mera Documentation

Release 2.3.2

mera

23/05/2025

CONTENTS

CHAPTER

ONE

MERA ERROR LIST

The following table summarizes common errors encountered during model quantization, conversion, and execution, along with their technical descriptions. Each entry clarifies the root cause and impact of the error to aid in debugging and resolution.

Table 1: Error List

Error Name	Module	Error Description
CHECK qparam.IsPerTensor()	MERA Interpreter	The operation expected per-tensor quantization (where a single scale and zero-point are applied to the entire tensor) but encountered an incompatible quantization scheme (such as per-channel quantization). Per-tensor quantization uses uniform scaling across all tensor values, while per-channel quantization applies separate parameters to each channel, typically seen in depthwise convolutional layers. This mismatch prevents the quantization process from proceeding as configured.
CHECK qparam.IsPerChannel()	MERA Interpreter	The operation expected per-channel quantization (where separate scale and zero-point parameters are applied to each channel of the tensor) but encountered an incompatible quantization scheme (such as per-tensor quantization). Per-channel quantization is typically required for depthwise convolutional layers and certain hardware accelerators that optimize for channel-wise scaling, while per-tensor quantization applies uniform parameters across all channels. This mismatch prevents proper quantization or execution of the model.
CHECK input.shape.size==output.shape.size	MERA Interpreter	The operation detected an inconsistency in tensor dimensions between a node's input and output, where the expected output shape does not match the computed result. This typically occurs when layer operations (such as reshapes, splits, or concatenations) are improperly configured or when the model's architecture enforces incompatible dimensional transformations between connected layers. The mismatch prevents successful execution of the affected node in the computational graph.

continues on next page

Table 1 – continued from previous page

Error Name	Module	Error Description
CHECK Implementation for node <NODE_TYPE>[<NODE_NAME>] is not defined.	MERA Interpreter	The interpreter encountered a node type that is not currently supported for execution, indicating either an incompatible layer operation or an unsupported framework-specific operator. This typically occurs when the model contains custom operations, experimental layers, or framework features that have not been implemented in MERA Interpreter. The operation cannot proceed until the unsupported node is modified or replaced with a compatible alternative.
ERROR Deserialization: <MSG>	MERA Interpreter	The deserialization process encountered an error while attempting to load the file, likely due to version incompatibility or file corruption. This typically occurs when trying to read a quantized model file that was created with an older version of the framework or when the serialized data structure has been modified or damaged. The operation cannot proceed without a valid, compatible model file in the current format.
CHECK fused_activation == ir::canonical::TFLiteFusedActType::NONE	MERA Interpreter	The interpreter encountered an operator with unsupported fused activation, indicating that the model uses combined operations (where linear computations and activations are merged into a single optimized node) that aren't implemented in the current runtime. This limitation occurs when the backend lacks specific handling for these composite operations, requiring either decomposition into separate primitive operations or implementation of the fused operator pattern.
CHECK n.axes.shape.size==1	MERA Interpreter	The operation expected a 1-dimensional axes parameter for the TFLite sum operator, but received an invalid or multi-dimensional input.
TODO	MERA Interpreter	The operation encountered a missing implementation for a required component or feature, marked as a placeholder (TODO) in the codebase. This indicates unimplemented functionality that was expected to be available during execution, typically arising during development of new features or support for untested use cases. The system cannot proceed until the specified component is properly implemented and integrated.
Histogram combination must encompass original histogram's range:	MERA Quantizer	The histogram-based quantization observer encountered invalid input data during range aggregation, likely due to infinite (inf) or Not-a-Number (NaN) values produced by preceding operators. This occurs when the calibration data or model computations generate numerical instabilities that corrupt the statistical analysis required for determining quantization parameters. The calibration process requires finite input ranges to properly calculate scale and zero-point values.

