Semantic Segmentation Optimization

By: SDDec25-01

SDDec25-01

Team Members:

Joey Metzen - CprE

Aidan Perry - CprE

Conner Ohnesorge - EE

Tyler Schaefer - SE

Advisor:

Dr. Namrata Vaswani

Client:

JR Spidell

Problem Statement

Problem

- People with disabilities face risks from undetected medical issues. Traditional methods lack real-time monitoring.
- Using eye movement tracking with semantic segmentation can detect warning signs and automatically reposition users to prevent incidents, improving safety needs.

Client

- Volunteered to help individuals with cerebral palsy.
- Create an assistive wheelchair technology.

Team

- Update the system to increase throughput.

Resources/Data Management

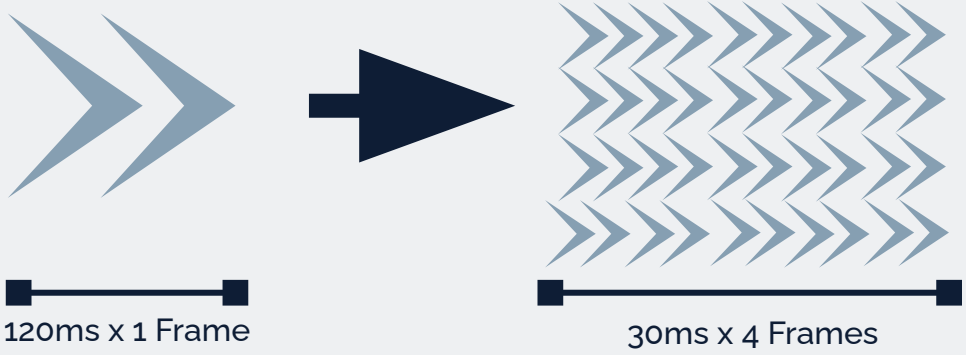
Hardware/Software:

- Kria Board Kv260
- Xilinx
- Vitis-AI
- Pytorch
- ONNX & ONNX-Runtime
- Petalinux

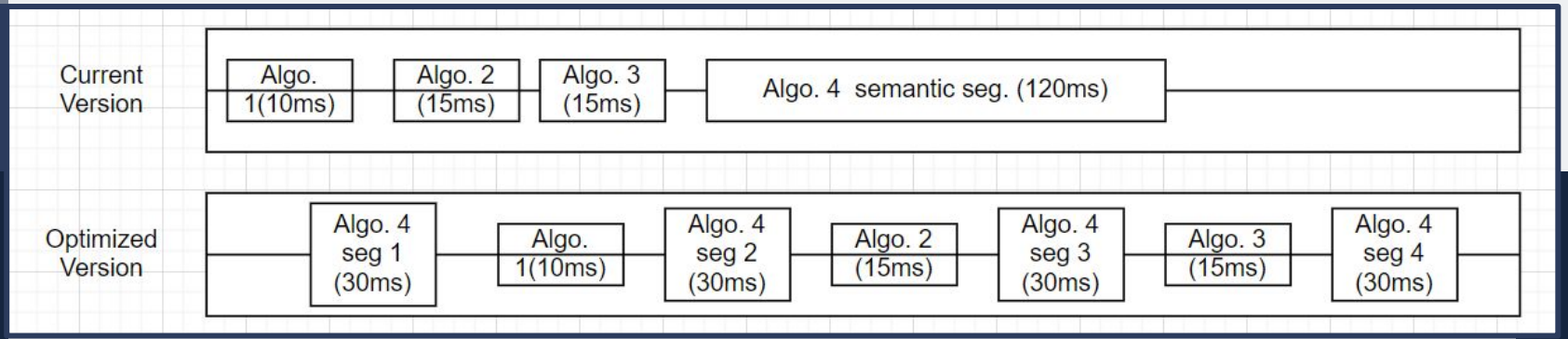
Mem Components:

- DDR4 RAM (Main Memory) [4GB]
- Quad-SPI Flash (Boot ROM) [64MB]
- Cortex-A53 L1 Data Cache
- Cortex-A53 L2 Cache [1MB]

