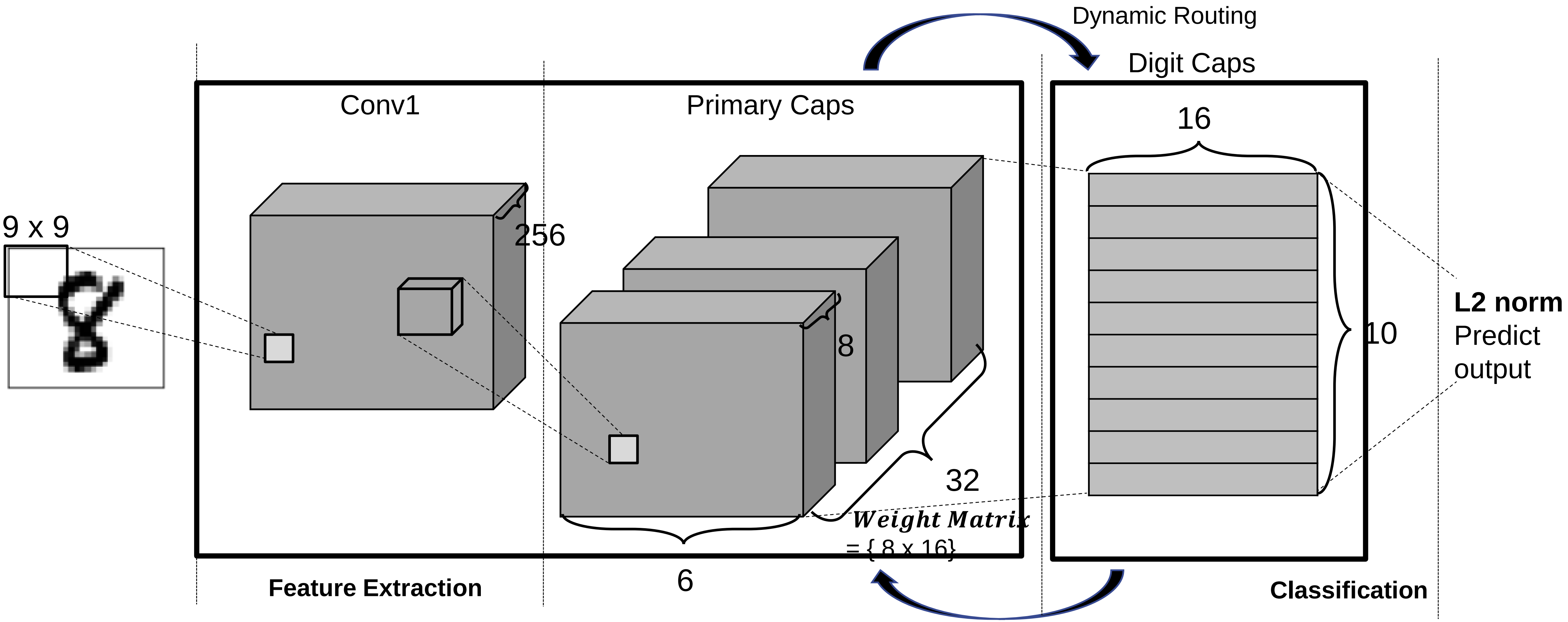
➔ extract the features without relations between objects



