CS6490 Assignment 2

Vishal Vijay Devadiga (CS21BTECH11061)

- A short summary on Scale-Sim
  - Introduction
  - Implementation Details
- Metrics and Results
  - Q3: Analyzing the impact of different DNNs and configurations
    - Mapping Efficiency and Compute Utilization
    - Total Runtime
    - DRAM Bandwidth Utilization and Read/Writes
    - SRAM Bandwidth Utilization and Read/Writes
  - Changing dataflows
    - Total Runtime
    - Mapping Efficiency and Compute Utilization
    - DRAM Bandwidth Utilization and Read/Writes
    - SRAM Bandwidth Utilization and Read/Writes
  - Changing buffer sizes
    - DRAM Bandwidth Utilization and Read/Writes
  - Changing systolic array sizes
    - Total Runtime
    - DRAM Bandwidth Utilization and Read/Writes
- References

A short summary on Scale-Sim

Introduction

Scale-Sim: SystoliC AcceLErator SIMulator

Scale-Sim is a cycle accurate simulator for DNN inference on systolic arrays.

It serves to explore both scale-up (increasing compute units) and scale-out (adding more chips) models, enabling the investigation of various systolic array architectures with different memory hierarchies, runtime behaviors, dataflows, and DRAM bandwidth requirements.

The simulator comprises two key components:

- Compute Unit: This unit is based on a parameterized systolic array, with size and aspect ratio as configurable parameters.
- Accelerator Memory System: It includes three double-buffered SRAM memories of specified sizes, dedicated to buffering matrices for two operands and one result.

Before initiating the simulation, Scale-Sim requires three inputs:

- Configuration File: This file outlines the parameters of the systolic array, memory system, dataflow, and DRAM bandwidth.
- Architecture File: It specifies all layers in the DNN along with their respective parameters.
- Logging Path: This path designates where the simulation results will be stored.

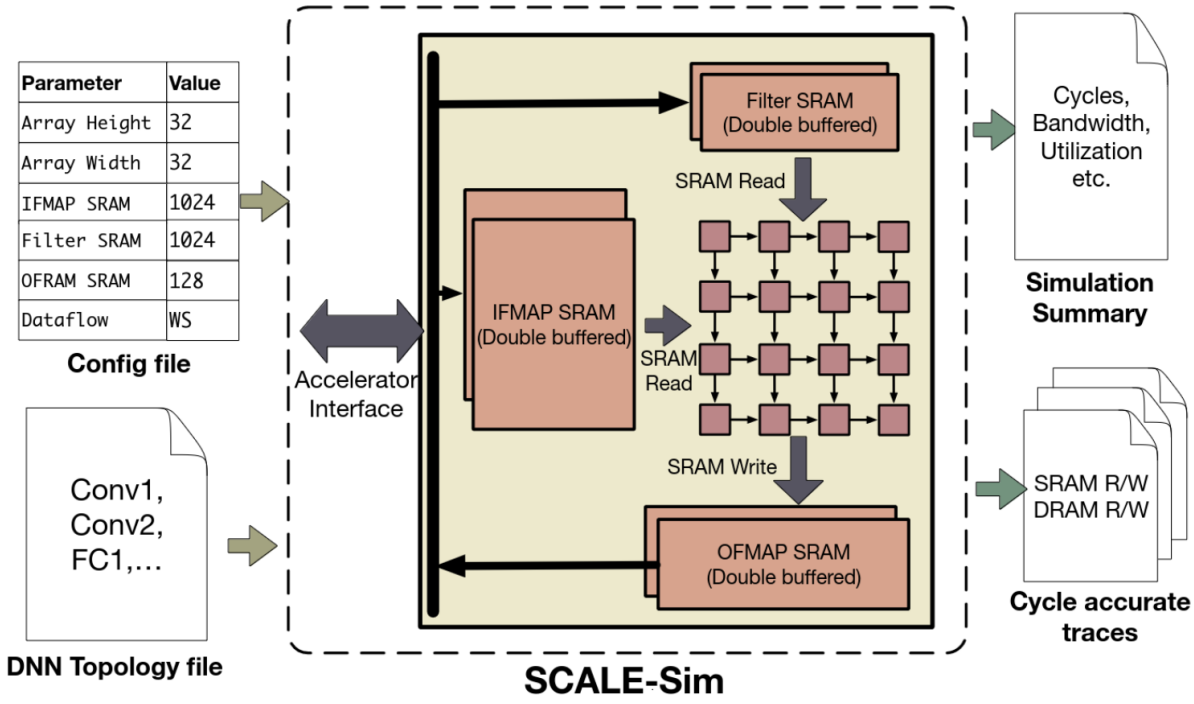


Figure 1: Reference: Scale - Sim Paper

Implementation Details

The simulator operates under the following assumptions:

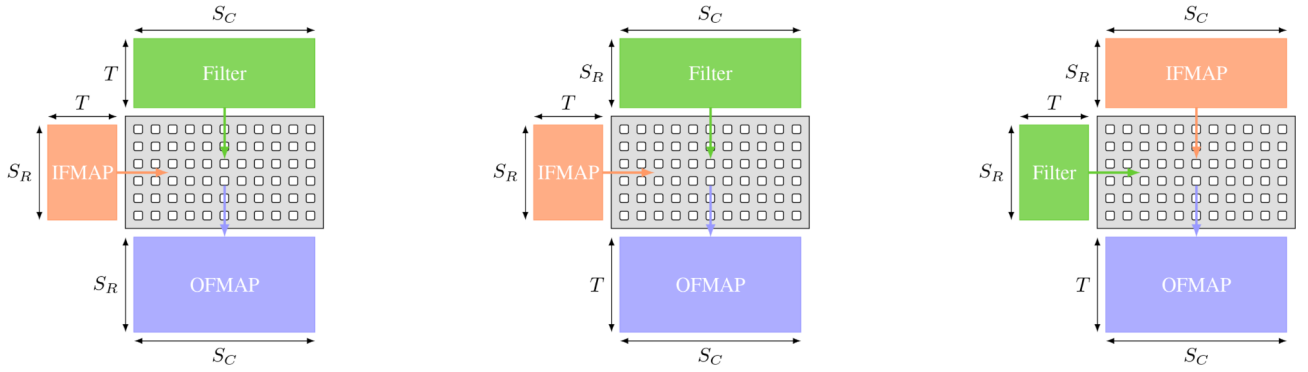
- The accelerator is consistently compute-bound.
- Processing elements (PEs) are maximally utilized.

It generates cycle-accurate read addresses for elements to ensure uninterrupted operation of the PE array. The resulting output trace includes SRAM write traffic for the output matrix. The simulator analyzes this traffic to calculate the total runtime and assesses array utilization on a per-cycle basis.

Both input matrices and generated elements are managed by dedicated SRAM buffers using a double-buffered mechanism. By parsing SRAM traces, the simulator optimizes the filling of these buffers to prevent any SRAM request from encountering a miss.

DRAM traces, which comprise prefetch requests to SRAM, inform the estimation of interface bandwidth requirements. Further parsing of trace data at both the SRAM and interface levels allows for the determination of compute efficiency and other relevant metrics.

Metrics and Results



(a) Output stationary dataflow.

(b) Weight stationary dataflow.

(c) Input stationary dataflow.

Fig. 5: Data Flow Mapping.

Figure 2: Reference: Scale - Sim Paper

TABLE III: Spatio-Temporal Allocation of DNN Dimensions

	Spatial Rows (S_R)	**Spatial Columns** (S_C)	**Temporal** (T)
Output Stationary	N_{ofmap}	N_{filter}	W_{conv}
Weight Stationary	W_{conv}	N_{filter}	N_{ofmap}
Input Stationary	W_{conv}	N_{ofmap}	N_{filter}

N_{filter} : Number of convolution filters
N_{ofmap}: Number of OFMAP pixels generated by filter
W_{conv} : Number of partial sums generated per output pixels

III. Analytical Model for Runtime

Figure 3: Reference: Scale - Sim Paper

S_R is spatial row, S_C is spatial column. T is the number of timesteps.

N_{ofmap} is the number of OFMAP pixels generated by the filter.

W_{CONV} is number of partial sums generated per output pixel.

N_{filter} is the number of convolutional filters.

If the size of the array is $R \times C$, then the folds are defined as:

- $F_R = \lceil \frac{S_R}{R} \rceil$
- $F_C = \lceil \frac{S_C}{C} \rceil$

Also, $T_F = 2R + C + T - 2$.

Thus, total runtime is $T_F \times F_R \times F_C$.

Note that N_{ofmap} is quite large.

The following metrics were used to analyze the experiments:

- Mapping Efficiency
- Compute Utilization
- DRAM Bandwidth Utilization for Input, Filter, and Output
- SRAM Bandwidth Utilization for Input, Filter, and Output
- DRAM Reads and Writes for Input, Filter, and Output
- SRAM Reads and Writes for Input, Filter, and Output

Overall around 167 graphs were generated for the above metrics.

Mapping Efficiency is defined as Utilized PEs / Total PEs per **iteration**. Thus it is a measure of how well the systolic array is utilized per iteration.

Compute Utilization is defined as Utilized PEs / Total PEs per **cycle**. Thus it is a measure of how well the systolic array is utilized per cycle.

Q3: Analyzing the impact of different DNNs and configurations

Six different DNNs were used: mobilenet, resnet18, alexnet, googlenet, fasterRCNN, and yolo_tiny.

Three different configurations were used: eyeriss, googleTPU, and scale.

All the DNNs were run on all the configurations.

Mapping Efficiency and Compute Utilization

Mapping utilization depends on S_R, S_C, R, C.

Compute utilization depends on S_R, S_C, R, C, T.

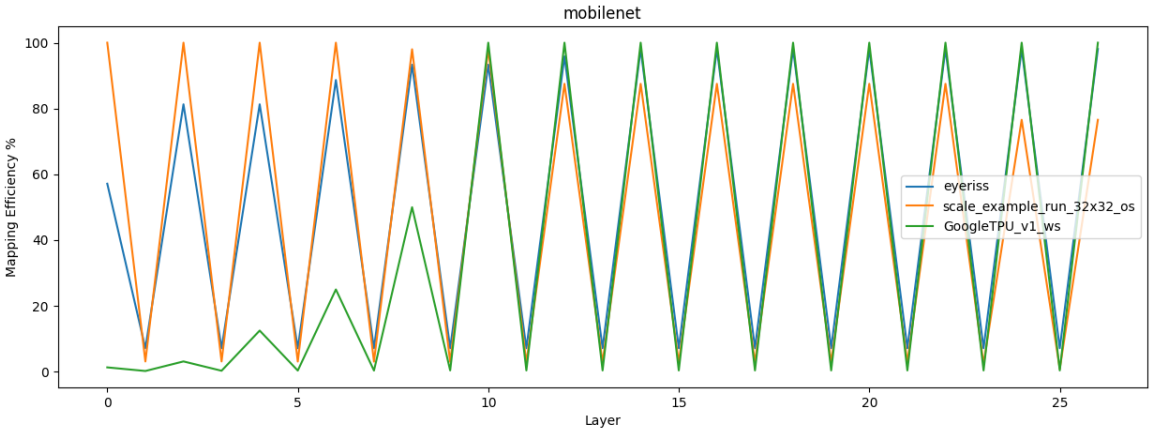


Figure 4: mobilenet_mapping

For mobilenet, the mapping efficiency alternates between big and small values due to the number of filters. The pattern of number of filters in mobilenet is like 32, 1, 64, 1, 128, 1, ... and so on. This causes the mapping efficiency to alternate between big and small values since the number of filters is the main factor in determining the mapping efficiency in OS and WS dataflows.

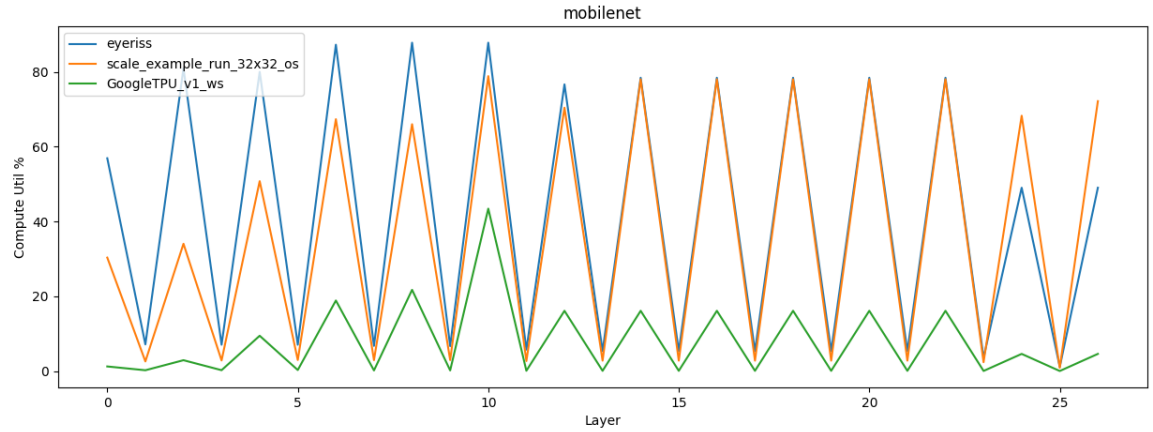


Figure 5: mobilenet_compute

We see a similar order in compute utilization as well.

The graph for GoogleTPU is lower in magnitude due to the fact that the systolic array is larger in GoogleTPU than in eyeriss and scale.

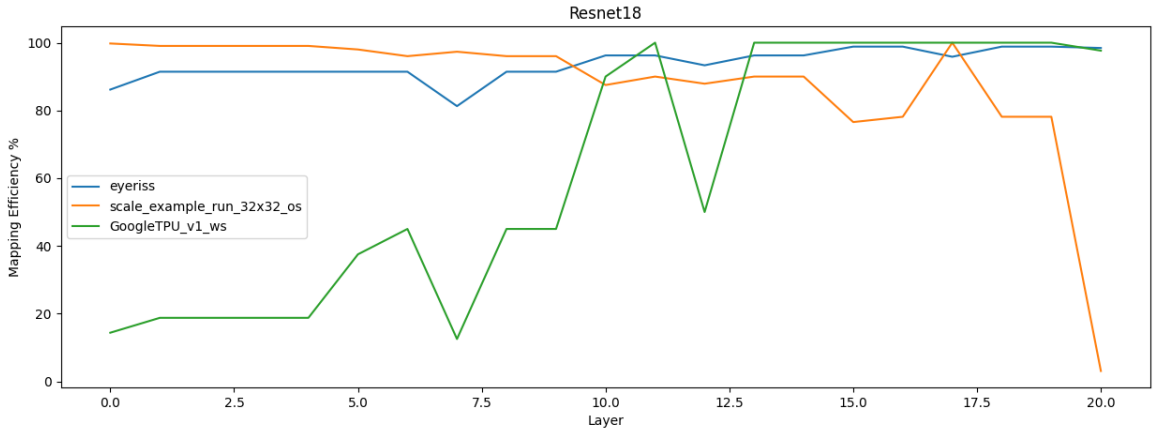


Figure 6: resnet18_mapping

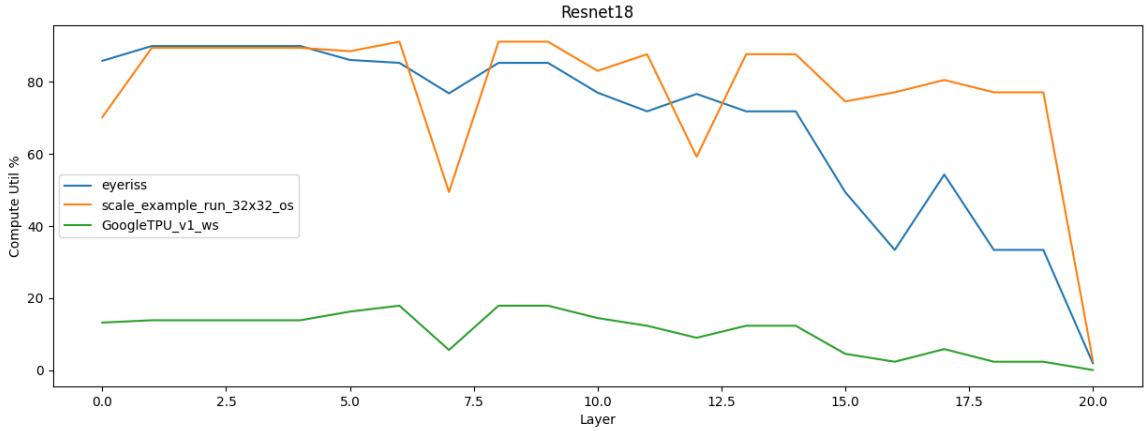


Figure 7: resnet18_compute

A similar order is observed in resnet18 as well.

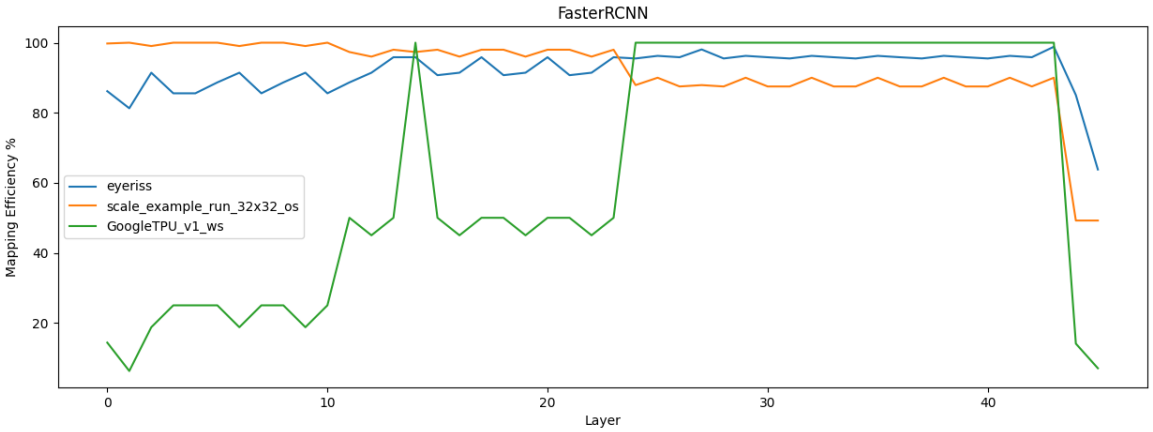


Figure 8: fasterRCNN_mapping

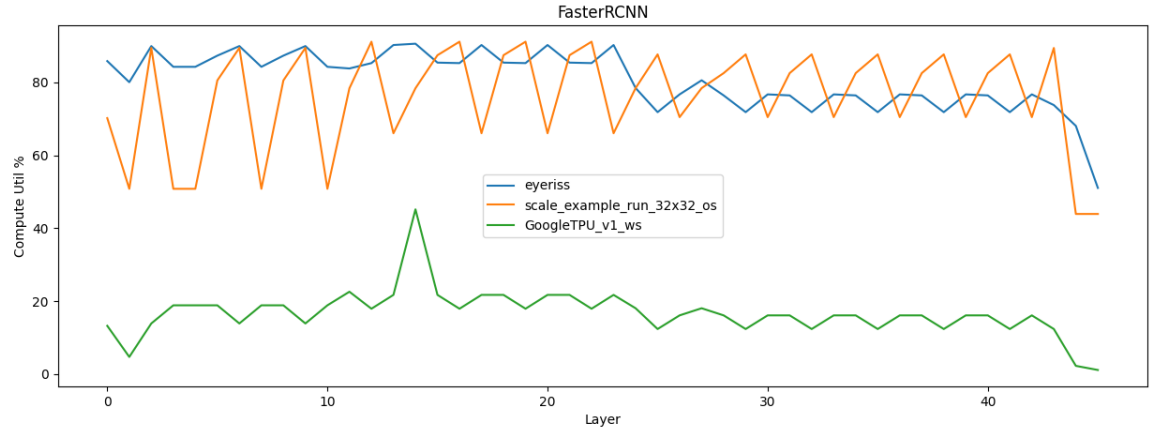


Figure 9: fasterRCNN_compute

A similar order is observed in fasterRCNN as well.

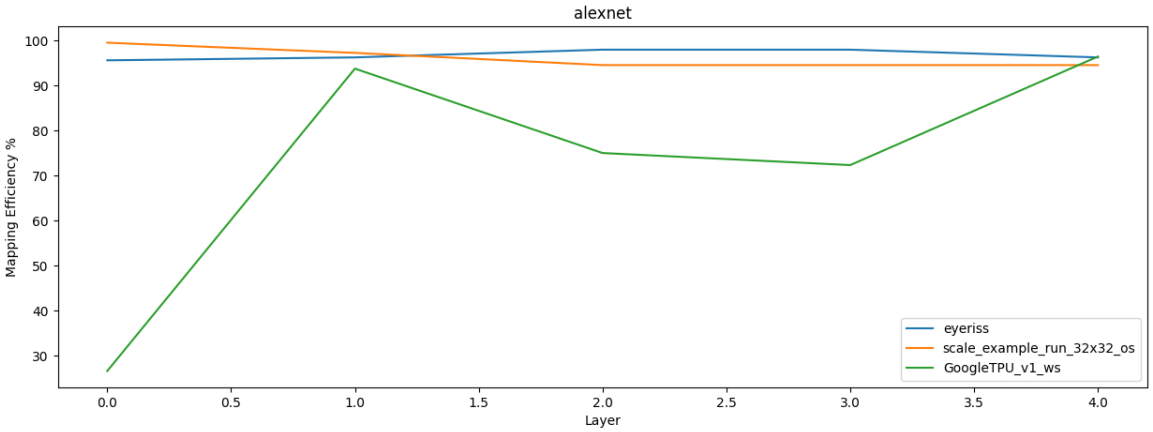


Figure 10: alexnet_mapping

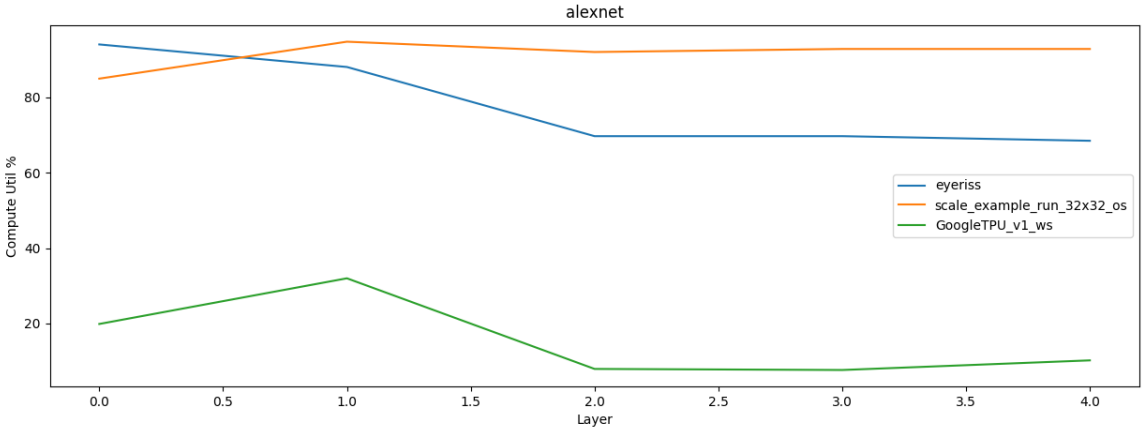


Figure 11: alexnet_compute

A similar order is observed in alexnet as well.

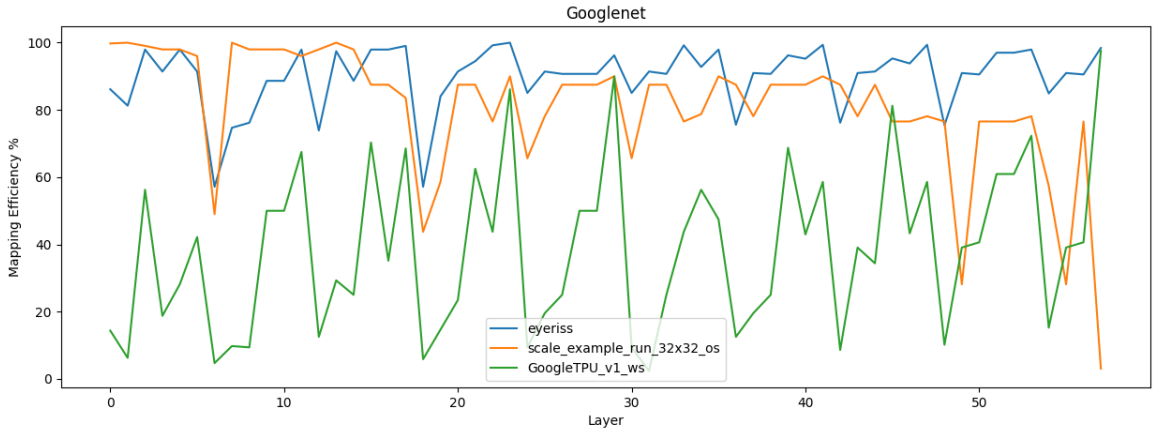


Figure 12: googlenet_mapping

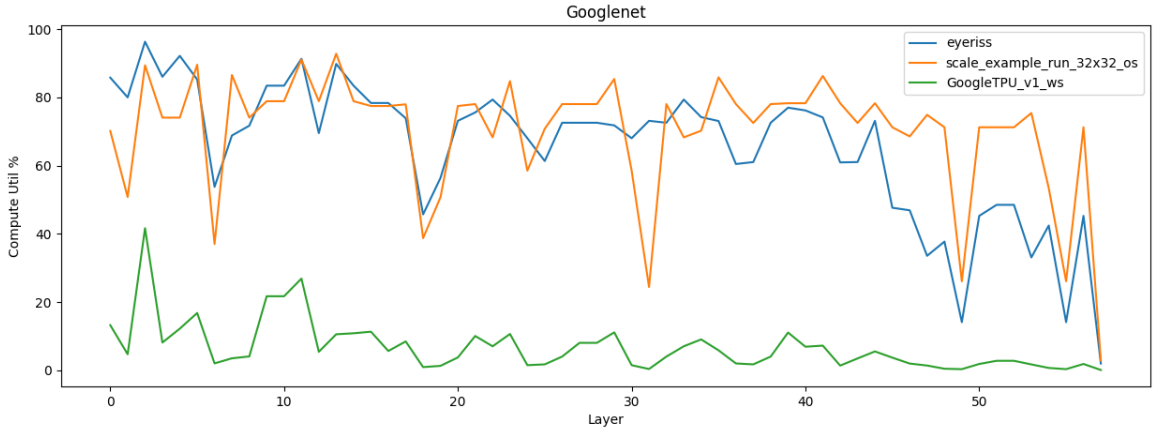


Figure 13: googlenet_compute

A similar order is observed in googlenet as well.

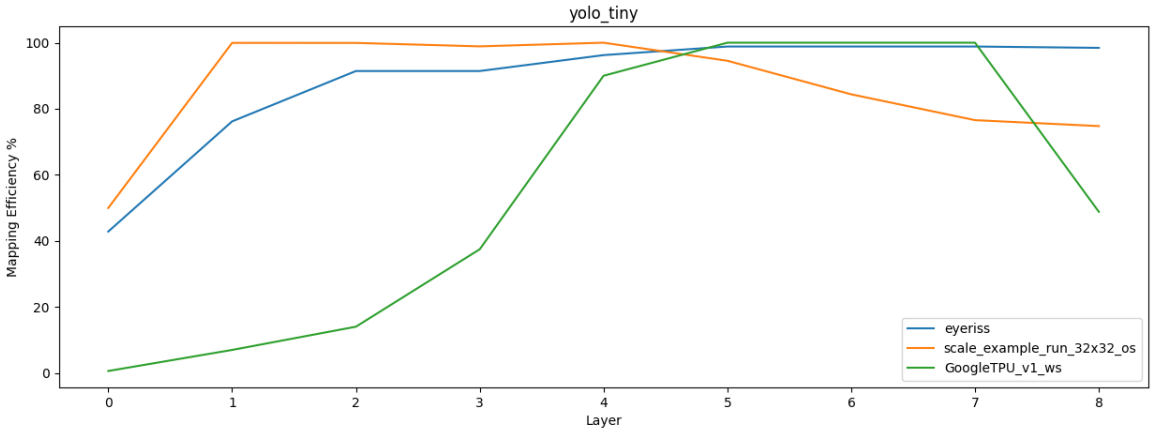


Figure 14: yolo_tiny_mapping

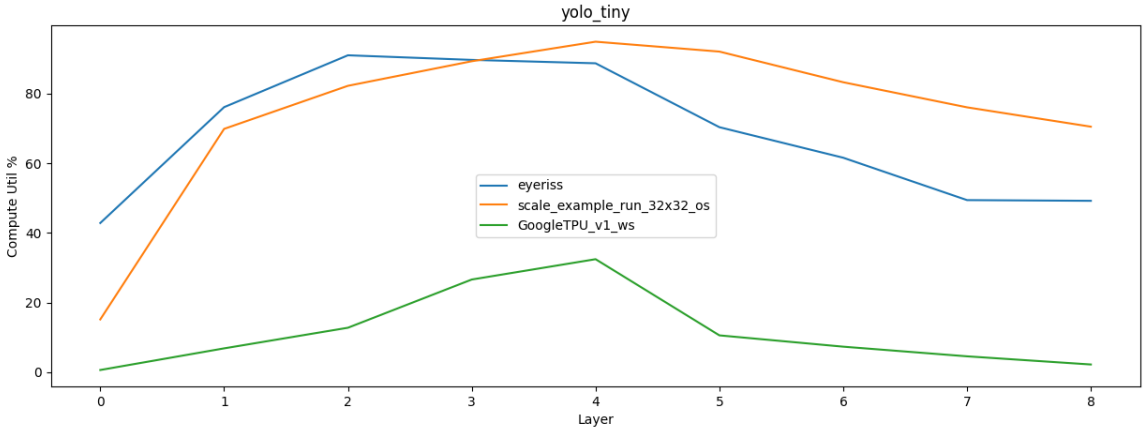


Figure 15: yolo_tiny_compute

A similar order is observed in yolo_tiny as well.

Total Runtime

The total runtime is calculated as $T_F \times F_R \times F_C$.

Thus it depends on S_R, S_C, R, C, T.

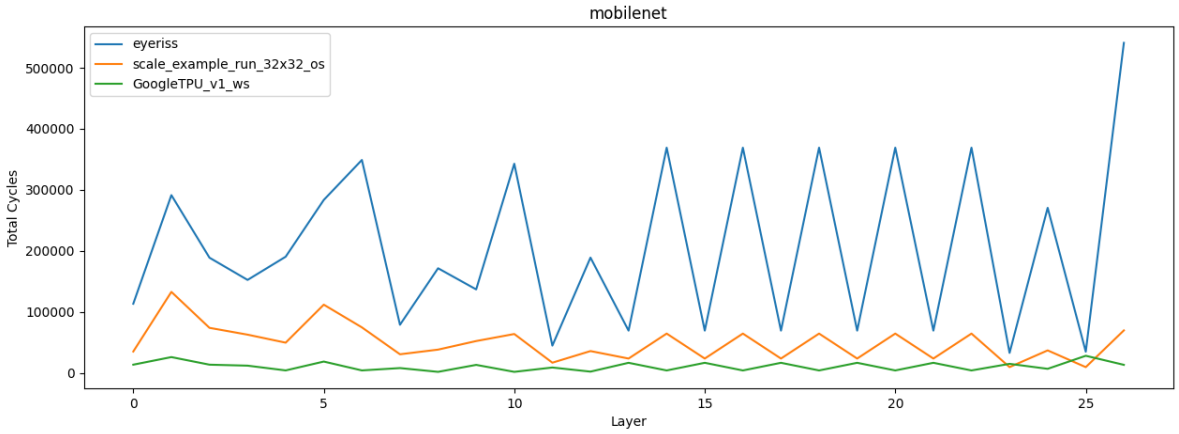


Figure 16: mobilenet_runtime

We can see that the total runtime is highest for eyeriss and lowest for GoogleTPU. This is because the systolic array is larger in GoogleTPU than in eyeriss and scale,

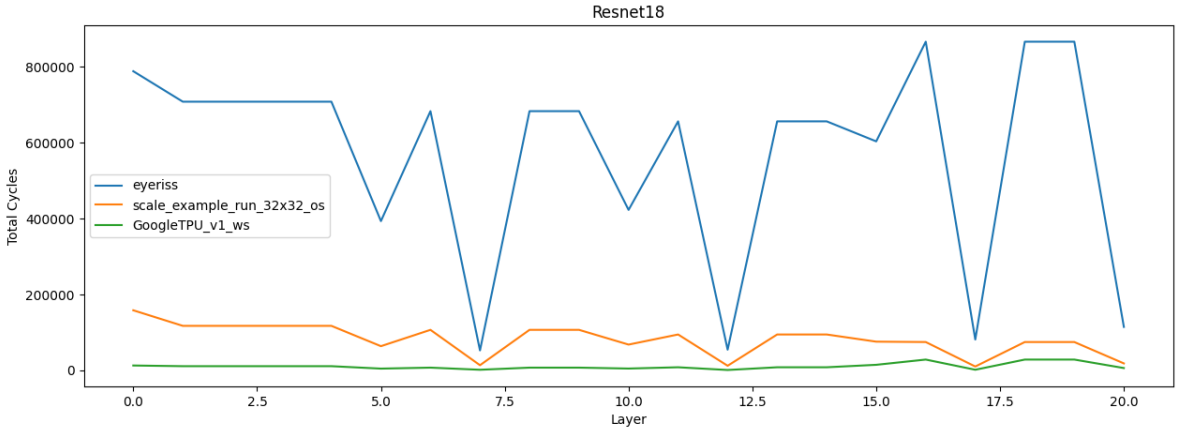


Figure 17: resnet18_runtime

A similar order is observed in resnet18 as well.

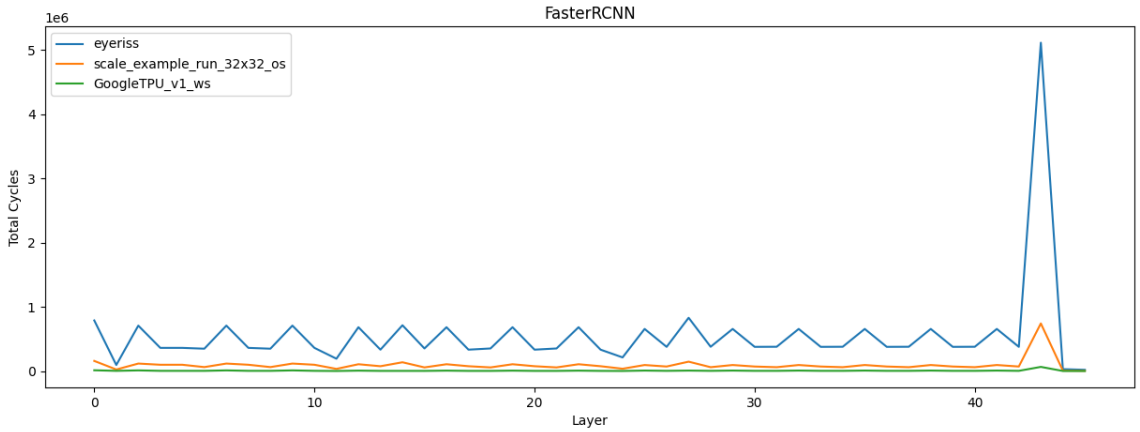


Figure 18: fasterRCNN_runtime

A similar order is observed in fasterRCNN as well.