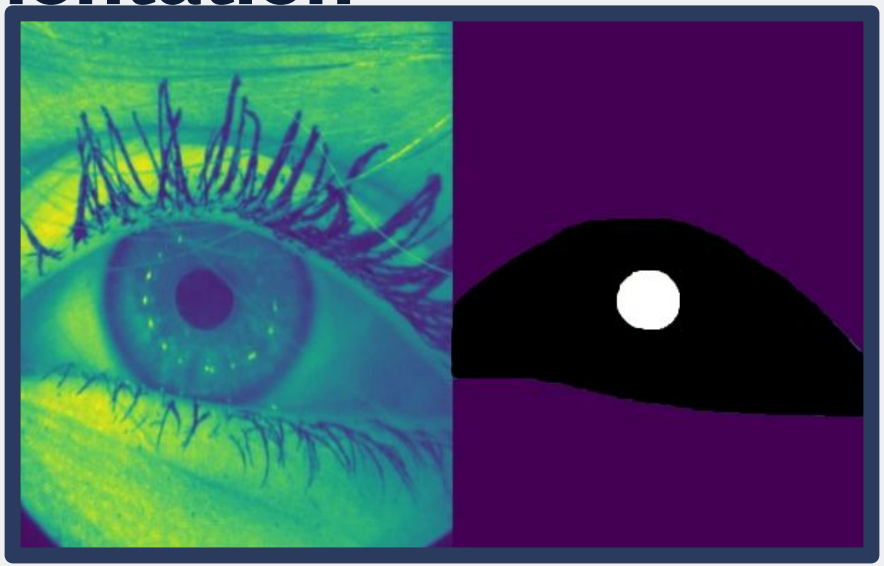
Project Overview



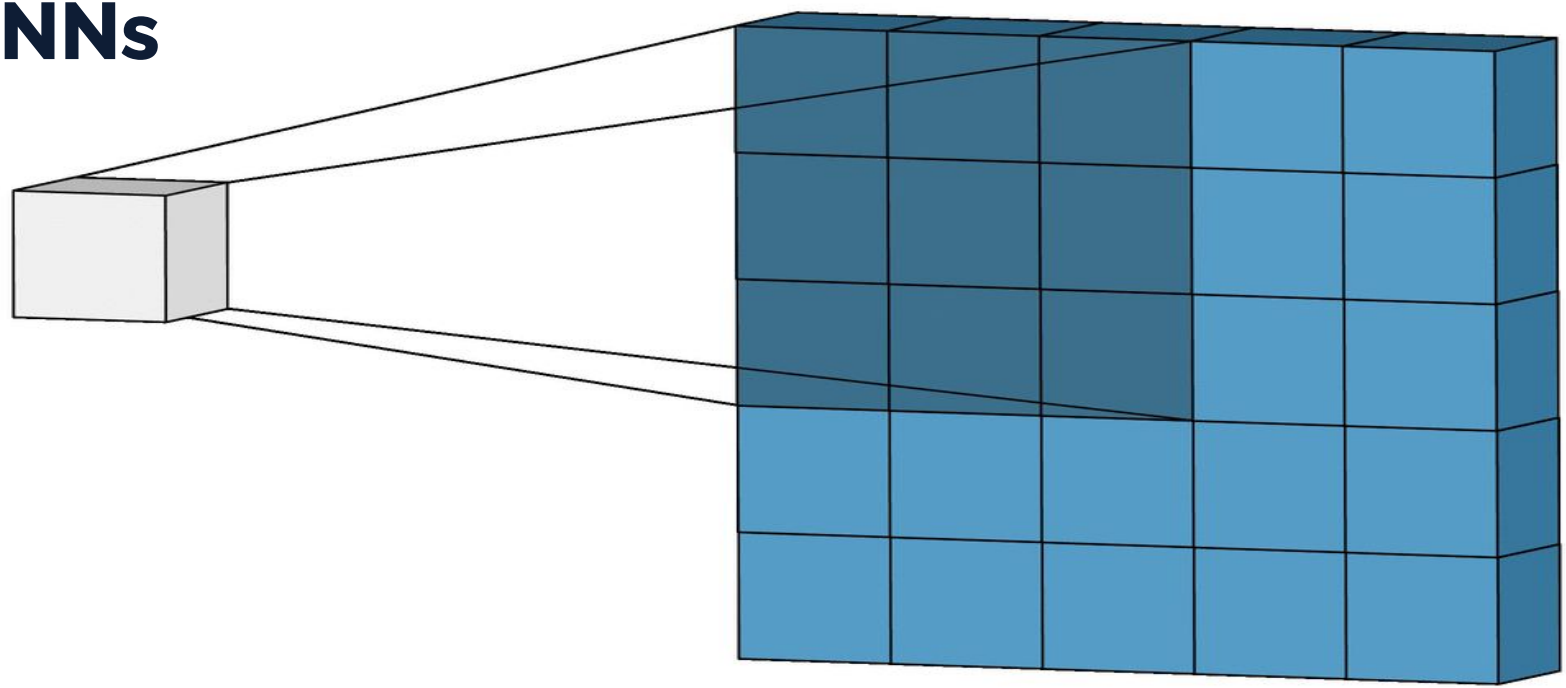
Increase throughput by splitting the U-net algorithm over 4 cores and across the DPU.