➔ Detect both images as human

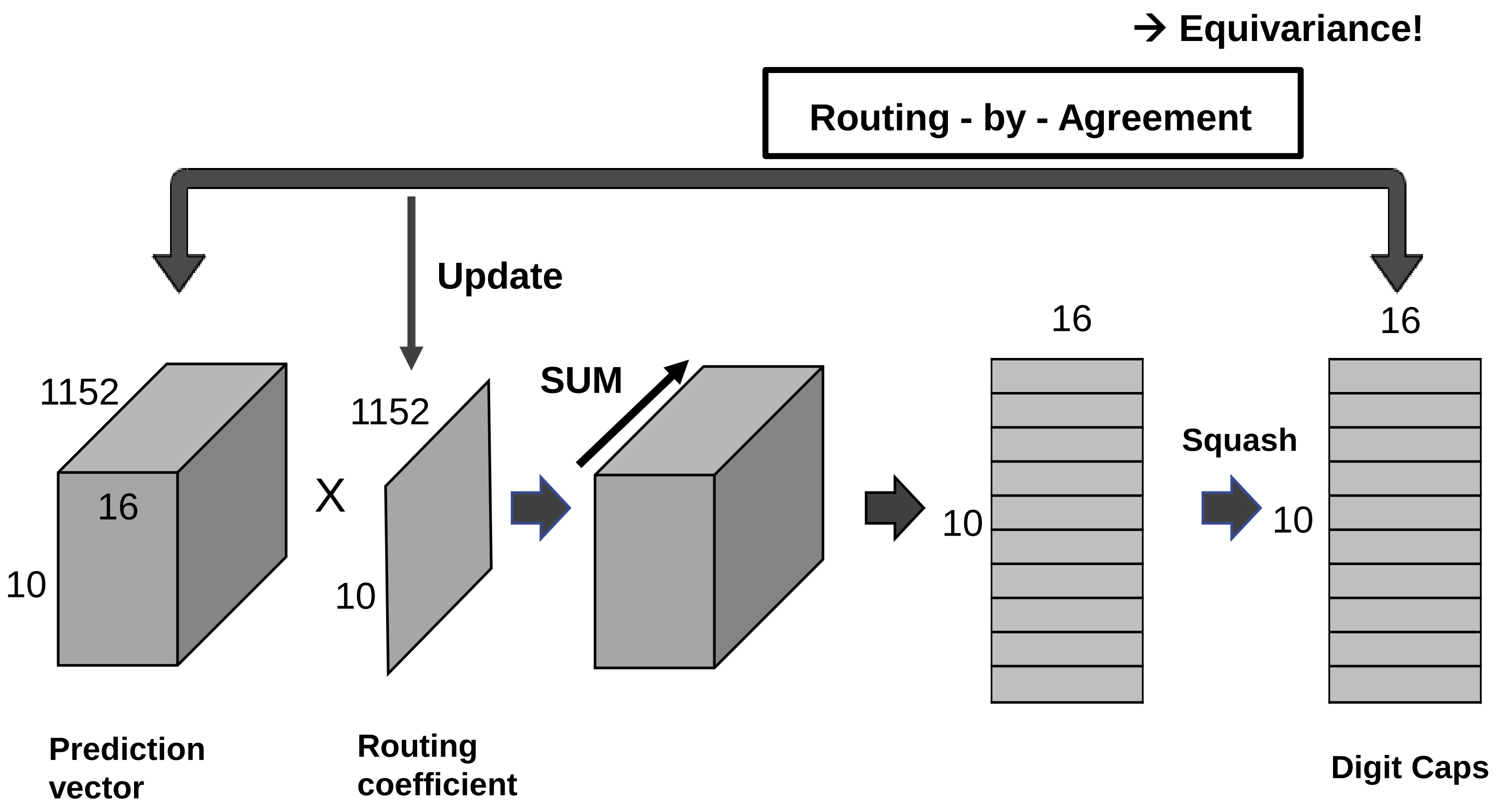
Introduction

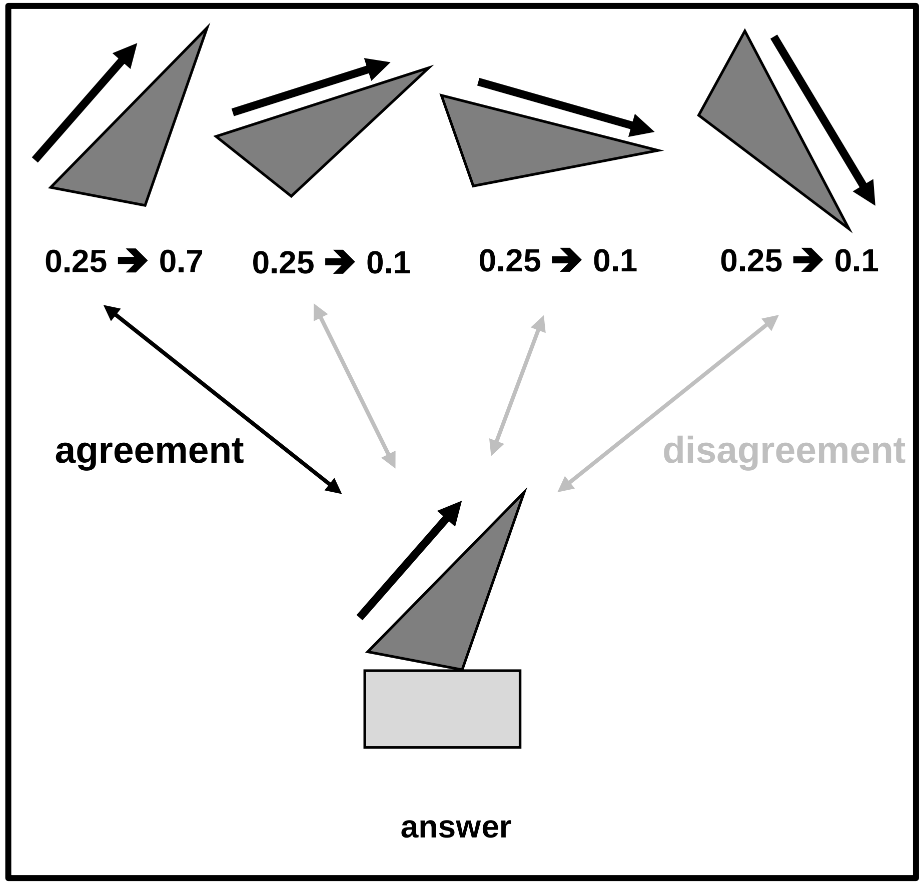
- Capsule Network (Caps Network) Structure



Introduction

- Dynamic Routing

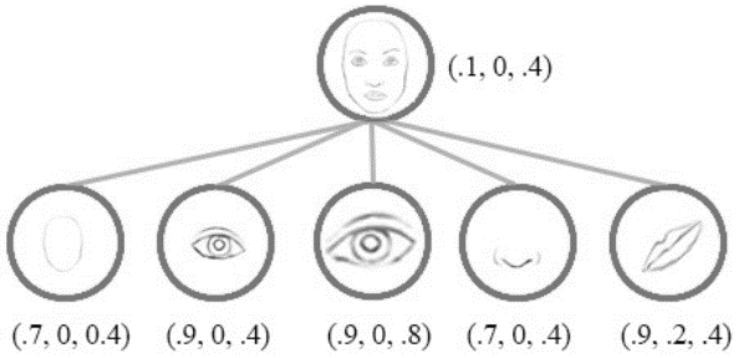




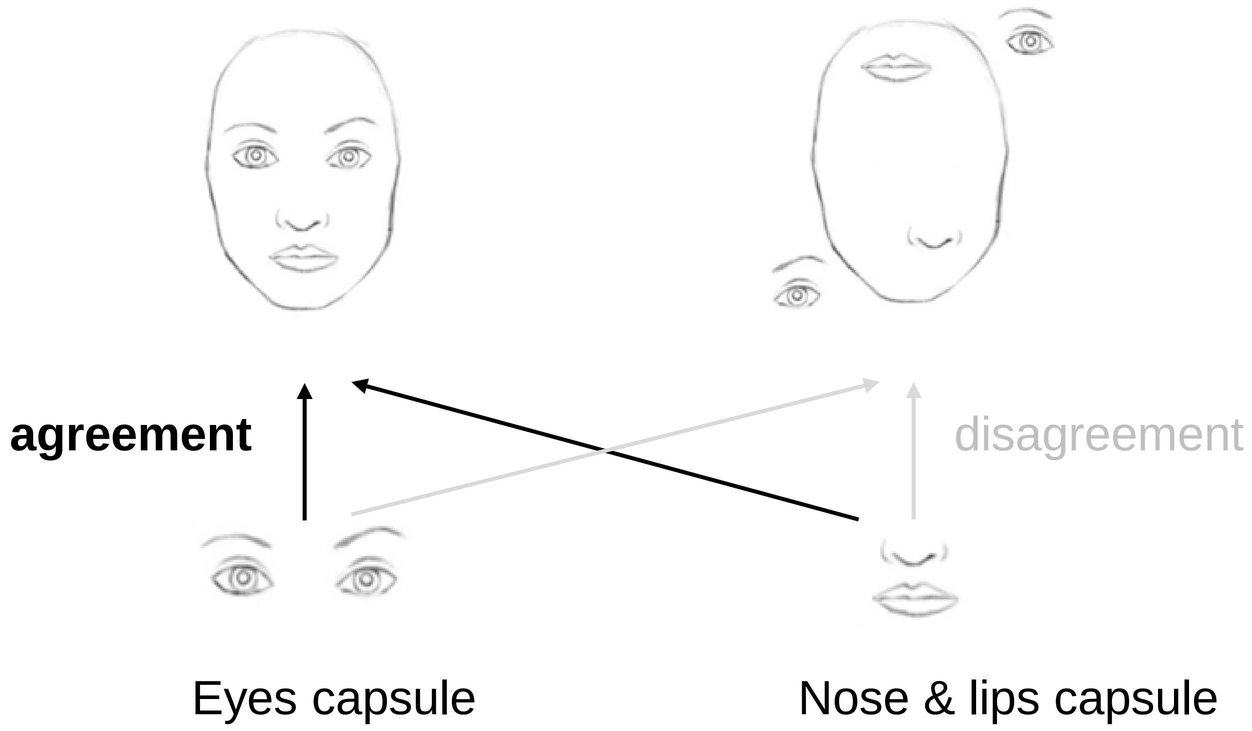
S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic routing between capsules,” in Proc. Conf. Neural Inf. Process. Syst. (NIPS), 2017

