**Documentation - TVM MetaWareNN Backend**

**Purpose:**

The purpose of this document is to give an overview of MetaWareNN Backend implementation and to explain workflow from the TVM framework.

**Overview:**

Machine Learning models are being loaded using the ML frameworks. The execution of the models is happening in the host environment (x86 / ARM platform). In order to enable graph execution in other platforms/devices, these ML frameworks added specific implementations to convert ML models to graph Intermediate Representation (IR) and then use the IR to execute the graph in expected devices(DSP, etc).

In TVM, many backends like TensorRT, Vitis AI, Ethos-N, Core ML, etc have unique implementations to run the executable graph. TVM has a backed specific implementation to check for the supported nodes in the graph and generate a TVM Relay compiled file from the ONNX / Caffe / Tensorflow / PyTorch / MXNet framework models. TVM will load the common runtime file into the backend specific implementation and the backend will convert into its own format to run in its device. Similarly, to execute the graph in Synopsys hardwares, we have added MetaWareNN Backend in the TVM framework to generate MetaWareNN Executable Graph.

**TVM - MetaWareNN Repository:**

* Forked [TVM](https://github.com/apache/tvm.git) repository from commit id [f64ddff4977cbb3edc8391a5ee4802e4870846c2](https://github.com/SowmyaDhanapal/tvm/commit/f64ddff4977cbb3edc8391a5ee4802e4870846c2) created a [metawarenn\_dev](https://github.com/SowmyaDhanapal/tvm/tree/metawarenn_dev) branch to incorporate MetaWareNN Backend related code changes.
* Added [MetaWareNN.cmake](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/cmake/modules/contrib/MetaWareNN.cmake) & Include MetaWareNN cmake in the [common CMakeLists.txt](https://github.com/SowmyaDhanapal/tvm/blob/20b9af5d901e32b399aae9d766a7a44117a4210b/CMakeLists.txt#L397) & [config.cmake](https://github.com/SowmyaDhanapal/tvm/blob/20b9af5d901e32b399aae9d766a7a44117a4210b/cmake/config.cmake#L205) file to compile backend files in TVM
* Added Python & Necessary C files in the TVM Framework to handle the MetaWareNN Backend - [Initial Commit](https://github.com/SowmyaDhanapal/tvm/commit/f235d66ddb9ad6b41a8e47ee4f16ad227e37b20b)
* Added code changes to generate high level MetaWareNN Graph, apply graph transformations(passes) & generate low level MetaWareNN Graph
* **In Default flow**, Converted the low level MetaWareNN Graph to the MetaWareNN Executable Graph and serialized it to a binary file
* **In Invoke NNAC flow**, Generated a MetaWareNN Graph proto from the low level MetaWareNN Graph and serialized it to a binary file. EVConvert python module which is integrated in MetaWareNN library will generate a Caffe prototxt & Caffemodel from MetaWareNN binary proto file, and finally evgencnn executable will generate a EV binary by taking the Caffe files

**Trigger Inference with MetaWareNN Backend:**

* [Register](https://github.com/SowmyaDhanapal/tvm/blob/20b9af5d901e32b399aae9d766a7a44117a4210b/src/relay/backend/contrib/metawarenn/codegen.cc#L97) the MetaWareNN backend codegen to the TVM registration API
* Created [partition for MetaWareNN](https://github.com/SowmyaDhanapal/tvm/blob/20b9af5d901e32b399aae9d766a7a44117a4210b/metawarenn_inference/mwnn_inference.py#L15) backend for the input model and build & run the same in the [inference script](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/metawarenn_inference/mwnn_inference.py)

**Build subgraph from TVM IR Module and its execution:**

[python/tvm/relay/op/contrib/metawarenn.py](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/python/tvm/relay/op/contrib/metawarenn.py) - Registers the operators supported by the MetaWareNN backend and Partition & annotate the subgraphs based on the supported operators

[src/relay/backend/contrib/metawarenn/codegen.cc](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/src/relay/backend/contrib/metawarenn/codegen.cc) - The external compiler

/ codegen tool takes a Relay expression/module(TVM) and compiles it into a runtime module(Backend) by serializing the expression to a json representation. No backend specific files are required here as we are handling the same during runtime

[src/runtime/contrib/metawarenn/metawarenn\_json\_runtime.cc](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/src/runtime/contrib/metawarenn/metawarenn_json_runtime.cc) - Build the backend specific MetaWareNN Graph from the JSON Representation. Apply the optimization passes & generate the MetaWareNN Executable network from the low level MWNNGraph. Fill the backend input buffers from TVM, Execute the network in backend and Fills the TVM output buffers from backend

[metawarenn\_lib/metawarenn\_graph.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_graph.h) - Contains MetaWareNN graph format(MWNNGraph) which stores the graph related information in MWNNNode’s, MWNNTensor’s and in some basic data types

[metawarenn\_lib/metawarenn\_node.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_node.h) - Basic class for all ops in MWNNGraph, contains the node information like name, type, inputs, outputs and & MWNNAttribute’s

[metawarenn\_attribute.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_attribute.h) - Contains the attribute information of the MWNNNode like name, data type and its corresponding value

[metawarenn\_value\_info.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_value_info.h) - Contains name, data type and dimensions of graph inputs/outputs information to handle locally

[metawarenn\_tensor.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_tensor.h) - Contains name, data type, dimensions and tensor values for all data initializers. Maintain index and offset for constant initializers to create a MetaWareNN executable graph

[metawarenn\_lib/op](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/op) directory - Operator type specific classes for Conv, DepthwiseConv, Relu,etc which have inputs and outputs of corresponding node (edge info) and its attributes(Padding, Strides,..)