What is Semantic Segmentation



CNNs



- Convolution = sliding window
- NN basically Turns func(3x3) into func(...func(func(3x3)))

U-net Semantic Segmentation cont.

Contracting Encoder

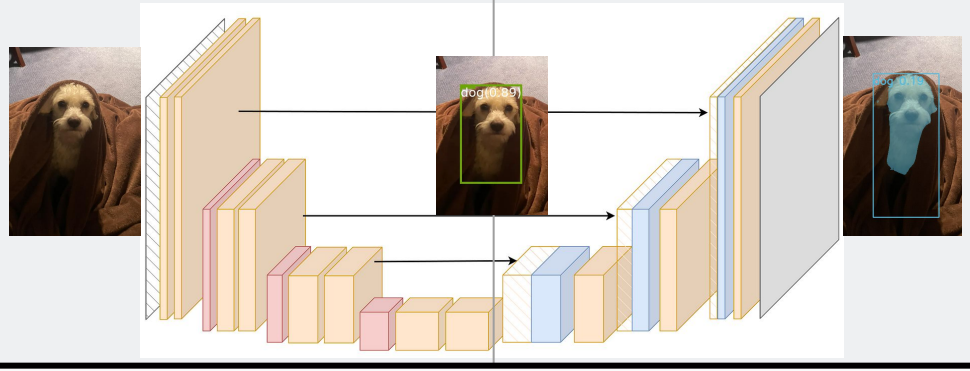
- Downsampling (i.e. 2x2 Max Pool) compensated by the doubles # channels
- Transmit to across to decoder

Includes spatial info

Expanding Decoder

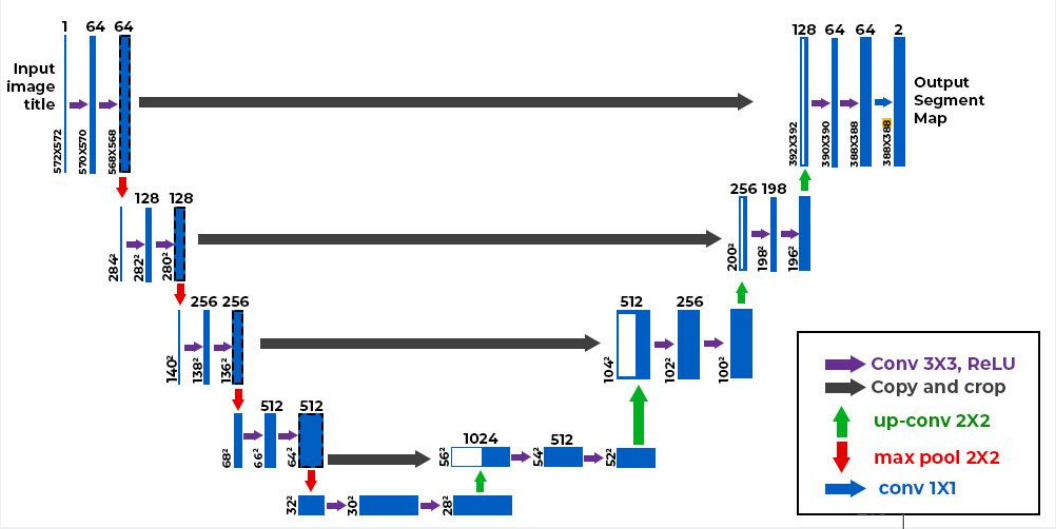
- Receive from encoder
- Upsampling (i.e. 2x2 Convolution) decreases the amount of channels

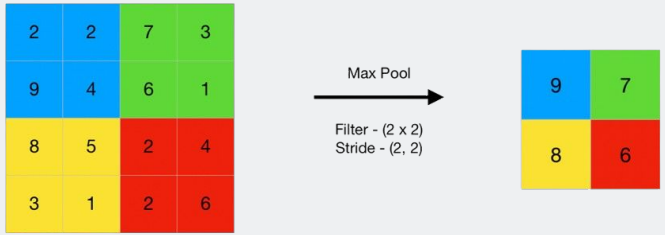
Includes semantic Info

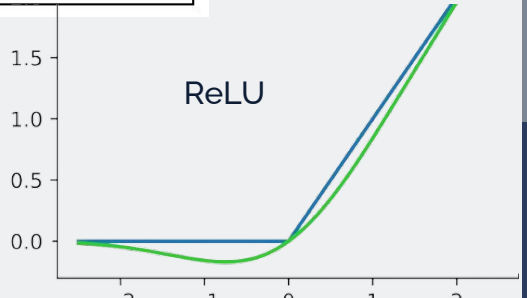


Each “forward step” applies a relu function to the output of a repeated convolutional layer application over input channels.