Introduction

- Capsule Network's Advantages
  - Equivariance (Max pooling ➔ Dynamic Routing)



➔ extract the features with property



Introduction

- Previous Work
  - 3D Object Recognition
  - Computation and EMA breakdown

Multi MNIST	Parameter	Test Error
CNN	35.4M	0.5 %
Capsule Network	15M	0.25 %

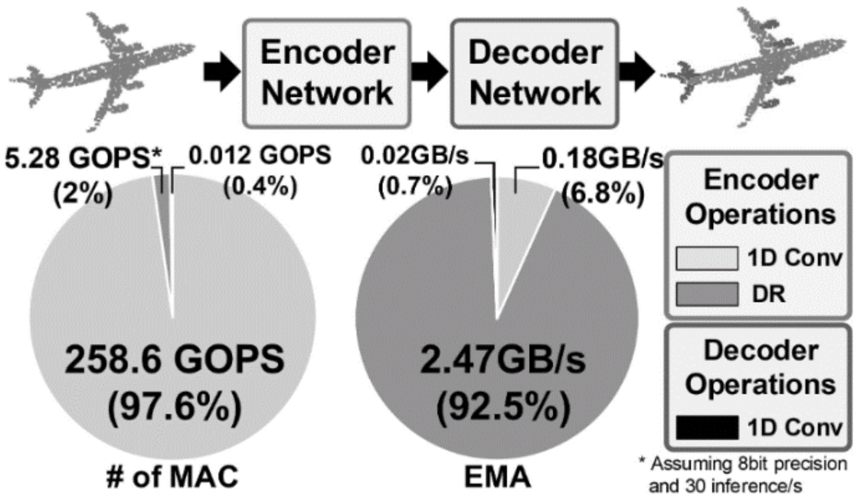


Fig. 1. Computation and EMA breakdown of the overall 3D-CapsNet.

MAC (Multiply Accumulate) :Conv에 집중된 MAC 연산
EMA (External Memory Access) :잦은 횟수의 EMA

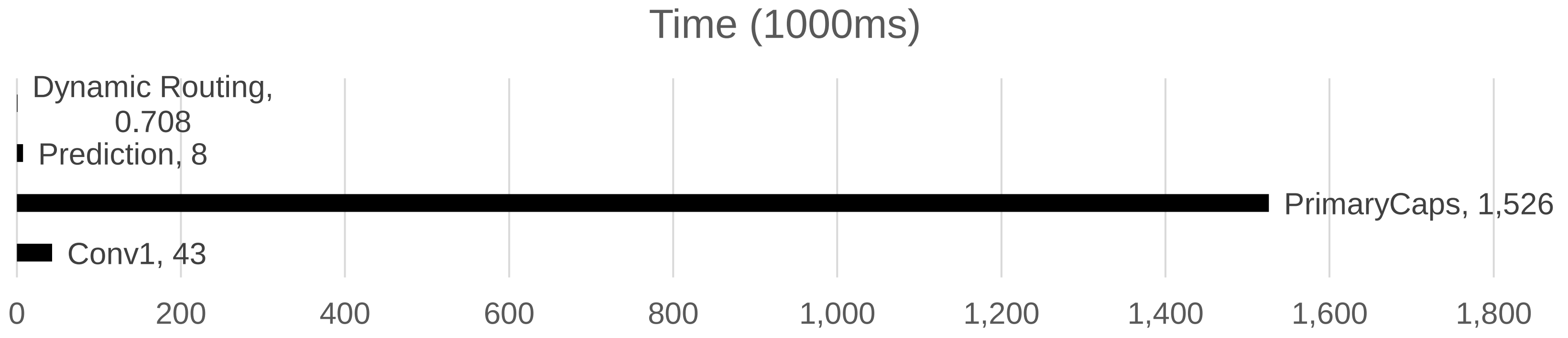
Reference:
- S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in Proc. Conf. Neural Inf. Process. Syst. (NIPS), 2017
- Lars Hertel, "Deep Convolutional Neural Networks as Generic Feature Extractors" IJCNN 2015
- Gwangtae Park, "A 1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment" TCSII 2020

