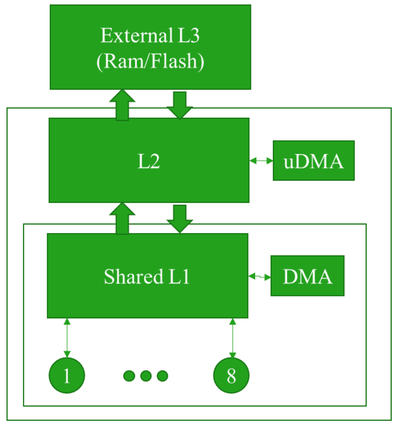
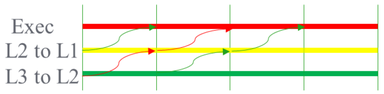
# Make complex neural networks “Tiny”

Eric Flamand, CTO, GreenWaves Technologies

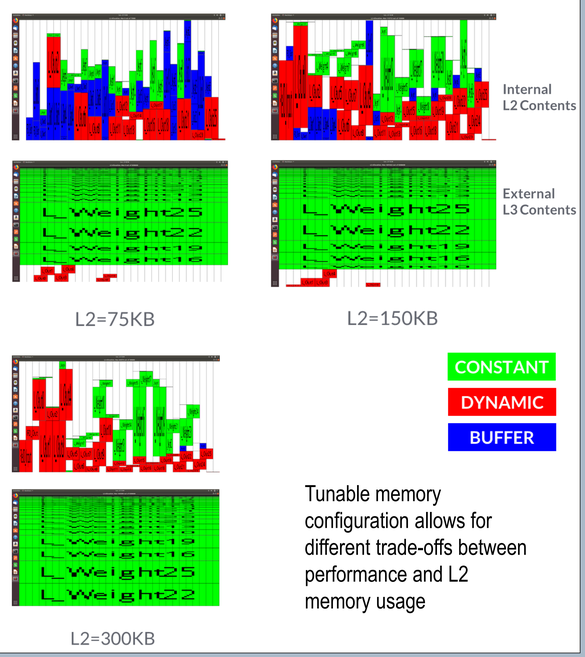
## GAP Autotiler - Explicit Memory Movement

Data caches are highly inefficient for streamed data, but they simplify programming model. With neural network and signal pricing tasks we generally know the location and size of all data moved during an operation at compile time. The GAP AutoTiler is a C library that generates C code for all data movement using DMAs, optimizing dataflow and keeping GAP’s multiple cluster cores busy with data reducing the energy cost of inference using significant neural networks.




## Memory configuration options




Tunable memory configuration allows for different trade-offs between performance and L2 memory usage

## AutoTiler CNN Model

AutoTiler Options

```
void vwwModel(unsigned int L1Memory, unsigned int L2Memory,
unsigned int L3Memory, unsigned int L3Flash)
{
     SetSymbolDynamics();

    SetUsedFilesNames(0, 2, "CNN_BasicKernels.h", "vww.h");
    SetGeneratedFilesNames("vwwKernels.c", "vwwKernels.h");
    AT_SetGraphCtrl(AT_GRAPH_MONITOR_CYCLES, AT_OPT_ON);

    SetMemoryDeviceInfos(4,
        AT_MEM_L1, L1Memory, "vww_L1_Memory", 0, 0,
        AT_MEM_L2, L2Memory, "vww_L2_Memory", 0, 0,
        AT_MEM_L3_HRAM, L3Memory, "vww_L3_Memory", 0, 0,
        AT_MEM_L3_HFLASH, L3Flash, "vww_L3_Flash",
"vww_L3_Flash_Const.dat", 0
    );
```

Kernel generation

```
    LoadCNNLibrary();

    // generator for CONV_2D_0_1_fusion
    CNN_GenControl_T gen_ctrl_S1_Conv2d_8x3x3x3_Relu6;
    CNN_InitGenCtrl(&gen_ctrl_S1_Conv2d_8x3x3x3_Relu6);
    CNN_SetGenCtrl(&gen_ctrl_S1_Conv2d_8x3x3x3_Relu6,
"ENABLEIM2COL", (void *)1);
CNN_ConvolutionPoolReLU("S1_Conv2d_8x3x3x3_Relu6",
&gen_ctrl_S1_Conv2d_8x3x3x3_Relu6, 1, 1, 1, 1, 1, 1, 1, 1, 3, 8,
238, 208,
        KOP_CONV_DP, 3, 3, 1, 1, 2, 2, 1,
        KOP_NONE, 0, 0, 0, 0, 0, 0, 0, 0, KOP_RELUN);

// And so on …

    // generator for FULLY_CONNECTED_0_83
    CNN_LinearReLU("S65_Linear_2x640x1x1", 0, 1, 1, 2, 2, 1, 1, 1,
1, 640, 2,
        KOP_LINEAR, KOP_NONE);
```