U-net Semantic Segmentation

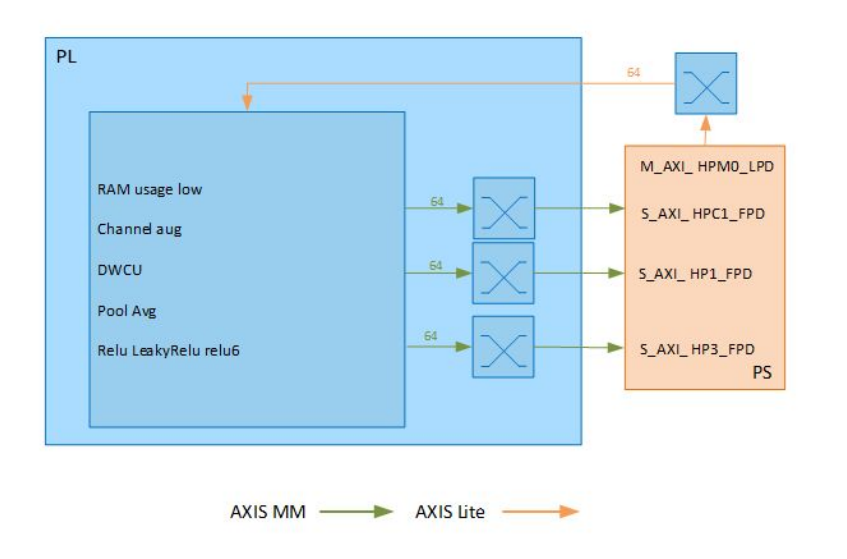






Quantization

32 fp -> 8b int
Less precise
More processable
DPU needs integer inputs
Post Training Quantization



Data Version Control

- S3 compatible store
  Versioned on top of Git
- Distributed (Supports cross country cooperation)
- Low Learning Curve

Questions before we continue?

Why Unet?

Why not Google drive?

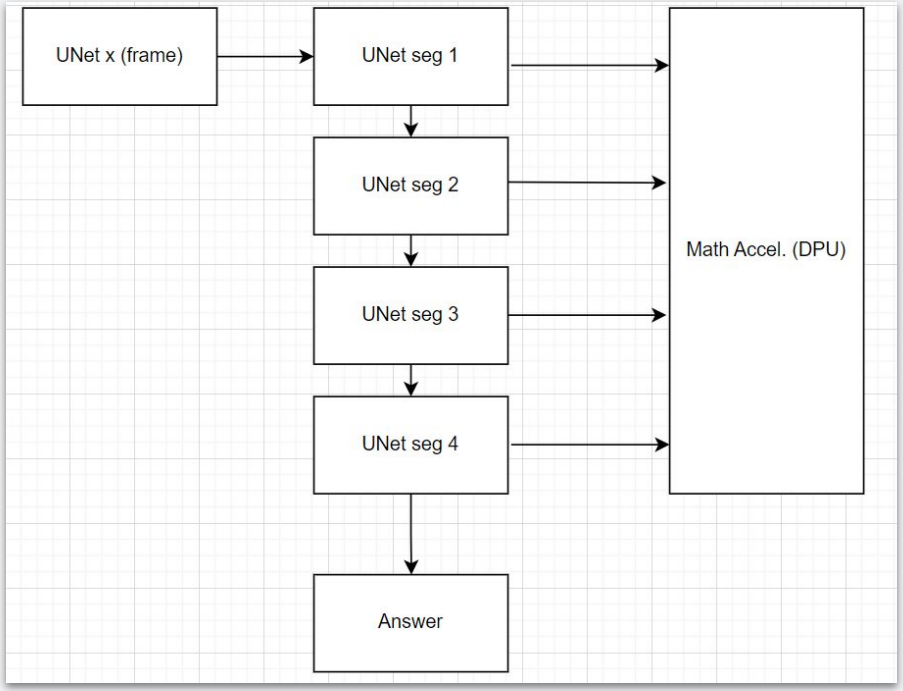
What is S3?