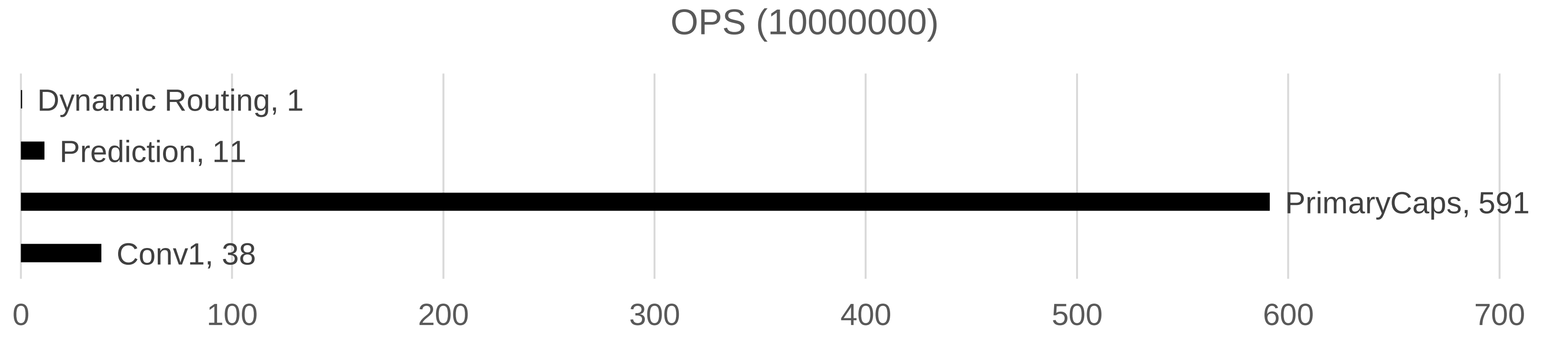
Proposed System

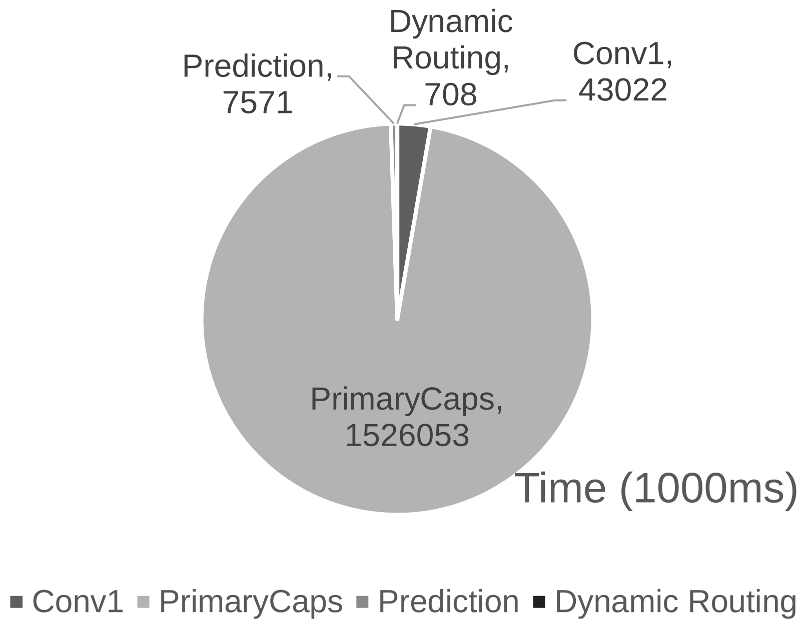
- Our Contributions
  - Analysis of Capsule Network inference algorithm in terms of the execution time and complexity
  - Optimization of the inference flow for embedded systems
    - Fixed−point optimization with mathematical approximations
  - Design of HLS-based components for accelerating some computationally-intensive kernels
  - Demonstration by designing a prototype system

Proposed System: Analysis of Capsule Network Inference Flow

- Main Bottlenecks: PrimaryCaps, Conv1, Prediction layer







Proposed System: Optimization of Inference Flow

- Mathematical Approximation
  - Squash Activation Function

$$\mathbf{V}_j = \frac{\|\mathbf{S}_j\|^2}{1+\|\mathbf{S}_j\|^2} \cdot \frac{\mathbf{S}_j}{\|\mathbf{S}_j\|}$$

$$\|\mathbf{S}_j\| = \; l_2 = \sqrt{|s_1|^2 + |s_2|^2 + |s_3|^2 + |s_4|^2 + \cdots\cdots + |s_n|^2}$$

$$l_\infty = max(|s_1|, |s_2|, |s_3|, |s_4|, \; \cdots , |s_n|)$$

$$l_1 = |s_1| + |s_2| + |s_3| + |s_4| + \cdots + |s_n|$$

- $l_2 \cong a * l_1 + b * l_\infty$
- Using linear regression to get coefficients

Reference : Gwangtae Park, "A 1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment", TCSII 2020

Proposed System: Optimization of Inference Flow

- Mathematical Approximation
  - Softmax Activation Function

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ij})}$$

- At Dynamic Routing 1st iteration, b_{ij} is initialized to zero.
- In inference, we don't have to repeat multiple Dynamic Routing iterations.
- So we can get c_{ij} without exponential function.

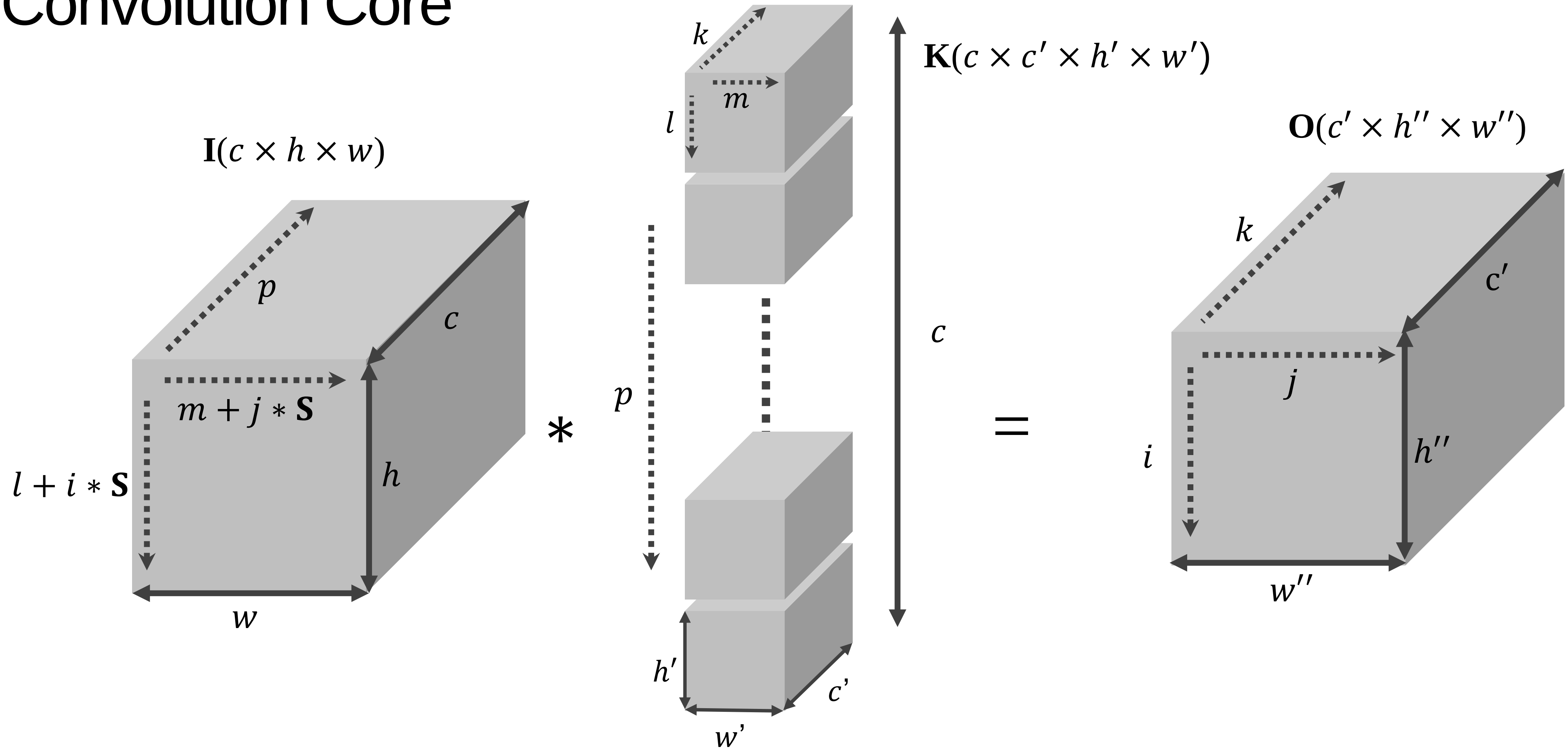
Proposed System: Optimization of Inference Flow