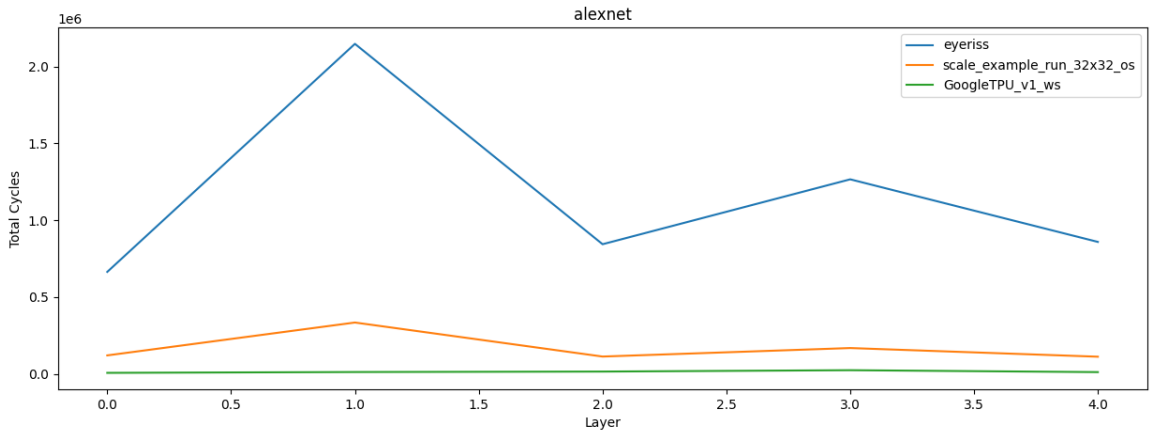


Figure 19: alexnet_runtime

A similar order is observed in alexnet as well.

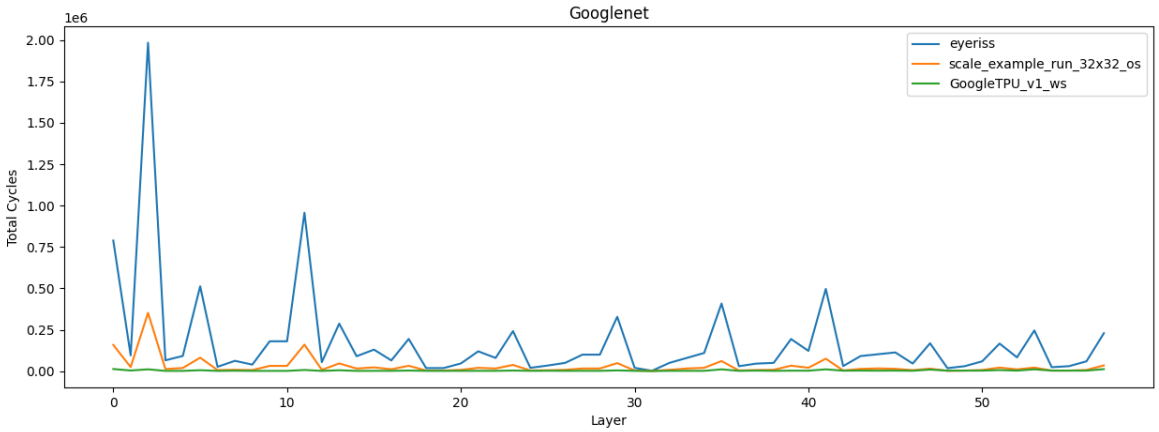


Figure 20: googlenet_runtime

A similar order is observed in googlenet as well.

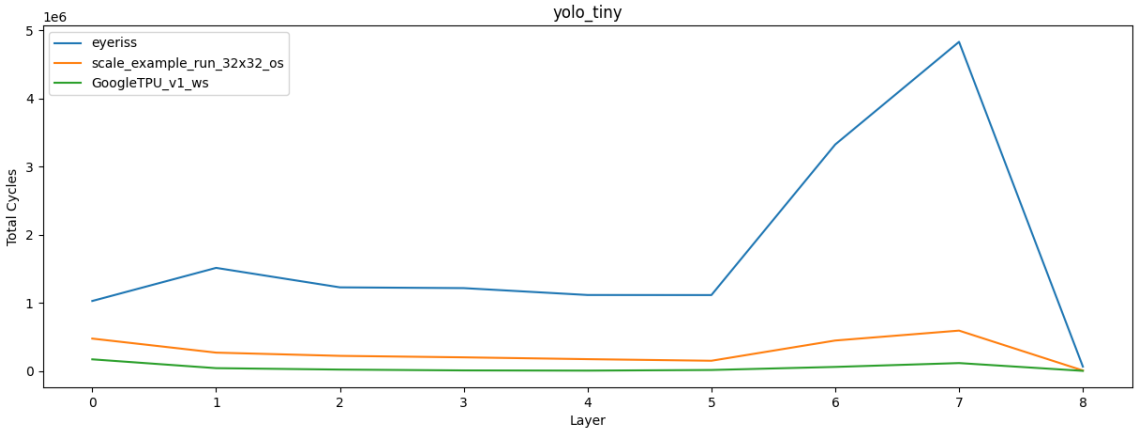


Figure 21: yolo_tiny_runtime

A similar order is observed in yolo_tiny as well.

DRAM Bandwidth Utilization and Read/Writes

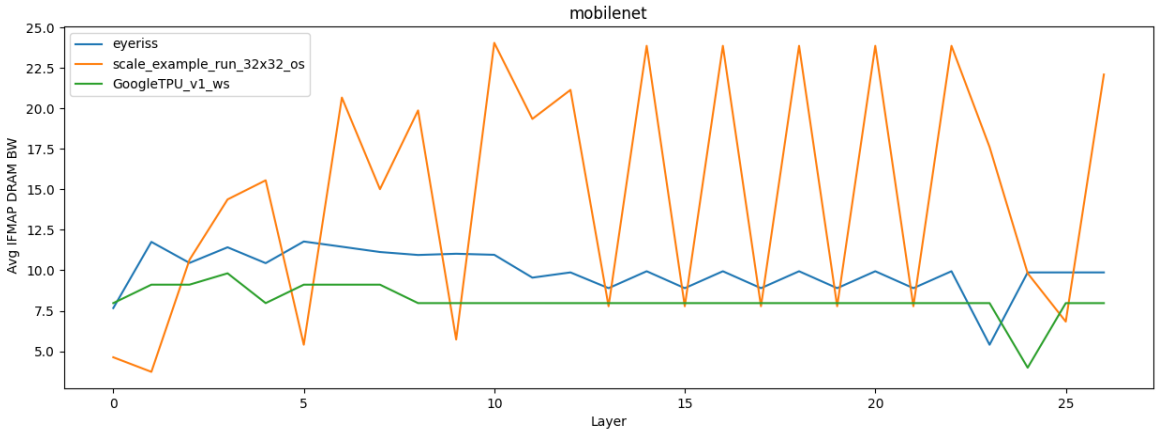


Figure 22: mobilenet_ifmap_dram_bw

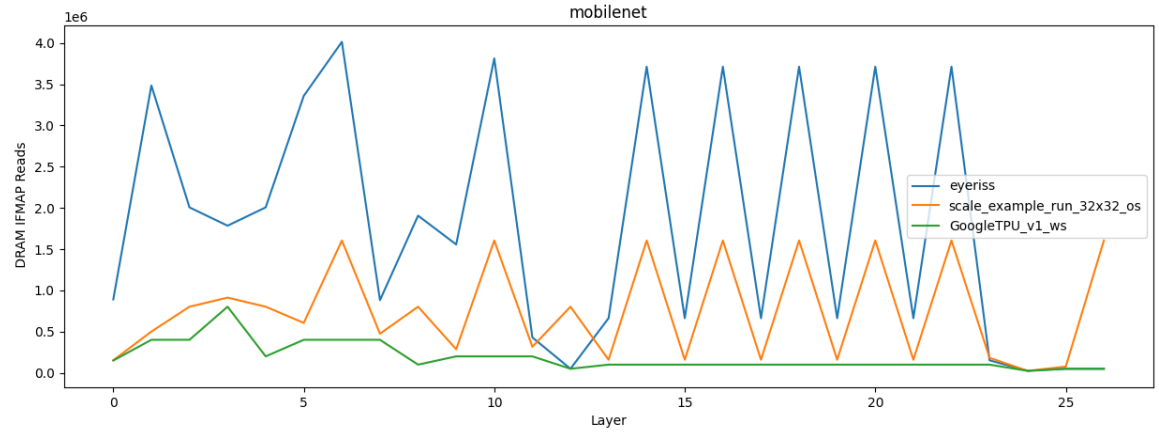


Figure 23: mobilenet_DRAM_IFMAP_READS

Since the eyeriss configuration has the smallest systolic array, the number of reads of input DRAM is the largest for eyeriss and the number of reads is the smallest for GoogleTPU.

The bandwidth is the largest for scale because of its dataflow.

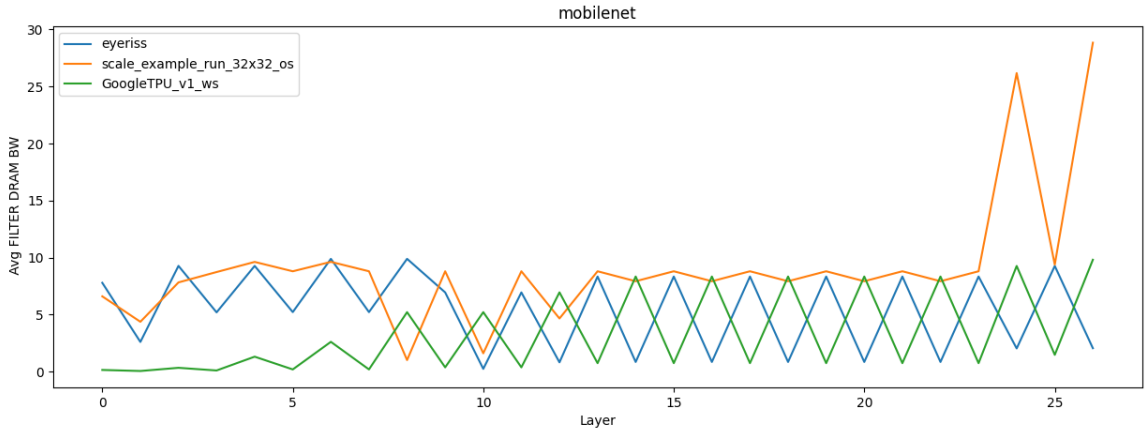


Figure 24: mobilenet_filter_dram_bw

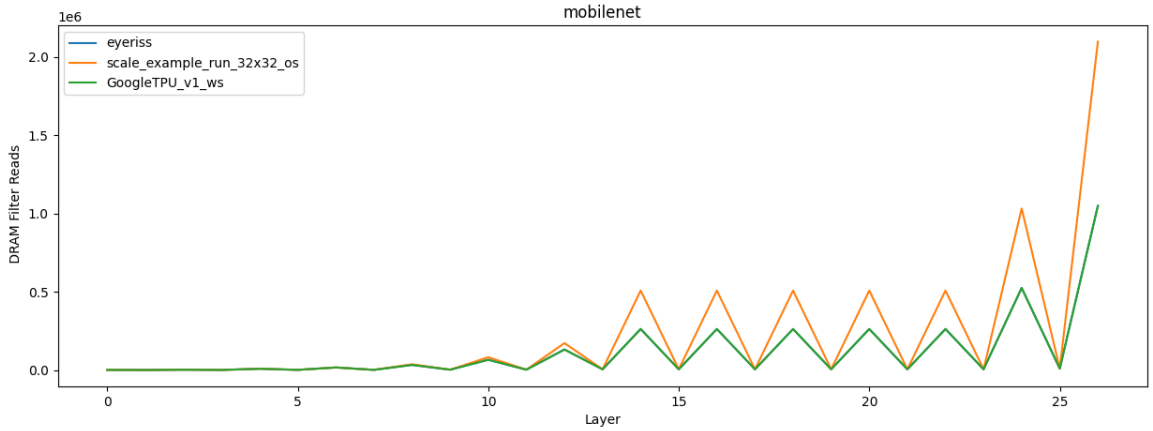


Figure 25: mobilenet_DRAM_Filter_READS

The number of reads and bandwidth of filter DRAM is the largest for scale because of its dataflow.

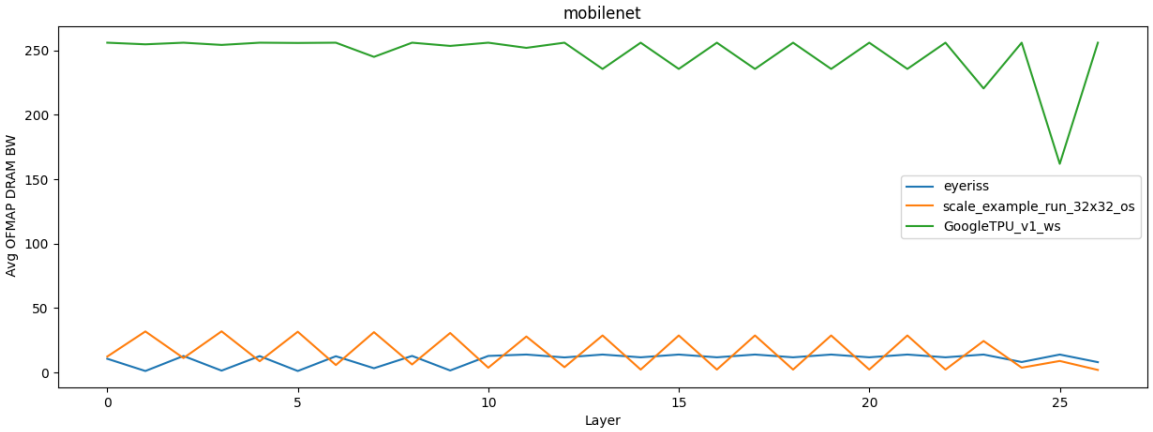


Figure 26: mobilenet_ofmap_dram_bw

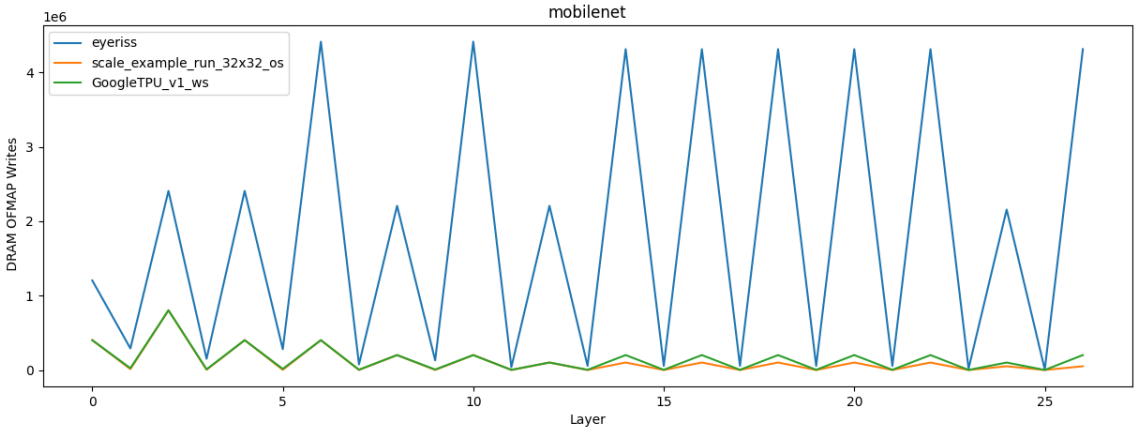


Figure 27: mobilenet_DRAM_OFMAP_WRITES

Since the eyeriss configuration has the smallest systolic array, the number of writes of output DRAM is the largest for eyeriss and the number of writes is the smallest for GoogleTPU.

Also the bandwidth is the largest for GoogleTPU because of its systolic array.

Similar orders are observed in resnet18, fasterRCNN, alexnet, googlenet, and yolo_tiny.