Task Decomposition and Decision Making

Member Roles

Role1 - Tyler	Dividing out the algorithm into 4 equal parts, code it in a way to have 4 segments to produce the same result and accuracy as prior design.	Skillset: equal math distribute, reasonable coding capability.
Role2 - Aidan	Implement algorithm into current codebase, (already has 4 pipelines) Understand how each thread works on a part of the equation.	Skillset: C++ expertise (OOP)
Role3 - Conner	Works with OS Docker environment to optimize all of the activities that need to happen. Splitting up the Onnx for loading into the MPU for Multicore Loading and Layering.	Skillset: OS Knowledge, comfort with scheduler (what is done when), Docker.
Role4 - Joey	Understand Hardware (Kira board) inside and out to give other members accurate benchmarks of what the board can handle.	Skillset: Hardware focused, intensive data sheet research.

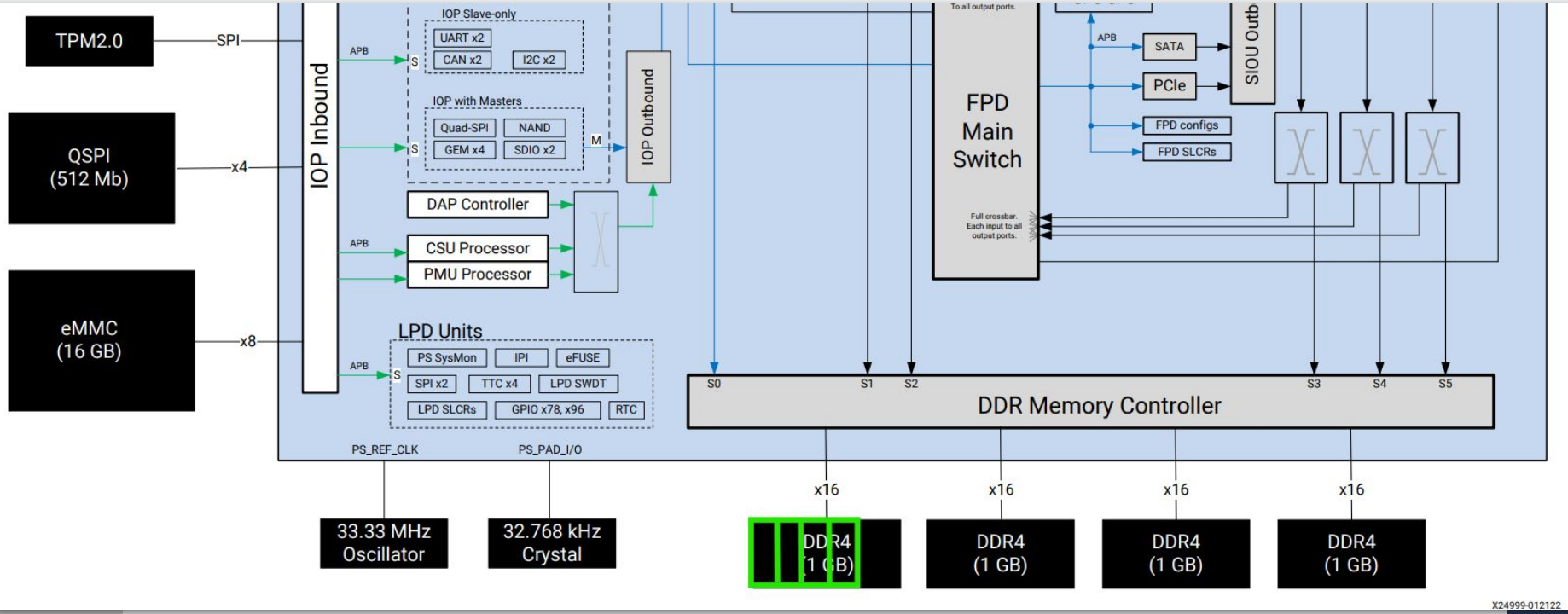
Proposed Pipelining



Tracked Metrics

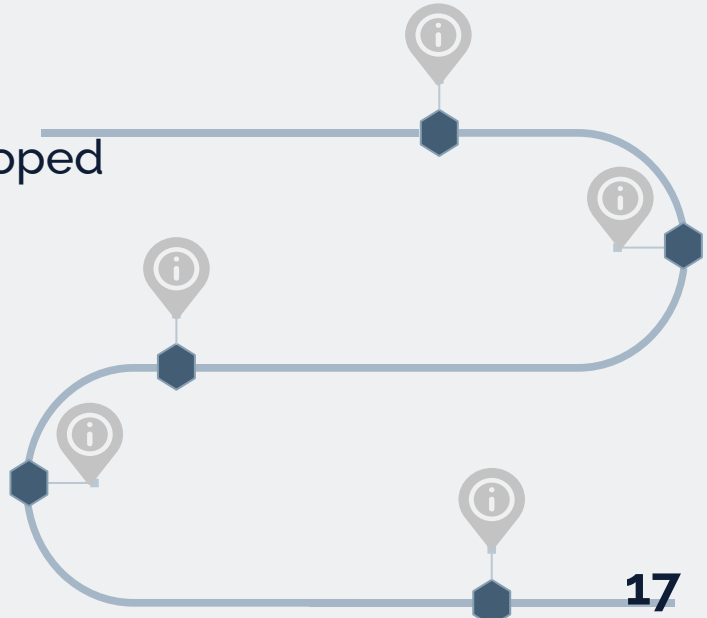
- Throughput
- Accuracy
- Resource Utilization

Emphasis on Resource Utilization



Milestones

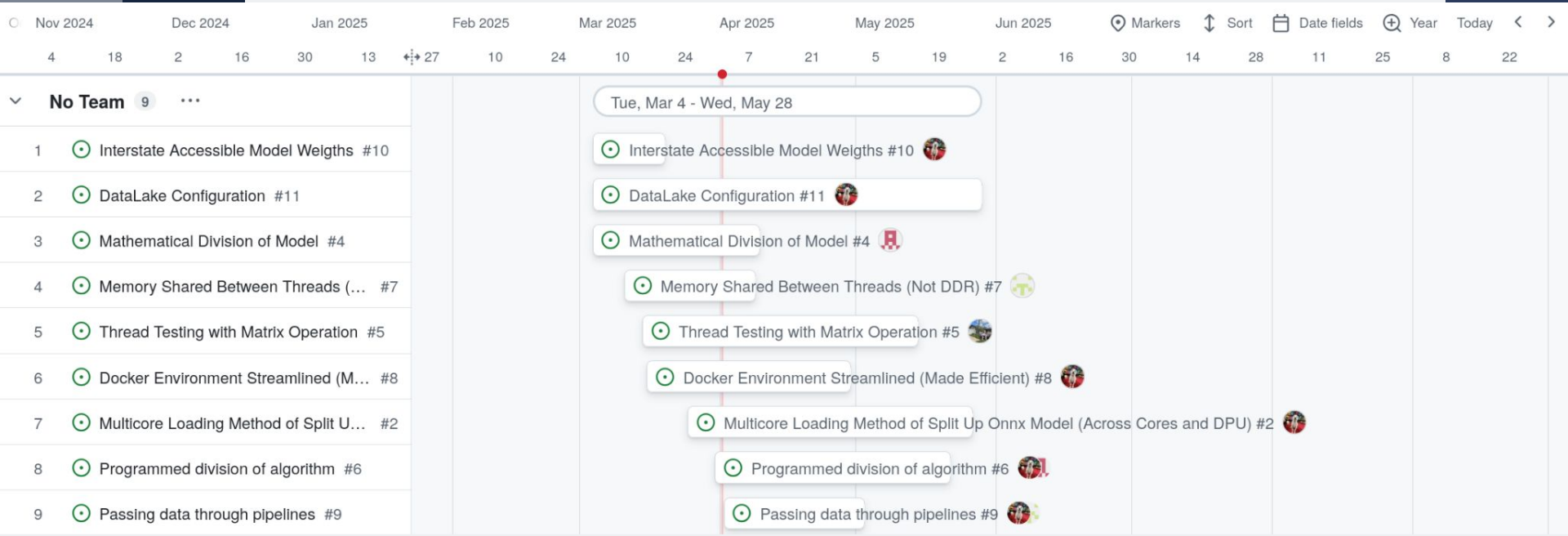
- Mathematical division of the Algorithm
- Loading of Split Algorithm weights onto DPU
- Pipelined Implementation of the Semantic Segmentation algorithm across the 4 developed threads.
- Increased Throughput over multiple frames