- Fixed-point Optimization / Mathematical Approximation
  - (A) Mathematical Approximation
  - (B) Fixed-point optimization
    - Input/Output and Parameters : Dynamic 8-bit fixed-point
    - Some accumulation adders : 16-bit fixed-point

	Original model (FP32)	A + B accuracy (FxP)	Accuracy loss
MNIST accuracy	99.1%	97.01%	2.1%

Proposed System: Design of Accelerating Components

- Convolution Core



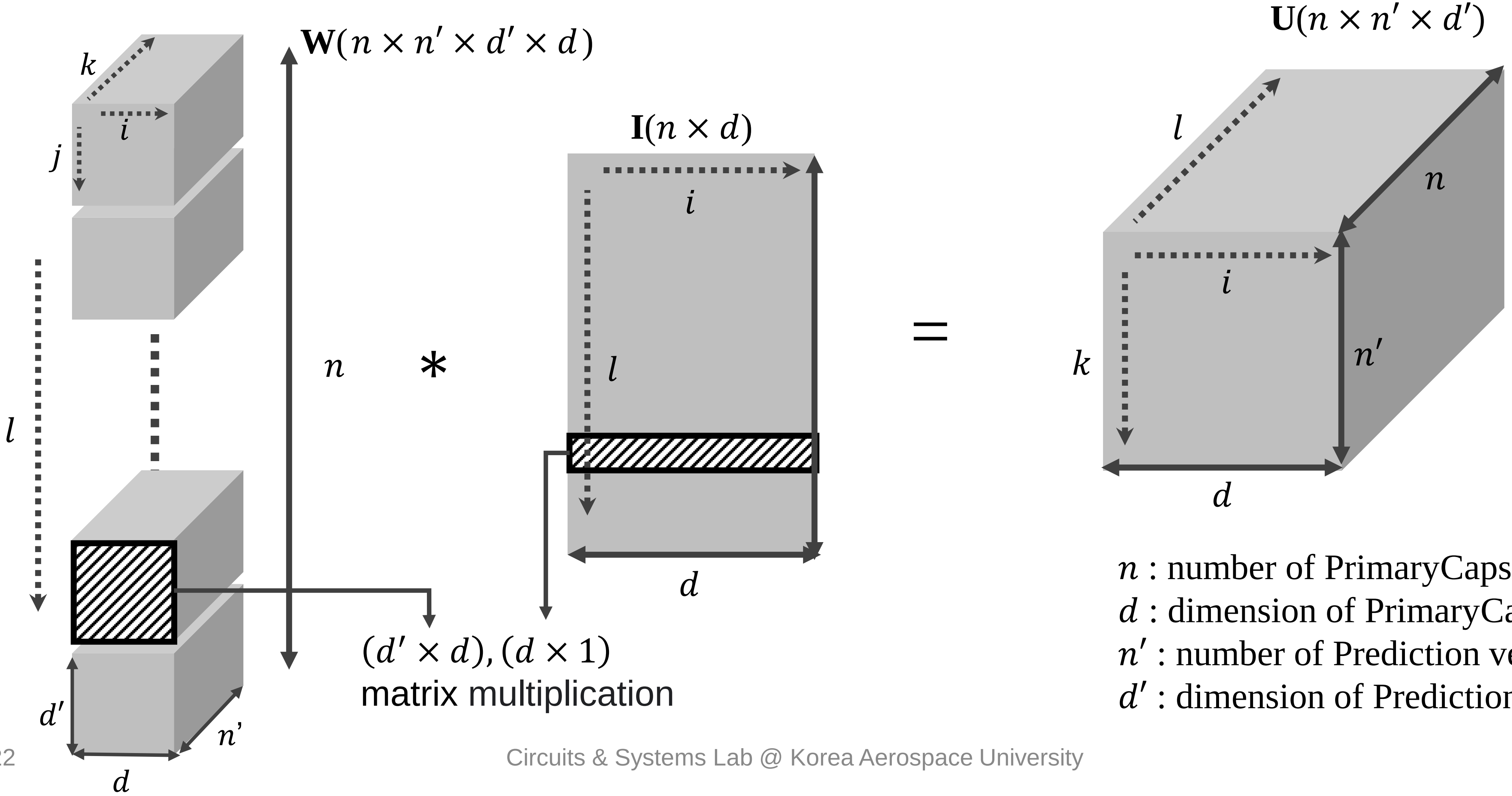
Proposed System: Design of Accelerating Components

- Convolution Core
  - For Accelerating Conv1 and PrimaryCaps layers

Convolution algorithm
Input : input feature **I**$(c \times h \times w)$, kernel **K**$(c \times c' \times h' \times w')$, bias **B**$(c')$, stride **S**
Output : output feature **O**$(c' \times h'' \times w'')$
Loop 0: O(↓) **for** i to from 0 h'' **do**
Loop 1: O(→) **for** j from 0 to w'' **do**
Loop 2: O(↗) **for** k from 0 to c' **do**
$O^{k,i,j} \leftarrow B^{k}$ or 0
Loop 3: I(↗) **for** p from 0 to c **do**
Loop 4: K(↓) **for** l from 0 to h **do**
Loop 5: K(→) **for** m from 0 to w **do**
$O^{k,i,j} \leftarrow O^{k,i,j} + K^{p,k,l,m} \times I^{p,l+iS,m+jS}$

Proposed System: Design of Accelerating Components

- Prediction Core



n : number of PrimaryCaps vectors
d : dimension of PrimaryCaps vectors
n' : number of Prediction vectors
d' : dimension of Prediction vectors