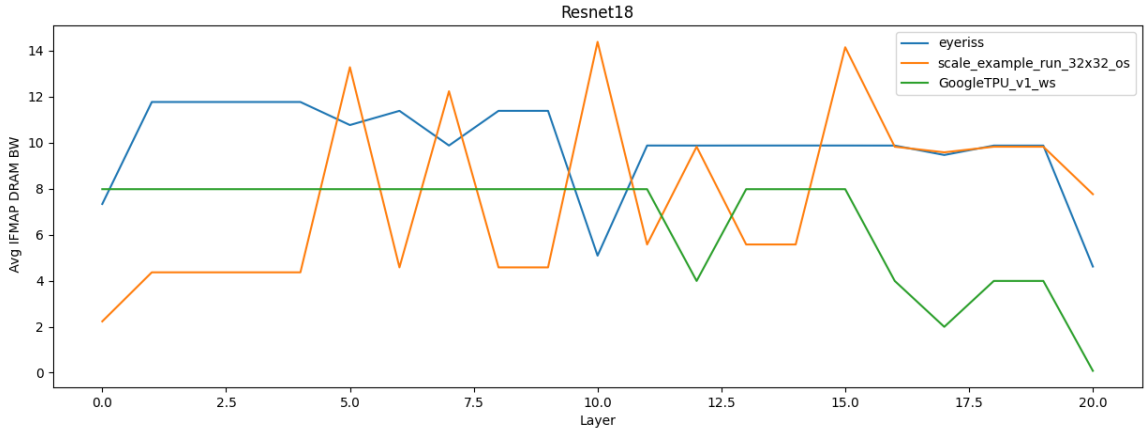


Figure 28: resnet18_ifmap_dram_bw

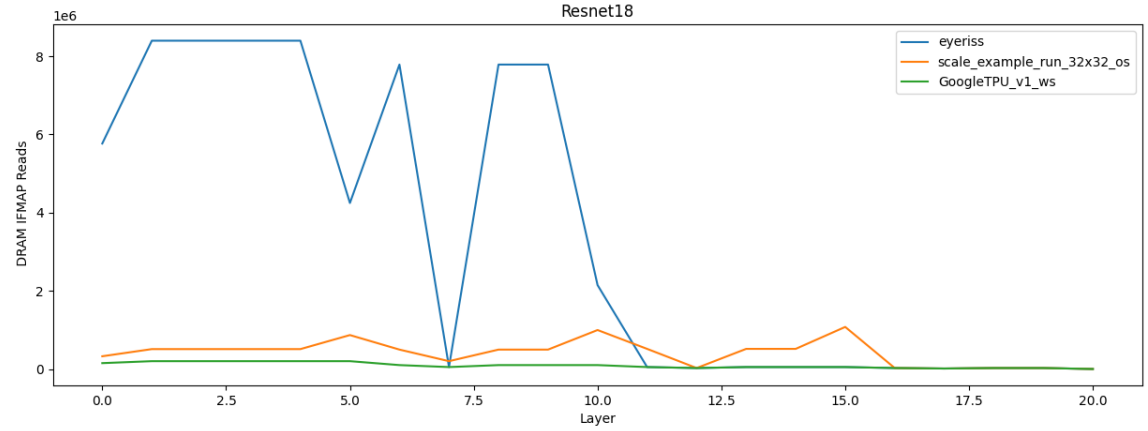


Figure 29: resnet18_DRAM_IFMAP_READS

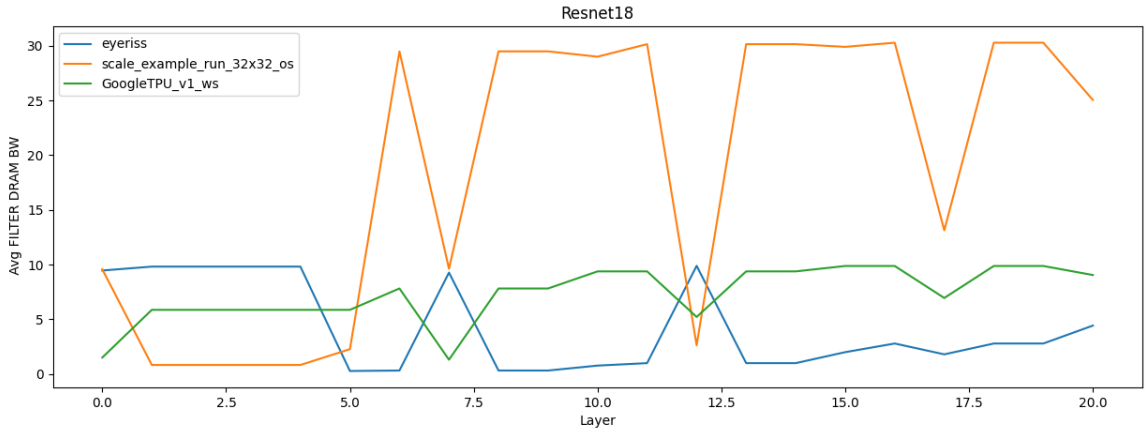


Figure 30: resnet18_filter_dram_bw

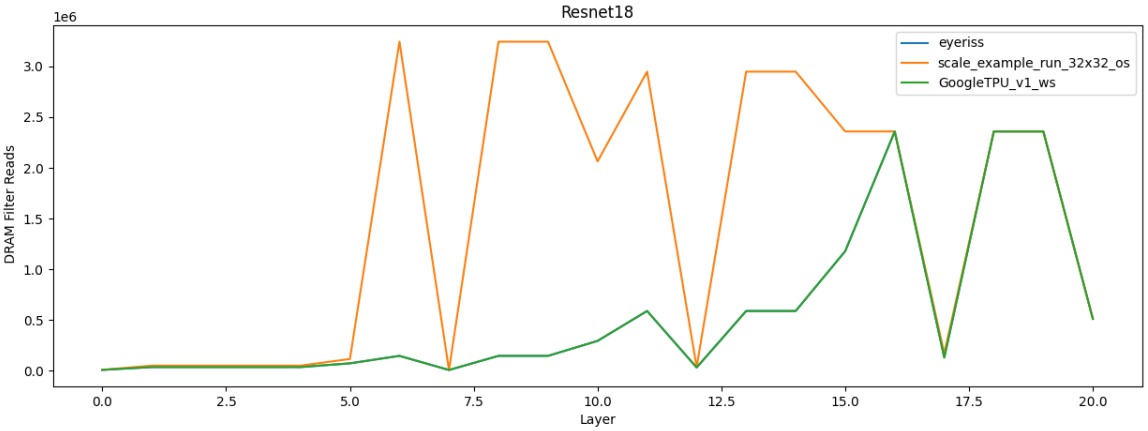


Figure 31: resnet18_DRAM_Filter_READS

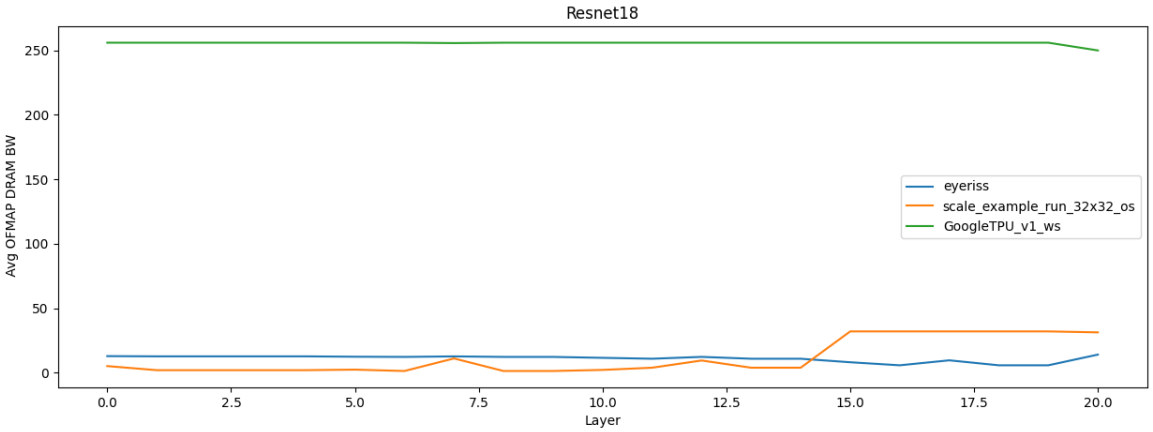


Figure 32: resnet18_ofmap_dram_bw

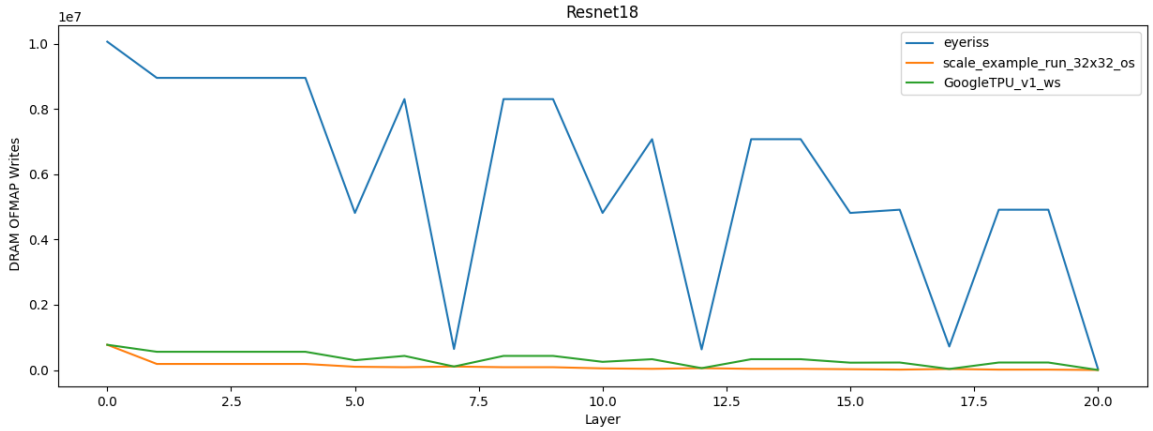


Figure 33: resnet18_DRAM_OFMAP_WRITES

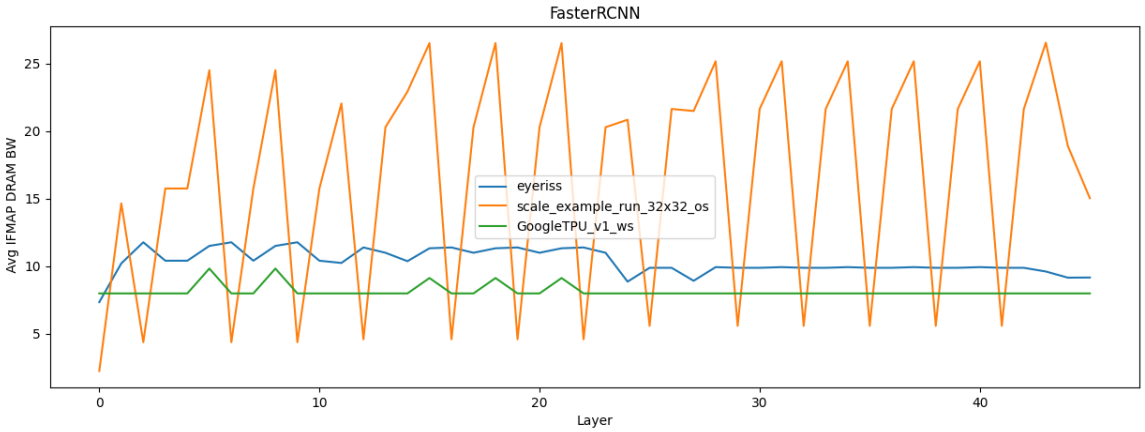


Figure 34: fasterRCNN_ifmap_dram_bw

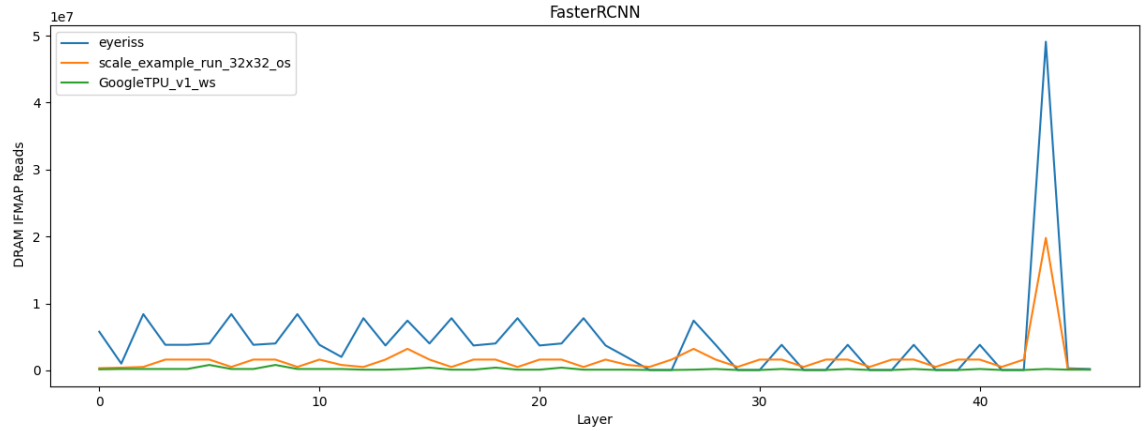


Figure 35: fasterRCNN_DRAM_IFMAP_READS

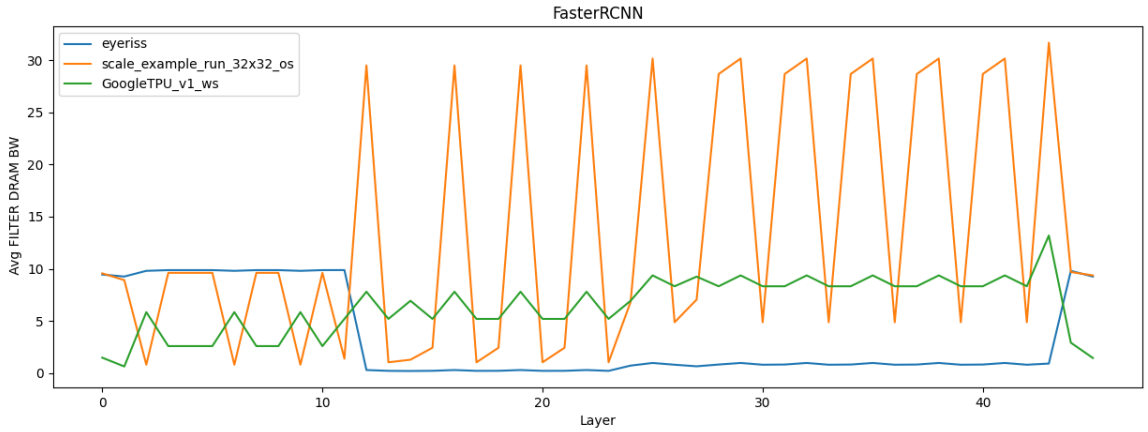


Figure 36: fasterRCNN_filter_dram_bw

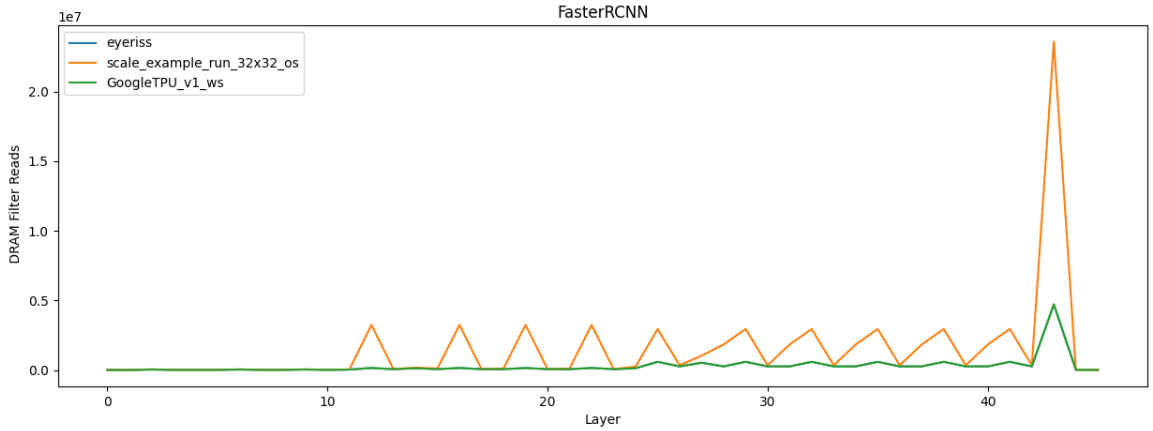


Figure 37: fasterRCNN_DRAM_Filter_READS

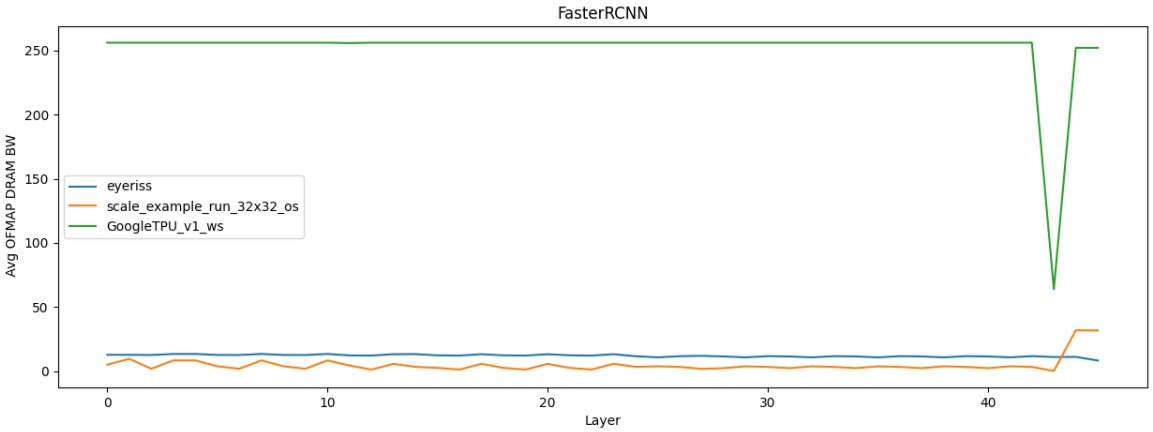


Figure 38: fasterRCNN_ofmap_dram_bw

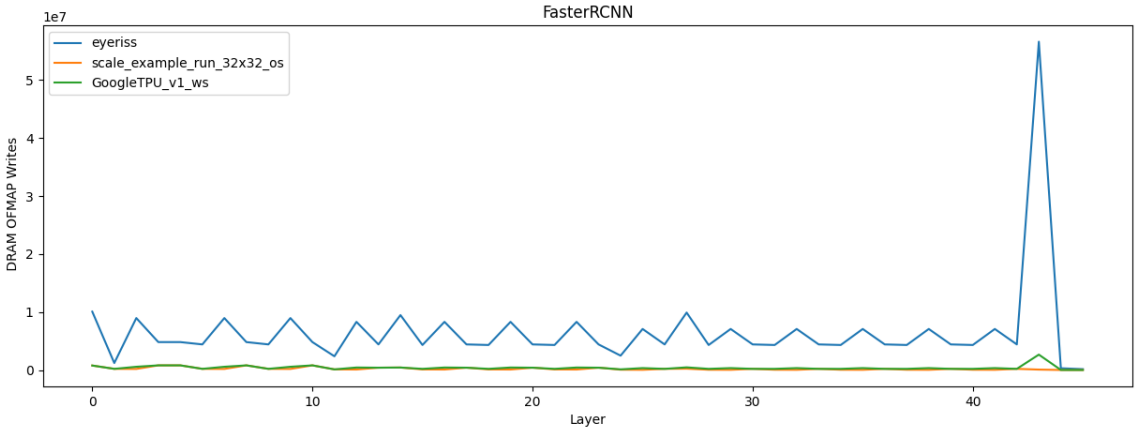


Figure 39: fasterRCNN_DRAM_OFMAP_WRITES

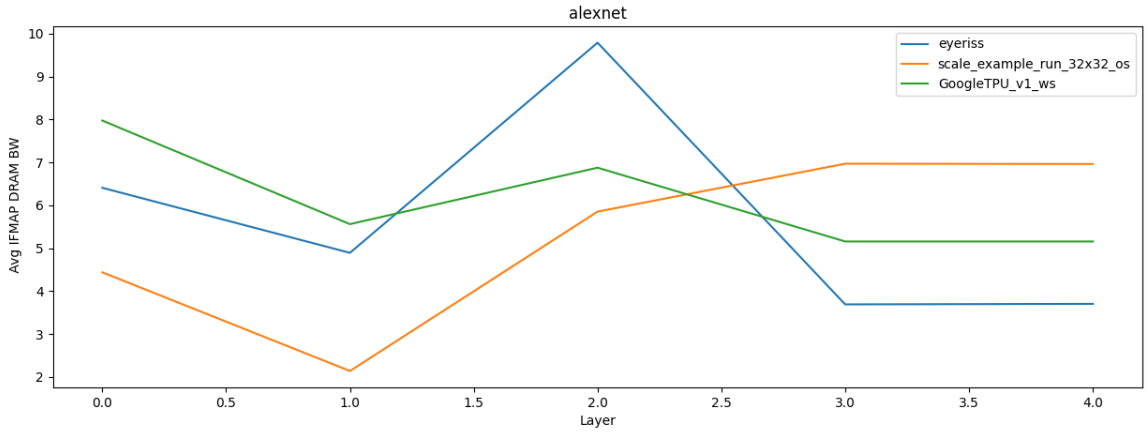


Figure 40: alexnet_ifmap_dram_bw

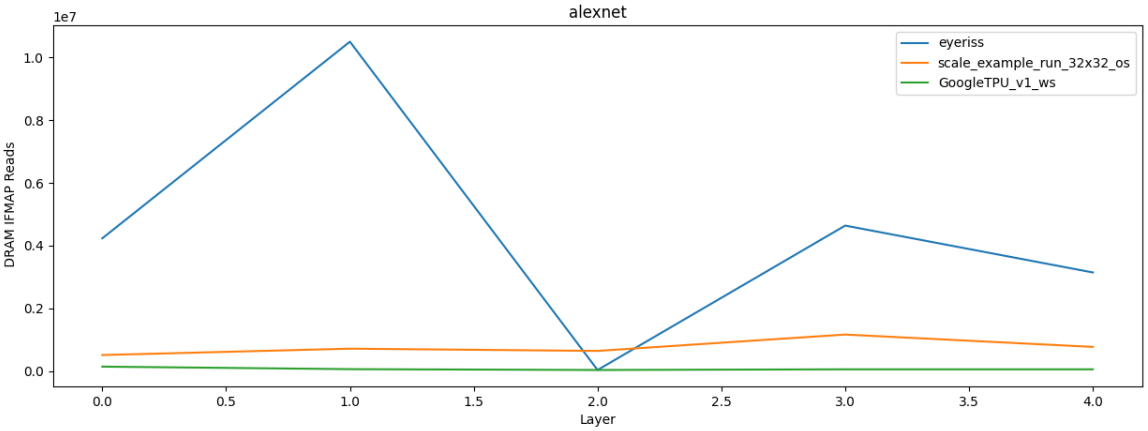


Figure 41: alexnet_DRAM_IFMAP_READS

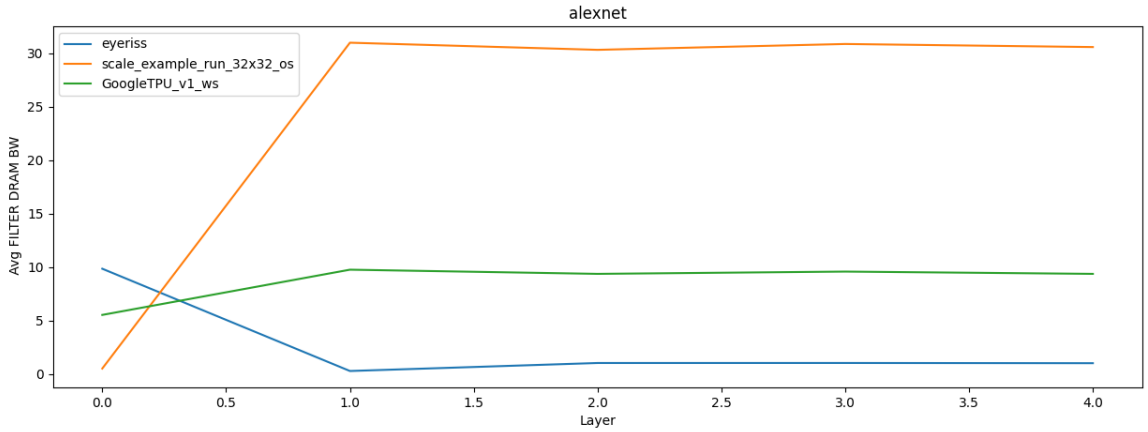


Figure 42: alexnet_filter_dram_bw

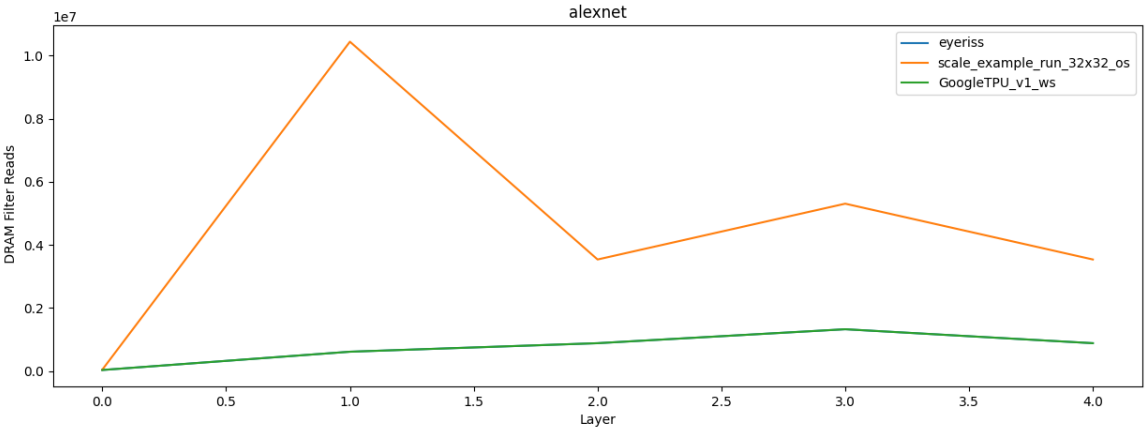


Figure 43: alexnet_DRAM_Filter_READS

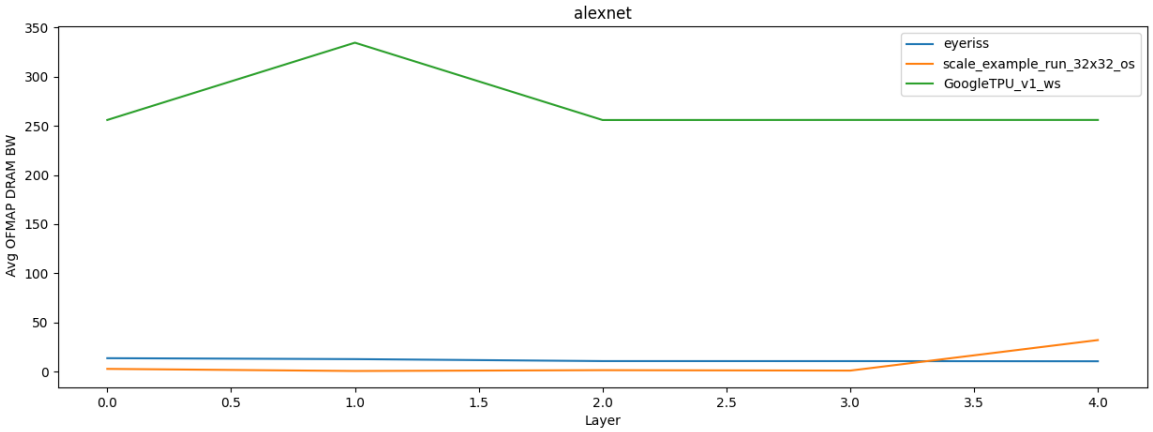


Figure 44: alexnet_ofmap_dram_bw

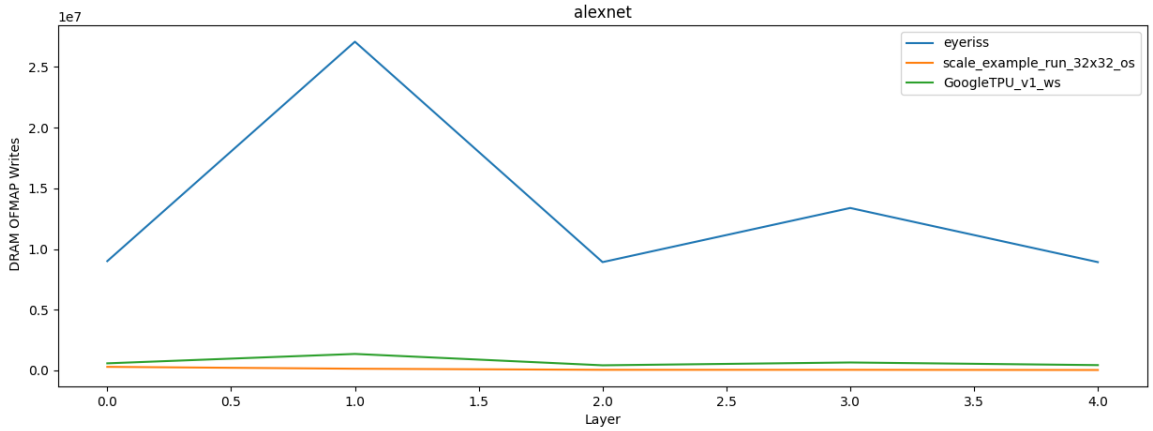


Figure 45: alexnet_DRAM_OFMAP_WRITES

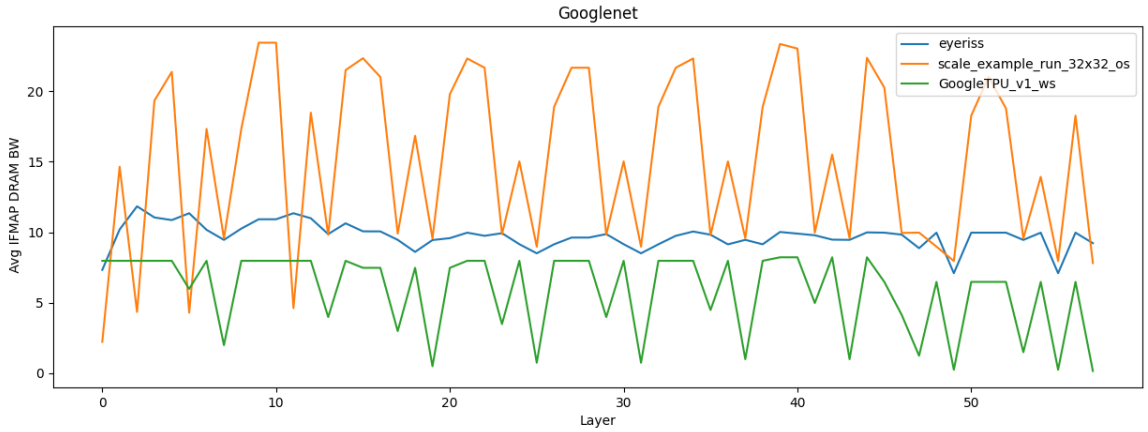


Figure 46: googlenet_ifmap_dram_bw

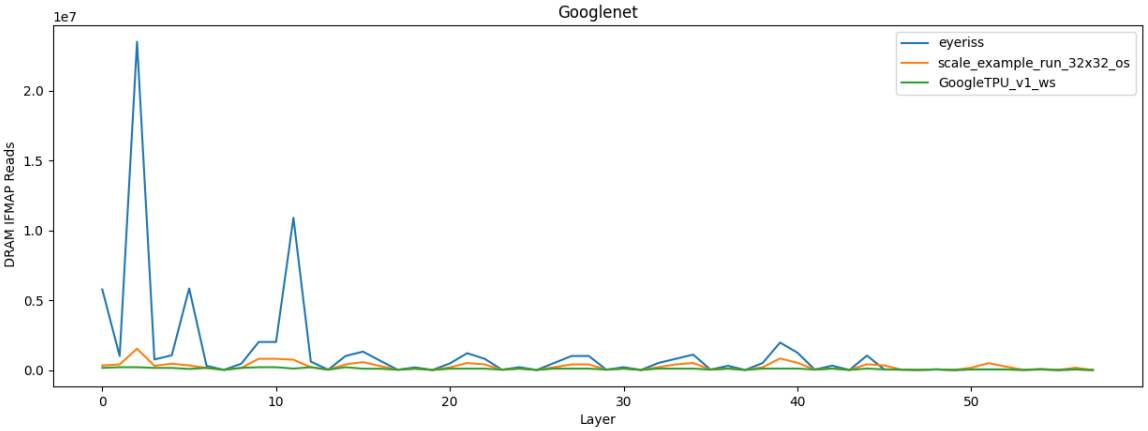


Figure 47: googlenet_DRAM_IFMAP_READS

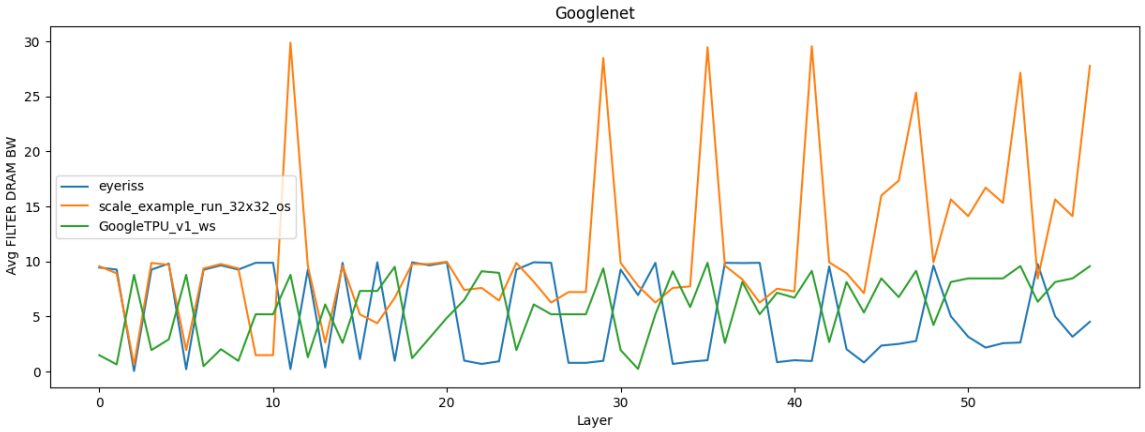


Figure 48: googlenet_filter_dram_bw

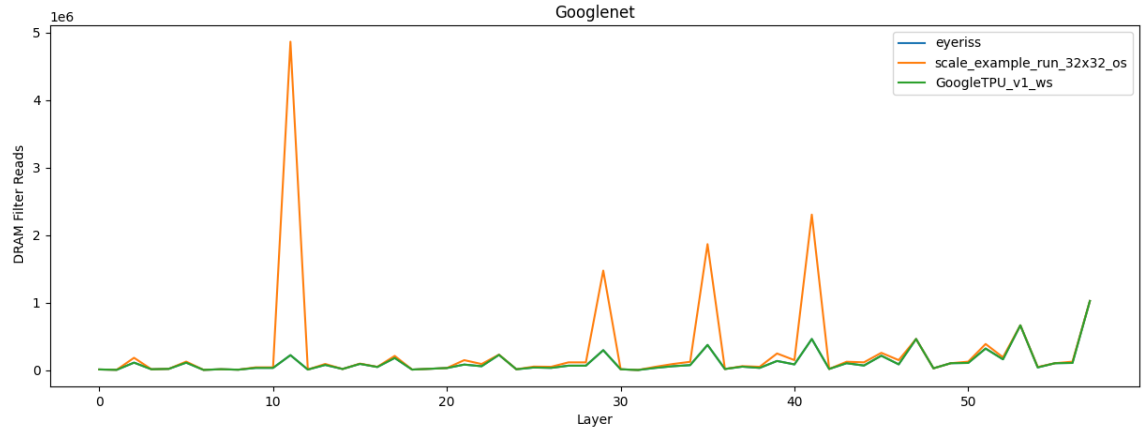


Figure 49: googlenet_DRAM_Filter_READS

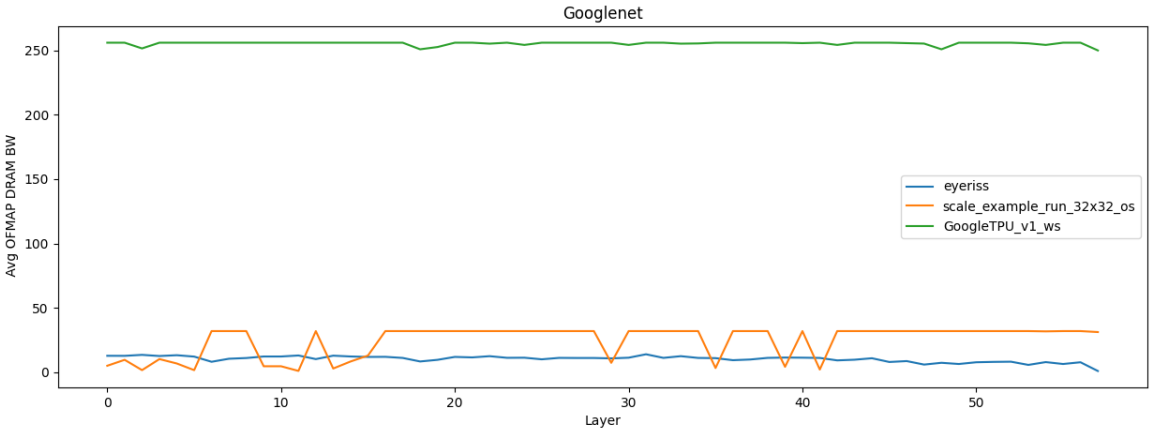


Figure 50: googlenet_ofmap_dram_bw

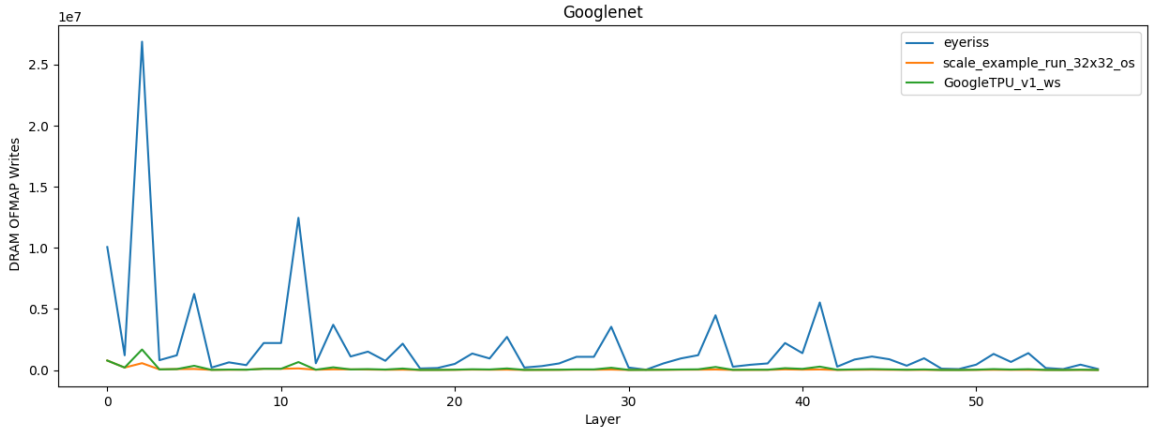


Figure 51: googlenet_DRAM_OFMAP_WRITES

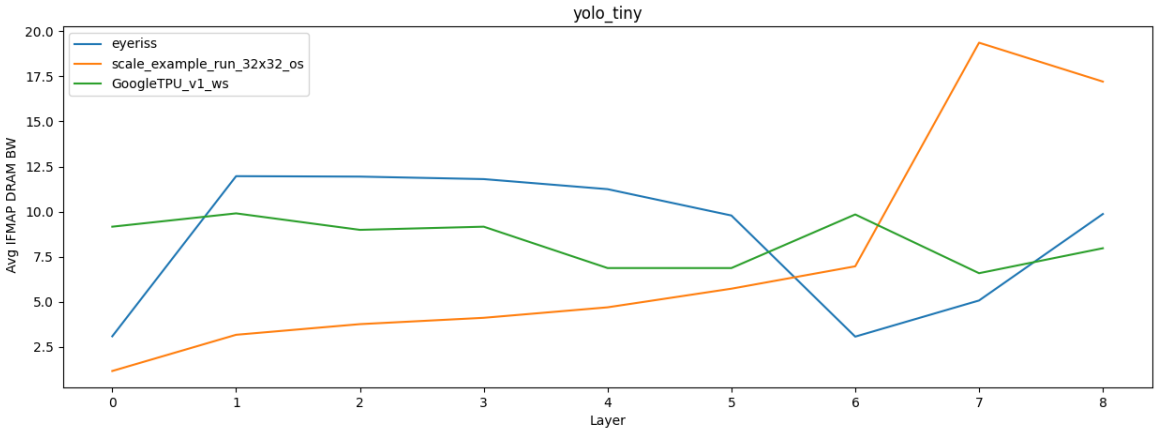


Figure 52: yolo_tiny_ifmap_dram_bw

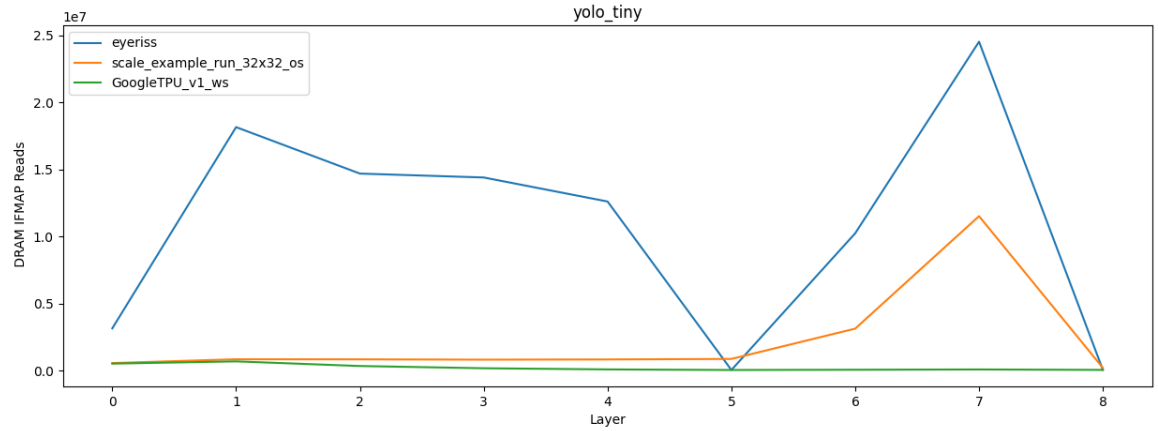


Figure 53: yolo_tiny_DRAM_IFMAP_READS

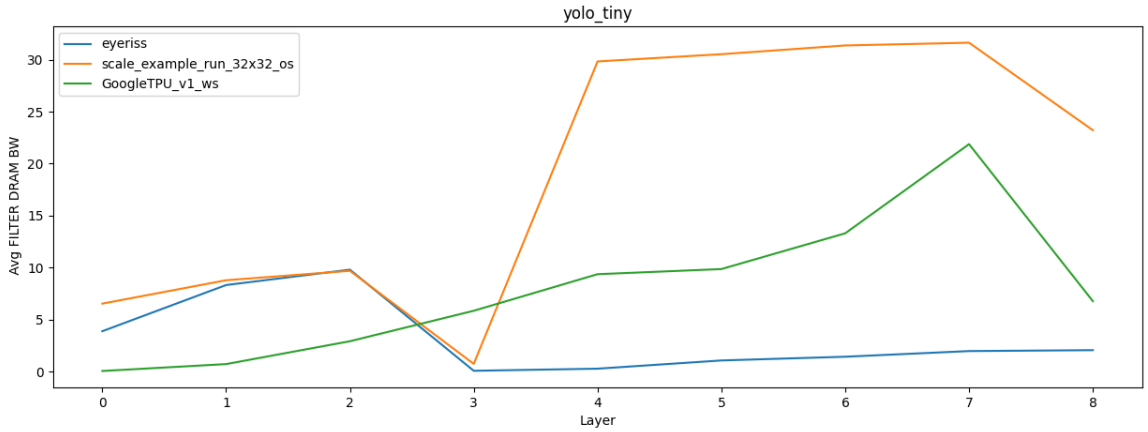


Figure 54: yolo_tiny_filter_dram_bw

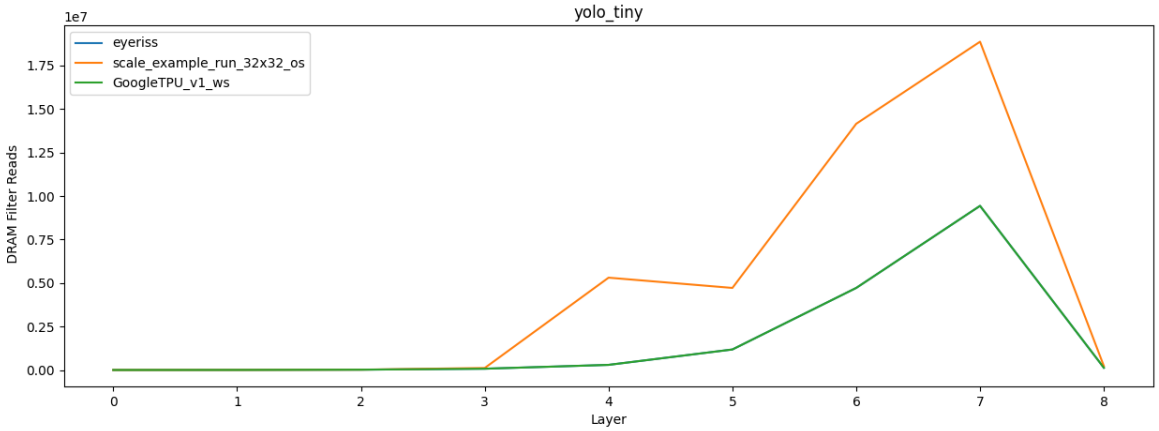


Figure 55: yolo_tiny_DRAM_Filter_READS

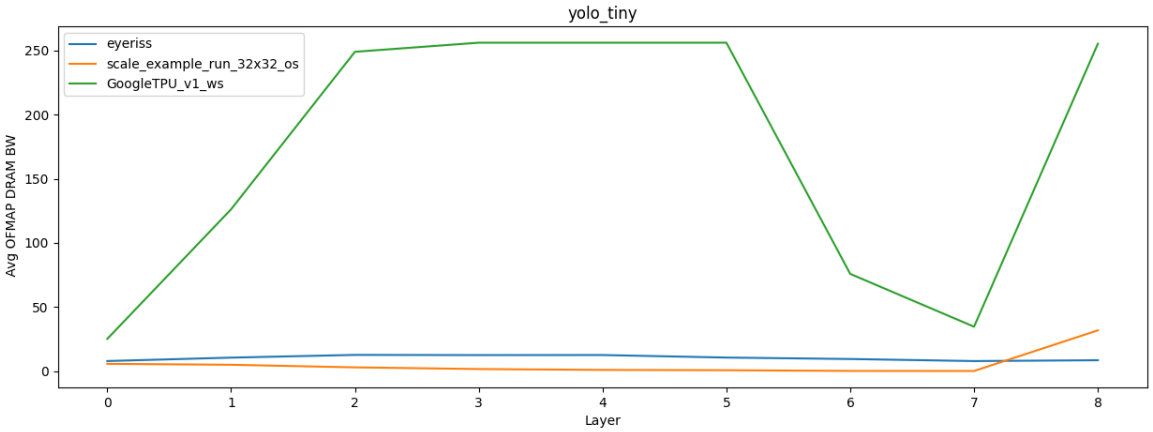


Figure 56: yolo_tiny_ofmap_dram_bw

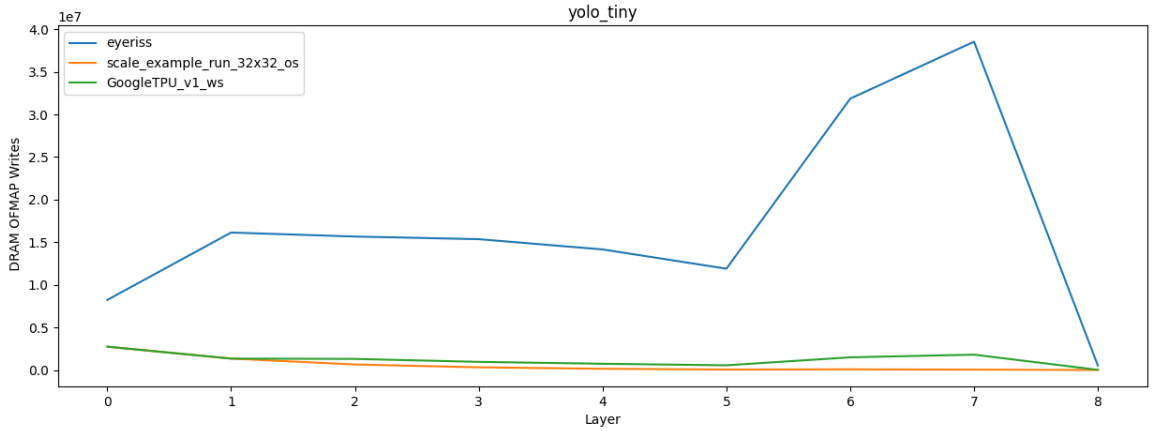


Figure 57: yolo_tiny_DRAM_OFMAP_WRITES

SRAM Bandwidth Utilization and Read/Writes

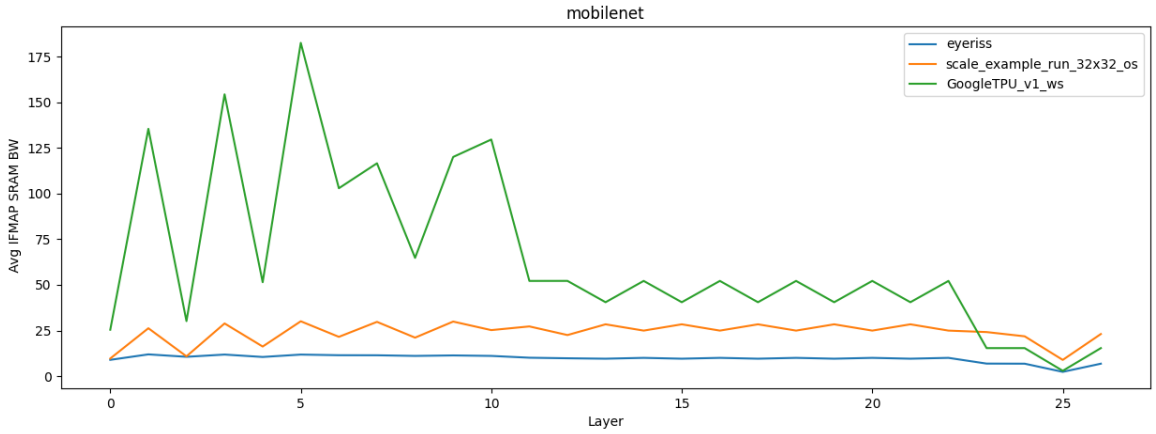


Figure 58: mobilenet_ifmap_sram_bw

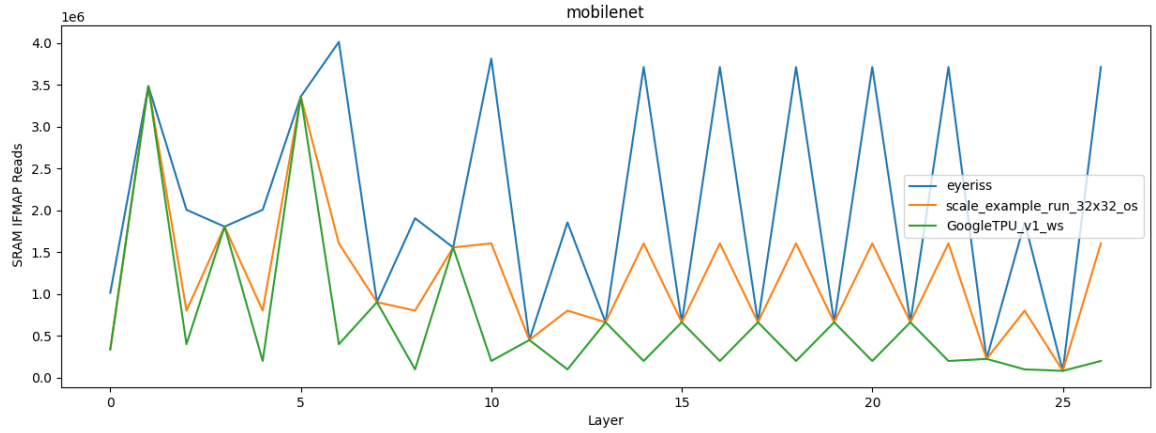


Figure 59: mobilenet_SRAM_IFMAP_READS

Here the reads of the input memory is largest for eyeriss because the systolic array is the smallest in eyeriss.

The bandwidth is the largest for GoogleTPU because of its systolic array.

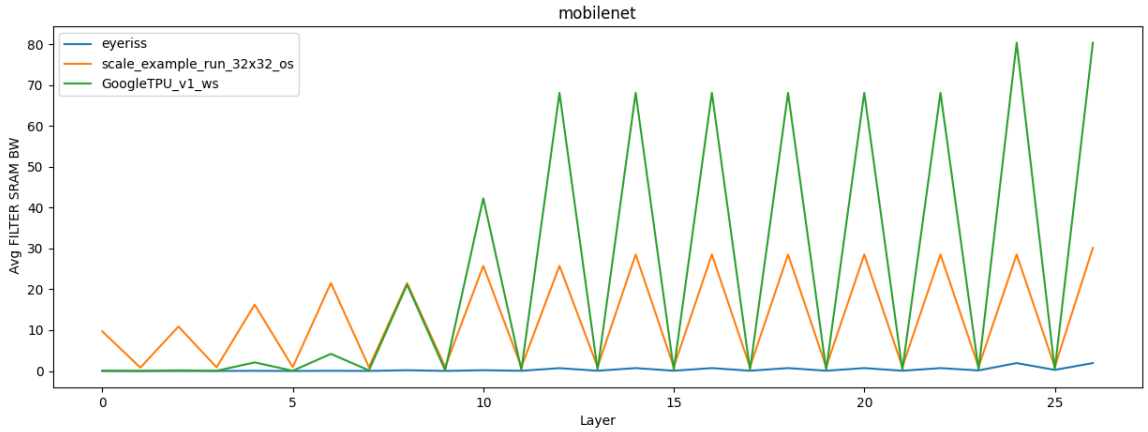


Figure 60: mobilenet_filter_sram_bw

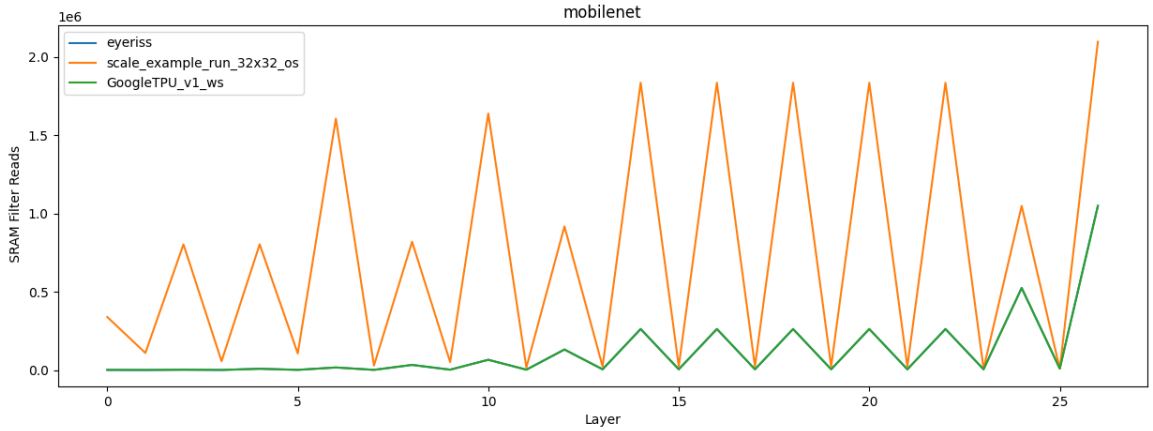


Figure 61: mobilenet_SRAM_Filter_READS

Bandwidth of the filter memory is the largest for GoogleTPU because of its systolic array.

Number of reads is the largest for scale because of its dataflow.

