**Documentation - TensorFlow Lite MetaWareNN Delegate**

**Purpose:**

The purpose of this document is to give an overview of MetaWareNN Delegate implementation and to explain workflow from the TensorFlow Lite 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 TFLite, delegates like Hexagon, NNAPI, etc have its specific implementation to build graphs from the TFLite model corresponding to each of the delegates. We are trying to have MetaWareNN IR common between ONNXRuntime, TFLite & GLOW frameworks.

**TFLite - MetaWareNN Repository:**

* Cloned TensorFlow repository (Tag [v2.3.1](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/tree/v2.3.1/)) and created a [metawarenn\_dev](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/tree/metawarenn_dev) branch to incorporate MetaWareNN delegate related code changes
* Added BUILD files for MetaWareNN delegate files and linked it to tensorflow lite shared object file to infer the ML model with MetaWareNN delegate
* 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 MetaWareNN Delegate:**

* [Create](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/08fa5dd4e2706366c828514857a1e4a0f473a157/tensorflow/lite/delegates/MetaWareNN/inference/inference_metawarenn.cpp#L25) the MetaWareNN delegate from the top level [inference script](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/inference/inference_metawarenn.cpp)
* [Modifies](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/08fa5dd4e2706366c828514857a1e4a0f473a157/tensorflow/lite/delegates/MetaWareNN/inference/inference_metawarenn.cpp#L26) the input graph based on the supported nodes in MetaWareNN delegate to NodeSubsets(subgraphs) and update it with MetaWareNN delegate information using TFLite interpreter
* Execute the subgraphs using [invoke call](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/08fa5dd4e2706366c828514857a1e4a0f473a157/tensorflow/lite/delegates/MetaWareNN/inference/inference_metawarenn.cpp#L49) through interpreter

**Build subgraph from TFLite model and its execution:**

[MetaWareNN\_delegate.cc](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/MetaWareNN_delegate.cc) - TFLiteMetaWareNNDelegate creation and triggers partition of graph to NodeSubsets based on the supported nodes list and MetaWareNN kernel registration

[MetaWareNN\_delegate\_kernel.cc](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/MetaWareNN_delegate_kernel.cc) - Implementation of Init, Prepare, Invoke for each NodeSubset in the partitioned graph which is triggered from common framework code flow

[builders](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/tree/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/builders) directory - Contains Build file & model builder source file to generate the multiple levels of MetaWareNN Graph structure from TFLite NodeSubset. Also has metawarenn library directory which contains all files related to MetaWareNN Graph handling

[model\_builder.cc](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/builders/model_builder.cc) - Implementation for creation of MetaWareNN High Level Graph Representation from TFLite SubGraph Nodes using structures like MWNNGraph, MWNNNode, MWNNTensor, etc. Creates Low Level MetaWareNN graph structure by registering and invoking basic metawarenn graph optimization passes.

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/metawarenn_dev/mwnnconvert/evconvert) python module to generate Caffemodel and Caffe Prototxt which is used by evgencnn executable to generate EV binary.

[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)

[inference/inference\_metawarenn.cpp](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/inference/inference_metawarenn.cpp) - Inference script which creates the MetaWareNN Delegate and invokes the MobileNetv2 model inference.

**Clone the Repository:**

* *git clone --recursive* [*https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow.git*](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow.git)
* *cd synopsys-tensorflow*
* *git checkout metawarenn\_dev*
* *Refer this* [*README.md*](https://github.com/foss-for-synopsys-dwc-arc-processors/synopsys-tensorflow/blob/metawarenn_dev/tensorflow/lite/delegates/MetaWareNN/README.md) *file to get the detailed steps to compile and run the inference for sample models*