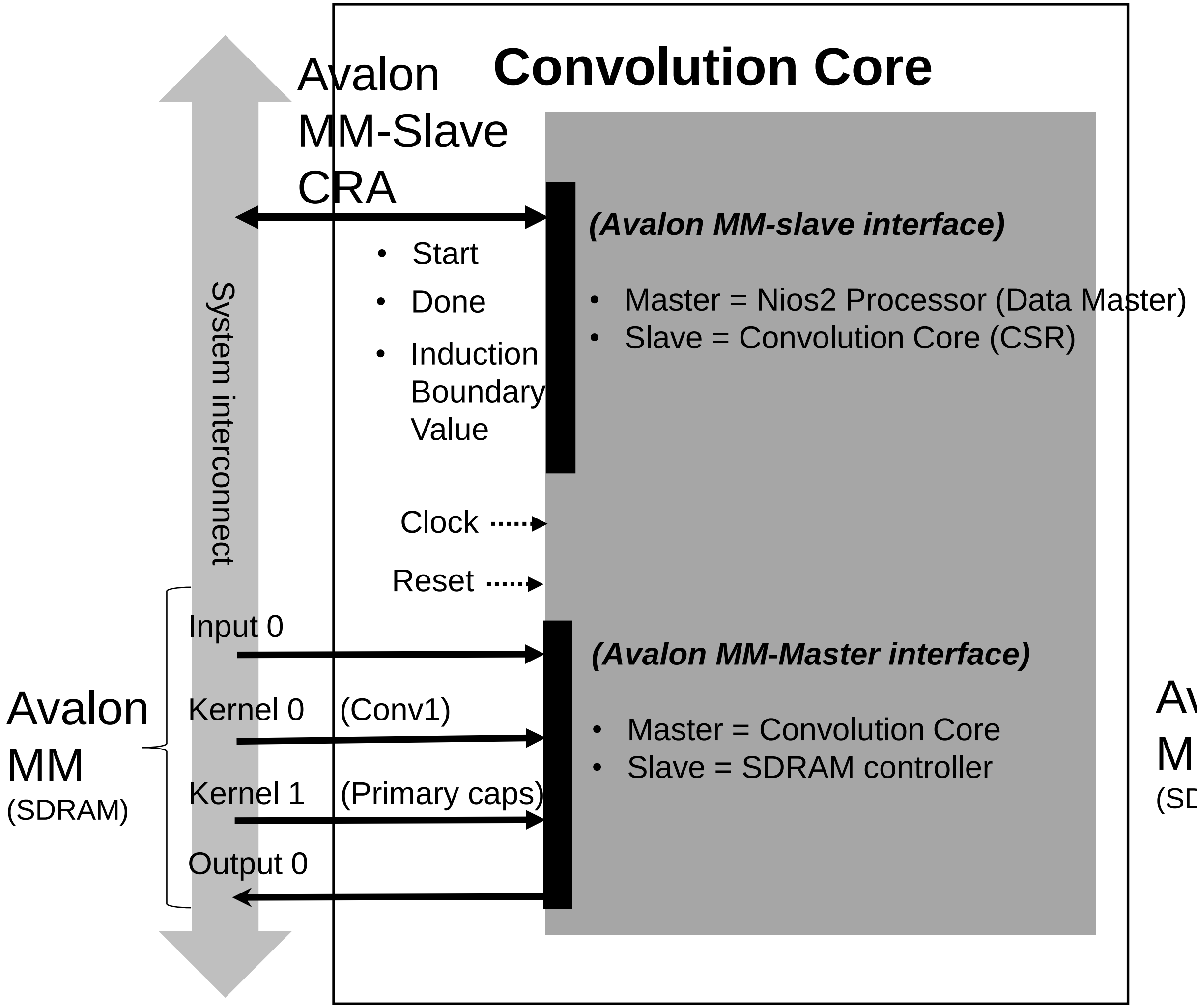
Proposed System: Design of Accelerating Components