Graph declaration

```
#define GRAPH
#ifdef GRAPH
    CreateGraph("vwwCNN",
        /* Arguments either passed or globals */
        CArgs(110,
```

Globals

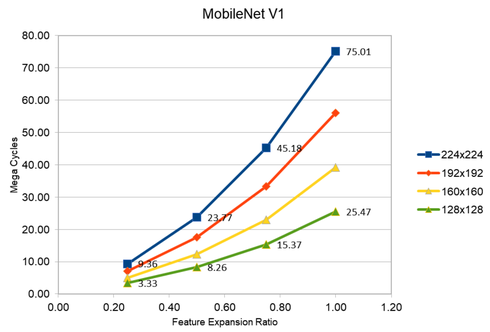
```
            TCArgInfo("signed char *__restrict__", "Input_1",
ARG_SCOPE_ARG_ALLOC, ARG_DIR_IN, AT_MEM_L2, AT_MEM_L2, 0),
            TCArgInfo("signed char *__restrict__", "Step1Weights",
ARG_SCOPE_GLOBAL, ARG_DIR_CONSTIN, AT_MEM_L3_HFLASH, AT_MEM_UNDEF,
ConstInfo("BUILD_MODEL_8BIT/tensors/Step1Weights.tensor", 1, 1, 8,
7)),
            TCArgInfo("signed char *__restrict__", "Step1Biases",
ARG_SCOPE_GLOBAL, ARG_DIR_CONSTIN, AT_MEM_L3_HFLASH, AT_MEM_UNDEF,
ConstInfo("BUILD_MODEL_8BIT/tensors/Step1Biases.tensor", 1, 1, 8,
5)),
// And so on ….
),
```

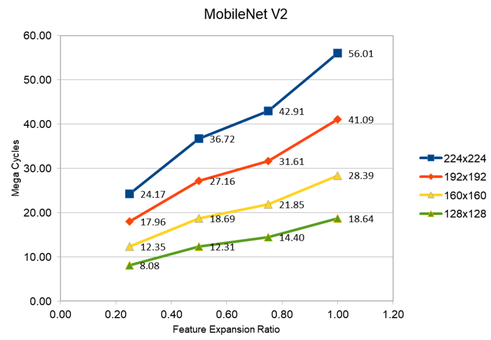
Locals

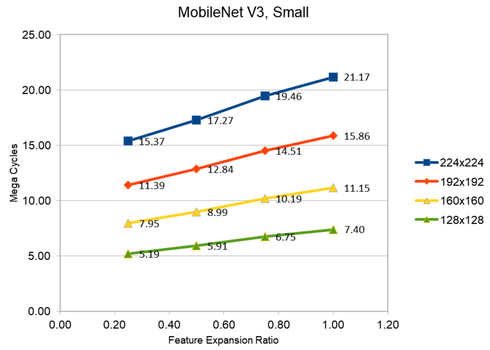
```
        /* Locals, allocated dynamically */
        CArgs(65,
            TCArgInfo("signed char *__restrict__", "OutputStep1",
ARG_SCOPE_LOCAL, ARG_DIR_INOUT, AT_MEM_UNDEF, AT_MEM_UNDEF, 0),
// And so on …
TCArgInfo("short int *__restrict__", "OutputStep65",
ARG_SCOPE_LOCAL, ARG_DIR_INOUT, AT_MEM_UNDEF, AT_MEM_UNDEF, 0)
        )
    );

    // Node S1_Conv2d_8x3x3x3_Relu6 inq 7 weightsq 7 outq 5 biasesq
5
    AddNode("S1_Conv2d_8x3x3x3_Relu6", Bindings(6, GNodeArg(GNA_IN,
"Input_1", 0), GNodeArg(GNA_IN, "Step1Weights", 0),
GNodeArg(GNA_IN, "Step1Biases", 0), GNodeArg(GNA_OUT,
"OutputStep1", 0), Imm(9), Imm(9)));
// And so on …
// Node FULLY_CONNECTED_0_83 inq 5 weightsq 7 outq 12
    AddNode("S65_Linear_2x640x1x1", Bindings(6, GNodeArg(GNA_IN,
"OutputStep64", 0), GNodeArg(GNA_IN, "Step65Weights", 0),
GNodeArg(GNA_IN, "Step65Biases", 0), GNodeArg(GNA_OUT,
"OutputStep65", 0), Imm(0), Imm(0)));
    // Node SOFTMAX_0_84 inq 12 outq 15
AddCallToNode("S2_Conv2d_12x8x1x1_Relu6", CALL_PROLOG,
"gv_vcd_configure", Bindings(2, Imm(1), Imm(0)));
CloseGraph();
#endif
}
```