Project Management Style

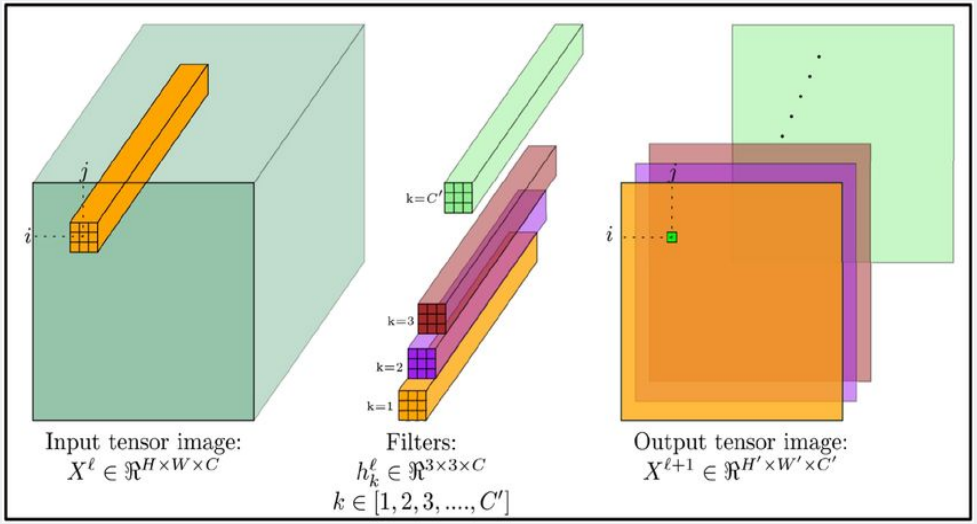
Waterfall & Agile

Gantt Chart


Nov 2024
Dec 2024
Jan 2025
Feb 2025
Mar 2025
Apr 2025
May 2025
Jun 2025
Markers
Sort
Date fields
Year
Today
4 18 2 16 30 13 27 10 24 10 24 7 21 5 19 2 16 30 14 28 11 25 8 22
No Team 9
1 Interstate Accessible Model Weigths #10
2 DataLake Configuration #11
3 Mathematical Division of Model #4
4 Memory Shared Between Threads (… #7
5 Thread Testing with Matrix Operation #5
6 Docker Environment Streamlined (M… #8
7 Multicore Loading Method of Split U… #2
8 Programmed division of algorithm #6
9 Passing data through pipelines #9
Tue, Mar 4 - Wed, May 28
Interstate Accessible Model Weigths #10
DataLake Configuration #11
Mathematical Division of Model #4
Memory Shared Between Threads (Not DDR) #7
Thread Testing with Matrix Operation #5
Docker Environment Streamlined (Made Efficient) #8
Multicore Loading Method of Split Up Onnx Model (Across Cores and DPU) #2
Programmed division of algorithm #6
Passing data through pipelines #9

Segmenting the Model

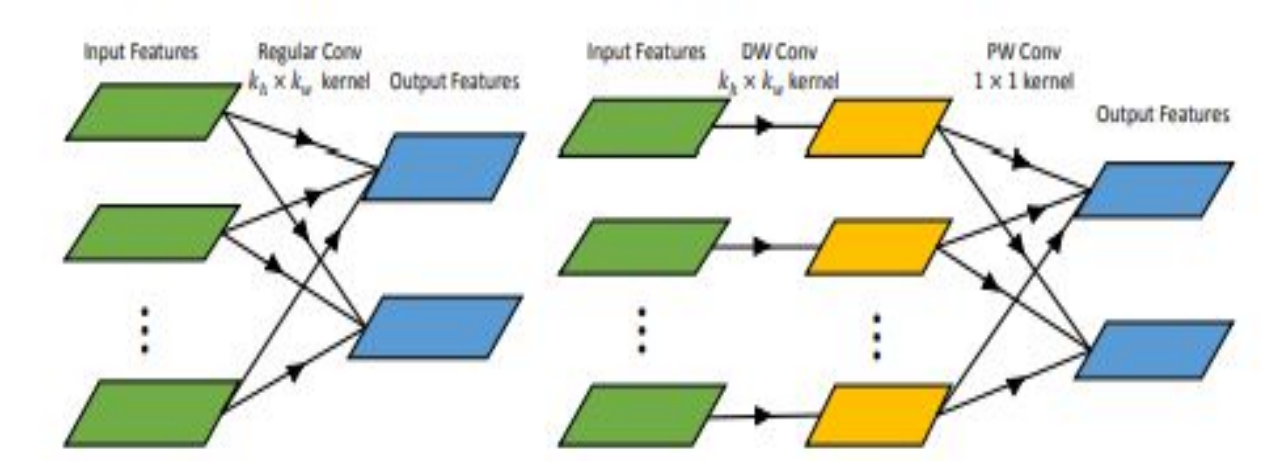
Computational Complexity of Model



Traditional Convolutions

O(H x W x Cin x K x Cout)