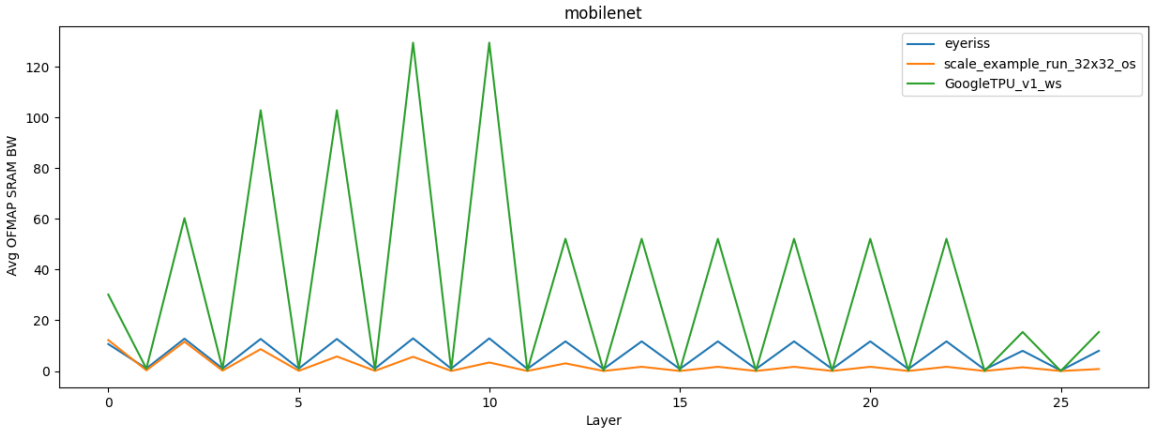


Figure 62: mobilenet_ofmap_sram_bw

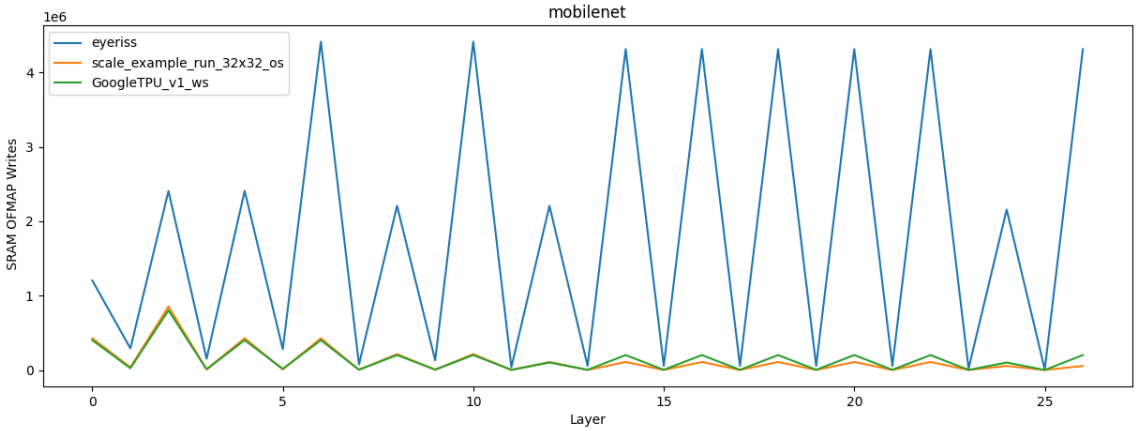


Figure 63: mobilenet_SRAM_OFMAP_WRITES

Here, the number of writes is largest for eyeriss because the systolic array is the smallest in eyeriss. The bandwidth is the largest for GoogleTPU because of its systolic array.

Similar orders are observed in resnet18, fasterRCNN, alexnet, googlenet, and yolo_tiny.