**Flow Chart:**

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

[Flow chart - TFLite MWNN Delegate](https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1&title=Flow%20chart%20-%20ONNX%20runtime%20EP%20and%20TFLite%20Delegates#R7V1tl5s2Fv41c87uB3MQIF4%2BZiaZJO20ne2kmXa%2FzBEg27QYsQJmxv31K4HACPAYO2A7psnJiZGEAOne5z73XiGu9JvV60eK4uVPxMfhlab6r1f6%2BytNA0DV2H%2B8ZF2UQFsULGjgi0abgofgbywKVVGaBT5OpIYpIWEaxHKhR6IIe6lUhiglL3KzOQnlq8ZogVsFDx4K26WPgZ8uRammqpuKTzhYLMtLO5qoWaGytShIlsgnL7Ui%2FcOVfkMJSYtfq9cbHPLRKwfm%2Bha9M53HH6K7Z%2B%2FX%2F9wt4A%2Fx51nR2e0%2Bp1TPQHGUDtu1mMwkXZcDhn02fuKQ0HRJFiRC4YdN6TUlWeRj3qvKjjZt7giJWSFghX%2FiNF0LYUBZSljRMl2FopY9BV3%2Fzs9XYHn4h%2BguP3j%2FKh2txVHPYRDDlZCMeviNdroQR0QX%2BK3%2BYNGOD0xNpsQgf8RkhdlNsgYUhygNnmXBQ0J%2BF1W7zRSxH2KW8sPfEkx%2Fcf%2FkWqCpIXJxWJx6pZkhu79rl%2F1YpPk4FAVzEqXS5Jn%2Fy0hZMUvy4X%2FHGgAzft1Ulr18ub37%2FOVD2Re7l6I7%2BRKsuHZZbZc46bWREeJXE62N4HApeFkGKX6IUT5HLwx9ZCGZB2F4Q0JC83N1bPqqz6VvQZEfMAGo1fnQ0h3W7XWSUvIXrtU4jglsuLfoPGOa4tc3J1vUapajiJMEQAIo4OKlBjeGKFvWkKYs20NC2GFNSPZQc2PCag57qrl1JmqOuJxQPJc0epmm3GC%2B451qt4sgXWau4pFVrrZJMpsTOkvWEYmTdTLzX7wZot4spsTDSUJowppVtSmOWNE8ZKaMKXdImH7fqrbqQR8bnuM5lm9YqmV6Ppg7uoEMT4XAZobftXX%2BTNLpIVNh%2FnicNjy5WRD6mCqexzTvztTrkNMDpIwukGpAnVep9qahDhD%2F2wlw%2Be2ZaMWxJXKTWEa2LYCHmmVJjKLORxC3w2%2BfLtx%2FaeYVJwZswlXp57%2F573wUej1zScLuCGLDWbux4j56gzEcDYznc2yyWe4AXN9yXHV%2FJd4HcFUFyIhbEbQ64podiKsfEXHNCSOu1RNxnTNB3CPr989Mv1nlQ%2BYmOH0DnmR1%2F26Nw4I7lk9BNCfCMkBwastwj2gapAGJ9jIGOwHXukjANR3FtGXAtc4QcO0JA67TE3BLwzk1xP3p8eefWWUe4toTcHv72I0L7wdIp4R317VsaKvQm7vAcpBp%2BhD6nma4DoaeDR0bOZ4%2Bt%2Bdb4J1xVbxAKebX%2BQmn6JEJNR%2FtW%2BEO8HImLuiFC3v0FAauXJCbB2EZDGNYnL%2BhGL0N89tEoeUDbLEQRxbkOeXTrn68%2B%2BWR%2FXebRV7j8fZ0EZzLtFiqZK50vcNc2R3mCh7RXJXx8Unaq9IO7TZY%2BpkYrMNjr30M0oARWFAfoGFV2ofY9o0xFdeGCnTUzR9d0mND7dBjzVLKFNWpmCeYchIF9M2igHNJo2w12Xu5nVNhaiROgxUbE8p%2