Computational Complexity Analysis



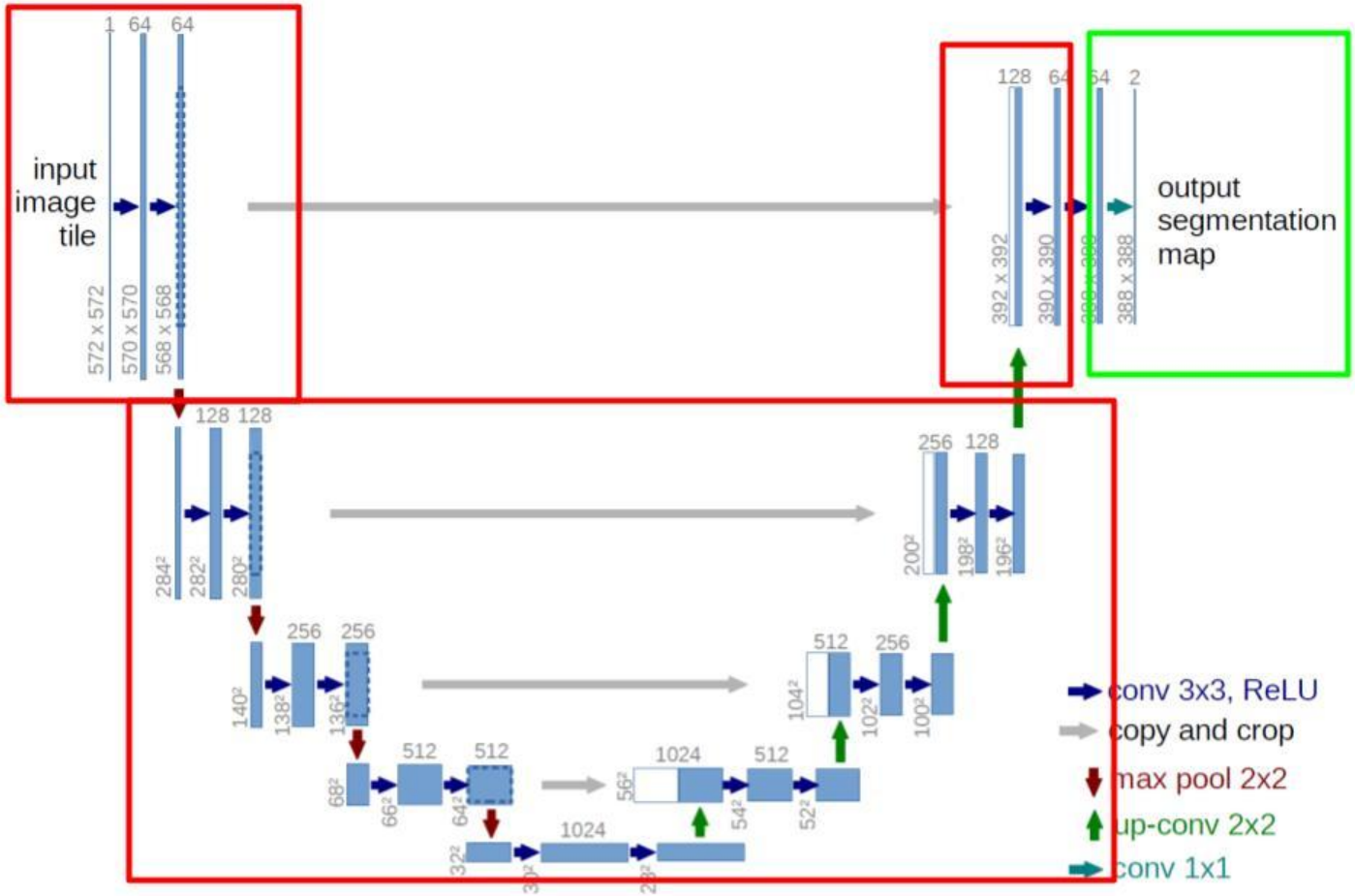
Depthwise Separable Convolutions

Depthwise

O(H x W x Cin x K)

Pointwise

O(H x W x Cin x Cout)


input image tile
output segmentation map
1 64 64
572 x 572
570 x 570
568 x 568
128 64 64 2
392 x 392
390 x 390
388 x 388
128 128
284²
282²
280²
256 128
200²
198²
196²
256 256
140²
138²
136²
512 256
104²
102²
100²
512 512
68²
66²
64²
1024 512
56²
54²
52²
1024
32²
30²
28²
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Conclusion

Problem Solved: Real-time monitoring for individuals with disabilities using eye tracking.

Outcome: Improve safety and throughput with pipelined U-Net on DPU.

Next Steps: Optimize Performance and Thorough Safety Testing.

Thank You :)

Questions ?