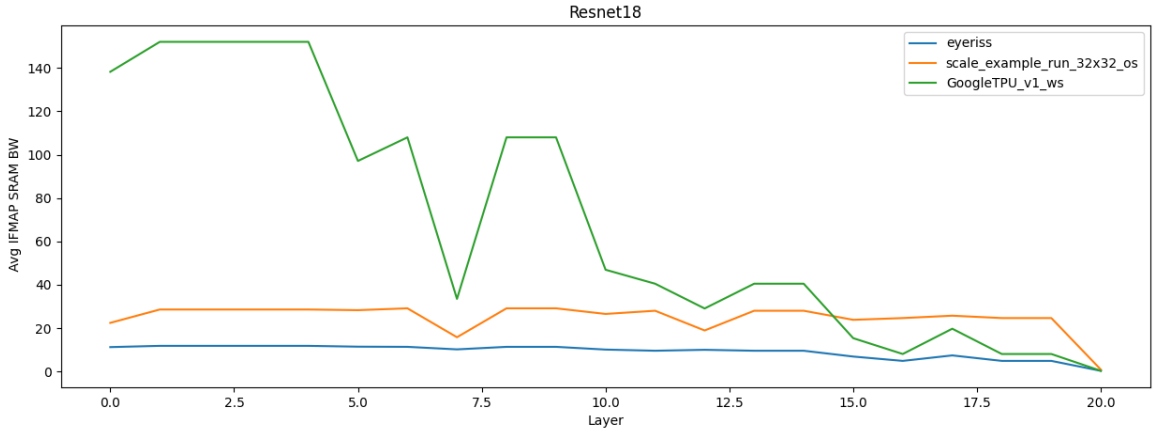


Figure 64: resnet18_ifmap_sram_bw

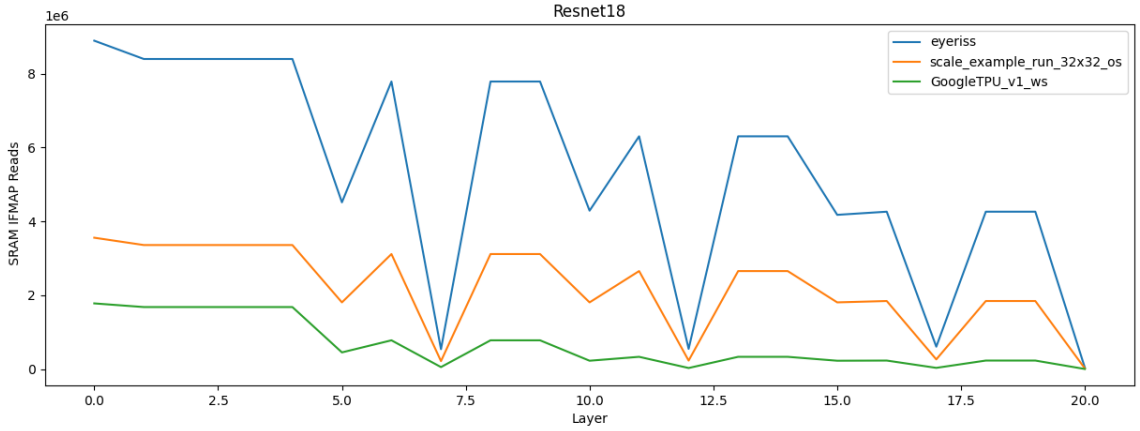


Figure 65: resnet18_SRAM_IFMAP_READS

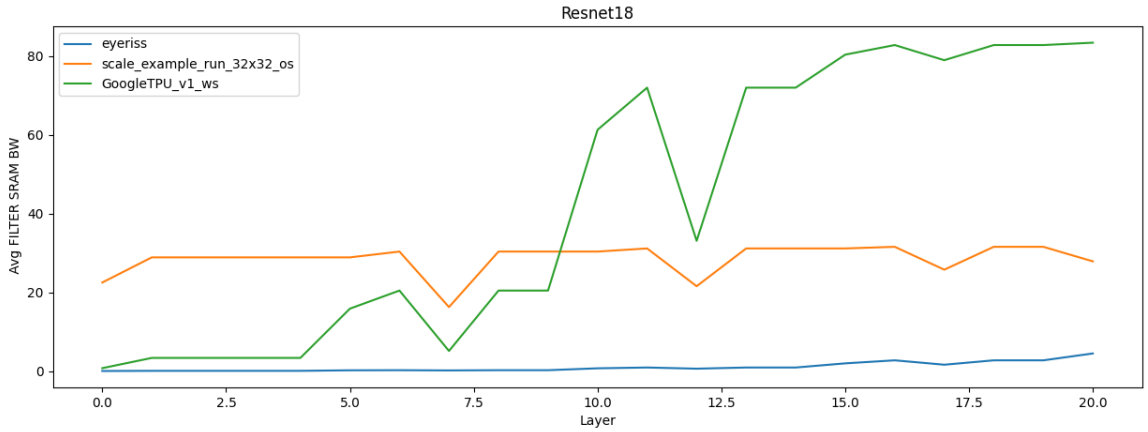


Figure 66: resnet18_filter_sram_bw

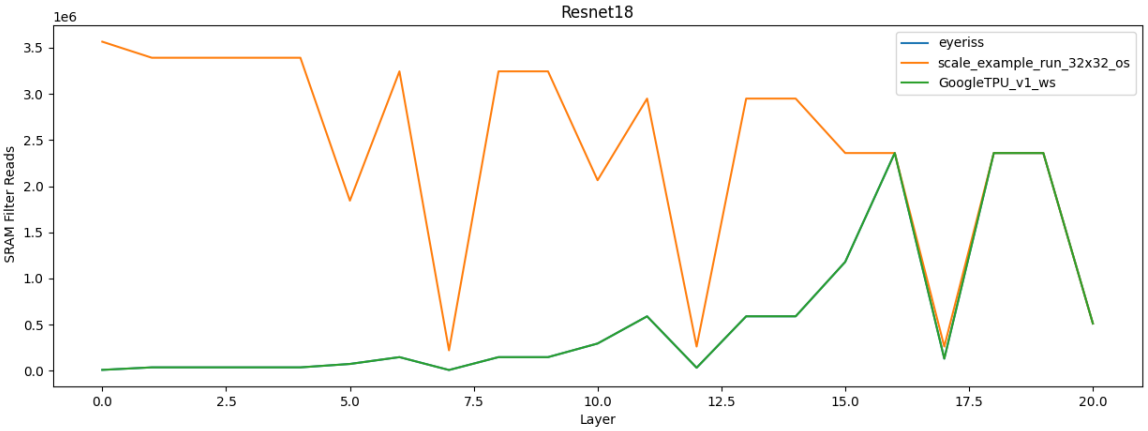


Figure 67: resnet18_SRAM_Filter_READS

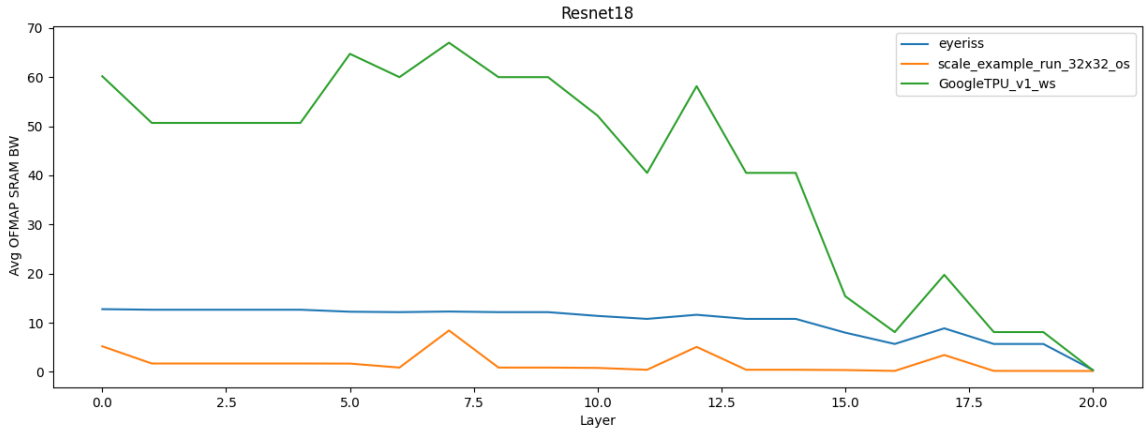


Figure 68: resnet18_ofmap_sram_bw

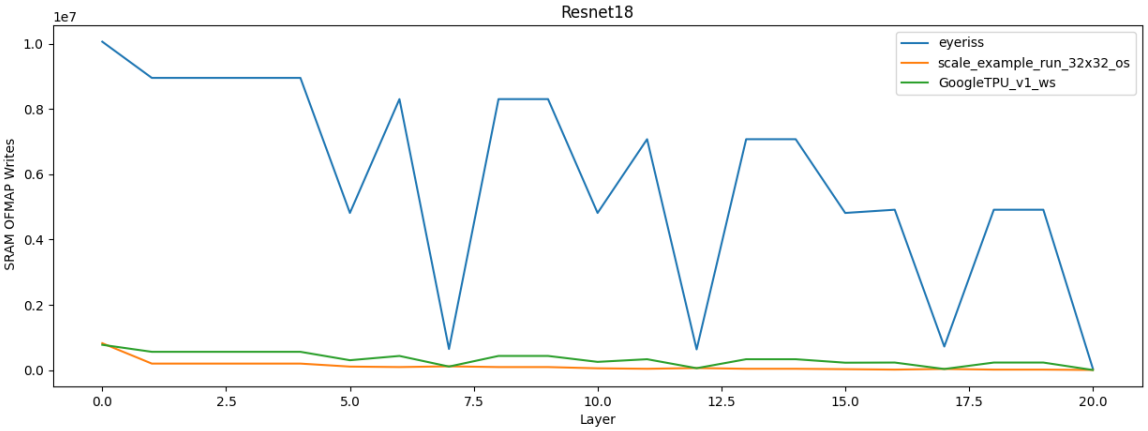


Figure 69: resnet18_SRAM_OFMAP_WRITES

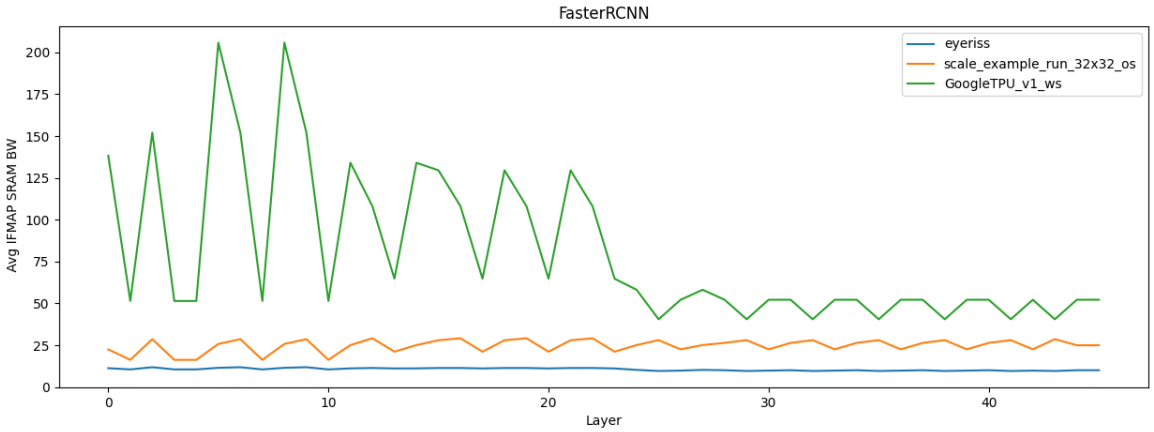


Figure 70: fasterRCNN_ifmap_sram_bw

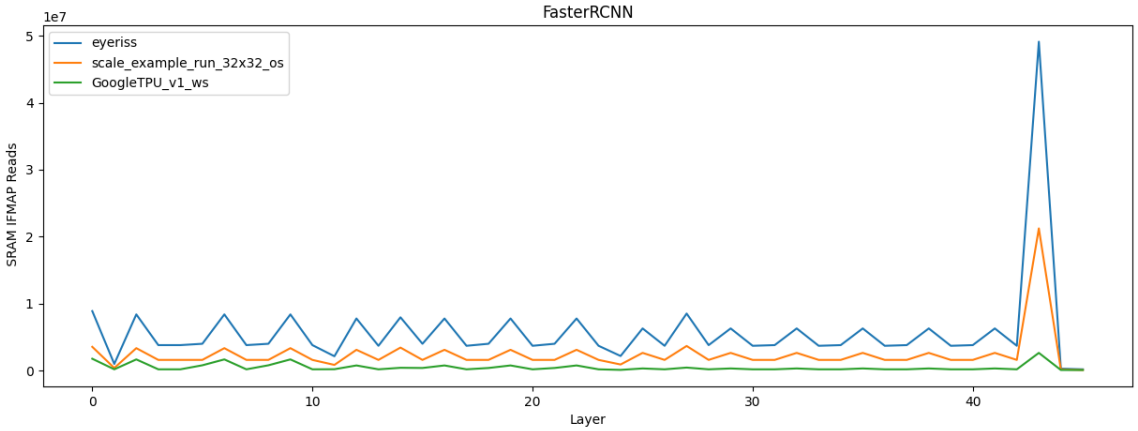


Figure 71: fasterRCNN_SRAM_IFMAP_READS

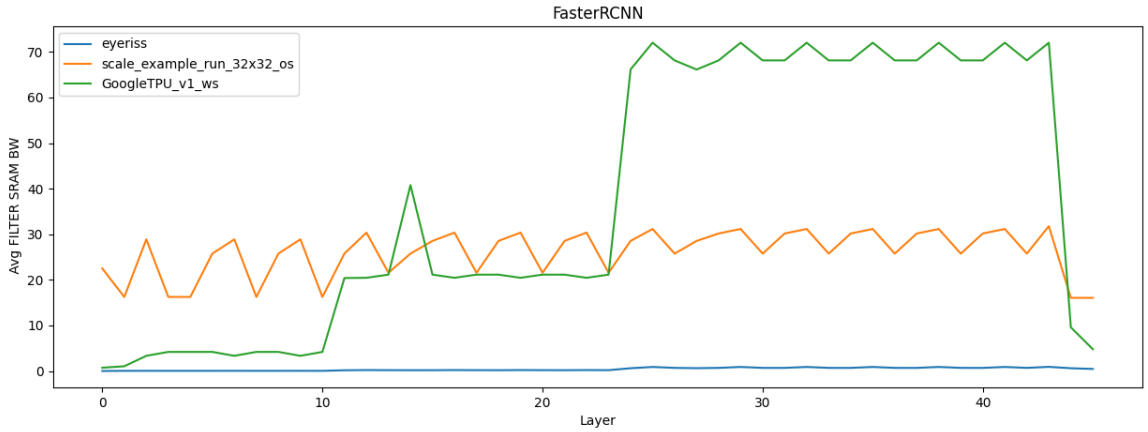


Figure 72: fasterRCNN_filter_sram_bw

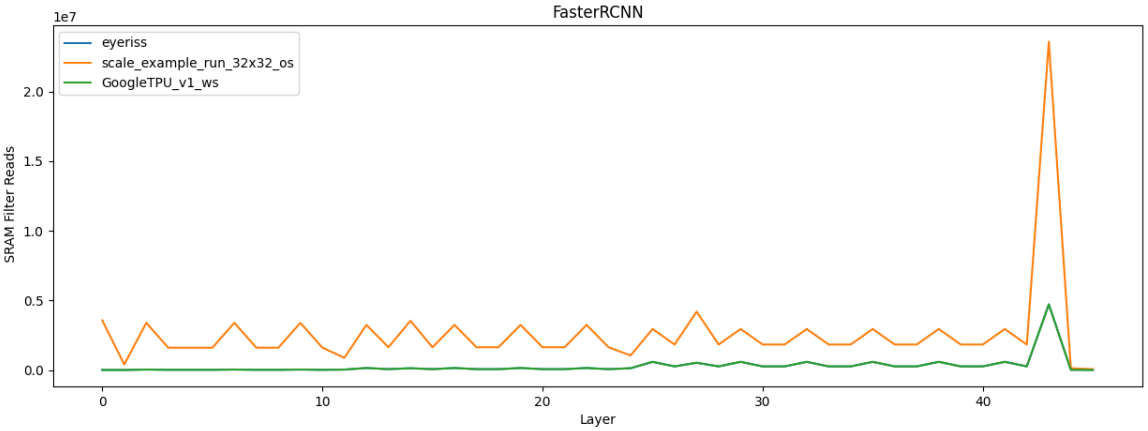


Figure 73: fasterRCNN_SRAM_Filter_READS

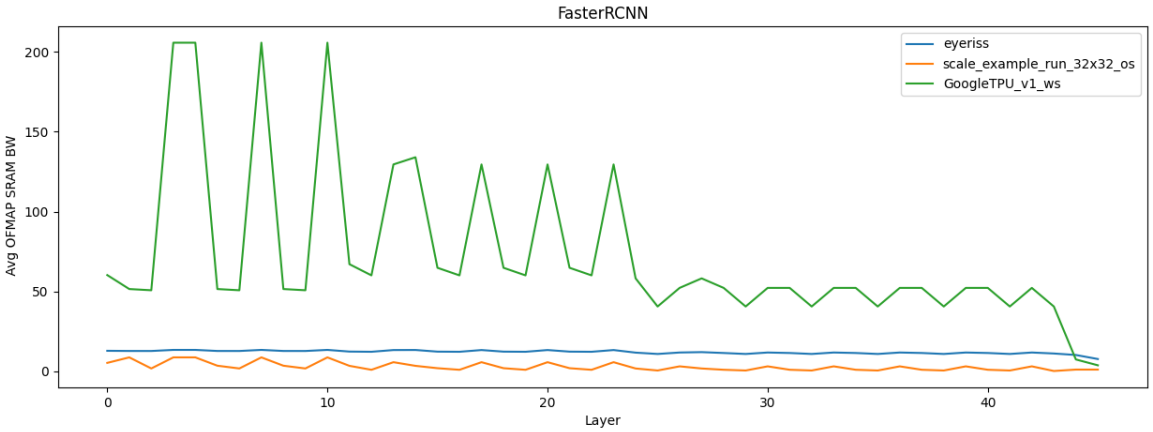


Figure 74: fasterRCNN_ofmap_sram_bw

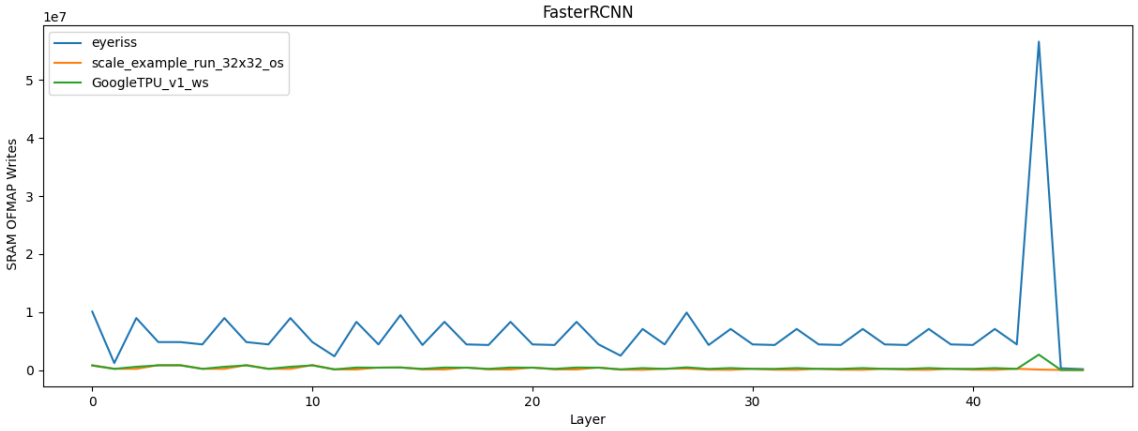


Figure 75: fasterRCNN_SRAM_OFMAP_WRITES

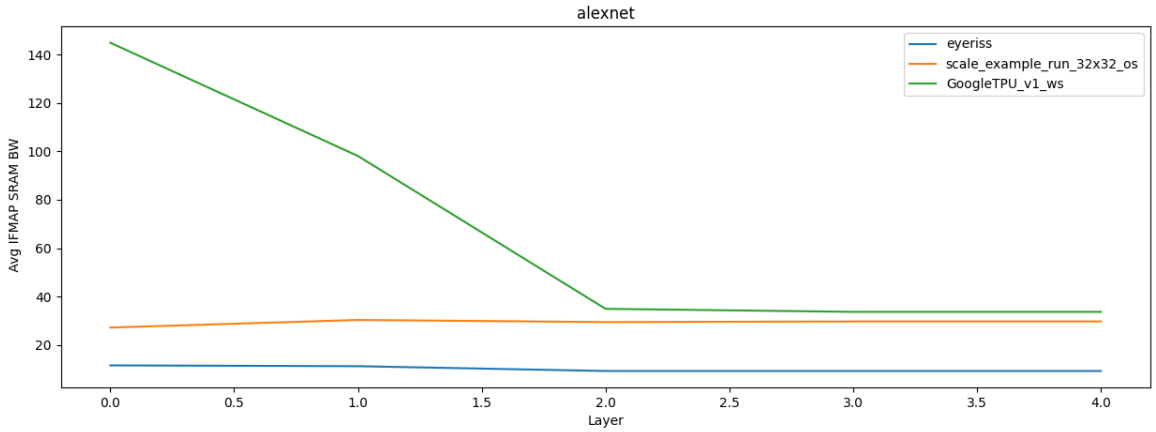


Figure 76: alexnet_ifmap_sram_bw

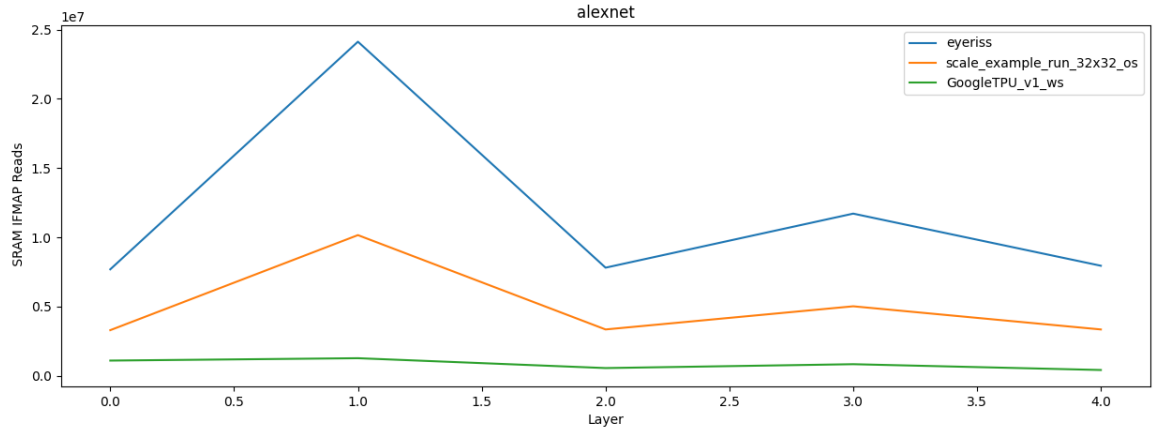


Figure 77: alexnet_SRAM_IFMAP_READS

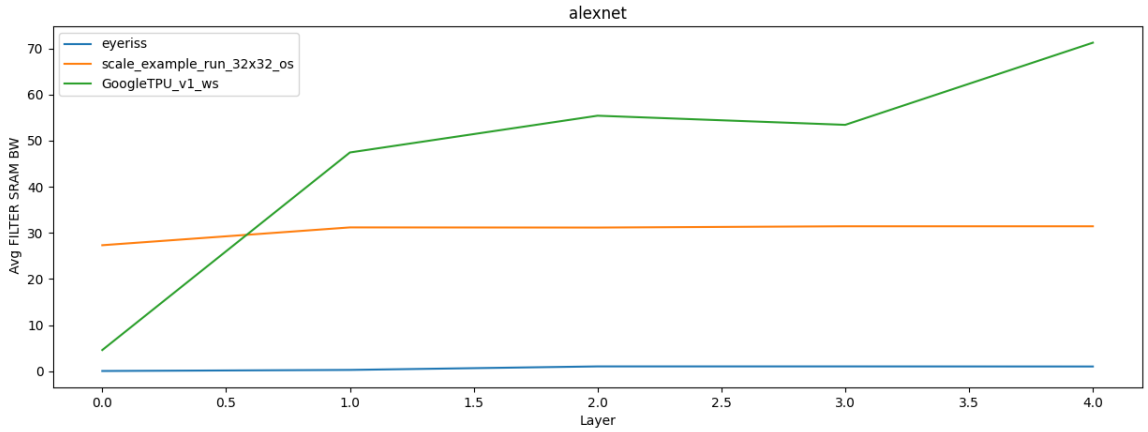


Figure 78: alexnet_filter_sram_bw

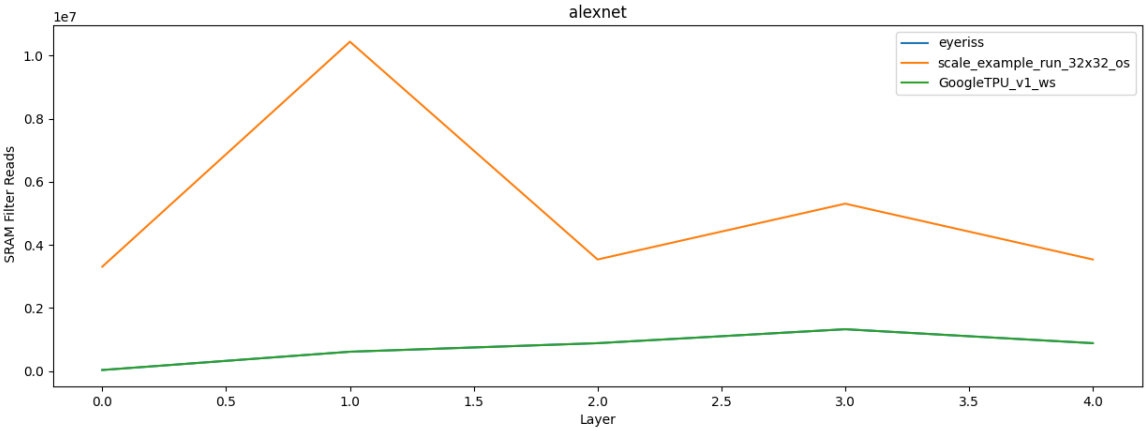


Figure 79: alexnet_SRAM_Filter_READS

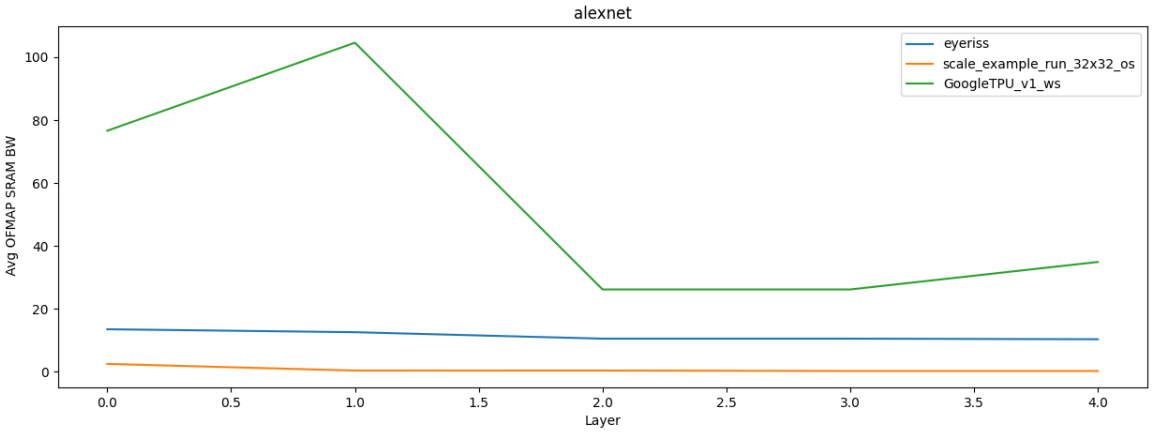


Figure 80: alexnet_ofmap_sram_bw

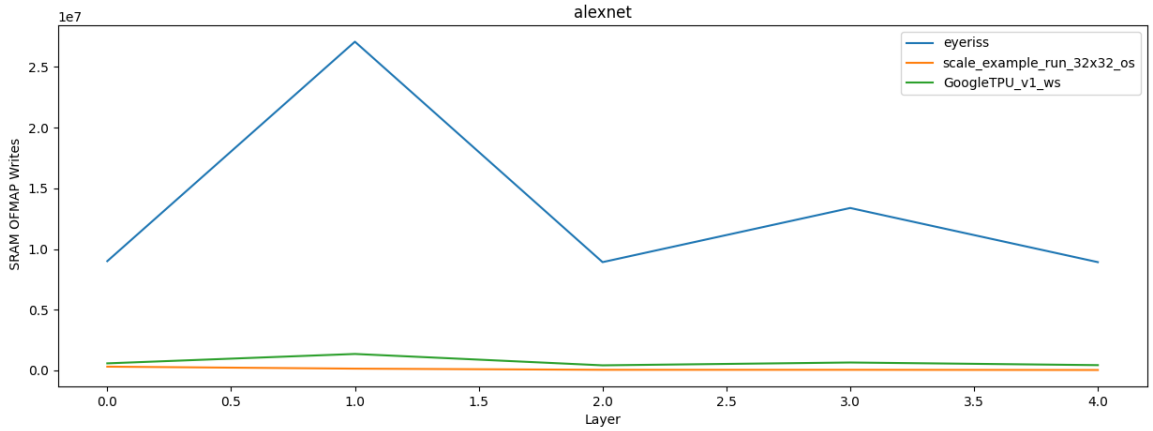


Figure 81: alexnet_SRAM_OFMAP_WRITES

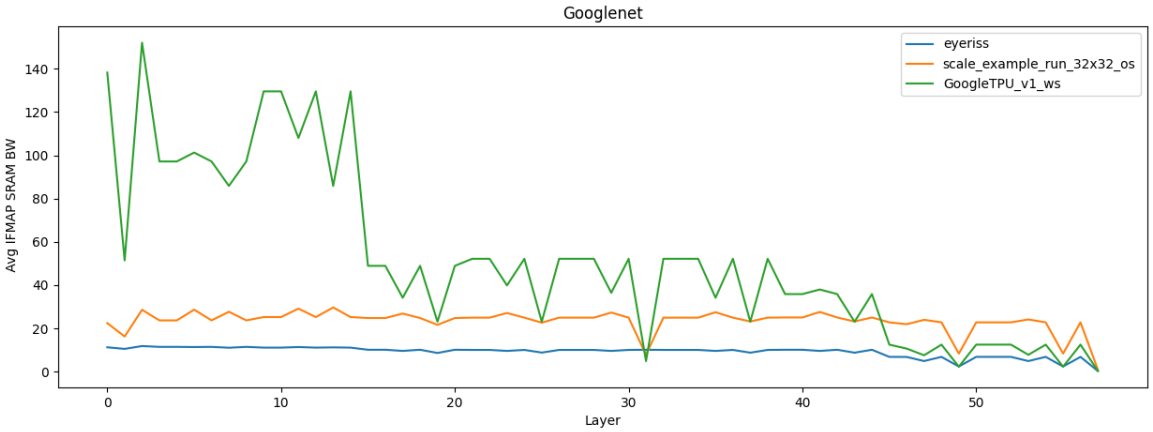


Figure 82: googlenet_ifmap_sram_bw

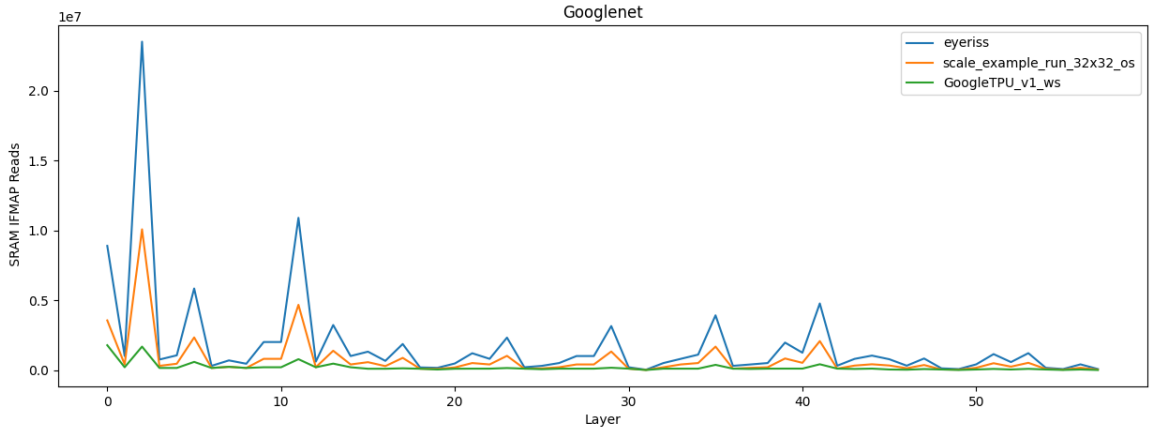


Figure 83: googlenet_SRAM_IFMAP_READS

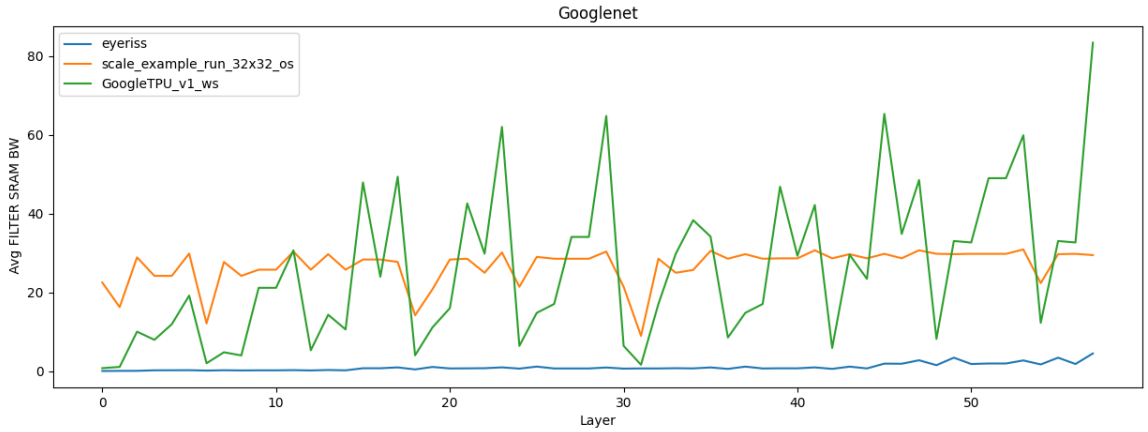


Figure 84: googlenet_filter_sram_bw

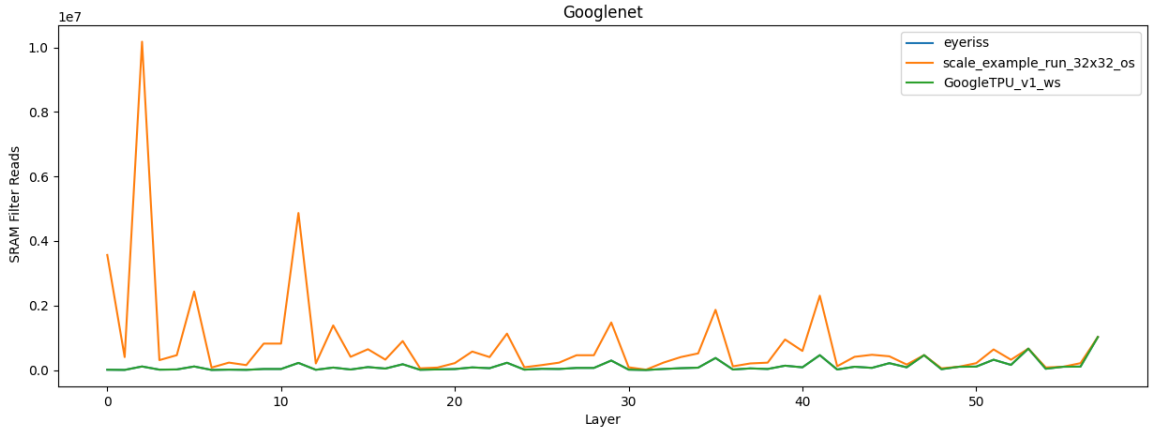


Figure 85: googlenet_SRAM_Filter_READS

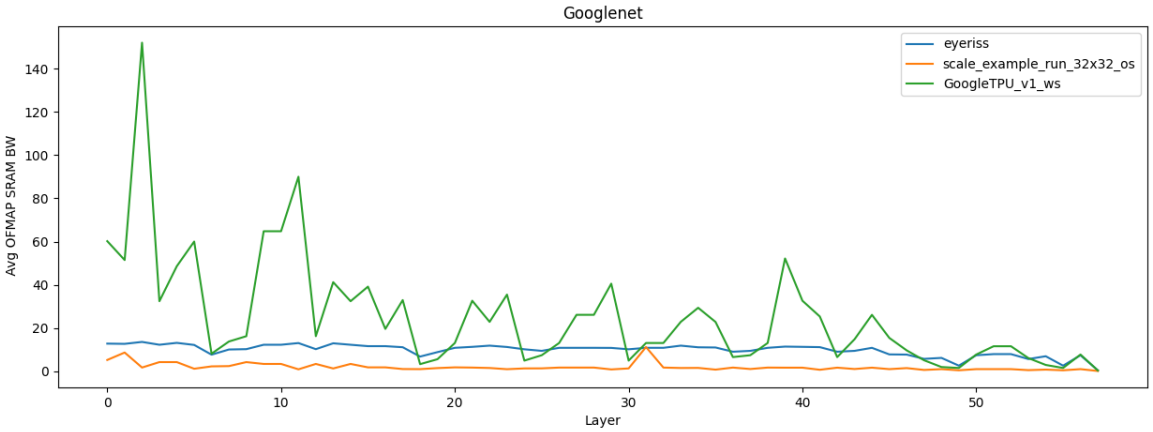


Figure 86: googlenet_ofmap_sram_bw

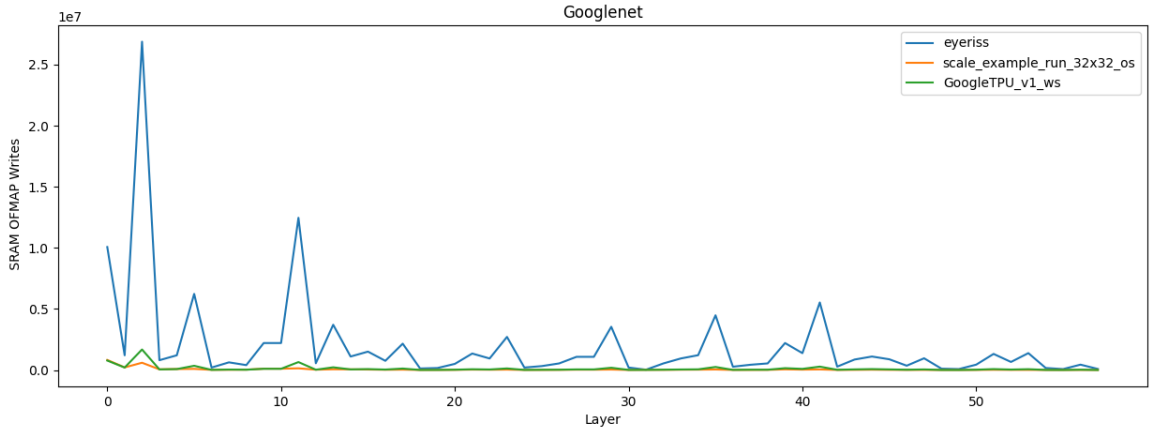


Figure 87: googlenet_SRAM_OFMAP_WRITES

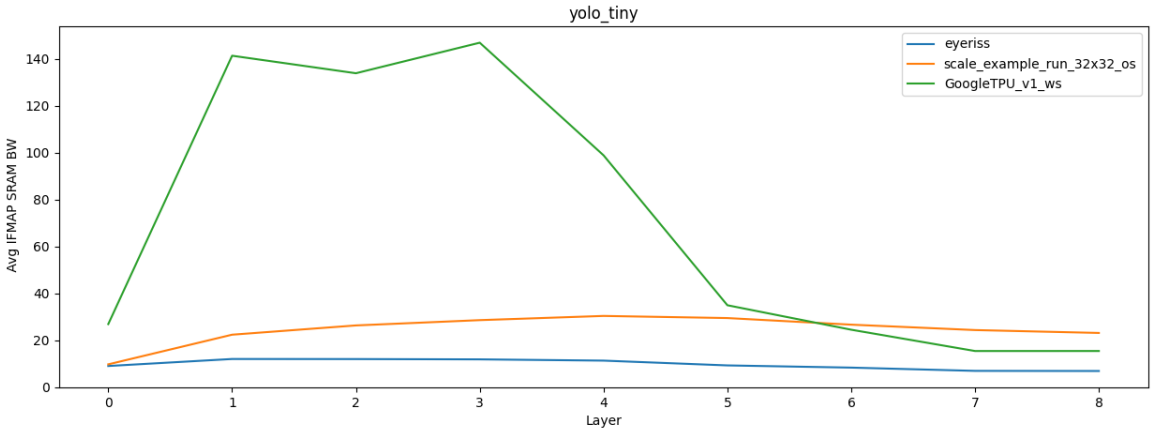


Figure 88: yolo_tiny_ifmap_sram_bw

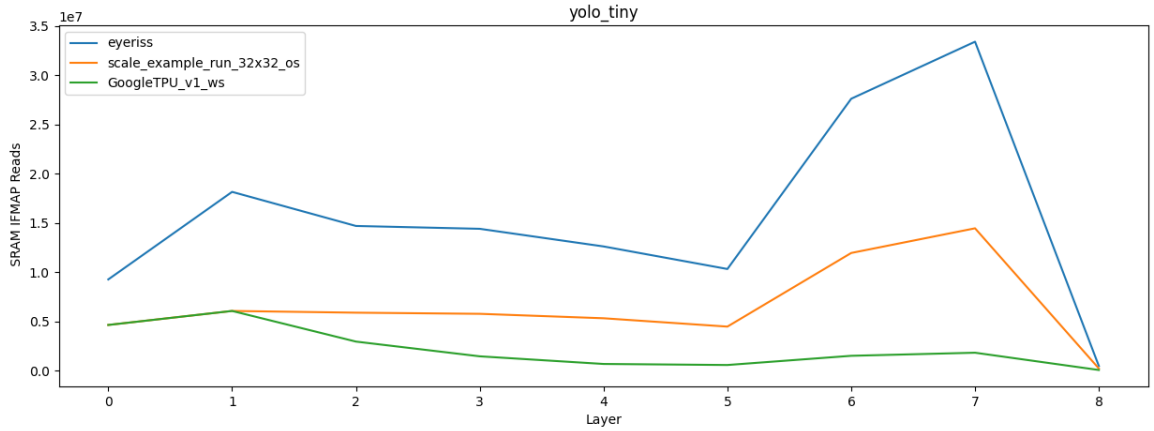


Figure 89: yolo_tiny_SRAM_IFMAP_READS

Changing dataflows

Three different DNNs were used: mobilenet, Resnet18, and FasterRCNN.

Three different dataflows for eyeriss configuration were used: OS, WS, and IS.

Total Runtime

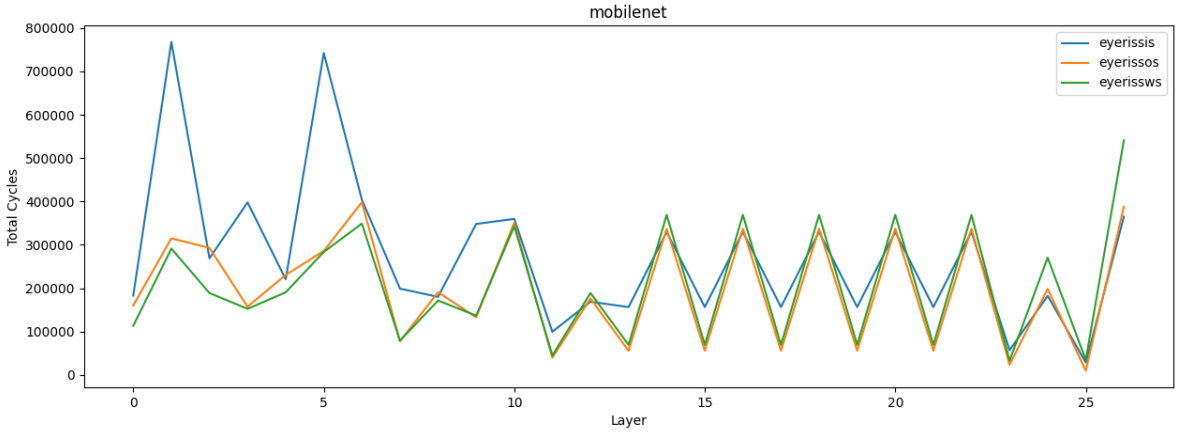


Figure 90: mobilenet_runtime

We can see that the total runtime is highest for IS in the initial layers. This is because the number of filters is not a factor in determining the mapping efficiency in IS dataflow, while it is the main factor in determining the mapping efficiency in OS and WS dataflows.

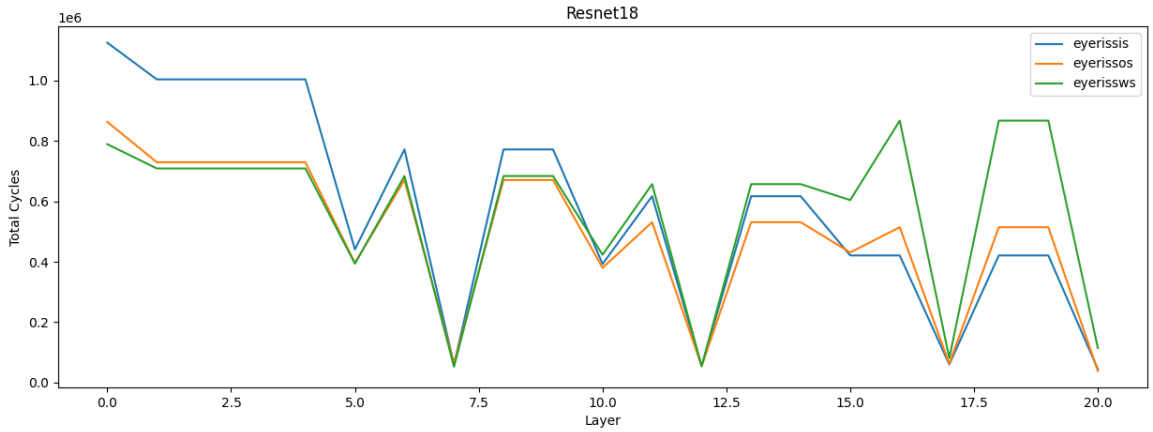


Figure 91: resnet18_runtime

A similar order is observed in resnet18 as well.

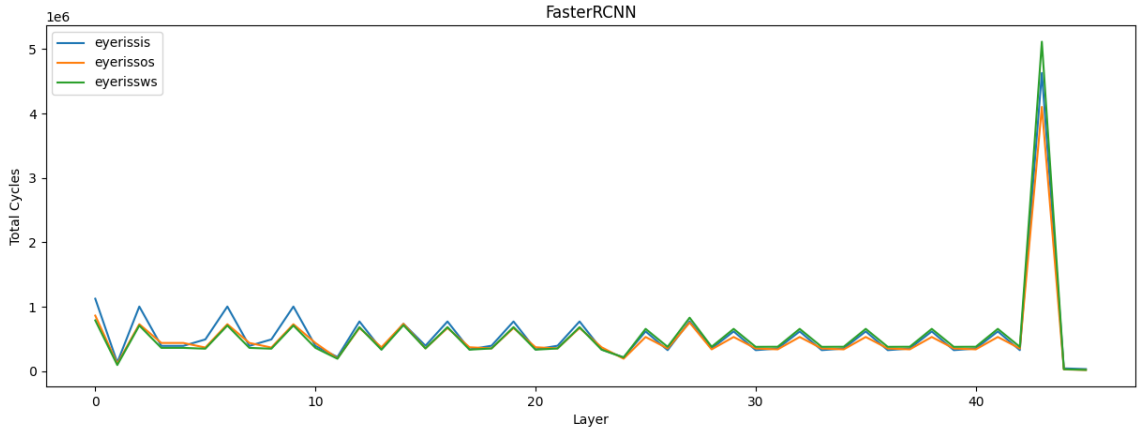


Figure 92: fasterRCNN_runtime

A similar order is observed in fasterRCNN as well.

Mapping Efficiency and Compute Utilization

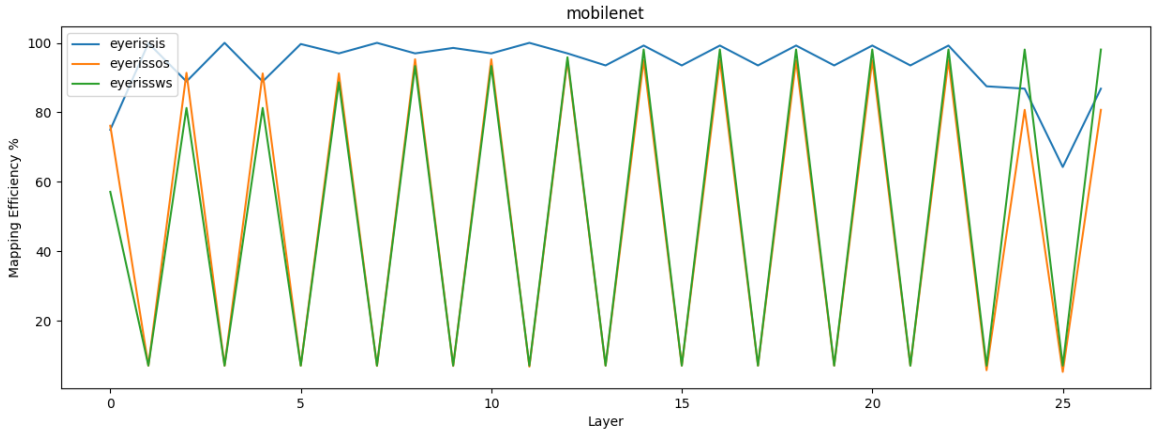


Figure 93: mobilenet_mapping

For mobilenet, the mapping efficiency is highest for IS dataflow and stays constant. This is because the number of filters is not a factor in determining the mapping efficiency in IS dataflow, while it is the main factor in determining the mapping efficiency in OS and WS dataflows.

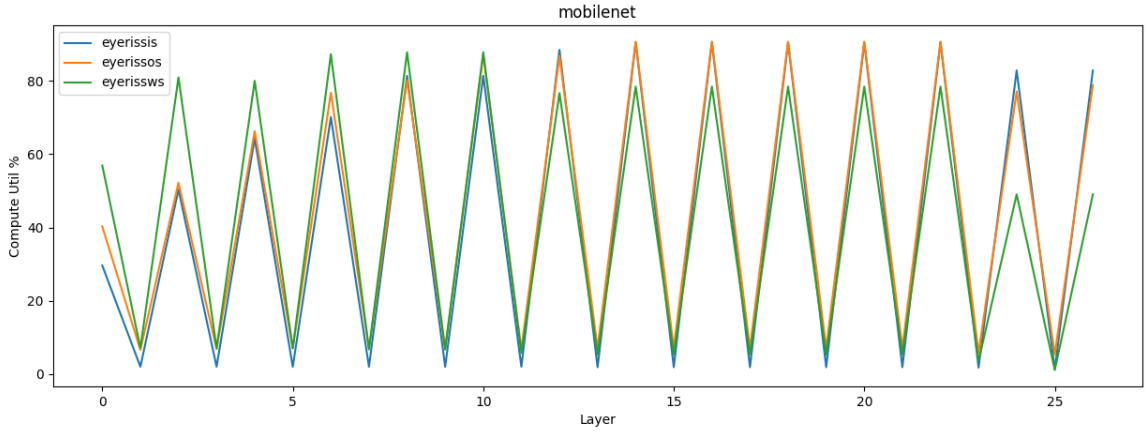


Figure 94: mobilenet_compute

Here, however we see that all three dataflows have similar compute utilization. This is due to compute utilization being dependent on T, that is the number of timesteps too.

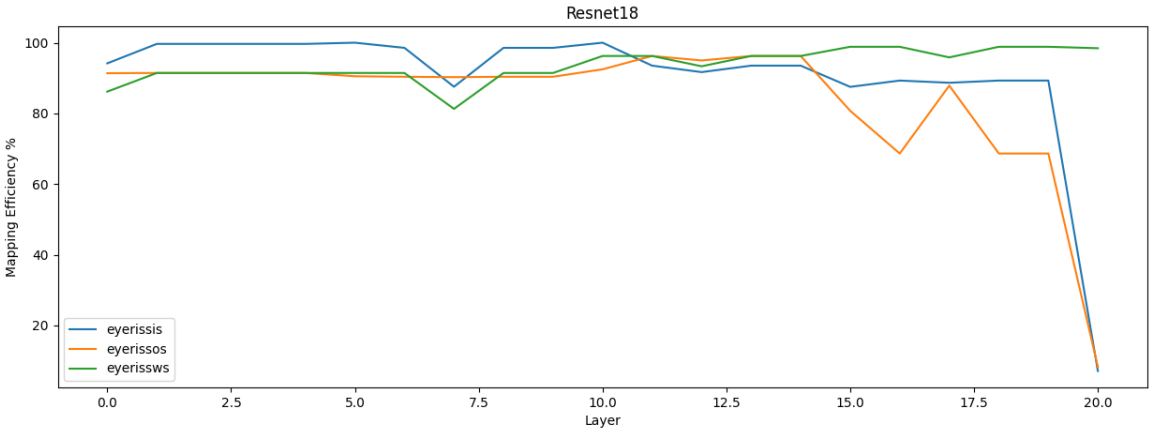


Figure 95: resnet18_mapping

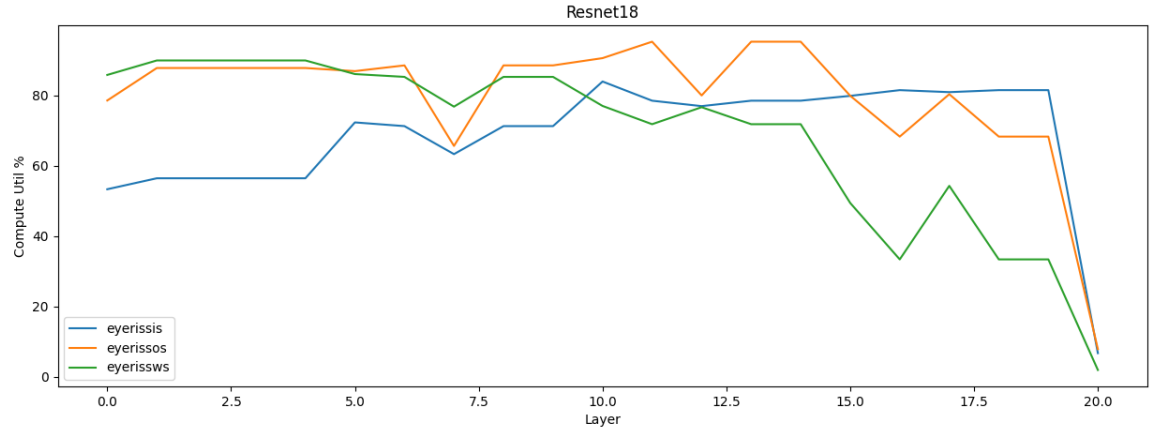


Figure 96: resnet18_compute

A similar order is observed in resnet18 as well.

A similar order is observed in fasterRCNN as well.

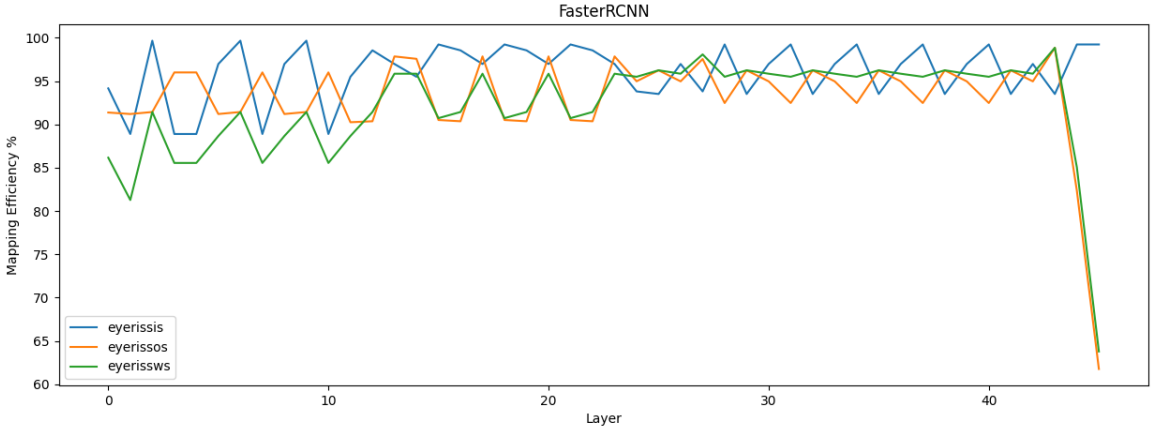


Figure 97: fasterRCNN_mapping

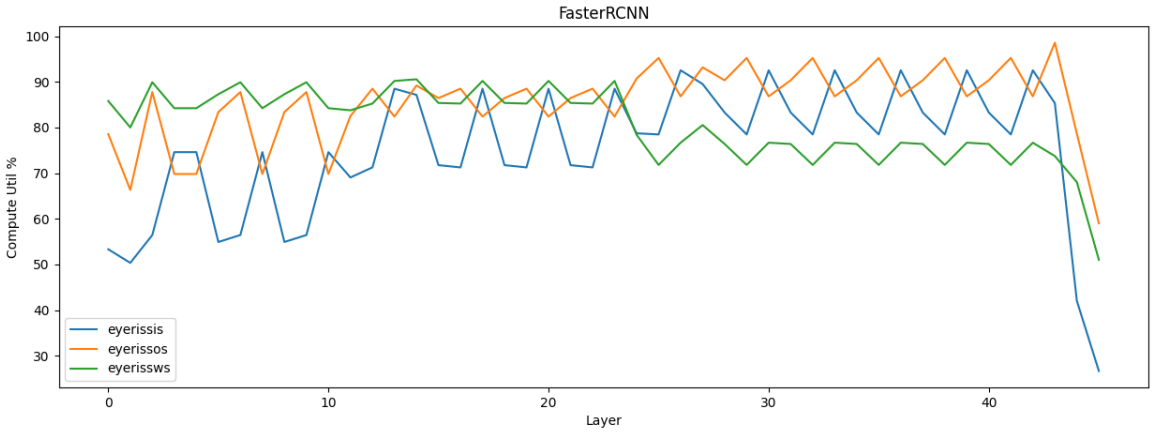


Figure 98: fasterRCNN_compute

DRAM Bandwidth Utilization and Read/Writes

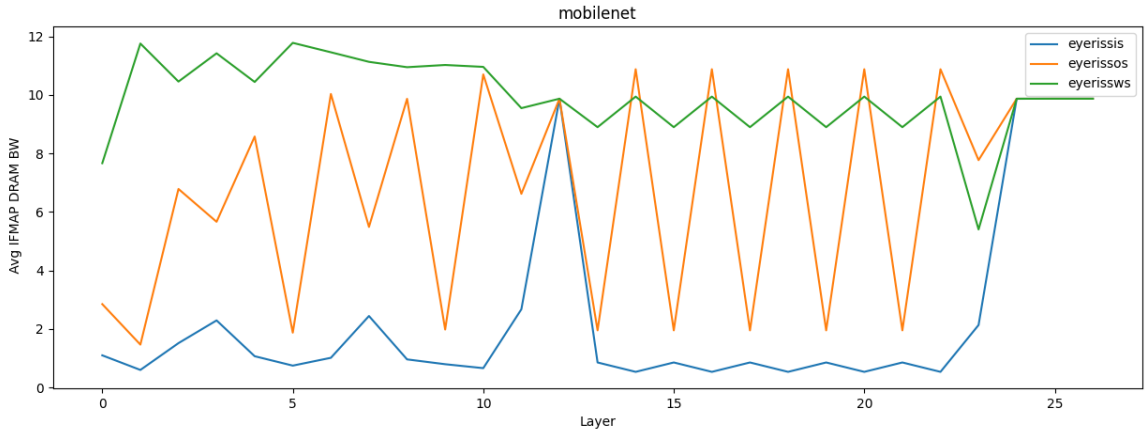


Figure 99: mobilenet_ifmap_dram_bw

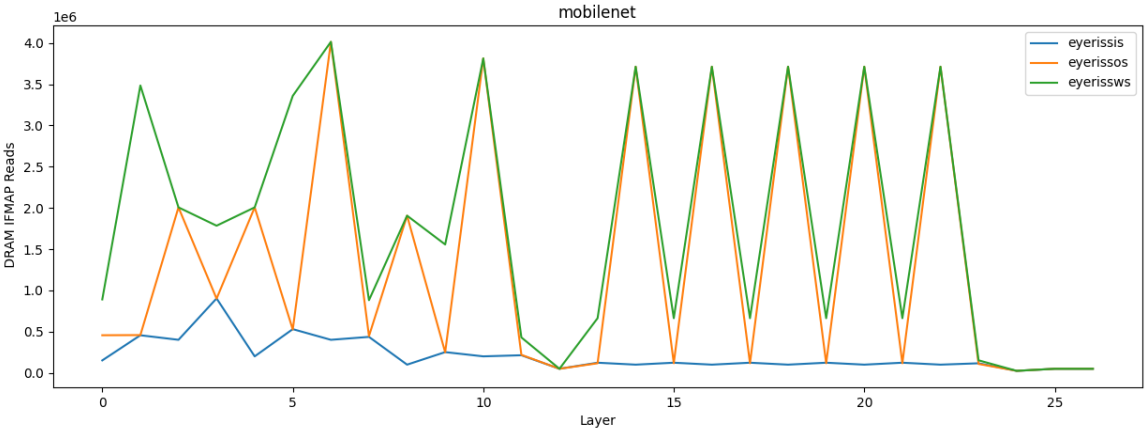


Figure 100: mobilenet_DRAM_IFMAP_READS

Here the reads and bandwidth of the input memory is lowest for the Input Stationary dataflow because the dataflow dictates that the input memory is read only the minimum number of times and then the same data is used for all the filters.

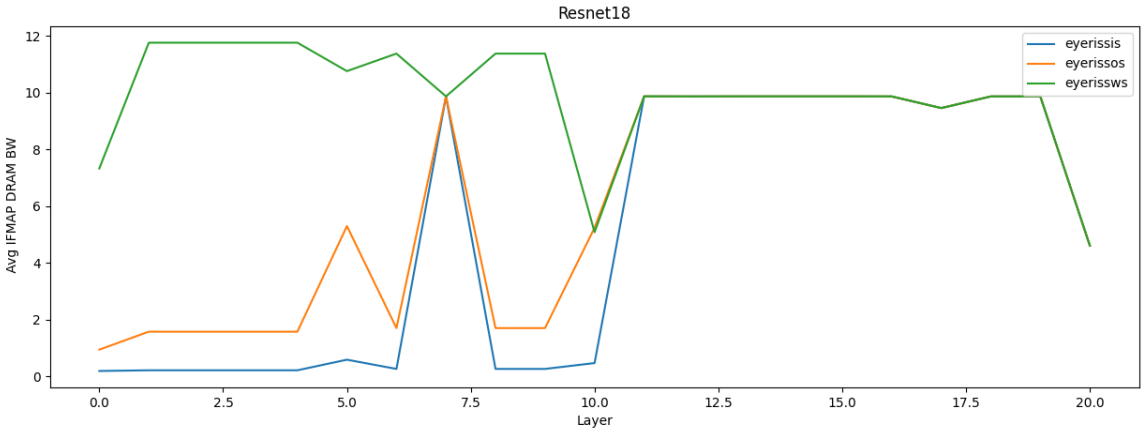


Figure 101: resnet18_ifmap_dram_bw

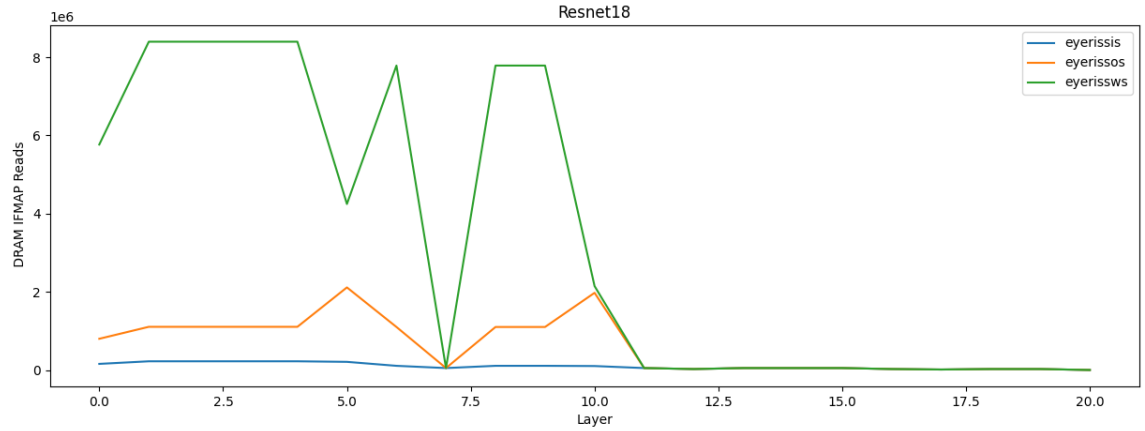


Figure 102: resnet18_DRAM_IFMAP_READS

A similar order is observed in resnet18 as well.

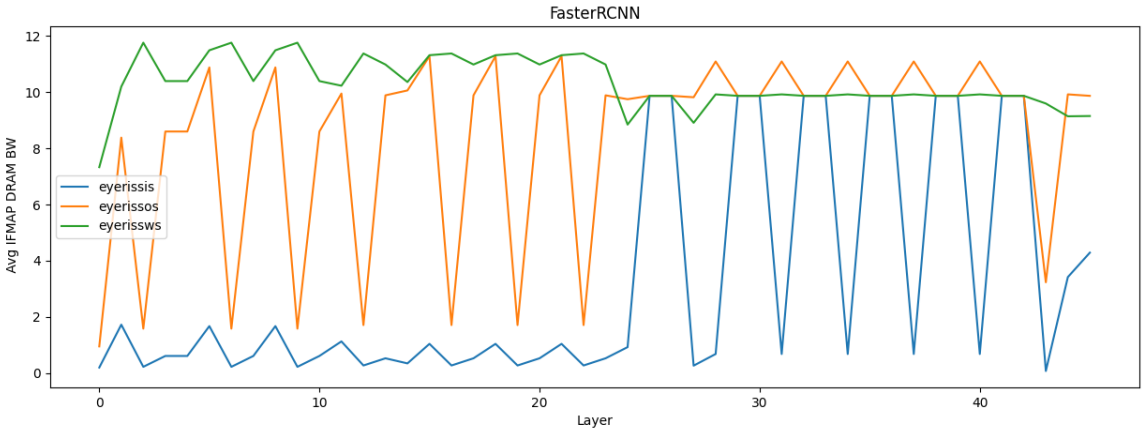


Figure 103: fasterRCNN_ifmap_dram_bw

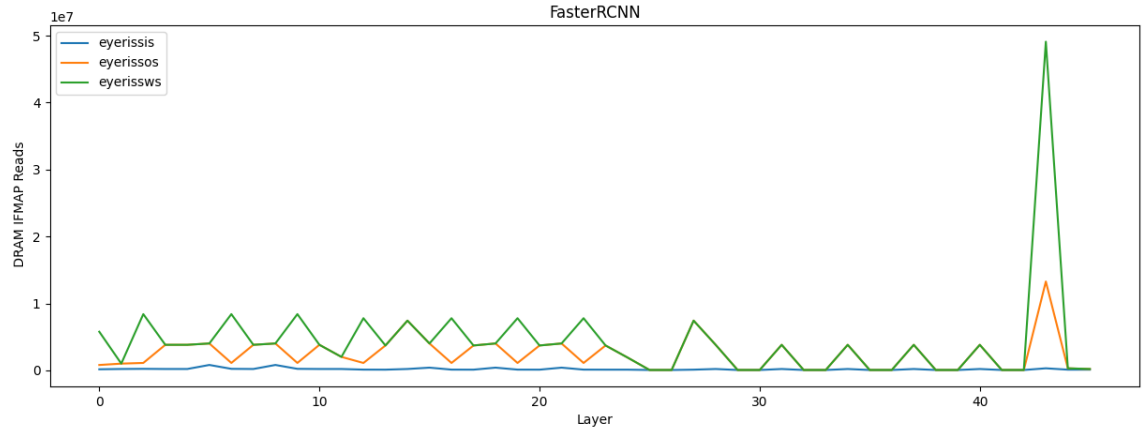


Figure 104: fasterRCNN_DRAM_IFMAP_READS

A similar order is observed in fasterRCNN as well.