BxyhJnlYoQosqymsPS9hkXFRn7N8NWcVBiA8I27q0o9WeBAmMl9E6KUMyFcOQwdXs8umhoneA6zEzV2DKqSvQN3cF7IsC14rw3HH8Uu%2Fwc54QEeiwaLCmFjc7QM3Hy5WMz5p0R1Gd2h9LUmyomYoGW7pdJU1OxppOnCGR1Hqj5UdSbLunYuvHitj1nDRrymjcd9JK1boQND6EyF06Te5Y8AAcbdgxfMA0QCGfpH24r5oueXaxsJxht%2BXcZ0HEICzavkwWrSuWnBmDdtvSAktTTPPELNqZMG6XcLwbt88l2jgwi66AxN9AQe5iM0xlczYok66tjP7%2BmLShKkCi0pJym5Dp%2Bxky6VIAJ8mktb7xxzKQcSZMWpty0Lj3pGmXETQWmHtDIg5GIrB56RS5lfN%2FiSKvHIFb%2FLz5zWs0%2FLz5pcRrzqi1gQl1dZ2ThJK1Cw0lN2ymZDIt0DP1fkwyrE05pKz1DSlrFxpSvkFMW1jze85909c0R%2BJbjs6iQiy%2F306KBwGD7zngbGhVzr3Uc7ut5xo4ffZIm3KQWesdZB7c7xWn3pMgt9JCcEwgL8yynYYoFHcqzmpIQ3Ub38C5pxy97i8NlxW9Ps0bExi5OjSgpnNmbSHftXXbVjXGTS3oeox4W9pcdcaj1%2FzoSRwqyXIMNv3h60cc3fD7Og2dvsyYsmUoplWj01BGzK6Xi%2BGp%2BfQhweVkiWJeOw%2Fx6zu%2Bz8MpobYx%2F5Zpqjbqmn9D195Da3hw1vuGqPXBAyJbTLUqm2rHHspUH5nuf%2FjK2rpBhPJRmRcrybblv0ah%2Bvr3HBGHxltrSxyjA46cE8NRuWTiGYWZGOO3kqFqkPAJqadIeE5ERfk0UYz4CxdZEkSLK5Fb7RafaYbxcp6Rp5HcjF%2FGi%2BMnLrZxvmCUj7QSu0rOP4CuD8xAbu7vueyVl1P3YCILHGHKHp1Pt3ip5p%2BJrE9kMXVFflC%2FM%2BGwMydU8L663j5zx7Qy89KMdqhig4aUJMMnXrbKoXEXwLoFGt%2B5ZcFYyGpqsoGFsP8yPcdS4N542hM8tRZ4XrpedC7lMQdGKhSGyX6O0tkYmRpw%2BHmS5lgu66BrqZaCeHk0iNNvmAmGHx5nDSkNFov8uf4hBc2ZjCLkiRQedxuGnEfe9YHT2Cb00pS56xznkzrTk2nhpU%2FqMYCwY%2FXRxjuL%2BY6Lh6sm4srI3NLC1G%2FR24A7l4w8RBOY0EMy8KKMSUChwHBg5hcS5FegWUz9jik%2F5CrrbxGjaArKfohsMBac4MIxyGUDWAOjQ36B%2FXiSGqQb%2F%2FwfI7zdoYvX6ZJEDef8KXY1MZm6ow87mff5BesO%2BjeqZOmzc%2Fgok%2FXxJotfNFoV2fvvyBEEjlKPsDXWnHYtOLW00v%2Bre4UA7B9k6%2BkU6mflFE7dIWGsNN2YUBw9B5REBedhU8QIqxvmOkLx%2F7KAFiEuwuNjD2JUr4o9uIu4Z5rFSic7njp2ytlTMLAj016GtI8EFFiX9yEAT%2B2IaDacJYYhuci5WHg6m2A3fmanexFHa%2FyKvSzlMsTPEKy5Dr1dGZXh4ZZ5XYuI%2FfbY6cx2HC8QZ4NzQNwTL1G82mvdUlcu9%2FQp1b4LHY3LWu9ymDFChZ1YMTjJnzVJCeVfVhj%2F3UB9wJWQHXvDO5avWlZbIKvUaUsgbc3VTXNExIGWnEUFatfGRsbpV0%2FqU97EWO%2B7i%2FHZ4Ie7BQoO%2FD7E1%2Fc%2FsrpPvzx82c1N3H0Ufrxtcs%2F