Class InferenceOp

Table of contents

- Defined in File inference.hpp

Nested Relationships

Nested Types

- Struct InferenceOp::DataMap
- Struct InferenceOp::DataVecMap

Inheritance Relationships

Base Type

- `public holoscan::Operator` (Class Operator)

Class Documentation

class InferenceOp : public holoscan::Operator

Inference Operator class to perform single/multi model inference.

==Named Inputs==

- **receivers** : multi-receiver accepting `nvidia::gxf::Tensor` (s)
  - Any number of upstream ports may be connected to this `receivers` port. The operator will search across all messages for tensors matching those specified in `in_tensor_names`. These are the set of input tensors used by the models in `inference_map`.

==Named Outputs==

- **transmitter** : `nvidia::gxf::Tensor` (s)
  - A message containing tensors corresponding to the inference results from all models will be emitted. The names of the tensors transmitted correspond to those in `out_tensor_names`.

==Parameters==

For more details on `InferenceOp` parameters, see Customizing the Inference Operator or refer to Inference.

- **backend**: Backend to use for inference. Set `"trt"` for TensorRT, `"torch"` for LibTorch and `"onnxrt"` for the ONNX runtime.
- **allocator**: Memory allocator to use for the output.
- **inference_map**: Tensor to model map.
- **model_path_map**: Path to the ONNX model to be loaded.
- **pre_processor_map**: Pre processed data to model map.
- **device_map**: Mapping of model (`DataMap`) to GPU ID for inference. Optional.
- **backend_map**: Mapping of model (`DataMap`) to backend type for inference.
- **temporal_map**: Mapping of model (`DataMap`) to a frame delay for model inference. Optional. Backend options: `"trt"` or `"torch"` . Optional.
- **in_tensor_names**: Input tensors (`std::vector&lt;std::string&gt;`). Optional.
- **out_tensor_names**: Output tensors (`std::vector&lt;std::string&gt;`). Optional.
- **infer_on_cpu**: Whether to run the computation on the CPU instead of GPU. Optional (default: `false`).
- **parallel_inference**: Whether to enable parallel execution. Optional (default: `true`).
- **input_on_cuda**: Whether the input buffer is on the GPU. Optional (default: `true`).
- **output_on_cuda**: Whether the output buffer is on the GPU. Optional (default: `true`).

- **transmit_on_cuda**: Whether to transmit the message on the GPU. Optional (default: `true`).
- **enable_fp16**: Use 16-bit floating point computations. Optional (default: `false`).
- **is_engine_path**: Whether the input model path mapping is for trt engine files. Optional (default: `false`).
- **cuda_stream_pool**: `holoscan::CudaStreamPool` instance to allocate CUDA streams. Optional (default: `nullptr`).

==Device Memory Requirements==

When using this operator with a `BlockMemoryPool`, `num_blocks` must be greater than or equal to the number of output tensors that will be produced. The `block_size` in bytes must be greater than or equal to the largest output tensor (in bytes). If `output_on_cuda` is true, the blocks should be in device memory (`storage_type`=1), otherwise they should be CUDA pinned host memory (`storage_type`=0).

Public Functions

HOLOSCAN_OPERATOR_FORWARD_ARGS (InferenceOp) InferenceOp()=default

virtual void setup(OperatorSpec &spec) override

Define the operator specification.

Parameters

spec – The reference to the operator specification.

virtual void initialize() override

Initialize the operator.

This function is called when the fragment is initialized by Executor::initialize_fragment().

virtual void start() override

Implement the startup logic of the operator.

This method is called multiple times over the lifecycle of the operator according to the order defined in the lifecycle, and used for heavy initialization tasks such as allocating memory resources.

virtual void compute(InputContext &op_input, OutputContext &op_output, ExecutionContext &context) override

Implement the compute method.

This method is called by the runtime multiple times. The runtime calls this method until the operator is stopped.

Parameters

- **op_input** – The input context of the operator.
- **op_output** – The output context of the operator.
- **context** – The execution context of the operator.

virtual void stop() override

Implement the shutdown logic of the operator.

This method is called multiple times over the lifecycle of the operator according to the order defined in the lifecycle, and used for heavy deinitialization tasks such as deallocation of all resources previously assigned in start.

struct DataMap

DataMap specification

Public Functions

DataMap() = default

inline explicit operator bool() const noexcept

inline void insert(const std::string &key, const std::string &value)

inline std::map<std::string, std::string> get_map() const

Public Members

std::map<std::string, std::string> mappings_

struct DataVecMap

DataVecMap specification

Public Functions

DataVecMap() = default

inline explicit operator bool() const noexcept

inline void insert(const std::string &key, const std::vector<std::string> &value)

inline std::map<std::string, std::vector<std::string>> get_map() const

Public Members

std::map<std::string, std::vector<std::string>> mappings_