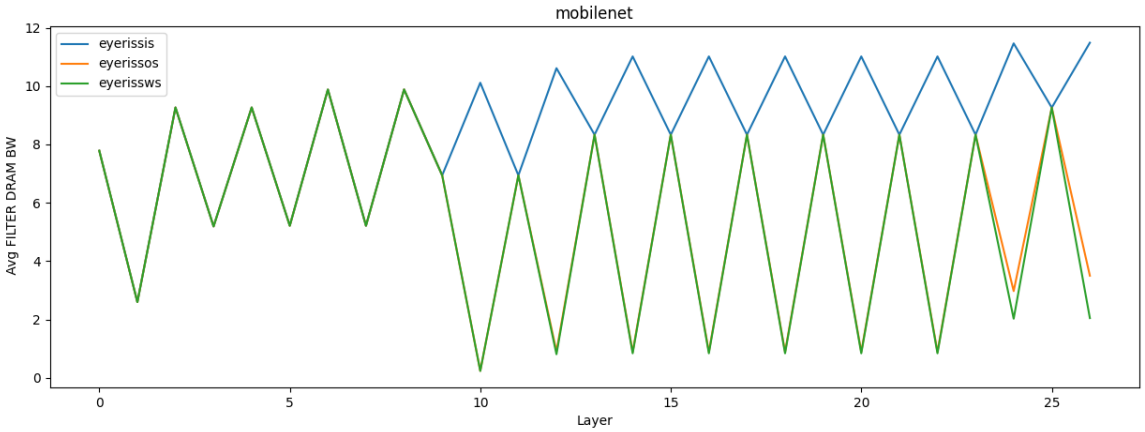


Figure 105: mobilenet_filter_dram_bw

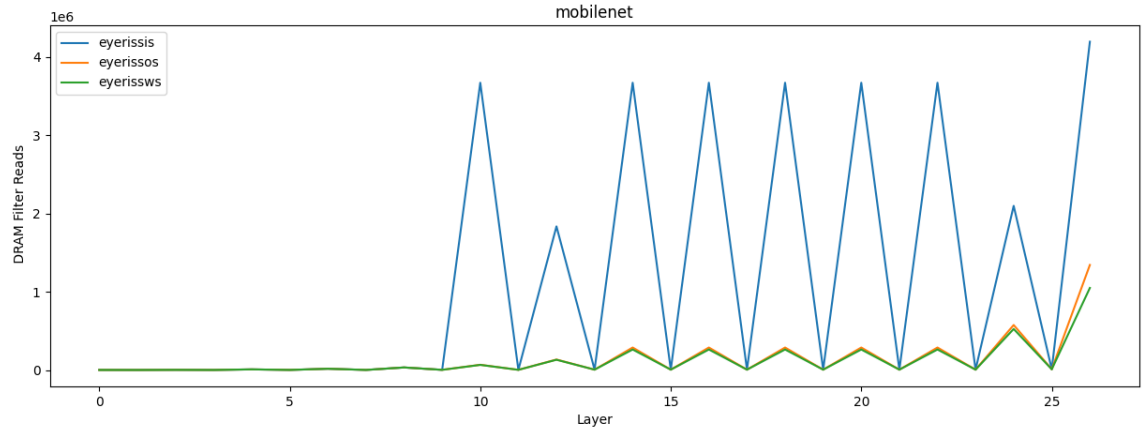


Figure 106: mobilenet_DRAM_Filter_READS

Here the reads and bandwidth of the filter memory is highest for the Input Stationary dataflow because the dataflow dictates that the filter memory is read the maximum number of times.

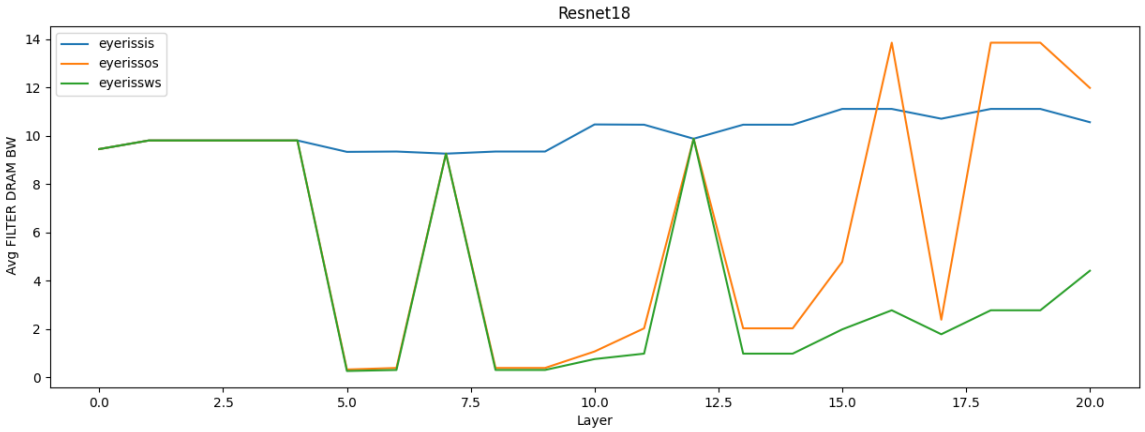


Figure 107: resnet18_filter_dram_bw

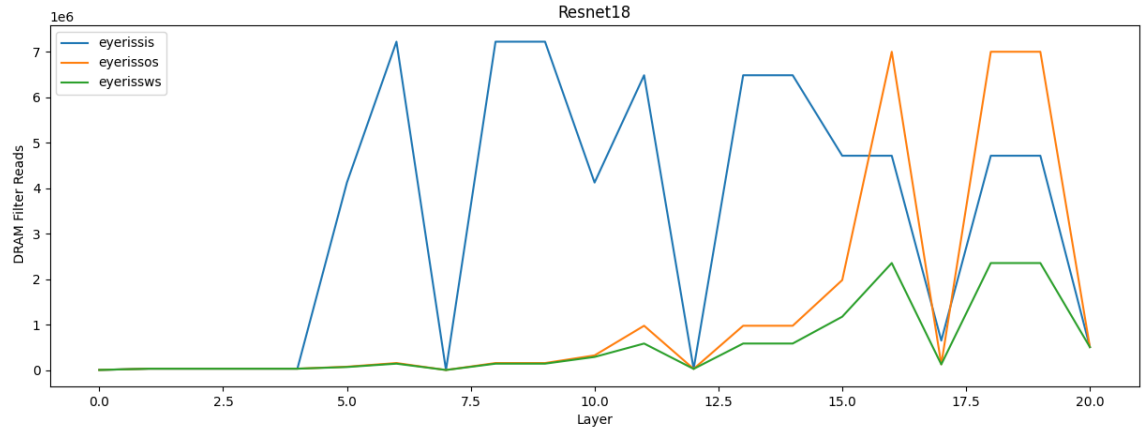


Figure 108: resnet18_DRAM_Filter_READS

A similar order is observed in resnet18 as well.

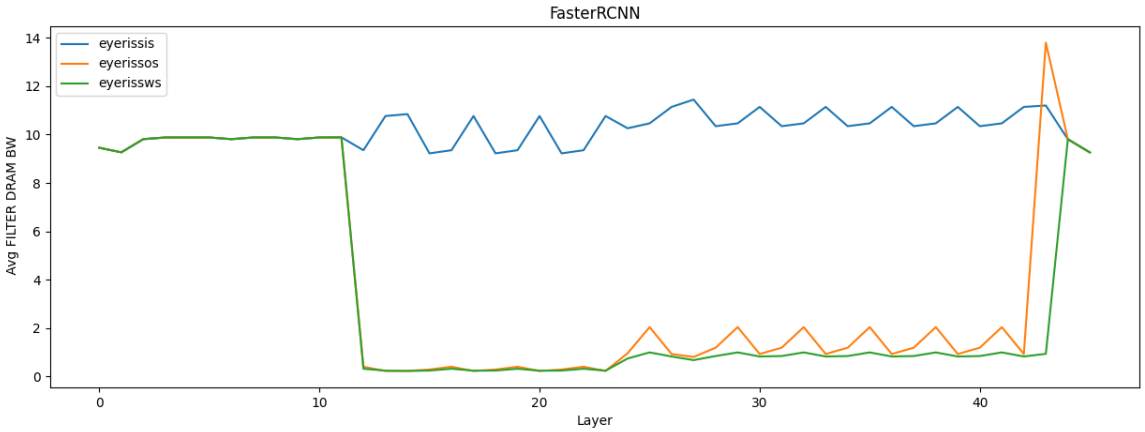


Figure 109: fasterRCNN_filter_dram_bw

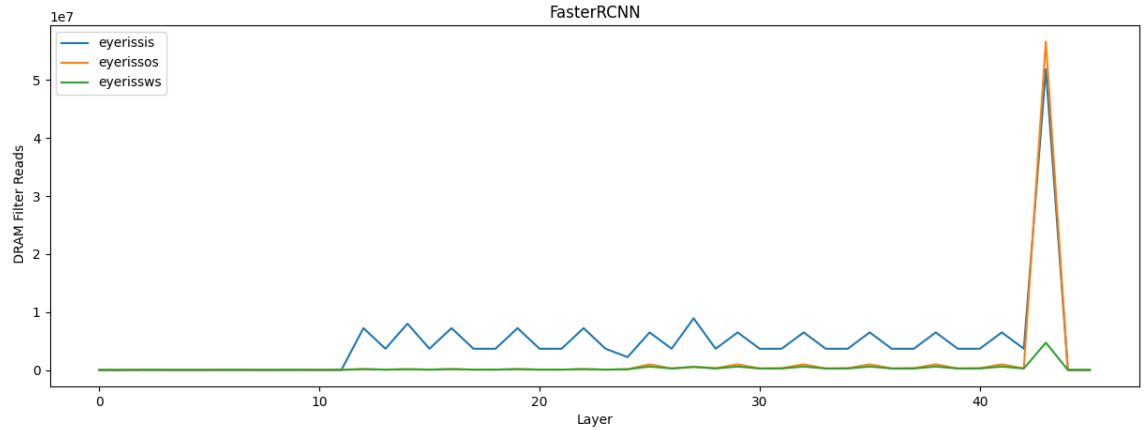


Figure 110: fasterRCNN_DRAM_Filter_READS

A similar order is observed in fasterRCNN as well.

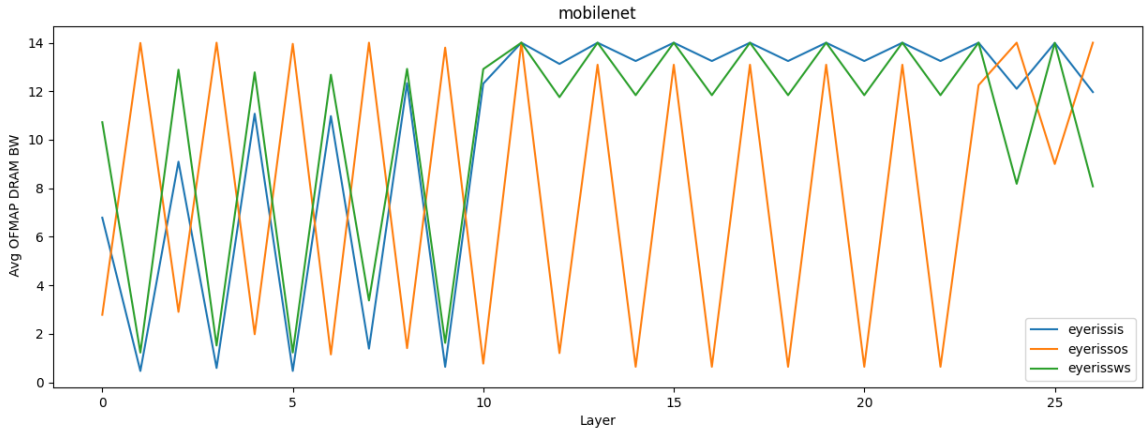


Figure 111: mobilenet_ofmap_dram_bw

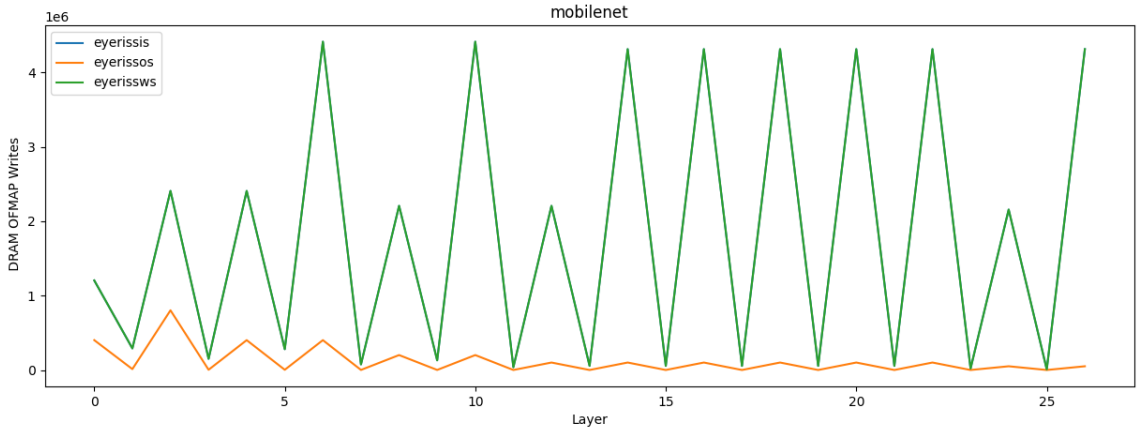


Figure 112: mobilenet_DRAM_OFMAP_WRITES

Here the bandwidth depends on the number of filters and the number of timesteps. Which is why the bandwidth varies like it does for OS graph. The WS and IS dataflow graphs vary similarly but in the opposite direction due to the same reason.

For the number of writes however, the number of writes is lowest for the Output Stationary dataflow because the dataflow dictates that the output memory is written only the minimum number of times.

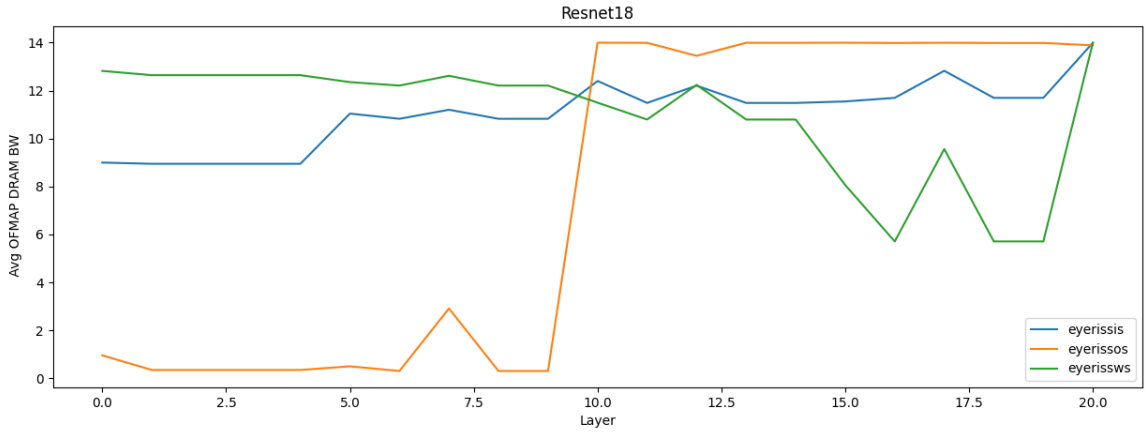


Figure 113: resnet18_ofmap_dram_bw

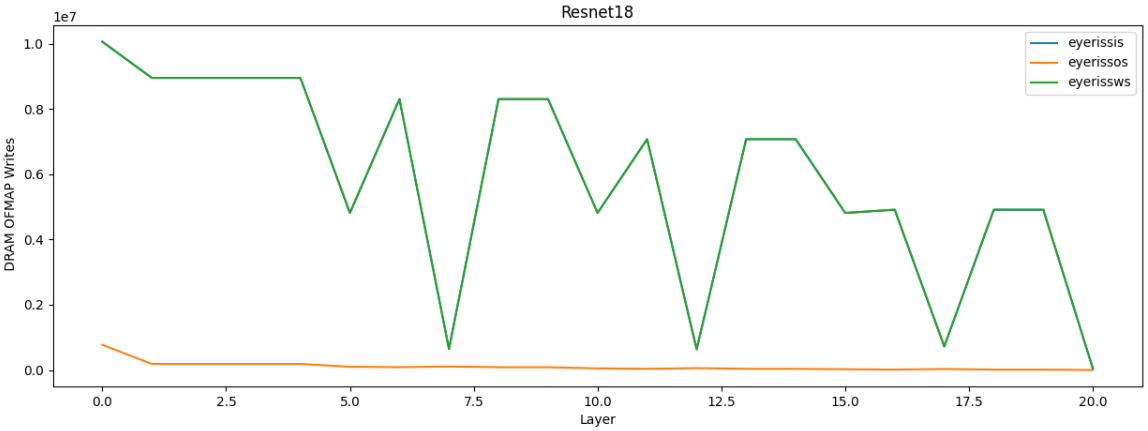


Figure 114: resnet18_DRAM_OFMAP_WRITES

A similar order is observed in resnet18 as well.

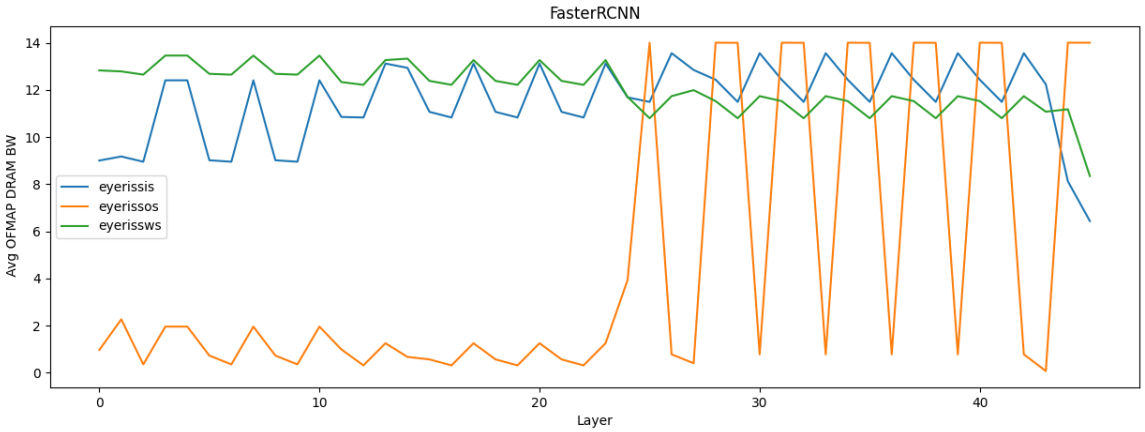


Figure 115: fasterRCNN_ofmap_dram_bw

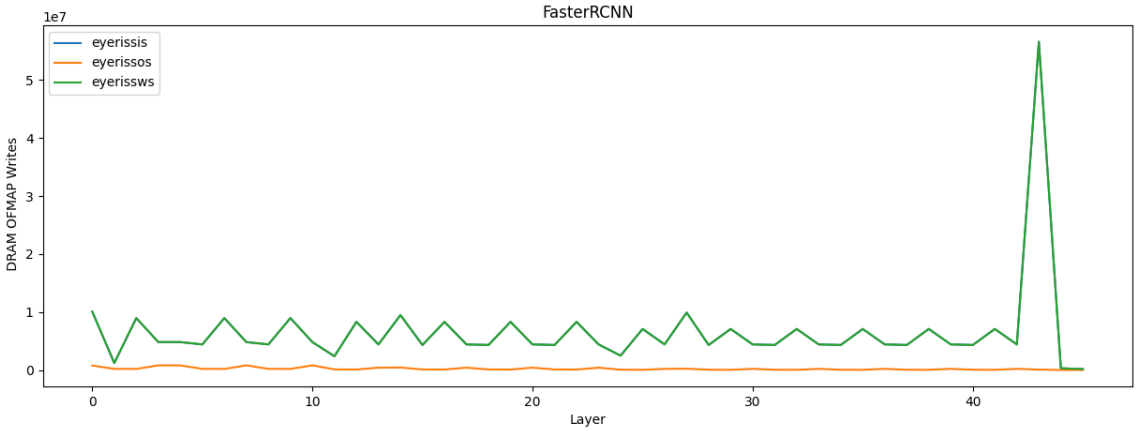


Figure 116: fasterRCNN_DRAM_OFMAP_WRITES

A similar order is observed in fasterRCNN as well.

SRAM Bandwidth Utilization and Read/Writes

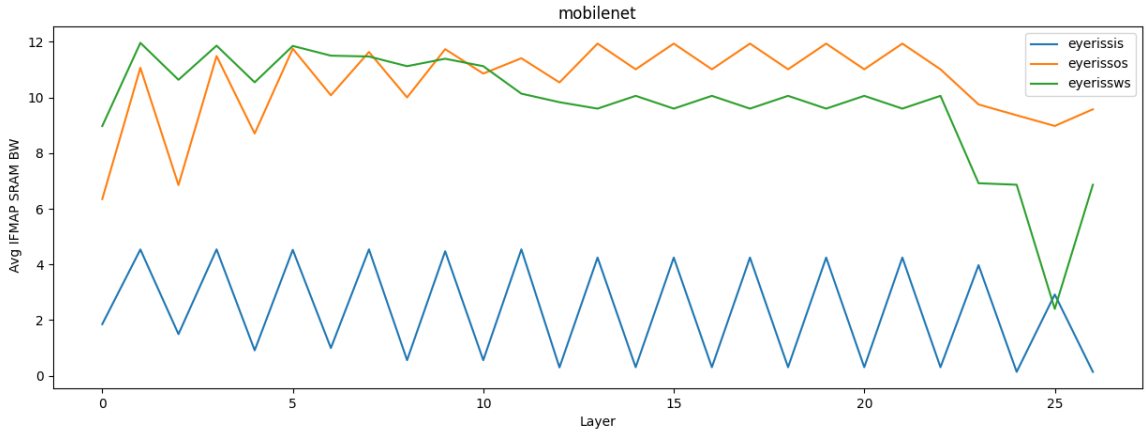


Figure 117: mobilenet_ifmap_sram_bw

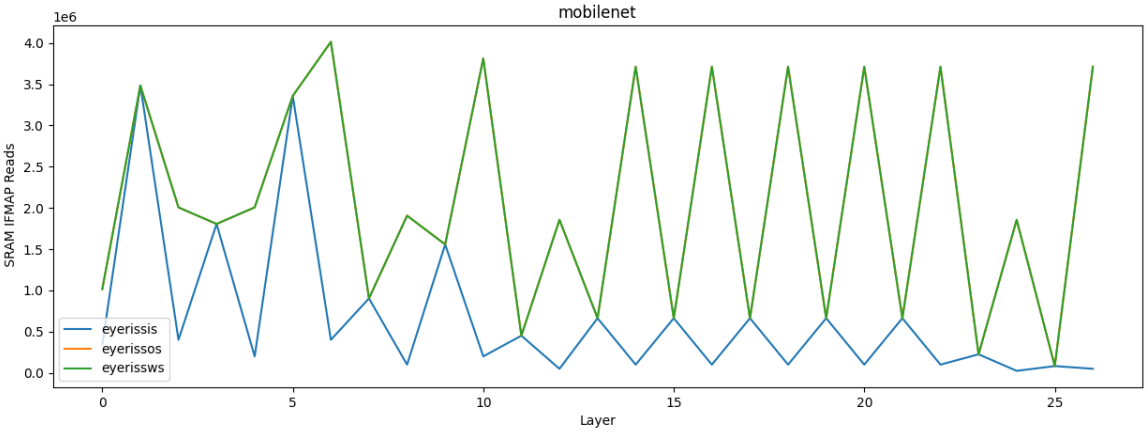


Figure 118: mobilenet_SRAM_IFMAP_READS

Here the reads and bandwidth of the input memory is the lowest for the Input Stationary dataflow because the dataflow dictates that the input memory is read only the minimum number of times and then the same data is used for all the filters.

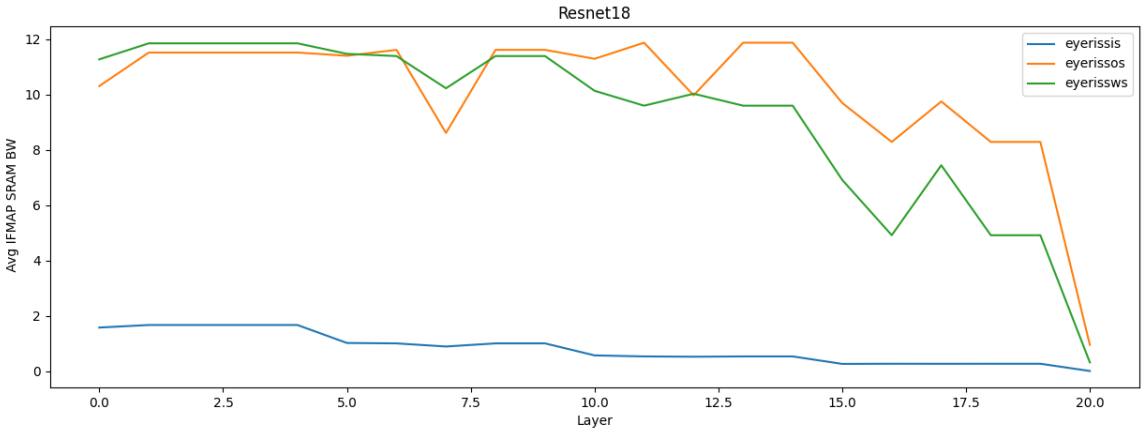


Figure 119: resnet18_ifmap_sram_bw

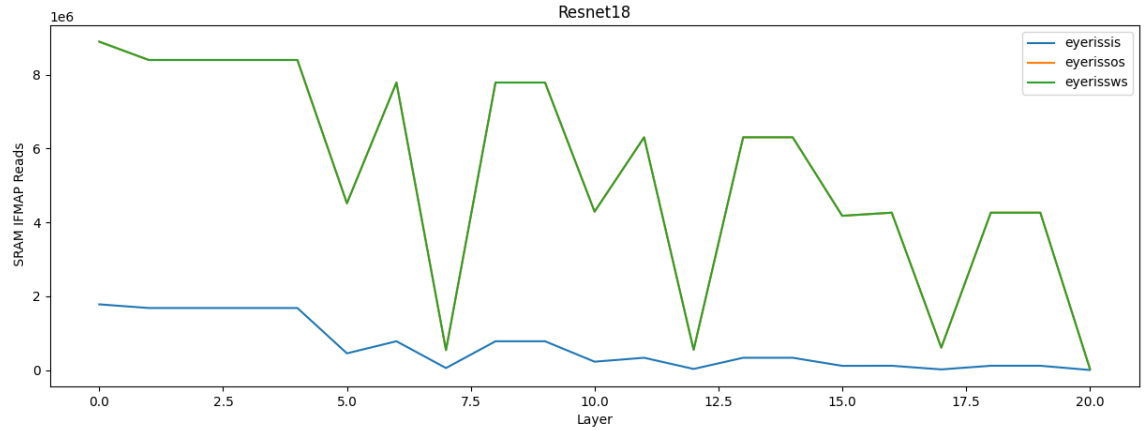


Figure 120: resnet18_SRAM_IFMAP_READS

A similar order is observed in resnet18 as well.

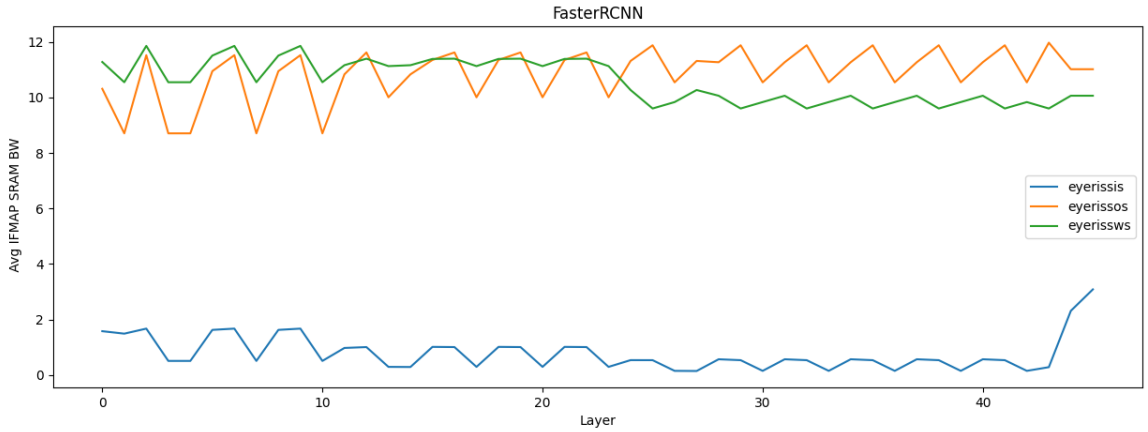


Figure 121: fasterRCNN_ifmap_sram_bw

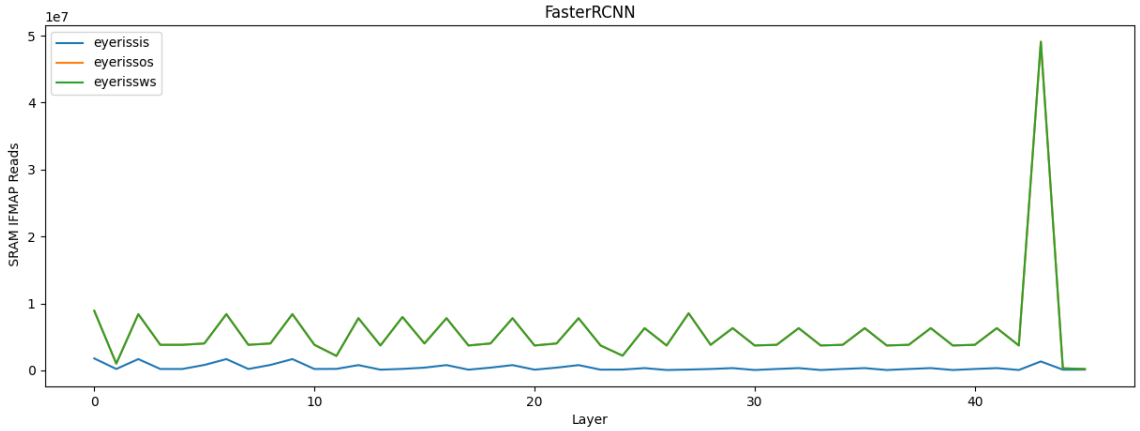


Figure 122: fasterRCNN_SRAM_IFMAP_READS

A similar order is observed in fasterRCNN as well.

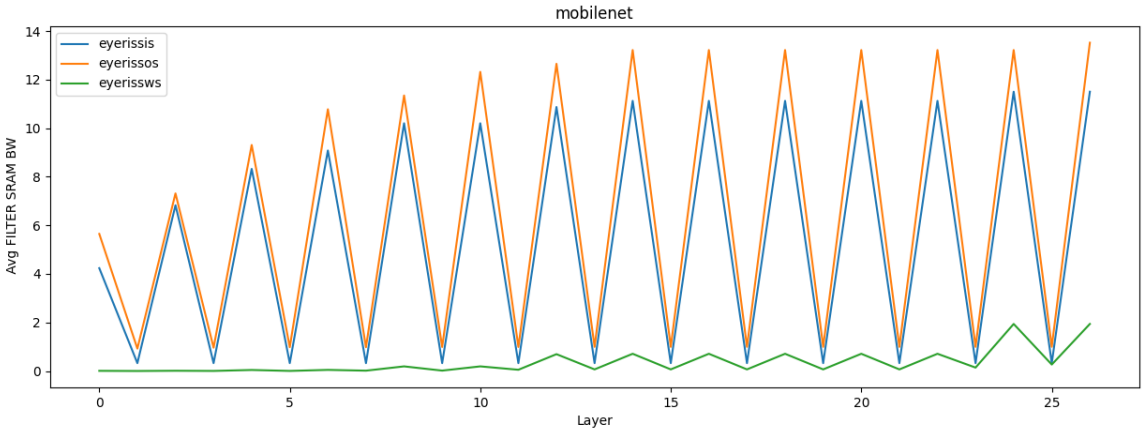


Figure 123: mobilenet_filter_sram_bw

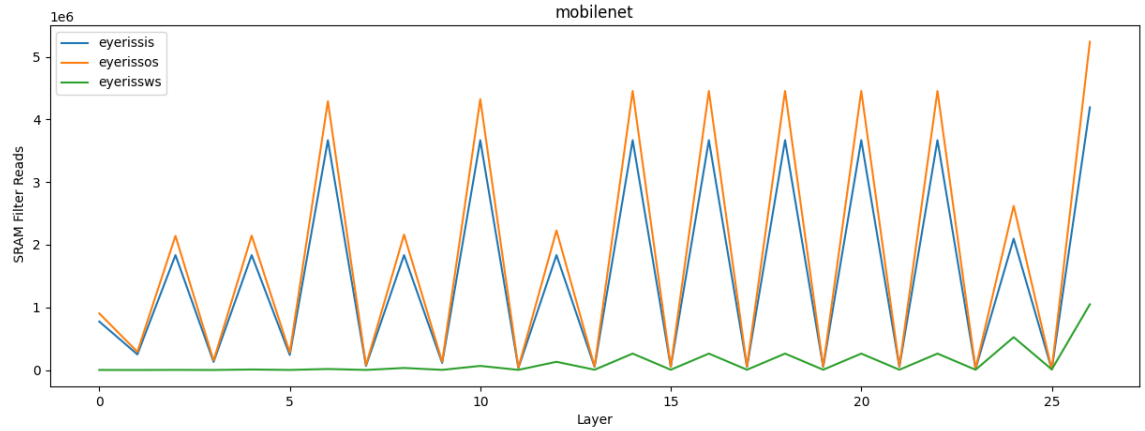


Figure 124: mobilenet_SRAM_Filter_READS

Here the reads and bandwidth of the filter memory is the lowest for Weight Stationary dataflow because the dataflow dictates that the filter memory is read the minimum number of times and then the same data is used for all the input.