0IxGMsyoNv65b50%2B%2BdMI%2BK0cPeaarmtBjqOM7QLcd24QYuQ40XReaCLjAcPw5j1AOGvTW%2BIK2Q8xrsWld%2B9xHGhTUufDYJAK95%2F7TNUVhwpfK2tDizFxEAw%2BF70TFKvD9AqQx0%2F6c4xfYGfN1XLmAwesr%2BJ73xXC5QIi8a1lpIhLhEfXFMGFzj3%2Bna8vprty4Nholn%2FIr9Ebf9YmGeSbG8NvI9J2UpZA93sL3DqI4S%2FPnQDWPO2kR6dy9zjtAYUg84aC32g3Hp43xlhtiwFiz1WlCTUtHY3JmhvwKtLZFSYHa8f4RsGzF1tsI0VzJOup3lw55NX%2BcpdGvQfp77XfNs2dHGxThByWI1MDHMKw6%2FABFtXZFBfKje0wDNrrcGp3HSuty28OzWWmtWw4b3dqmt%2FKKfwBAY61X8YitldftJdyNbbCBajUkf%2BTXrQxtyhaz7xYHpUB%2B5xbzlH6Cb2gexI5qWoZhWthHwAYac%2BkMD2MPMdPhQGDYvrmvn7A5eCrbPP2FaYRD4SlwV3ZIR6G9%2Fa2cB%2F0cPTOU3C%2BROhyxuND9C%2BzmJl6gwuCj7YV7xqHhzC1flPgsKK%2Bbsfnalc4fROKMyUhc5xeVTh8BNuBJTXh1cJL350tfdrcJv6ytMg6BiffZKt44zCVikCwtICP3gltZpvEBxLxIX9jWlHJJTuUlOLrSYbS2JJGO%2BaEbY8qbLhh9N12AZ%2F9lttEh5Fdcj7m1IERE2IqsVPk%2BZjveVkTn8g9rFqhTdFd1Q%2Bb5fVGKk5hEvuhmQ3Ly9eEHfTl4EMC6zE0NoKWoZgOwOhiPphqK1QFX5VvHR4ErEVmspcAa6ZugQYOnk5axGgEl0JXD1I%2BZlCkN2SSNS2kzdhuXy9rX%2BBDj8lvsi%2BWLpfoWS2IT7JHCYlQmYHSQh%2BN9ou2kO9eYitUAeb0LIKYaSGlLmkqzKKmxmZJ2qEWQcdeLMIPI4mUG9SxNqScLG7soAb0r3GLpSrkzaF0uzSP6SvC04ZZRzdluK%2BUMbaW2JNygrTRwymhu174lxzZUZgy211l94u9IRUnpgYR88iojJRAjf2Eji2M2x7nTk7%2ByXM9YqNfI%2BwtzlNlJSjuZY12DRVF%2F8tqFDrJsjkZMZV5qGh2RkK5N0sajpW0nYjtot0sG%2Fqj0HK2CcF2cVvm0DIZ1g4sEDp8xn9BWjdxJ0zRKdcWN8sqI0BUK5eoXMeS8Pv%2BGq6gMGSphOmNP63H713U%2Bl9uZEEFeLaRQqg6YhEWie7V2a3llSlGUMC1ald1HuGrwQqgvX71%2BustUaZHL76w57NxLLweb%2B4vlb6huRt4PkjhEYtSDKAxqF56HBKX1G2pyhduA4XfIJXxRBDLkGIYIhcx5K5lQFNEP5PHE6ZV4N1a0rshFEDG3FUdevr9B8%2B0zmT60BbSBK2f8XqWmG0q5AqHC%2Ba6gQxcyVJ7tt0DDy2cjfnxxbmbzv%2BP%2FPi%2ByKJ1%2FnakdyLBlUL2MhutryiG9x8DKVG2sUa2WKL%2FxbfWuZZnmEFj7dA2yP35fvPO%2FfLz7gin90Qv1mXacAZ2HQfxp5MGFsBFg0Uy91%2BhqQ4hr5%2Bi23%2Fy9IVES%2BJgWLktMCO%2BDIV0Oy%2FleehFJZaLSHUbdwApQWPvbgCb8vIeWe5QTIonmlKsx2l1pvKs3XK28Lx%2FPURbyiwlHS5kYYYJ2a4E10Ds%2BAMyolGI7Q%2FlElJC0zqP5zBSb7esf%2Fg8%3D)