**Documentation - GLOW MetaWareNN Backend**

**Purpose:**

The purpose of this document is to give an overview of MetaWareNN Backend implementation and to explain workflow from the GLOW 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 GLOW, many backends like Habana, NNPI, OpenCL, etc have unique implementations for each of the mentioned backends to convert the ONNX/Caffe model to Backend specific IR which is in the form of a compiled function. Similarly, to execute the graph in Synopsys hardwares, we have added MetaWareNN Backend in GLOW framework to generate MetaWareNN IR.

GLOW loads ONNX and Caffe models and converts them into GLOW Function and being passed onto the backends.

**GLOW - MetaWareNN Repository:**

* Forked [GLOW](https://github.com/pytorch/glow.git) repository from commit id [916b8914e0585c220b6186a241db0845c8eff5a9](https://github.com/SowmyaDhanapal/glow/commit/916b8914e0585c220b6186a241db0845c8eff5a9) created a [metawarenn\_dev](https://github.com/SowmyaDhanapal/glow/tree/metawarenn_dev) branch to incorporate MetaWareNN Backend related code changes.
* Added backend specific BUILD files along with MetaWareNN Backend source files - [Initial Commit](https://github.com/SowmyaDhanapal/glow/commit/76aeb5c7a1b0437f7a093997fcf37a160fa2a12c) (New files added in recent code)
* Added code changes to generate high level MetaWareNN Graph, apply graph transformations(passes) & generate low level MetaWareNN Graph and created InferenceEngine & ExecutionContext using [metawarenn\_lib](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/onnx_conversion)
* **FLAGS to Control Code Flow**
  + By default, *INFERENCE\_ENGINE* is disabled which will create output ONNXProto directly from MWNNGraph and dumps the ONNX model to *op\_onnx\_models*
  + Enable *INFERENCE\_ENGINE* flag to convert MWNNGraph to ExecutableGraph and cache it to EXEC\_DUMPS path and then create Inference Engine & Execution Context and finally dumps the output ONNX model in *op\_onnx\_models*
  + Enable *INVOKE\_NNAC* flag to generate a MetaWareNN Graph proto from the low level MetaWareNN Graph and serialize 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 using the Caffe files
    - [Note] : INVOKE\_NNAC flag is Outdated and not tested after MWNNGraph update to ONNX format

**Trigger MetaWareNN Backend:**

* [Register](https://github.com/pytorch/glow/blob/76aeb5c7a1b0437f7a093997fcf37a160fa2a12c/lib/Backends/MetaWareNN/MetaWareNNFactory.cpp#L5) the MetaWareNN backend to the GLOW backend factory
* Run inference with image classifier executable with ***backend set as MetaWareNN*** to invoke the backend specific implementations and graph format conversion to low level IR.

**Build subgraph from GLOW Function and its execution:**

[MetawareNN.cpp](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/lib/Backends/MetaWareNN/MetaWareNN.cpp) - Base file for MetaWareNN Backend which parses the supported ops json file to load ops supported by MetaWareNN Backend and creates set of subgraphs

[MetaWareNNDeviceManager.cpp](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/lib/Backends/MetaWareNN/MetaWareNNDeviceManager.cpp) - One of the top level files which adds the list of subgraphs(GLOW Functions) to be [stored](https://github.com/SowmyaDhanapal/glow/blob/188595979bbdd1ed084294a653e2cd112f6dd376/lib/Backends/MetaWareNN/MetaWareNNDeviceManager.cpp#L45) and initiates subgraph inference through [runFunction](https://github.com/SowmyaDhanapal/glow/blob/188595979bbdd1ed084294a653e2cd112f6dd376/lib/Backends/MetaWareNN/MetaWareNNDeviceManager.cpp#L18) function

[MetaWareNNFactory.cpp](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/lib/Backends/MetaWareNN/MetaWareNNFactory.cpp) - Register MetaWareNN Backend in the Glow Backend Factory

[MetaWareNNFunction.cpp](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/lib/Backends/MetaWareNN/MetaWareNNFunction.cpp) - Creates a MetaWareNN Graph and has INFERENCE\_ENGINE macro to create ExecutableGraph or cache it from the EXEC\_DUMPS path. Creates Inference Engine and Execution Context and uses it for final execution. If INFERENCE\_ENGINE is diabled, then directly creates ONNXProto from MWNNGraph and dumps it to onnx model