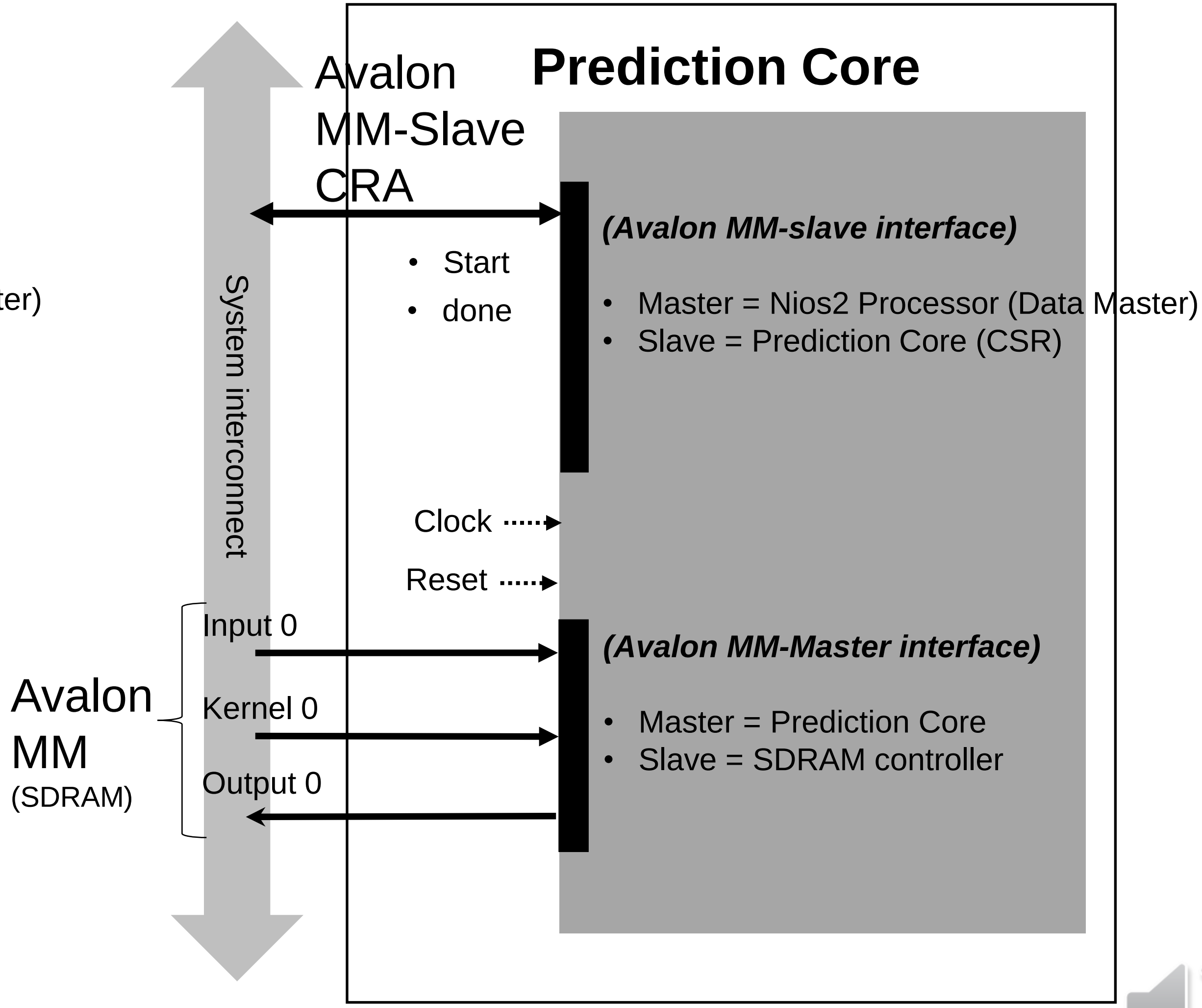
- Prediction Core

Prediction-vector algorithm

Input : input capsules $\mathbf{I}(n \times d)$, digit weight $\mathbf{W}(n \times n' \times d' \times d)$
Output : prediction vector $\mathbf{U}(n \times n' \times d')$
Loop 0: **for** l from 0 to n **do**
Loop 1:   **for** k from 0 to n' **do**
Loop 2:   **for** j from 0 to d' **do**
            $U^{l,k,j} \Leftarrow 0$
Loop 3:   **for** i from 0 to d **do**
                $U^{l,k,j} \Leftarrow U^{l,k,j} + W^{l,k,j,i} \times I^{l,i}$
n : number of PrimaryCaps vectors
d : dimension of PrimaryCaps vectors
n' : number of Prediction vectors
d' : dimension of Prediction vectors

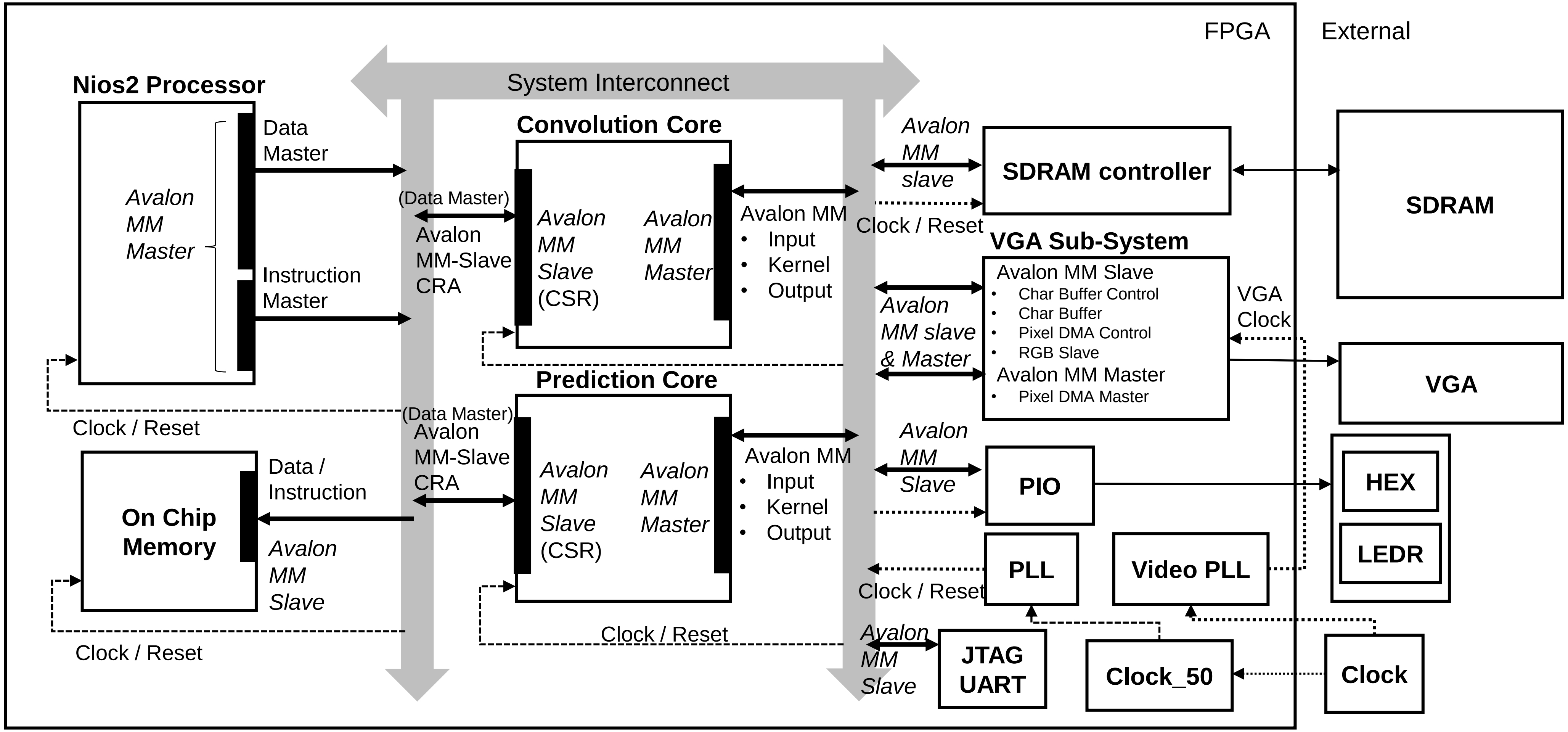
Proposed System: Design of Accelerating Components





Proposed System: Prototype Acceleration System

- Overall Architecture



Results: Resource Usage

- Estimated Resource Usage (Cyclone V, 5CSEMA5F31C6)
  - Convolution Core

	ALUTs	FFs	RAMs	DSPs
Conv Core(util%)	9,023 (8%)	13,676 (6%)	43 (8%)	14 (13%)
Available	109,572	219,144	514	112

  - Prediction Core

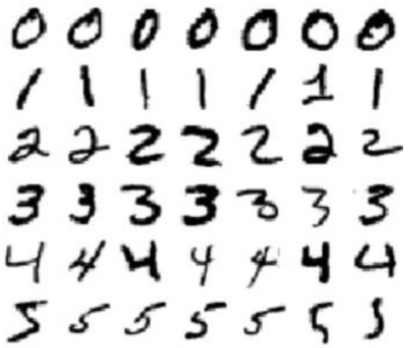
	ALUTs	FFs	RAMs	DSPs
Prediction core	5,231 (5%)	13,477 (6%)	80 (16%)	0.5 (1%)
Available	109,572	219,144	514	112