continues on next page

Table 1 – continued from previous page

Error Name	Module	Error Description
CHECK **data_**.size==(**pre_axis_** * **axis_** * **post_axis_**)	MERA Quantizer	The operation encountered an invalid tensor shape during observer processing, where the input dimensions were incompatible with the expected observation requirements. This typically occurs when statistical observers (used for quantization range calibration) receive malformed tensors that violate shape constraints. The shape mismatch prevents proper collection of activation statistics needed for accurate quantization.
CHECK Must have accumulation domain set.	MERA Quantizer	The quantization process expected tensor operations to use int32 as the accumulation domain (for preserving intermediate calculation precision) but instead encountered operations configured for activation domains (typically int8/uint8).
CHECK Unhandled Quantize() constant layout: <LAYOUT>	MERA Quantizer	The quantization process failed because the specific memory layout of the constant tensor is not supported. Constants must follow strict layout requirements (such as contiguous memory, specific stride patterns, or dimension ordering) to be properly quantized, but the encountered tensor violates these constraints. This typically occurs when constants are created with unconventional storage formats or optimized layouts that aren't compatible with the quantization process.
CHECK Unhandled axis for layout <LAYOUT>	MERA Quantizer	The specified axis value is invalid for the tensor's current memory layout, indicating a mismatch between the requested dimensional operation and the actual data organization. This typically occurs when operations (like reductions or broadcasts) reference non-existent dimensions or misinterpret stride patterns, particularly with transposed, sliced, or non-contiguous tensors where logical and physical layouts diverge. The operation cannot proceed until either the axis parameter is corrected or the tensor is reorganized to match the expected layout.
Histogram observer can only use PER_TENSOR mode	MERA Quantizer	The histogram observer encountered an unsupported quantization mode, as it only operates with PER_TENSOR quantization (where a single scale/zero-point is applied to the entire tensor). This limitation occurs because histogram-based range calibration requires uniform statistical analysis across all tensor values, which isn't compatible with PER_CHANNEL or other granular quantization schemes that maintain separate parameters for different tensor segments. The observer cannot proceed until configured for pure per-tensor operation.
No quantized model available. Needs to call QuantizeTransform()	MERA Quantizer	The operation attempted to use an API method designed for quantized models, but was called on a non-quantized input. This occurs when the model has not undergone the necessary quantization transformation process. The system requires explicit quantization via the transform() method to convert the model into the supported quantized representation before this operation can proceed.

continues on next page

Table 1 – continued from previous page

Error Name	Module	Error Description
CHECK Missing quantization transform recipe(s) for nodes [<NODE_LIST>]	MERA Quantizer	The operation found a layer type with no quantization implementation, blocking full model conversion. This occurs when the framework lacks quantization logic for the layer's specific operations. The process requires either implementing support for this layer or replacing it with a quantizable alternative.
MERA Core config file not found: <PATH>	Framework Frontend	The operation failed to locate the required configuration file at the specified path.
Shape contains an undefined dimension	Framework Frontend	The ONNX parser encountered a tensor shape with undefined dimensions, which occurs when the model contains dynamic shapes or placeholder values that weren't resolved during export. This prevents proper tensor allocation and validation during model parsing.
Input type not supported yet	Framework Frontend	The ONNX parser encountered an input tensor type that is not currently supported by MERA, indicating either a custom data type or an unsupported ONNX feature. This prevents the model from being properly ingested and processed.
Symbolic dimension has not been defined for this tensor	Framework Frontend	The ONNX parser encountered a tensor with undefined symbolic dimensions, indicating unresolved dynamic shape variables that must be explicitly specified. To resolve this, provide the missing dimension definitions through the shape_mapping argument when calling from_onnx(), which allows proper shape inference and validation of the computational graph.
Only one ONNX operator set is supported at a time	Framework Frontend	The ONNX parser encountered multiple default operator set versions, which violates the specification requiring exactly one global operator set version declaration. This typically occurs when merging models from different ONNX versions or manual editing of the protobuf file, preventing consistent versioned operation handling.
Error: constant_segment index is not valid	Framework Frontend	The ExecuTorch parser encountered an invalid constant segment index while processing the model, indicating either corruption in the serialized data or an out-of-bounds access attempt. This prevents proper loading of constant tensor data required for execution.
Extended header length <LENGTH> is less than minimum required length <MIN_LENGTH>	Framework Frontend	The ExecuTorch parser rejected the extended header due to insufficient length, indicating the header section is either corrupted or improperly serialized. The actual header size falls below the framework's minimum required length for valid metadata storage, preventing model initialization.
Torch front-end: DataType not supported yet	Framework Frontend	The ExecuTorch parser encountered an unsupported data type during TorchScript model conversion, indicating either a custom type or framework feature not yet implemented in the ExecuTorch frontend. This blocks successful model parsing and deployment.