[metawarenn\_lib/optimizer/metawarenn\_optimizer.h](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/metawarenn_optimizer.h) - Base class for available MetaWareNN passes and implementation for following passes has been added

* [fuse\_relu.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/fuse_relu.cc) - Fuses Relu with Conv layer and sets Activation attribute in Conv & removed the Rely layer
* [fuse\_batchnorm.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/fuse_batchnorm.cc) - Fuses the BatchNorm layer weights into the Conv weights & biases and removed the BatchNorm layer
* [calculate\_offset.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/calculate_offset.cc) - Calculates the total bytes needed by each initializer to store the information in MetaWareNN Executable Graph
* [convert\_layout.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/convert_layout.cc) - Weights and inputs from CHW to HWC and vice versa are being handled to have a common layout
* [remove\_reshape.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/remove_reshape.cc) - Removes reshape node and adjust the input & output data as removing the edge & node
* [remove\_transpose.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/remove_transpose.cc) - Removes transpose node and adjust the input & output data as removing the edge & node

[metawarenn\_lib/optimizer/pass\_manager.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/optimizer/pass_manager.cc) - Pass manager registers above MetaWareNN passes and has a passes list which will trigger the optimizations from each pass & updates the MWNNGraph to get an optimized low level graph

[metawarenn\_lib/metawarenn\_utils.cc](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/metawarenn_utils.cc) - In Current implementation, convert\_to\_mwnn\_format() loops each [MWNN graph nodes](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/262fe11e9b92611172904165df5a7686aa295d87/metawarenn_graph.h#L162)(computational nodes like Conv, Relu) and converts the inputs, weights, bias of nodes to MLI format which also handles memory allocation of input, output and initializers. Maintains a tensor map for each computational node. Invokes the MLI kernels for each computational node and fills the output in the tensor map to get the inputs for the upcoming nodes.

We have checked the possibility of writing the input of a subgraph to shared memory and read the output of a subgraph from shared memory and its linkage to the consecutive subgraphs which runs in other backends too by invoking VDK from GLOW. This section of code is planned to push after finalising the Synopsys device inference API’s.

[metawarenn\_lib/mwnnconvert](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/mwnnconvert) directory - Holds the python and protobuf related files to create and read a serialized MWNNGraph proto and generates a Caffemodel, Caffe prototxt & EV binary file

* [evconvert](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/mwnnconvert/evconvert) directory - Contains [mwnn2ev](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/mwnnconvert/evconvert/mwnn2ev) python module which parses the MWNNGraph from serialized graph proto and creates Caffe prototxt & Caffemodel from the input graph
* [input/pkl](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/mwnnconvert/input/pkl) directory - Pickle file to pass the intermediate input buffer for the second subgraph rather than a jpeg image
* [mwnn\_protobuf](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/mwnnconvert/mwnn_protobuf) directory - Contains protobuf file and its wrapper files in cpp & python language
* [mwnn\_convert.sh](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/mwnnconvert/mwnn_convert.sh) - Shell script to initiate the nnac python script & to invoke evgencnn executable from the cpp code.
* [nnac.py](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/metawarenn_dev/mwnnconvert/nnac.py) - Python script to invoke mwnn2ev module

[metawarenn\_lib/kernels](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/metawarenn_dev/kernels) - Contains the MLI reference kernels for Conv, DepthwiseConv, FusedRelu, AveragePool implementation files and its dependency files from [MLI repository](https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_mli/tree/ww42) - Tag ww42 (shared by Synopsys)

[mwnn\_inference.py](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/metawarenn_inference/mwnn_inference.py) - Inference script which partitions the graph in the MetaWareNN Backend and triggers the build & run call for the input MobileNetv2 model

**Clone the Repository:**

1. *git clone --recursive https://github.com/SowmyaDhanapal/tvm.git tvm*
2. *cd tvm*
3. *git checkout metawarenn\_dev*
4. *git submodule sync*
5. *git submodule update*
6. *sudo apt-get update*
7. *sudo apt-get install -y python3 python3-dev python3-setuptools gcc libtinfo-dev zlib1g-dev build-essential cmake libedit-dev libxml2-dev*

**Modifications to make in the repository:**

*Refer this* [*README.md*](https://github.com/SowmyaDhanapal/tvm/blob/metawarenn_dev/metawarenn_inference/README.md) *file to get the detailed steps to compile and run the inference for sample models*

**Flow chart:**

Workflow diagram of the MetaWareNN backend has been depicted in this flow diagram link

[Flow chart - TVM MWNN Backend](https://viewer.diagrams.net/?page-id=6jtj7vH_vbAAvlTVk0RX&highlight=0000ff&edit=_blank&layers=1&nav=1&page-id=6jtj7vH_vbAAvlTVk0RX#G11bZE5KFItbOyN-k8hOc2PsEUJGo4c6Rj)