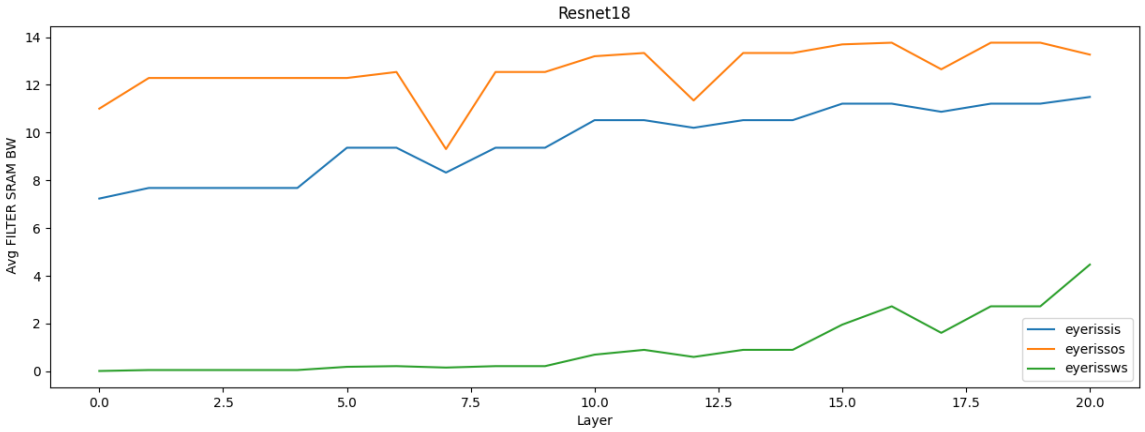


Figure 125: resnet18_filter_sram_bw

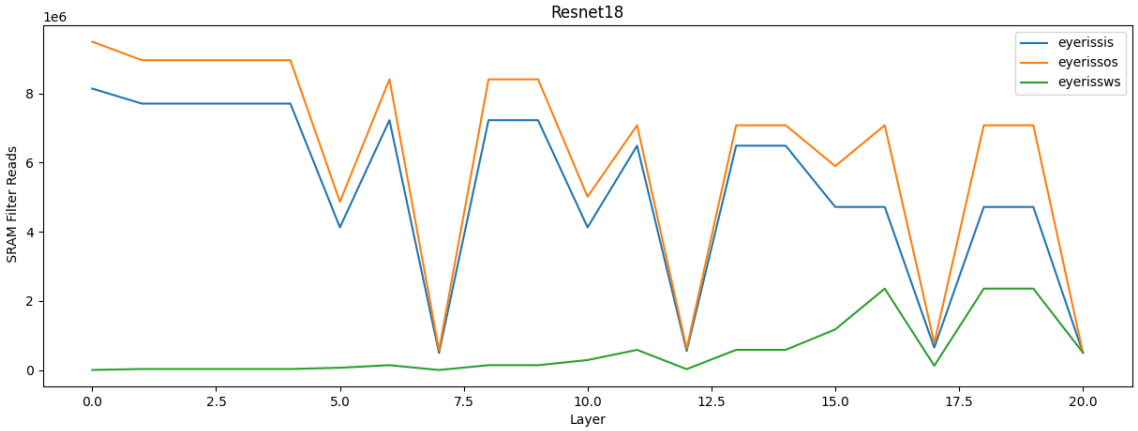


Figure 126: resnet18_SRAM_Filter_READS

A similar order is observed in resnet18 as well.

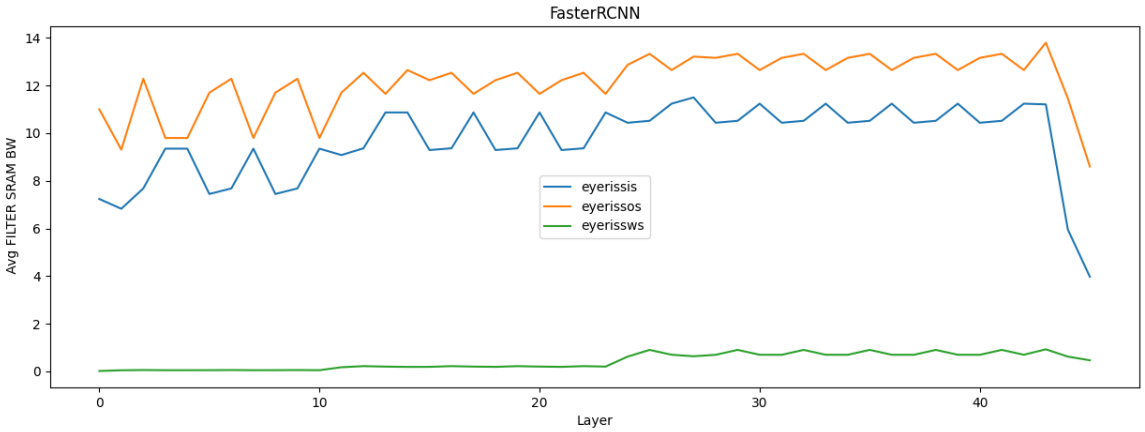


Figure 127: fasterRCNN_filter_sram_bw

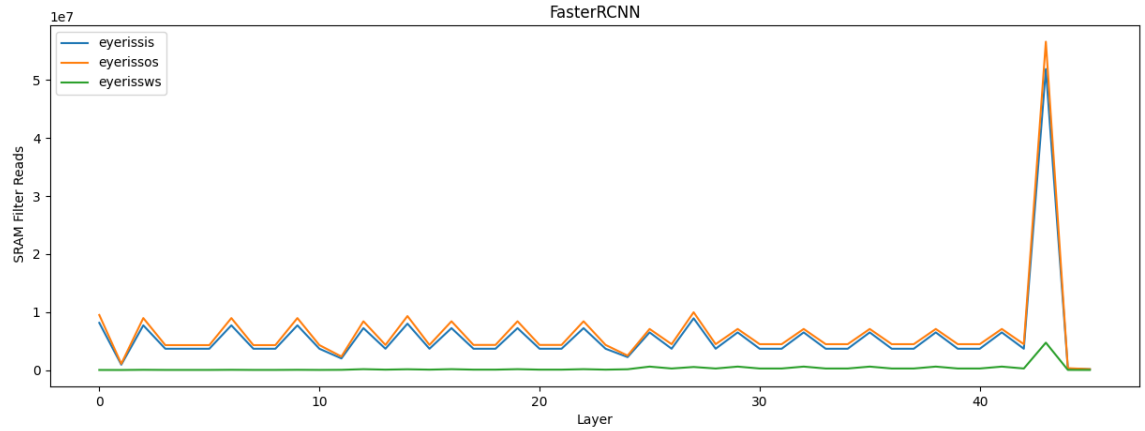


Figure 128: fasterRCNN_SRAM_Filter_READS

A similar order is observed in fasterRCNN as well.

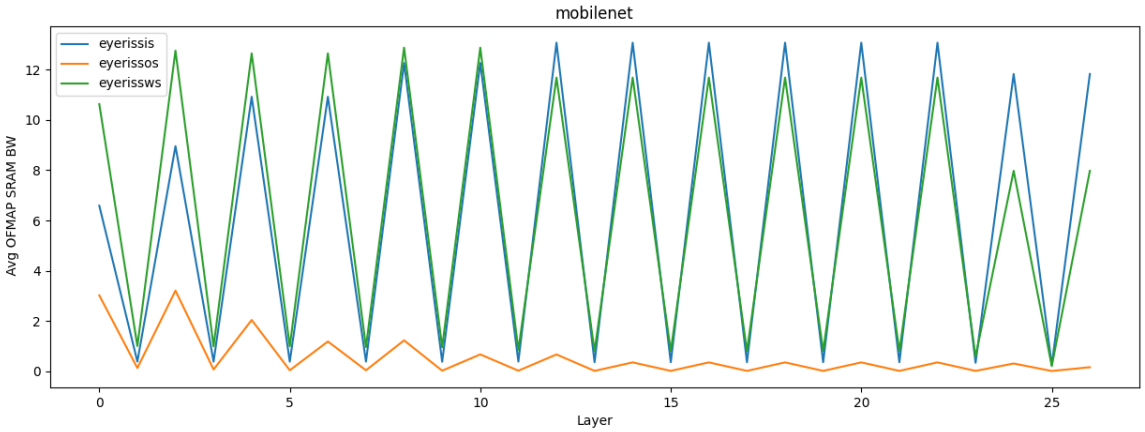


Figure 129: mobilenet_ofmap_sram_bw

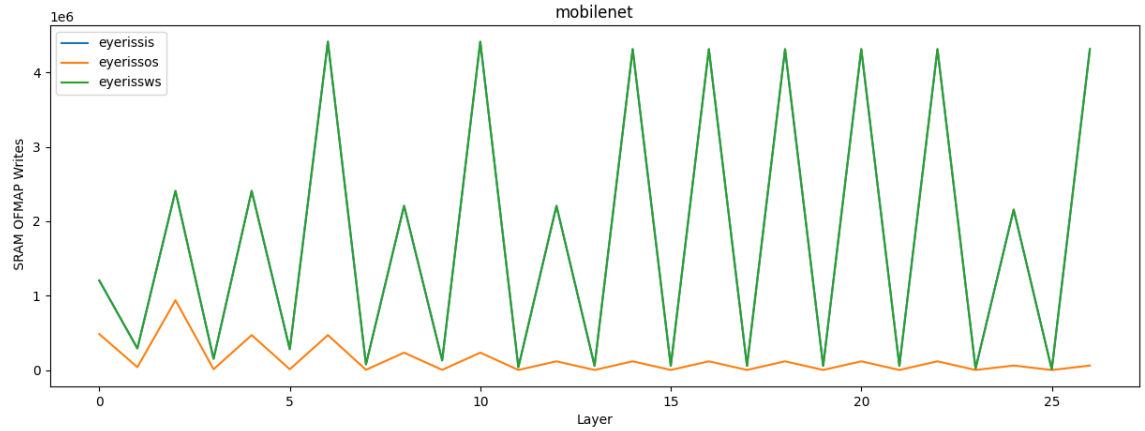


Figure 130: mobilenet_SRAM_OFMAP_WRITES

Here the reads and bandwidth of the output memory is the lowest for the Output Stationary dataflow because the dataflow dictates that the output memory is written only the minimum number of times.

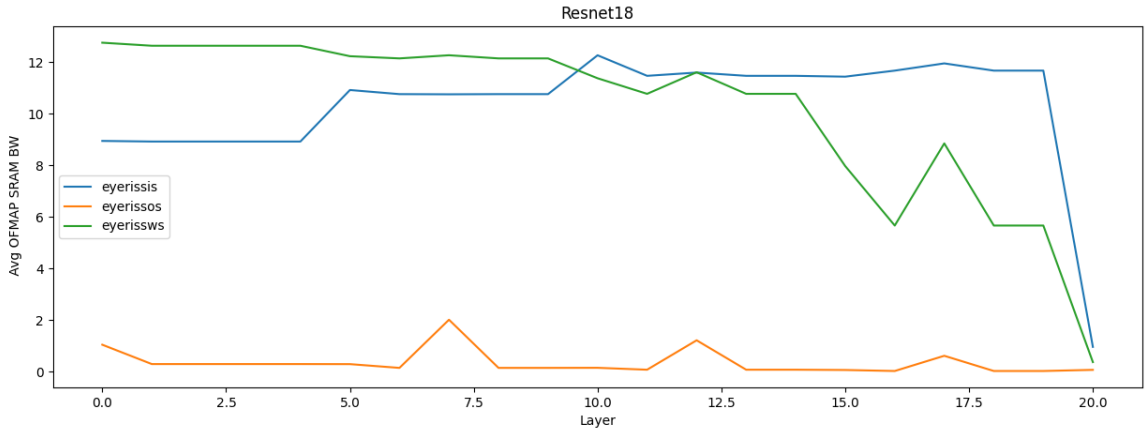


Figure 131: resnet18_ofmap_sram_bw

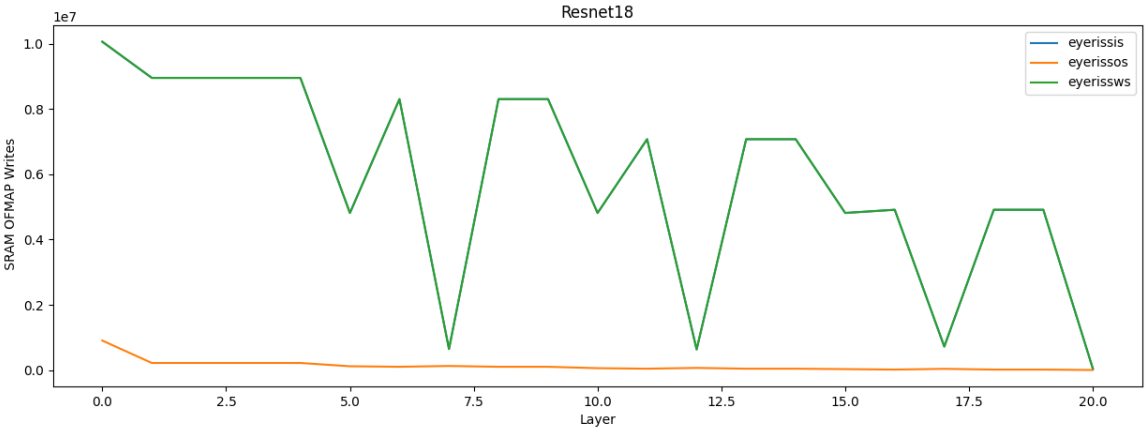


Figure 132: resnet18_SRAM_OFMAP_WRITES

A similar order is observed in resnet18 as well.

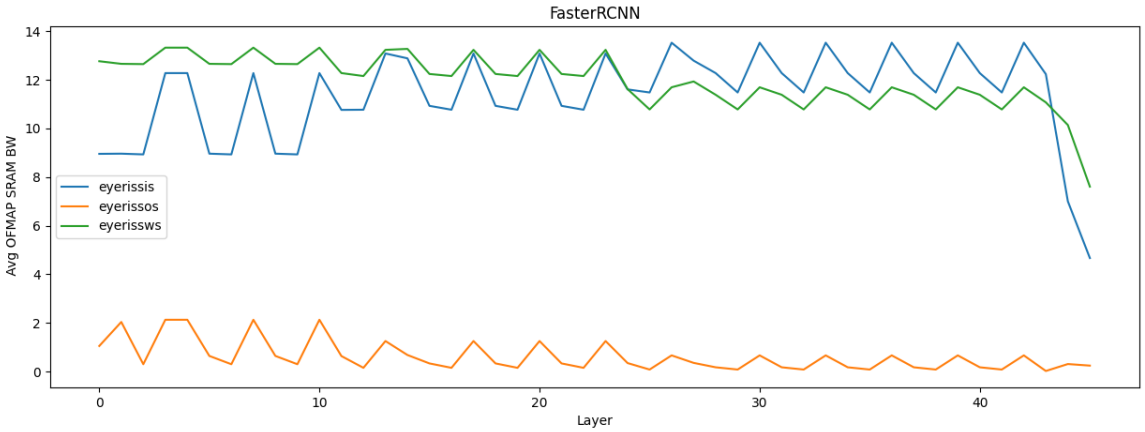


Figure 133: fasterRCNN_ofmap_sram_bw

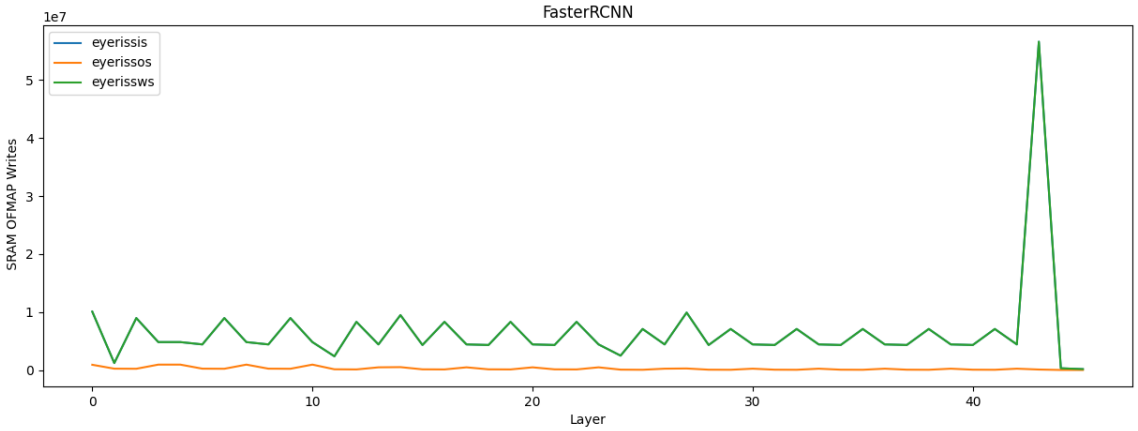


Figure 134: fasterRCNN_SRAM_OFMAP_WRITES

A similar order is observed in fasterRCNN as well.

Changing buffer sizes

Three different DNNs were used: mobilenet, resnet18, and FasterRCNN.

8 different memory sizes were used for eyeriss configuration where input, filter, and output memory sizes were varied in [54KB, 108KB].

Mapping efficiency and compute utilization were constant for different memory sizes.

Total runtime was also constant for different memory sizes.

Strangely, SRAM bandwidth utilization and reads/writes were also constant for different memory sizes.

However, DRAM bandwidth utilization and reads/writes were not constant for different memory sizes.

DRAM Bandwidth Utilization and Read/Writes

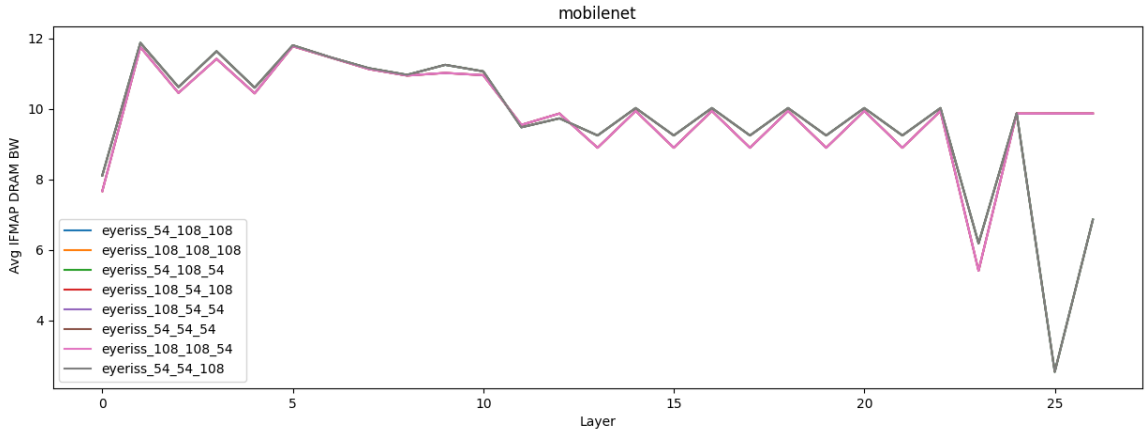


Figure 135: mobilenet_ifmap_dram_bw

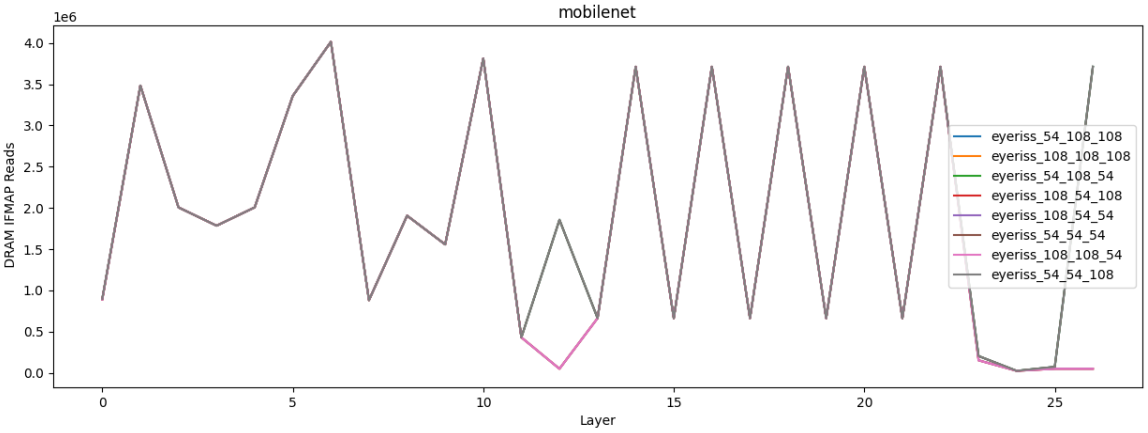


Figure 136: mobilenet_DRAM_IFMAP_READS

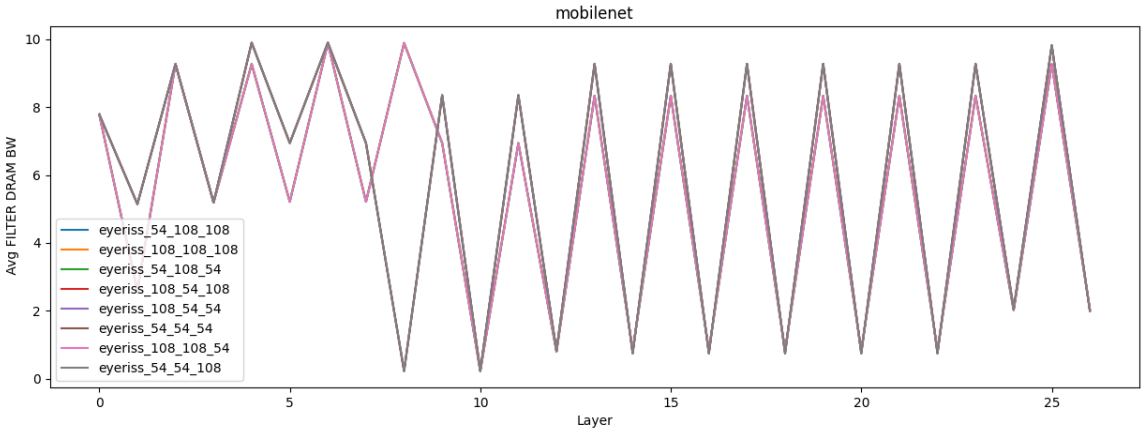


Figure 137: mobilenet_filter_dram_bw

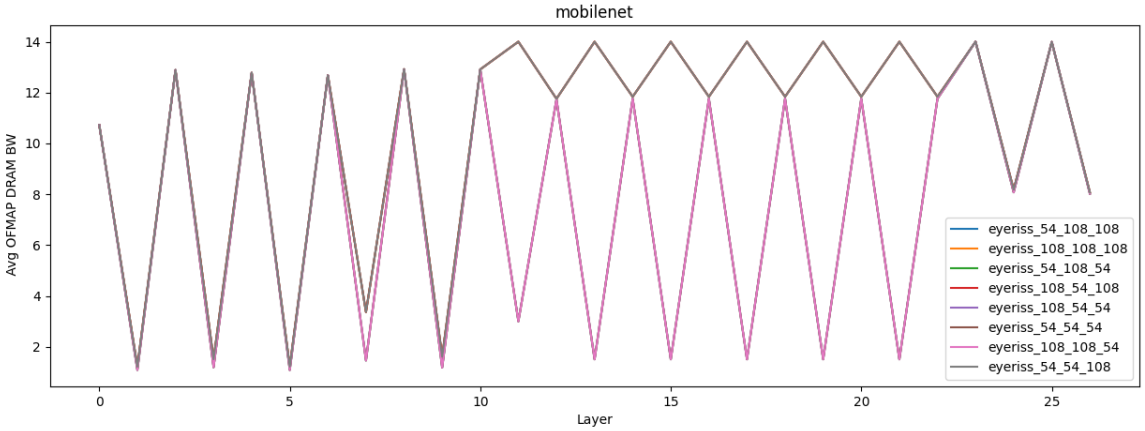


Figure 138: mobilenet_ofmap_dram_bw

Similar orders are observed in resnet18.

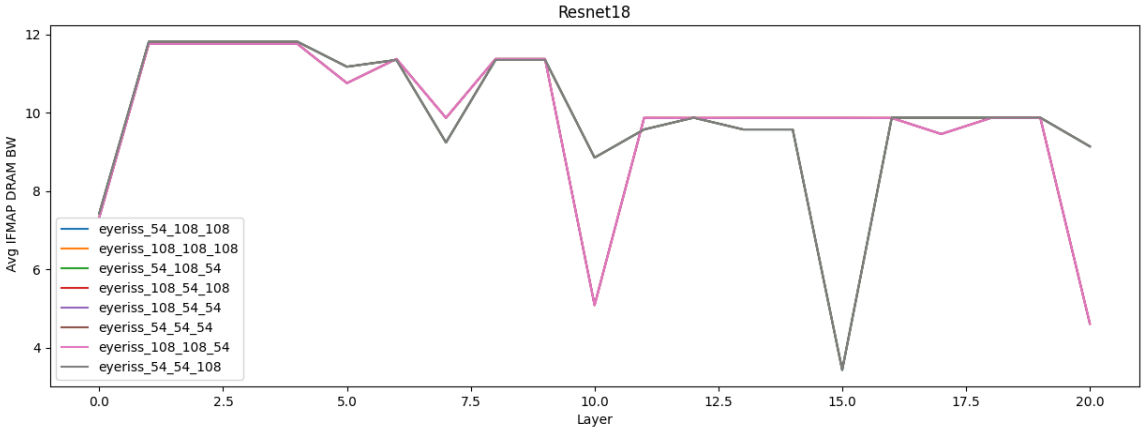


Figure 139: resnet18_ifmap_dram_bw

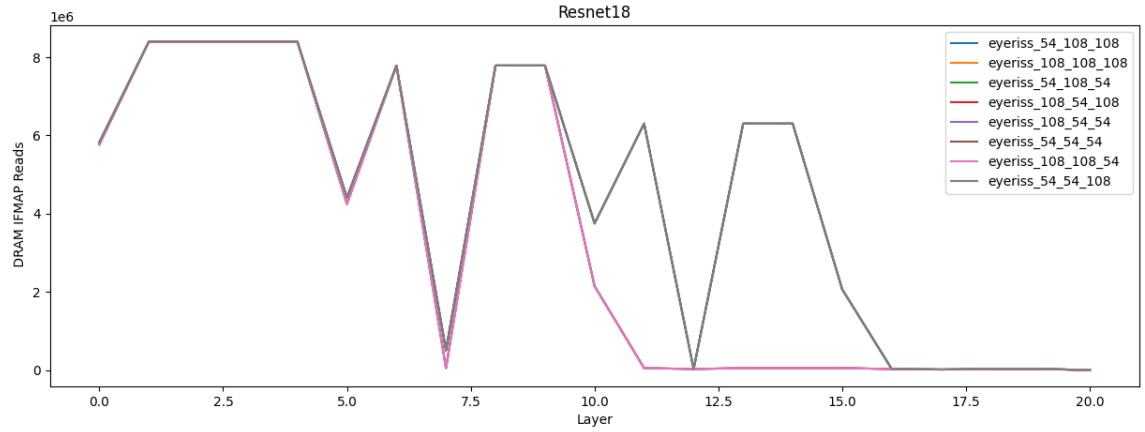


Figure 140: resnet18_DRAM_IFMAP_READS

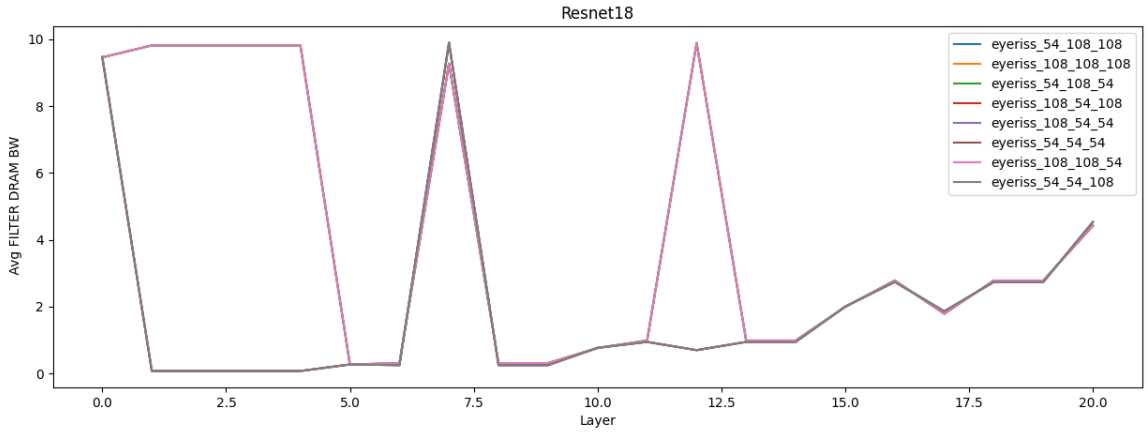


Figure 141: resnet18_filter_dram_bw

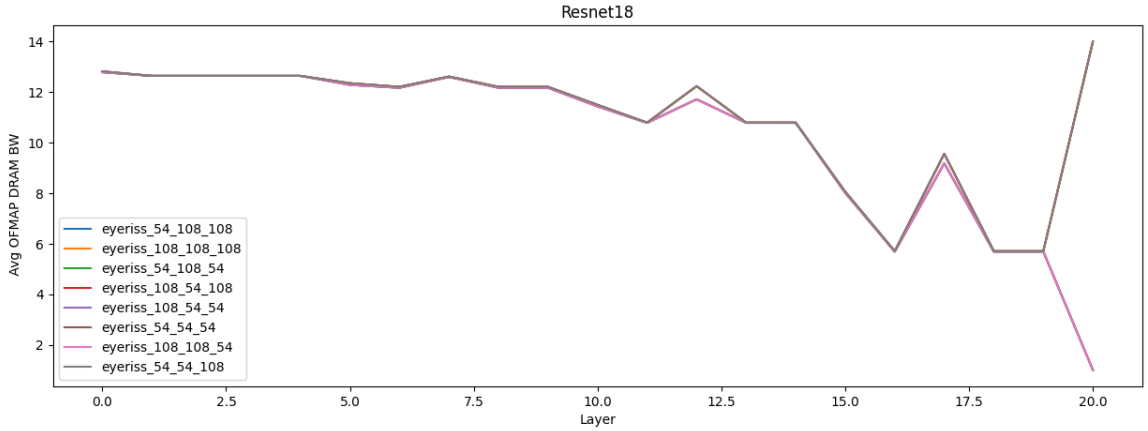


Figure 142: resnet18_ofmap_dram_bw

Similar orders are observed in fasterRCNN.

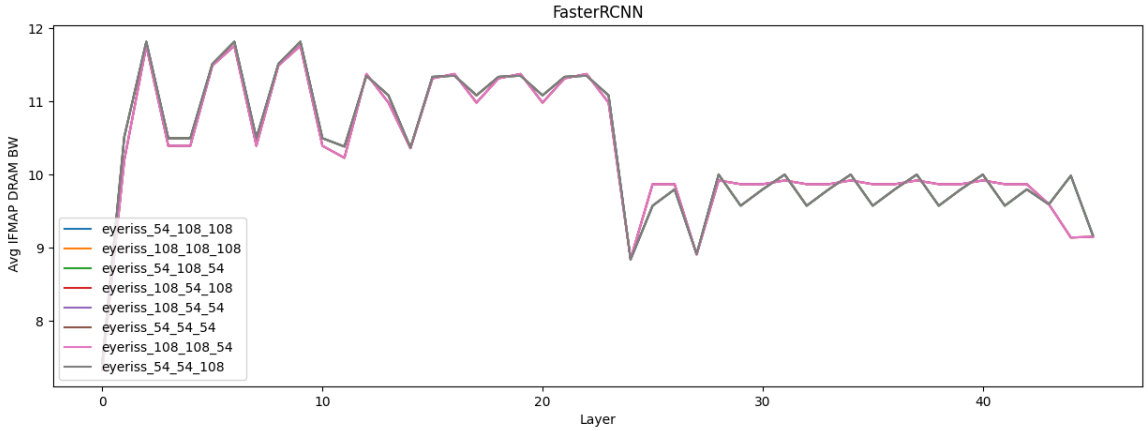


Figure 143: fasterRCNN_ifmap_dram_bw

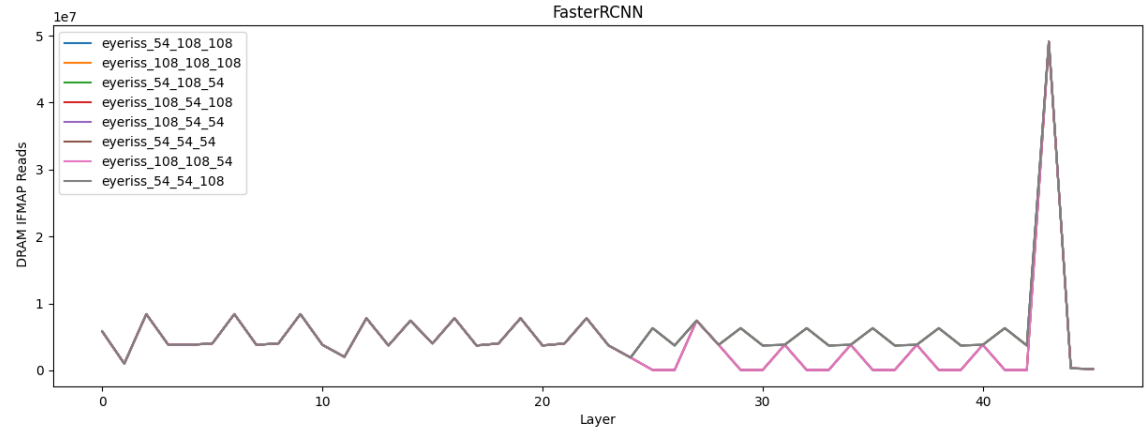


Figure 144: fasterRCNN_DRAM_IFMAP_READS

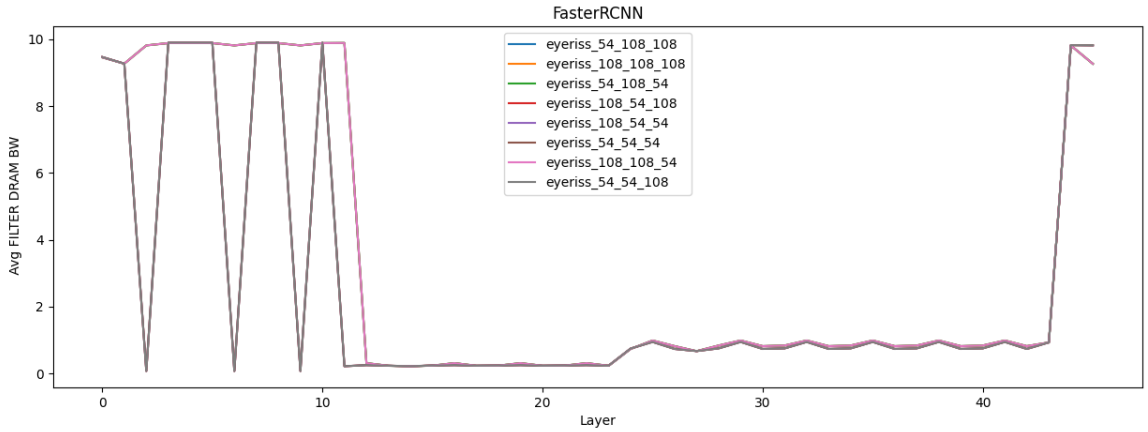


Figure 145: fasterRCNN_filter_dram_bw

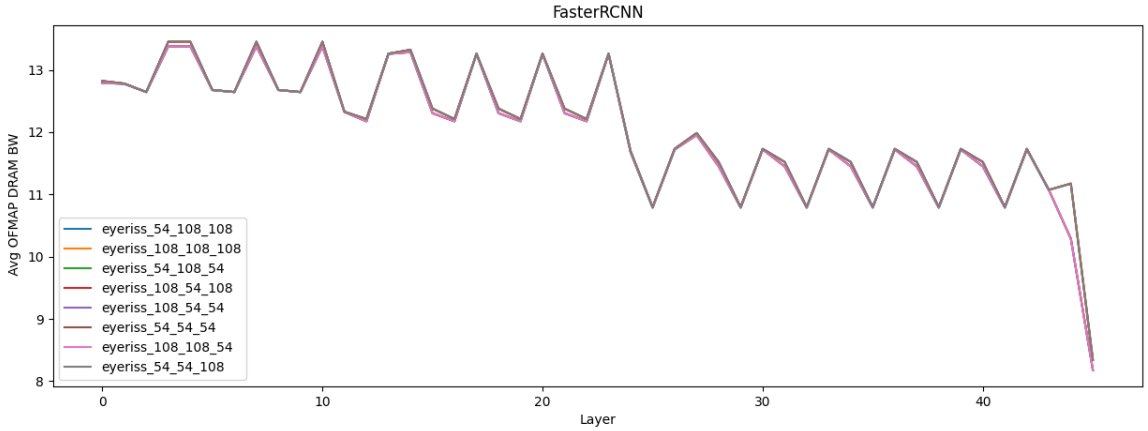


Figure 146: fasterRCNN_ofmap_dram_bw

Changing systolic array sizes

Three different DNNs were used: mobilenet, resnet18, and FasterRCNN.

9 different systolic array sizes were used for eyeriss configuration where systolic array height and width were varied in [12, 24, 48] and [14, 28, 56] respectively.

