Program Listing for File inference.hpp

Return to documentation for file (include/holoscan/operators/inference/inference.hpp)

```
/* * SPDX-FileCopyrightText: Copyright (c) 2022-2024 NVIDIA CORPORATION & AFFILIATES. All rights reserved. * SPDX-License-Identifier: Apache-2.0 * * Licensed under the Apache License, Version 2.0 (the "License"); * you may not use this file except in compliance with the License. * You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License. */ #ifndef HOLOSCAN_OPERATORS_INFERENCE_HPP #define HOLOSCAN_OPERATORS_INFERENCE_HPP #include <map> #include <memory> #include <string> #include <vector> #include "holoscan/core/io_context.hpp" #include "holoscan/core/io_spec.hpp" #include "holoscan/core/operator.hpp" #include "holoscan/core/operator_spec.hpp" #include "holoscan/utils/cuda_stream_handler.hpp" #include <holoinfer.hpp> #include <holoinfer_buffer.hpp> #include <holoinfer_utils.hpp> namespace HoloInfer = holoscan::inference; namespace holoscan::ops { class InferenceOp : public holoscan::Operator { public: HOLOSCAN_OPERATOR_FORWARD_ARGS(InferenceOp) InferenceOp() = default; void setup(OperatorSpec& spec) override; void initialize() override; void start() override; void compute(InputContext& op_input, OutputContext& op_output, ExecutionContext& context) override; void stop() override; struct DataMap { DataMap() = default; explicit operator bool() const noexcept { return !mappings_.empty(); } void insert(const std::string& key, const std::string& value) { mappings_[key] = value; } std::map<std::string, std::string> get_map() const { return mappings_; } std::map<std::string, std::string> mappings_; }; struct DataVecMap { DataVecMap() = default; explicit operator bool() const noexcept { return !mappings_.empty(); } void insert(const std::string& key, const std::vector<std::string>& value) { for (auto const& val : value) mappings_[key].push_back(val); } std::map<std::string, std::vector<std::string>> get_map() const { return mappings_; } std::map<std::string, std::vector<std::string>> mappings_; }; private: Parameter<DataVecMap> inference_map_; Parameter<DataMap> model_path_map_; Parameter<DataVecMap>
```

```
pre_processor_map_; Parameter<DataMap> device_map_; Parameter<DataMap>
temporal_map_; Parameter<std::vector<std::string>> in_tensor_names_;
Parameter<std::vector<std::string>> out_tensor_names_;
Parameter<std::shared_ptr<Allocator>> allocator_; Parameter<bool> infer_on_cpu_;
Parameter<bool> parallel_inference_; Parameter<bool> input_on_cuda_;
Parameter<bool> output_on_cuda_; Parameter<bool> transmit_on_cuda_;
Parameter<bool> enable_fp16_; Parameter<bool> is_engine_path_;
Parameter<std::string> backend_; Parameter<DataMap> backend_map_;
Parameter<std::vector<IOSpec*>> receivers_; Parameter<std::vector<IOSpec*>>
transmitter_; // Internal state std::unique_ptr<HoloInfer::InferContext>
holoscan_infer_context_; std::shared_ptr<HoloInfer::InferenceSpecs>
inference_specs_; std::map<std::string, std::vector<int>> dims_per_tensor_; const
std::string module_{"Inference Operator"}; CudaStreamHandler
cuda_stream_handler_; }; } // namespace holoscan::ops #endif/*
HOLOSCAN_OPERATORS_INFERENCE_HPP */
```