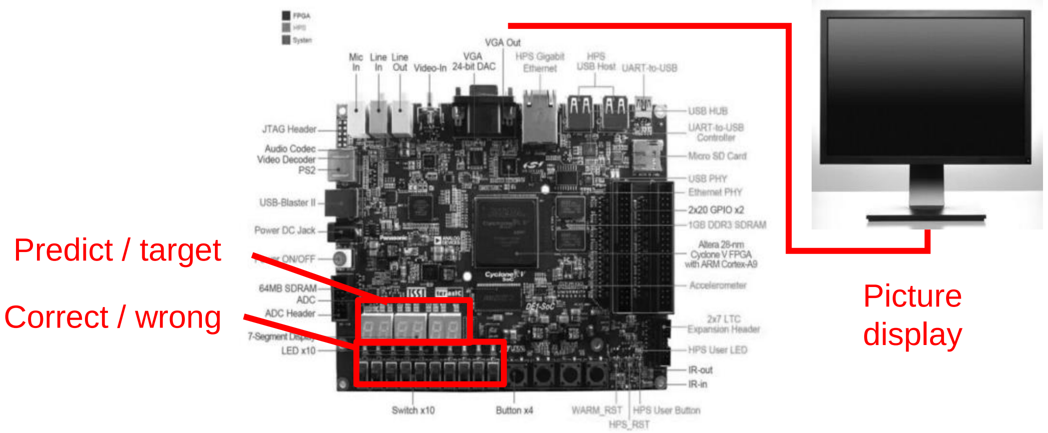
  - Fitter Summary

Device	5CSEMA5F31C6
Logic utilization	10,215 / 32,070 (32%)
Total registers	20,581 (----)
Total pins	241 / 457 (53%)
Total block memory bits	1,212,712 / 4,065,280 (30%)
Total RAM Blocks	261 / 397 (66%)
Total DSP Blocks	19 / 87 (22%)

Results: Implementation of Accelerator on DE1-SoC

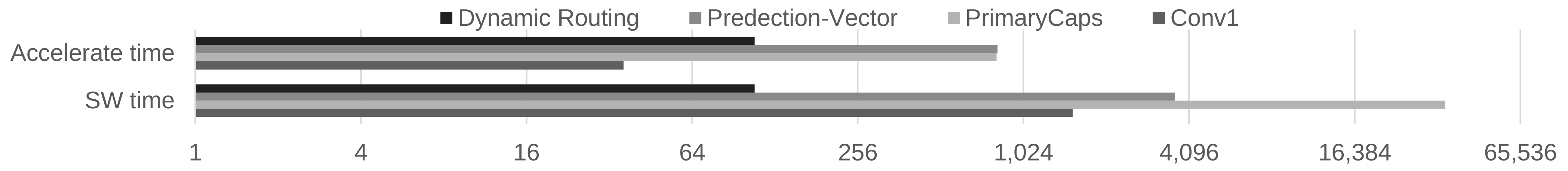
- Dataset: MNIST

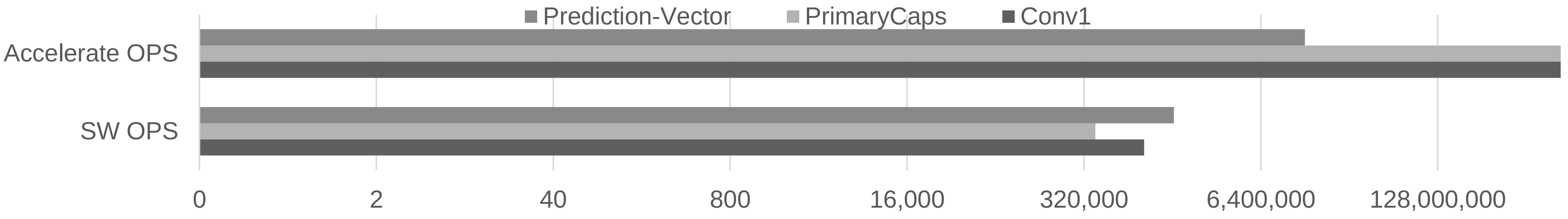




Evaluation and Discussion

Layer	Operation Count (OP)	Time (Not Accelerated)	Speed (OP/s)	Time (Accelerated)	Accelerate Speed (OP/s)	Speedup
Conv1	381,542,400	1,545 ms	246,953,009 OPS	36 ms	10.6 GOPS	43x
PrimaryCaps	5,913,331,816	34,932 ms	169,281,226 OPS	817 ms	10.9 GOPS	64.4x
Prediction-Vector	111,006,720	3,645 ms	30,454,518 OPS	825 ms	0.135 GOPS	4.4x
Multiplication : 4 Operations / Add : 1 Operations						





Evaluation and Discussion

- Comparison with Previous Work

	TCSII 2020	This Work
Target Model (Application)	3D Capsule Network (Point Segmentation)	2D Capsule Network (Image Classification)
Clock Frequency	200 MHz	50 MHz
Precision	4/8 bit fixed-point	8bit Dynamic fixed-point
# of DSP	278	19
Peak Performance	96 GOPS	10.9 GOPS
Performance(GOP)/DSP	0.39	0.78

Implemented on Cyclone V

Reference : Gwangtae Park, “A 1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment”, TCSII 2020

Conclusion

- We have optimized the Capsule Network inference flow for embedded systems
  - **39.2x speed up (FP➔FxP)**
  - **Accuracy loss: as little as 2% (FP➔FxP)**
  - **Memory usage reduction : 93.36 %**
- We have accelerated Conv1, PrimaryCaps and Prediction-vector layer using HLS
  - **Increased Operation Per seconds of main bottleneck : 0.17(GOPs)➔10.9 (GOPs)**
  - **23.9x speed up (FxP)**
- The proposed Capsule Network acceleration system have been successfully demonstrated with the high performance per DSP ratio
  - **2x higher in terms of GOP/DSP**

Appendix: Model Optimization

Before / After Layer	Before Optimization (FP32)			After Optimization (Our Work: FxP8 + Model Reduction)		
	Input (Byte)	Parameters (Byte)	Outputs (Byte)	Input (Byte)	Parameters (Byte)	Outputs (Byte)
Conv1	3,136	83,968	406,400	784	1,312	6,400
PrimaryCaps	406,400	21,234,688	4,608	6,400	332,032	1,152
Prediction Vectors	4,608	5,898,240	36,864	1,152	1,474,560	9,216
Dynamic Routing	36,864	0	640	9,216	0	160
Total size of Parameters	27,216,896 byte			1,807,904 byte		

Memory Usage Reduction : 93.36 %

Idea 및 결과

- Main Idea
  - Reduction of Computational Complexity
    - 8bit Fixed Point Optimization
    - Mathematical Approximation
  - HLS(High Level Synthesis)-based Acceleration
- Result
  - 39.2x speedup
  - Main Bottleneck (Conv1, PrimaryCaps Layer)
    - 0.17 GOPs ➔ 10.9 GOPs
  - 2x Higher in terms of **GOP/DSP**