If INVOKE\_ NNAC is enabled, serialize it to a binary file using MetaWareNN GraphProto and invokes nnac python script which takes the MetaWareNN serialized binary and invokes [EVConvert](https://github.com/SowmyaDhanapal/metawarenn_lib/tree/onnx_conversion/mwnnconvert/evconvert) python module to generate Caffemodel and Caffe Prototxt which is used by evgencnn executable to generate EV binary. [Note] INVOKE\_NNAC is outdated

[metawarenn\_inference/test\_regression\_onnx.py](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/mwnn_inference/test_regression_onnx.py) - Python Regression script to run inference for all the ONNX models in [models.txt](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/mwnn_inference/models.txt) using image-classifier object

[metawarenn\_inference/test\_regression\_quantized\_tflite.py](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/mwnn_inference/test_regression_quantized_tflite.py) - Python Regression script to run inference for all the quantized TFLite models in [tflite\_quantized\_models.txt](https://github.com/SowmyaDhanapal/glow/blob/metawarenn_dev/mwnn_inference/tflite_quantized_models.txt) using image-classifier object

**MetaWareNN Library Documentation:**

Refer to [this document](https://github.com/SowmyaDhanapal/metawarenn_lib/blob/onnx_conversion/docs/Documentation-MetaWareNNLibrary-Structures.docx) for the details on MetaWareNN Library class structures, Optimizers, Serializations of Executable Graph, Inference Engine & Execution Context generation.

**Clone the Repository:**

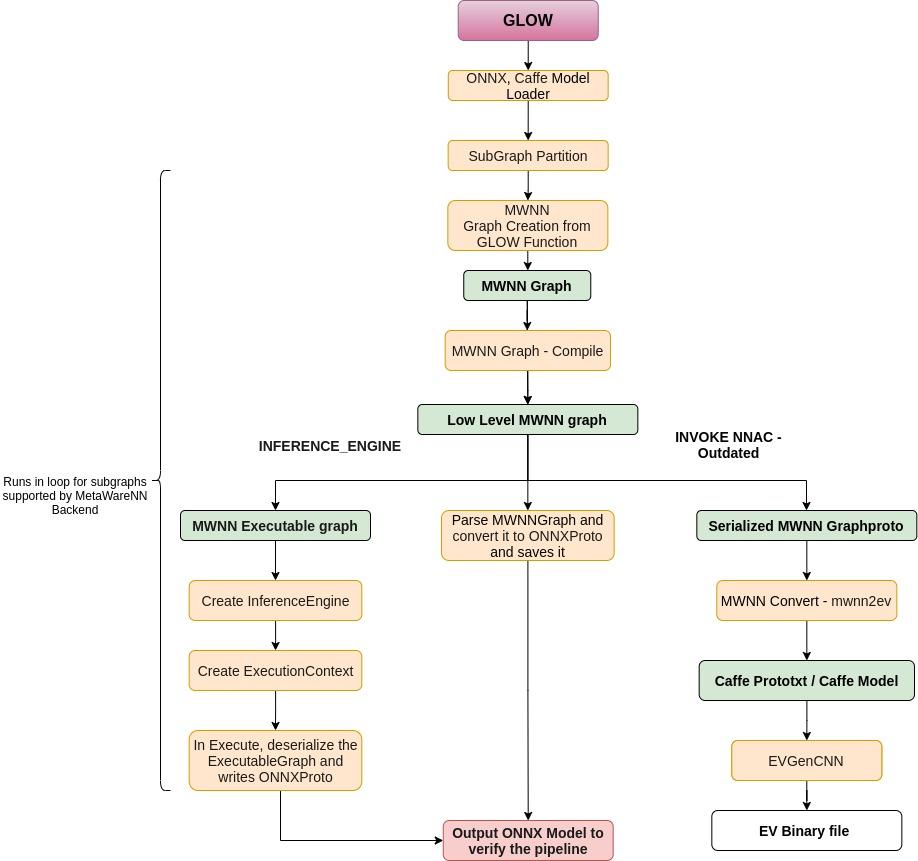
* *git clone --recursive* [*https://github.com/SowmyaDhanapal/glow.git*](https://github.com/SowmyaDhanapal/glow.git)
* *cd glow*
* *git checkout metawarenn\_dev*
* *git submodule update --init --recursive*

**Modifications to make in the repository:**

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

**Flow chart:**

The workflow of the MetaWareNN Backend in Glow has been depicted in the following diagram,

****