Mapping efficiency and compute utilization were mostly constant for different systolic array sizes.

Total Runtime

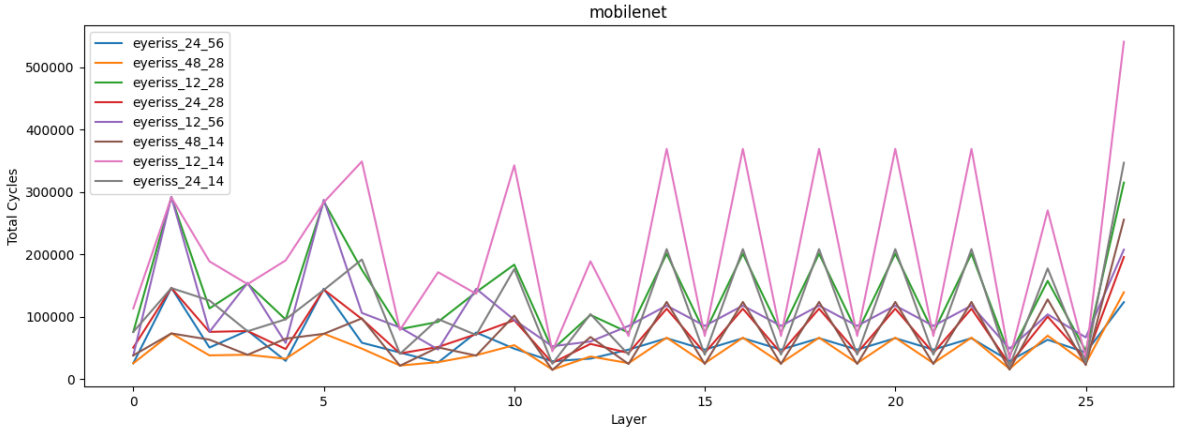


Figure 147: mobilenet_runtime

We can see that the total runtime is highest for the smallest systolic array and lowest for the largest systolic array. This is because the number of PEs is the main factor in determining the total runtime.

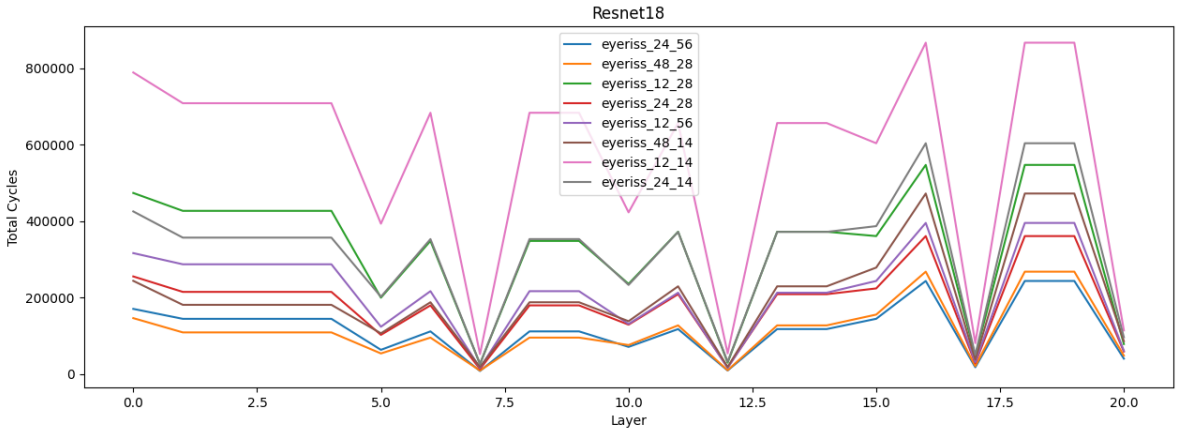


Figure 148: resnet18_runtime

A similar order is observed in resnet18 as well.

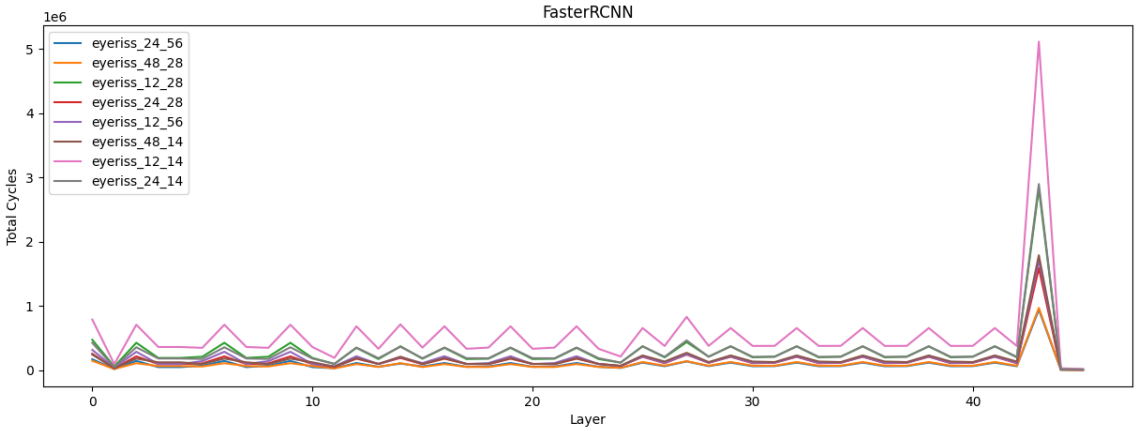


Figure 149: fasterRCNN_runtime

A similar order is observed in fasterRCNN as well.

DRAM Bandwidth Utilization and Read/Writes

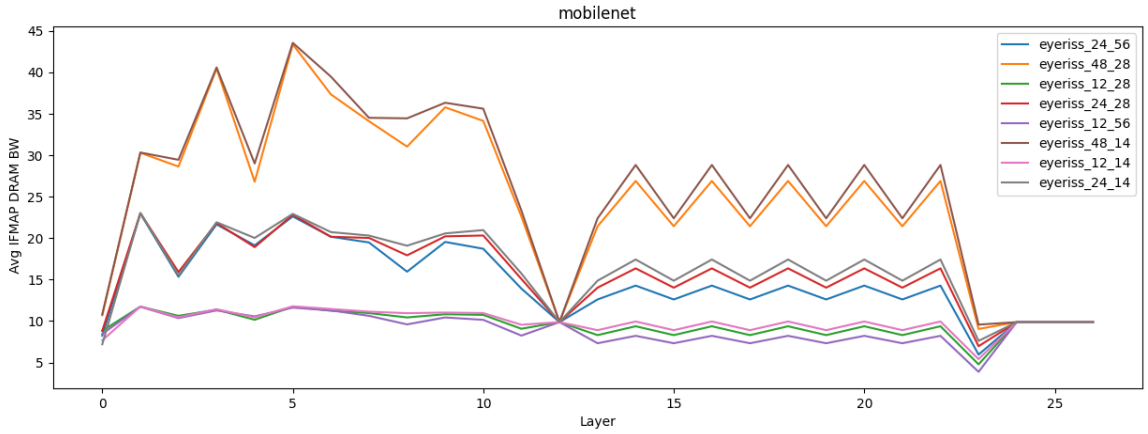


Figure 150: mobilenet_ifmap_dram_bw

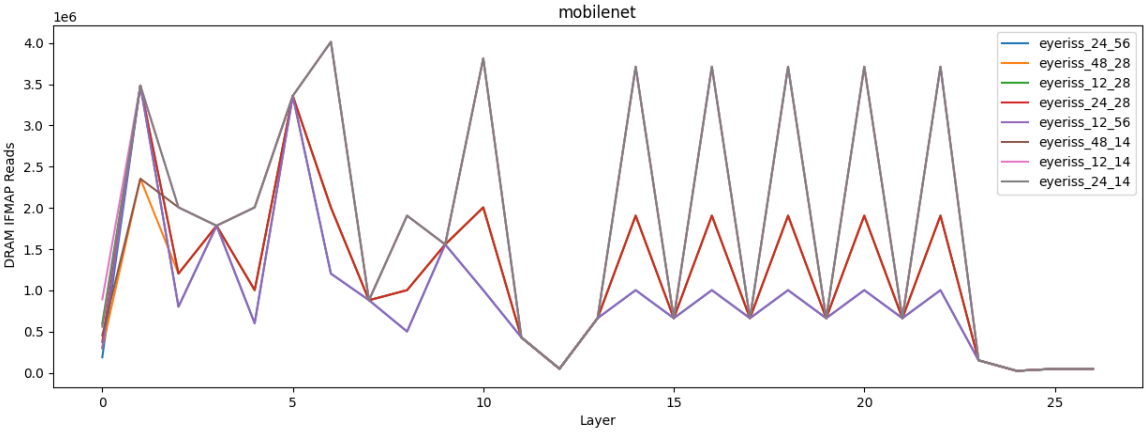


Figure 151: mobilenet_DRAM_IFMAP_READS

Here the bandwidth and reads of the input DRAM memory is dependent on the height of the systolic array. All configurations with the same height have similar bandwidth and reads.

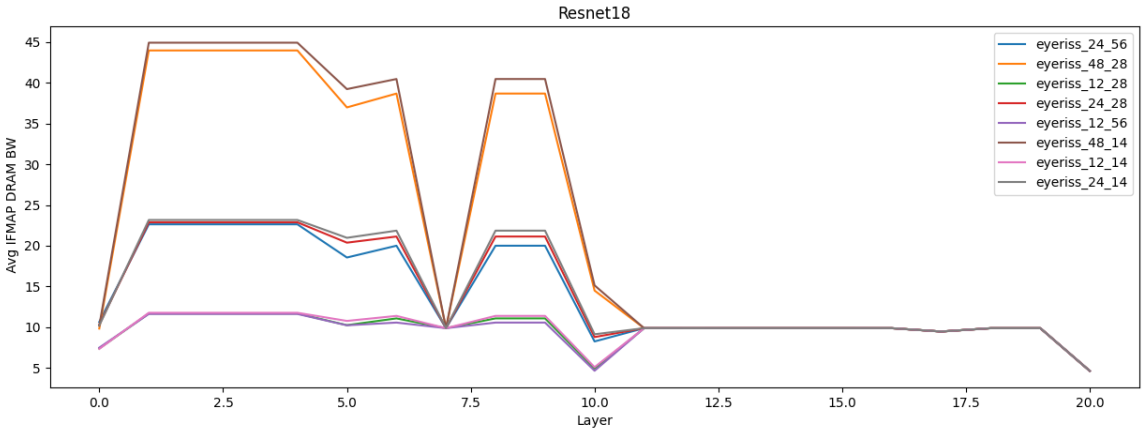


Figure 152: resnet18_ifmap_dram_bw

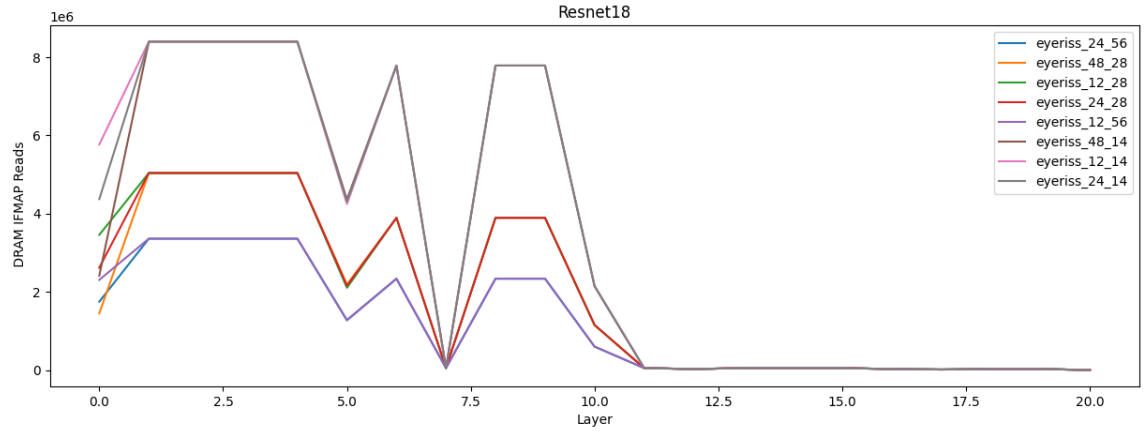


Figure 153: resnet18_DRAM_IFMAP_READS

A similar order is observed in resnet18 as well.

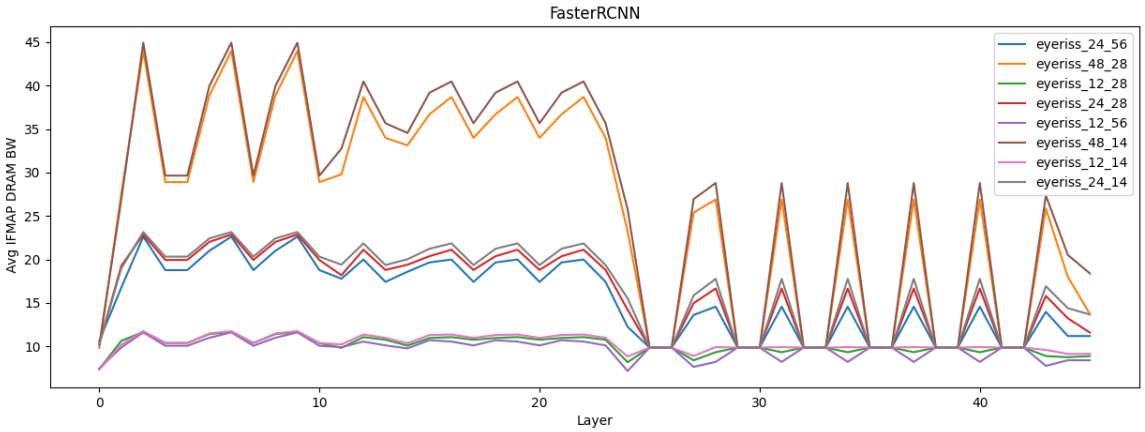


Figure 154: fasterRCNN_ifmap_dram_bw

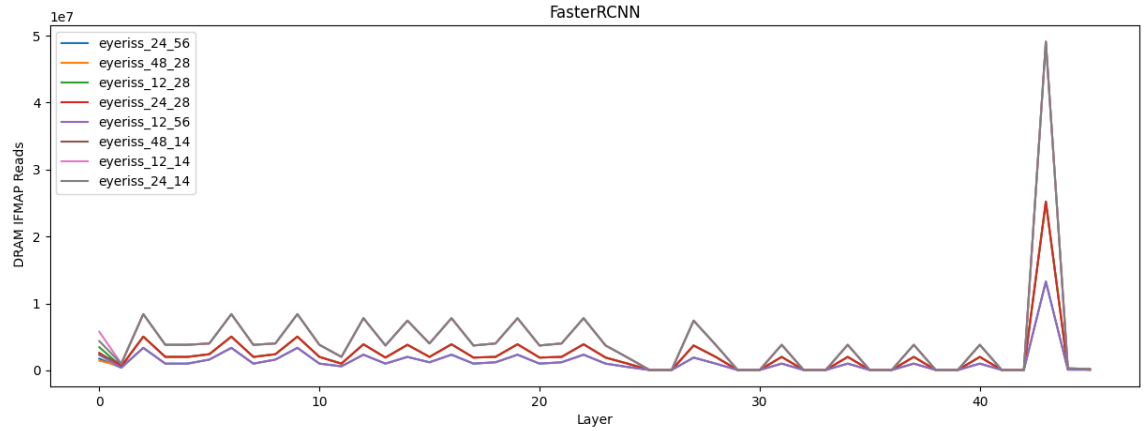


Figure 155: fasterRCNN_DRAM_IFMAP_READS

A similar order is observed in fasterRCNN as well.

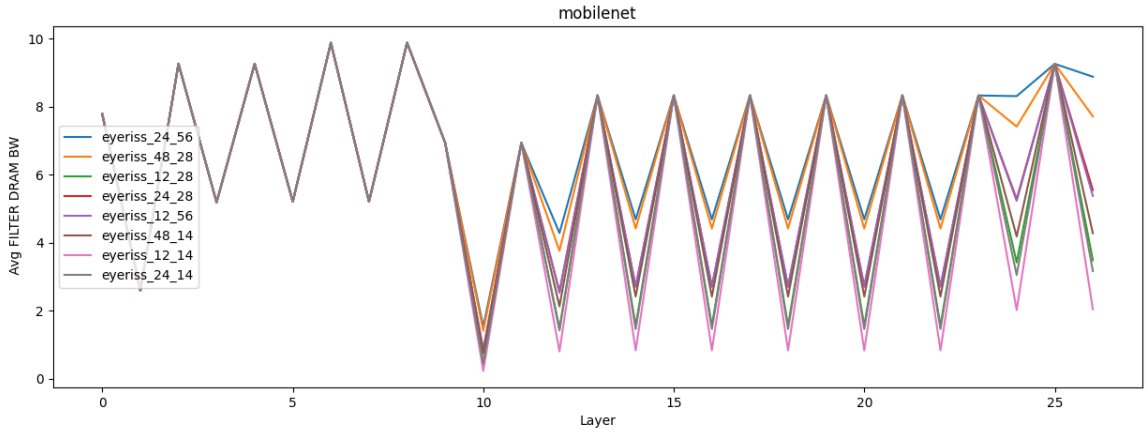


Figure 156: mobilenet_filter_dram_bw

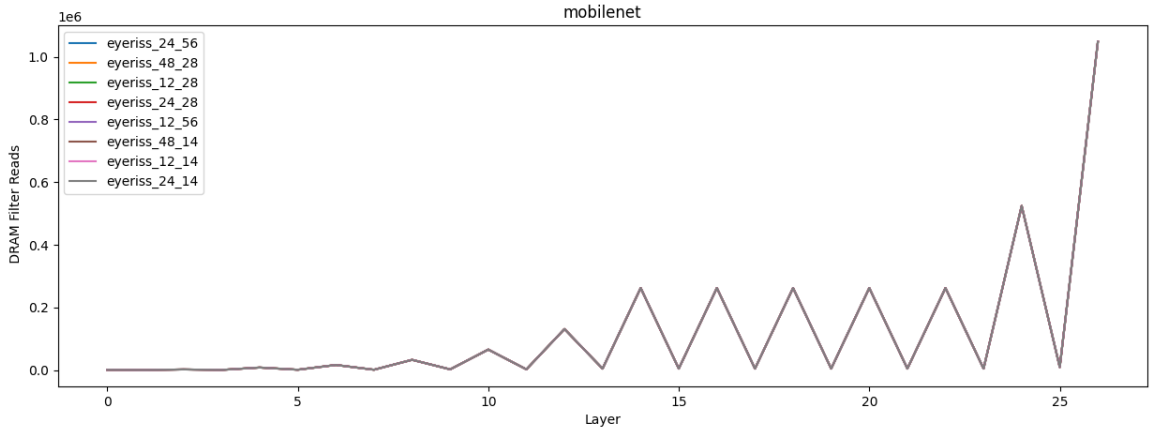


Figure 157: mobilenet_DRAM_Filter_READS

Here the reads of the filter DRAM memory remains constant for different systolic array sizes. However, the bandwidth is dependent on the area of the systolic array. All configurations with the same area have similar bandwidth.

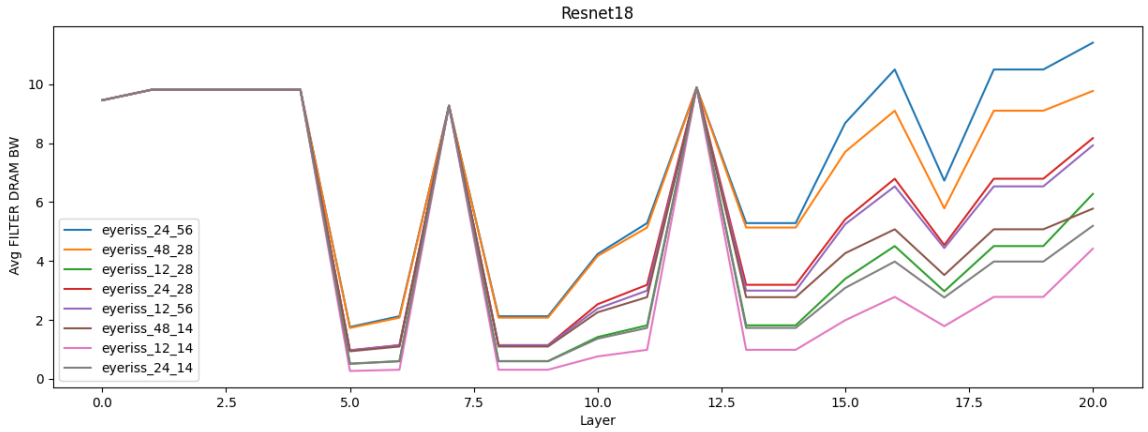


Figure 158: resnet18_filter_dram_bw

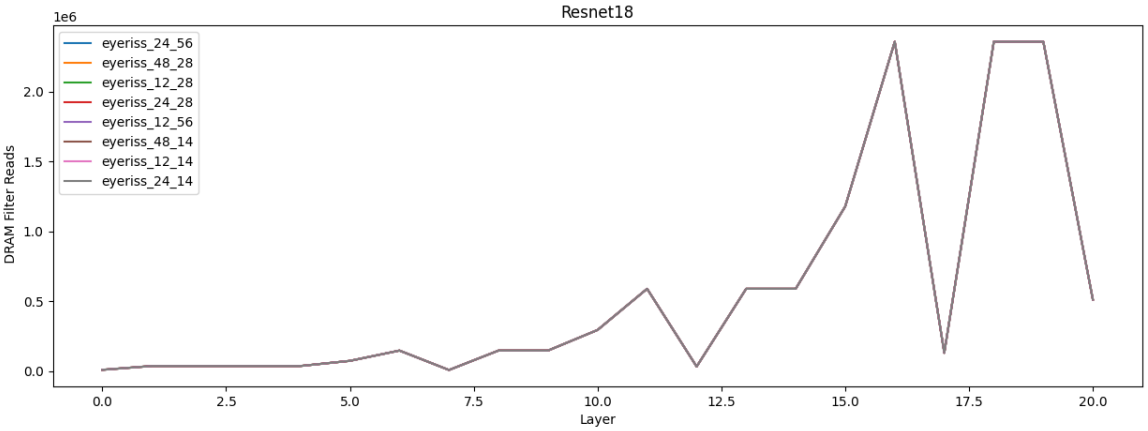


Figure 159: resnet18_DRAM_Filter_READS

A similar order is observed in resnet18 as well.

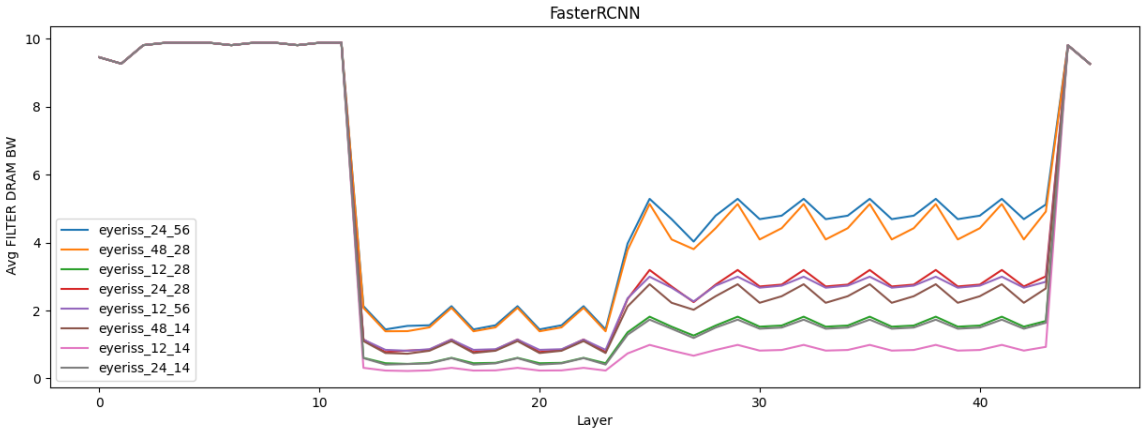


Figure 160: fasterRCNN_filter_dram_bw

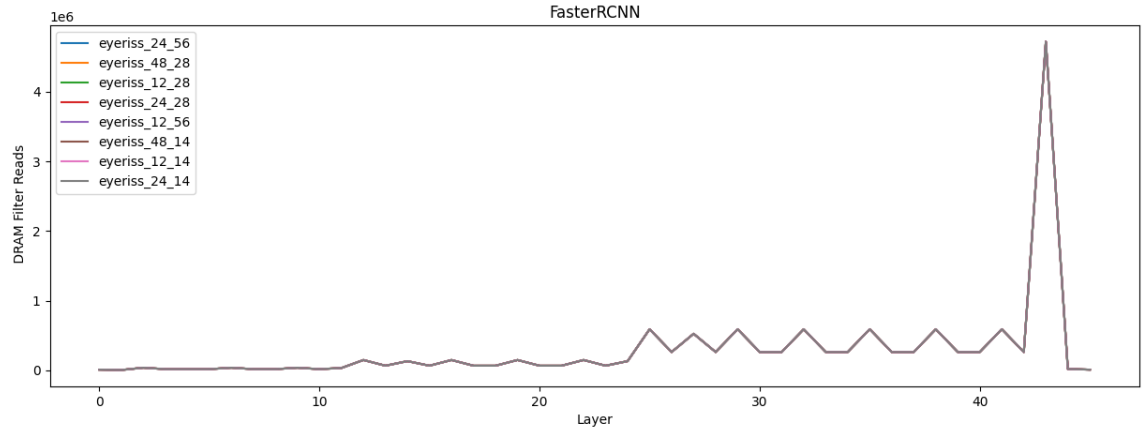


Figure 161: fasterRCNN_DRAM_Filter_READS

A similar order is observed in fasterRCNN as well.

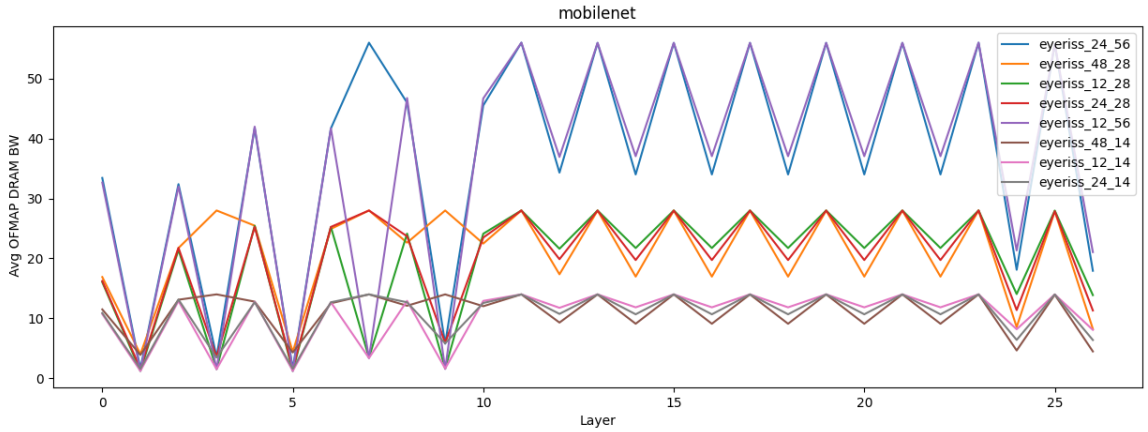


Figure 162: mobilenet_ofmap_dram_bw

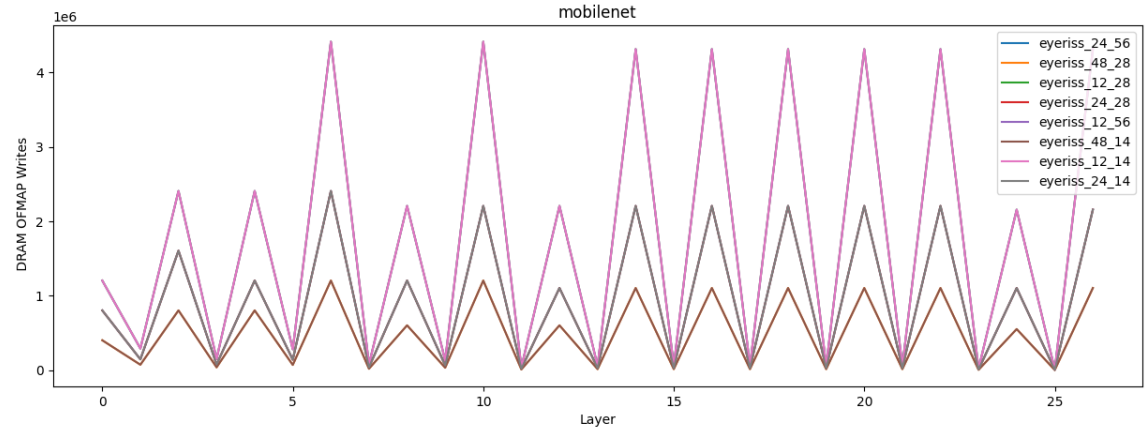


Figure 163: mobilenet_DRAM_OFMAP_WRITES

Here writes of the output DRAM memory depend on the height of the systolic array. All configurations with the same width have similar writes.

Also the bandwidth of the output DRAM buffer is dependent on the width of the systolic array. All configurations with the same width have similar bandwidth.

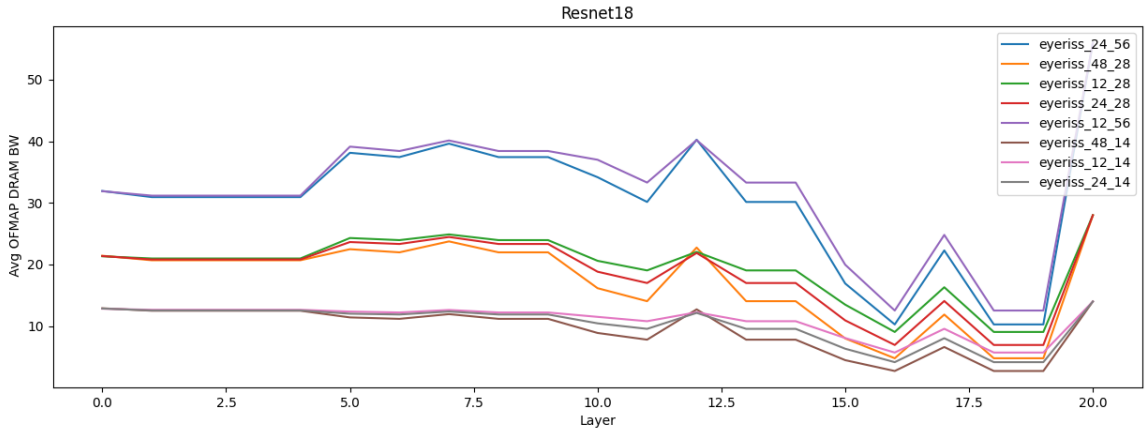


Figure 164: resnet18_ofmap_dram_bw

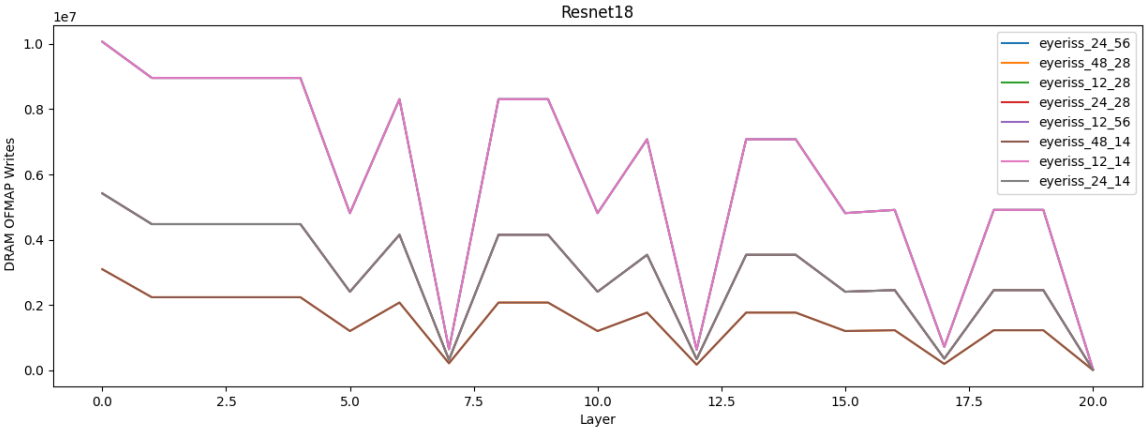


Figure 165: resnet18_DRAM_OFMAP_WRITES

A similar order is observed in resnet18 as well.

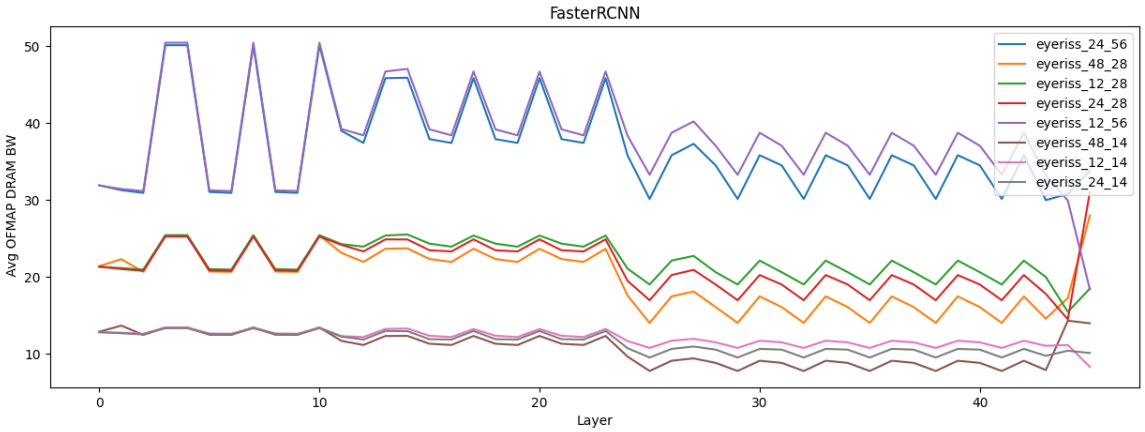


Figure 166: fasterRCNN_ofmap_dram_bw

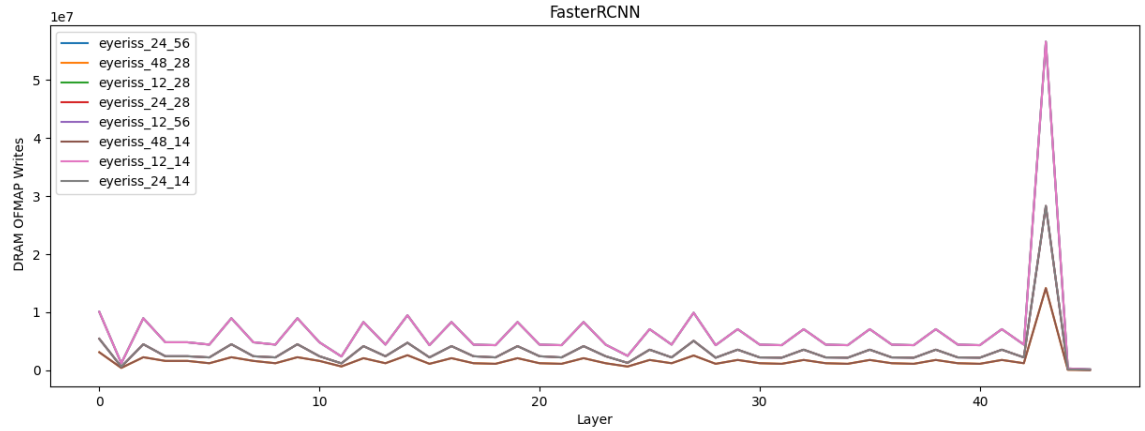


Figure 167: fasterRCNN_DRAM_OFMAP_WRITES

A similar order is observed in fasterRCNN as well.

References

Note that the Scale-Sim paper and the Scale-Sim Github repository are the primary sources of information for this report.

Also the text below are clickable links to the respective sources.

- Scale - Sim Paper
- Scale - Sim Github