Program Listing for File holoinfer_utils.hpp

Return to documentation for file (modules/holoinfer/src/include/holoinfer_utils.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_INFER_UTILS_API_H #define _HOLOSCAN_INFER_UTILS_API_H #include <sys/utsname.h> #include <filesystem> #include <map> #include <string> #include <vector> #include "gxf/core/entity.hpp" #include "gxf/core/gxf.h" #include "gxf/core/parameter.hpp" #include "gxf/cuda/cuda_stream.hpp" #include "gxf/cuda/cuda_stream_id.hpp" #include "gxf/cuda/cuda_stream_pool.hpp" #include "gxf/multimedia/video.hpp" #include "gxf/std/allocator.hpp" #include "gxf/std/clock.hpp" #include "gxf/std/codelet.hpp" #include "gxf/core/parameter_parser_std.hpp" #include "gxf/std/receiver.hpp" #include "gxf/std/tensor.hpp" #include "gxf/std/timestamp.hpp" #include "gxf/std/transmitter.hpp" #include "holoinfer_buffer.hpp" #include "holoinfer_constants.hpp" namespace holoscan { namespace inference { cudaError_t check_cuda(cudaError_t result); gxf_result_t _HOLOSCAN_EXTERNAL_API_ report_error(const std::string& module, const std::string& submodule); void _HOLOSCAN_EXTERNAL_API_ raise_error(const std::string& module, const std::string& submodule); InferStatus inference_validity_check(const Mappings& model_path_map, const MultiMappings& pre_processor_map, const MultiMappings& inference_map, std::vector<std::string>& in_tensor_names, std::vector<std::string>& out_tensor_names); InferStatus processor_validity_check(const MultiMappings& processed_map, const std::vector<std::string>& in_tensor_names, const std::vector<std::string>& out_tensor_names); bool is_platform_aarch64(); void timer_init(TimePoint& _t); gxf_result_t timer_check(TimePoint& start, TimePoint& end, const std::string& module); void string_split(const std::string& line, std::vector<std::string>& tokens, char c); using node_type = std::map<std::string,

```
std::map<std::string, std::string>>; static const std::map<std::string, holoinfer_datatype> kHoloInferDataTypeMap = { {"kFloat32", holoinfer_datatype::h_Float32}, {"kInt32", holoinfer_datatype::h_Int32}, {"kInt8", holoinfer_datatype::h_Int8}, {"kUInt8", holoinfer_datatype::h_UInt8}, {"kInt64", holoinfer_datatype::h_Int64}}; InferStatus parse_yaml_node(const node_type& in_config, std::vector<std::string>& names, std::vector<std::vector<int64_t>>& dims, std::vector<holoinfer_datatype>& types); } // namespace inference } // namespace holoscan #endif
```