continues on next page

Table 1 – continued from previous page

Error Name	Module	Error Description
Torch front-end: Unhandled Constant data type	Framework Frontend	The ExecuTorch frontend encountered a constant tensor with an unsupported data type during TorchScript model conversion, indicating either a specialized numeric type or custom constant value that lacks handling in the parser. This prevents complete translation of the model's static data elements.
Torch/EXIR Edge operator not supported yet	Framework Frontend	The ExecuTorch runtime encountered an unsupported edge operator during execution, indicating either a newly introduced PyTorch operation or a specialized kernel that hasn't been implemented in the edge deployment target. This prevents the model from running on the specified edge device.
Torch front-end: non-tensor inputs/outputs are not supported	Framework Frontend	The ExecuTorch frontend encountered non-tensor inputs or outputs during TorchScript model conversion, which violates the framework's requirement that all model boundaries must use tensor types. This typically occurs when passing Python primitives (like integers or lists) directly across the model interface, preventing successful export to the edge runtime.
PatternMatchRewrite: the rewrite does not preserve required node	MERA Deploy	The pattern matching rewrite failed because it did not explicitly include a required output node in its replacement outputs, despite this node being consumed by operations outside the rewritten subgraph. This occurs when transformation rules neglect to propagate interface nodes that external graph segments depend on, breaking the model's dataflow integrity.
Found cycles while performing topological sort of subgraphs	MERA Deploy	The operation detected a cyclic dependency during topological sorting of a computational subgraph, indicating circular references between nodes that prevent valid execution ordering. This violates the requirement for directed acyclic graph (DAG) structures in model execution plans, stalling further processing until the cycle is resolved.
Vela optimized model to source: expected only one subgraph	MERA Deploy	When Ethos-U target is enabled the MERA compiler will identify subgraphs that can be accelerated with the NPU. It is expected that when MERA invokes the Arm Vela compiler to generate assembly only one subgraph will be present and that this whole subgraph will be merged by Vela into a single Ethos Vela optimized node. This error indicates that one or more nodes have been incorrectly identified as supported by the Ethos-U NPU. This error indicates a bug on MERA software stack that should not be solved by the user but reported to EdgeCortix.
No subgraphs found in model	MERA Deploy	Indicates an error processing the Vela optimized model because this model does not contain any subgraph. Can be either an error on either MERA or Arm Vela compiler.

continues on next page

Table 1 – continued from previous page

Error Name	Module	Error Description
No Ethos-U custom operators found in subgraph	MERA Deploy	MERA compiler identified that a subgraph can be accelerated with the Ethos-U NPU but after processing this subgraph with Arm Vela compiler no Arm Vela optimized custom nodes were found on the graph. This typically indicates that there is a bug on MERA compiler and should not be fixed by the user but reported to EdgeCortix.
More than one Ethos-U custom operator found in subgraph	MERA Deploy	MERA compiler identified that a subgraph can be accelerated with the Ethos-U NPU but after processing this subgraph with Arm Vela compiler both Arm Vela optimized custom nodes and CPU nodes were found on the graph. This typically indicates that there is a bug on MERA compiler and should not be fixed by the user but reported to EdgeCortix.
Error converting runtime plan to source: buffer belongs to several arenas	MERA Deploy	When MERA compiles a model using several targets as for example ARM Cortex-M + Ethos-U55, several subgraphs for each of these targets will be created. The graph that connects these subgraphs for either CPU or Ethos-U is detected as no supported when two subgraphs for the same target share the same input tensor. This situation is not currently supported by MERA because restrictions on how the NPU generally overwrite its inputs tensors as part of the memory plan generated by Arm Vela compiler.
Error converting the ONNX model to canonical IR	MERA Deploy	The model conversion from ONNX to canonical intermediate representation failed due to incompatible operators, unsupported attributes, or invalid graph structure that couldn't be properly translated. This prevents further processing or optimization of the model in the target framework.
Error converting the TFLite model to canonical IR	MERA Deploy	The model conversion from TFLite to canonical intermediate representation failed due to incompatible operators, unsupported attributes, or invalid graph structure that couldn't be properly translated. This prevents further processing or optimization of the model in the target framework.
Depthwise transposed conv2d is not supported by tflite	TFLite Export	TFLite doesn't support depthwise transposed convolutions, preventing conversion or execution of models using this operation.
TFLite exporter: Operator code not supported yet	TFLite Export	The TFLite exporter encountered an unsupported operator type, indicating the operation lacks an implementation for conversion to TFLite's flatbuffer format. This blocks model export until the operator is either implemented or replaced.
Operator conversion to tflite not supported yet	TFLite Export	Conversion to TFLite format failed because this operator type isn't currently supported in the exporter. The operation lacks a translation rule to TFLite's operator set, preventing model export.