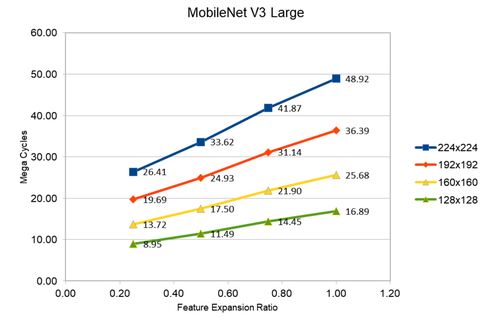
Node bindings

## MobileNet on GAP9



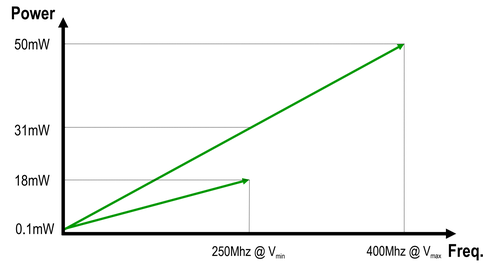

## Neural Networks on GAP9

Performance measurements of several significant neural networks running on GAP9. Comparison with Raspberry Pi3 and Intel i7

| Network | MCycles | MOps | **Ops / cycle** | FPS 400Mhz |
|---|---|---|---|---|
| MobileNetV1 | 73.96 | 571.12 | 7.72 | 5.41 |
| MobileNetV2 | 54.85 | 293.54 | 5.35 | 7.29 |
| MobileNetV3Large | 48.81 | 280.76 | 5.75 | 8.19 |
| MobileNetV3Small | 21.13 | 124.42 | 5.89 | 18.93 |
| InceptionV1 (GoogleNet) | 556.14 | 3268.66 | 5.88 | 0.72 |
| SqueezeNetV1 | 187.23 | 839.29 | 4.48 | 2.14 |
| TinyDarknet | 92.74 | 493.53 | 5.32 | 4.31 |
| PlainDarknet | 112.04 | 500.29 | 4.47 | 3.57 |
| Darknet19 Byte | 554.44 | 2980.43 | 5.38 | 0.72 |
| ResNet18 Byte | 338.7 | 1741.12 | 5.14 | 1.18 |

| Network | Input RGB | FPS Other | Which other |
|---|---|---|---|
| MobileNetV1 | 224x224 | 2.6 | RP3, 1.2 Ghz/4 Cores |
| MobileNetV2 | 224x224 | | |
| MobileNetV3Large | 224x224 | | |
| MobileNetV3Small | 224x224 | | |
| InceptionV1 (GoogleNet) | 224x224 | 1.3 | RP3, 1.2 Ghz/4 Cores |
| SqueezeNetV1 | 224x224 | 4.8 | RP3, 1.2 Ghz/4 Cores |
| TinyDarknet | 224x224 | | |
| PlainDarknet | 256x256 | 7.14 | i7 4GHz/1 Core |
| Darknet19 Byte | 256x256 | 1.5 | i7 4GHz/1 Core |
| ResNet18 Byte | 256x256 | 1.75 | i7 4GHz/1 Core |

GAP9 power consumption at dynamically adjustable frequency and core voltage points




## Summary

**Significant, useful neural networks can be run efficiently on an ultra low power budget.**

**TinyML can do big things!**