

# **Contents**

1	Einle	eitung	1						
2	Nam	nespace Index 3							
	2.1	Namespace List	3						
3	Hiera	erarchical Index 5							
	3.1	Class Hierarchy	5						
4	Clas	es Index	7						
	4.1	Class List	7						
5	File	Index	9						
	5.1	File List	9						
6	Nam	nespace Documentation	11						
	6.1	anonymous_namespace{TensorflowAndroidJNIUtils.cpp} Namespace Reference	11						
	6.2	FilterManagementLibrary Namespace Reference	11						
		6.2.1 Detailed Description	11						
	6.3	FilterManagementLibrary::PipeSystem Namespace Reference	12						
		6.3.1 Detailed Description	12						
	6.4	FilterManagementLibrary::TFIntegration Namespace Reference	12						
		6.4.1 Detailed Description	12						
	6.5	google Namespace Reference	12						
	6.6	google::protobuf Namespace Reference	13						
	6.7	RoadSignAPI Namespace Reference	13						
	6.8	tensorflow Namespace Reference	13						
	6.9	tensorflow::android Namespace Reference	13						

ii CONTENTS

7	Clas	s Docu	mentation	1	15
	7.1	RoadS	ignAPI::Cl	lassifiedSignsGrouper Class Reference	15
		7.1.1	Detailed	Description	15
		7.1.2	Member	Function Documentation	16
			7.1.2.1	initByPipeSetup()	16
			7.1.2.2	process()	16
		7.1.3	Member	Data Documentation	16
			7.1.3.1	castedWorkingDataSet	17
			7.1.3.2	horizontalRangePercentage	17
			7.1.3.3	imageWidth	17
	7.2	RoadS	ignAPI::Do	etectedSignCombination Class Reference	17
		7.2.1	Detailed	Description	18
		7.2.2	Construc	ctor & Destructor Documentation	18
			7.2.2.1	DetectedSignCombination()	18
			7.2.2.2	~DetectedSignCombination()	18
		7.2.3	Member	Function Documentation	18
			7.2.3.1	addDetectedSign()	19
			7.2.3.2	getDetectedSignsAmount()	19
			7.2.3.3	getGestimatedPolePositionX()	19
			7.2.3.4	getSignsInCombination()	19
			7.2.3.5	insertSorted()	20
		7.2.4	Member	Data Documentation	20
			7.2.4.1	estimatedPolePositionX	20
			7.2.4.2	signs	20
	7.3	RoadS	ignAPI::De	etectedSignDescriptor Struct Reference	20
		7.3.1	Detailed	Description	21
		7.3.2	Member	Data Documentation	21
			7.3.2.1	classifierApprovedClassID	21
			7.3.2.2	classifierConfidence	21
			7.3.2.3	detectionPredictedClassID	21

CONTENTS

		7.3.2.4	detectorConfidence	21
		7.3.2.5	lowerRight	21
		7.3.2.6	upperLeft	22
7.4	RoadS	SignAPI::De	etectionBasedImageSlicer Class Reference	22
	7.4.1	Detailed	Description	22
	7.4.2	Member	Function Documentation	23
		7.4.2.1	expandBox()	23
		7.4.2.2	initByPipeSetup()	23
		7.4.2.3	process()	24
	7.4.3	Member	Data Documentation	24
		7.4.3.1	castedWorkingDataSet	24
		7.4.3.2	expandPercentage	24
		7.4.3.3	minExpandPixels	24
7.5	anonyr	mous_nam	nespace{TensorflowAndroidJNIUtils.cpp}::IfstreamInputStream Class Reference	25
	7.5.1	Construc	ctor & Destructor Documentation	25
		7.5.1.1	IfstreamInputStream()	25
		7.5.1.2	~IfstreamInputStream()	25
	7.5.2	Member	Function Documentation	25
		7.5.2.1	Read()	25
	7.5.3	Member	Data Documentation	26
		7.5.3.1	ifs	26
7.6	tensor	flow::andro	oid::LimitingFileInputStream Class Reference	26
	7.6.1	Construc	etor & Destructor Documentation	26
		7.6.1.1	LimitingFileInputStream()	27
		7.6.1.2	~LimitingFileInputStream()	27
	7.6.2	Member	Function Documentation	27
		7.6.2.1	Read()	27
		7.6.2.2	Skip()	27
	7.6.3	Member	Data Documentation	27
		7.6.3.1	bytes_left	27

iv CONTENTS

		7.6.3.2	errno	28
		7.6.3.3	fd	28
7.7	FilterM	lanagemen	tLibrary::Logger Class Reference	28
	7.7.1	Detailed [	Description	28
	7.7.2	Member F	Function Documentation	29
		7.7.2.1	getTimeString()	29
		7.7.2.2	printfln()	30
		7.7.2.3	setLogTag()	30
	7.7.3	Member [	Data Documentation	31
		7.7.3.1	logTag	31
7.8	RoadS	SignAPI::Mo	bilenetV2RoadSignClassificator Class Reference	31
	7.8.1	Detailed [	Description	32
	7.8.2	Construct	or & Destructor Documentation	32
		7.8.2.1	MobilenetV2RoadSignClassificator() [1/2]	32
		7.8.2.2	MobilenetV2RoadSignClassificator() [2/2]	33
	7.8.3	Member F	Function Documentation	33
		7.8.3.1	applyImageVectorFromOpenCVMat()	33
		7.8.3.2	initByPipeSetup()	34
		7.8.3.3	isInUnwantedClasses()	34
		7.8.3.4	onNNEvaluationFinished()	35
		7.8.3.5	process()	35
	7.8.4	Member [	Data Documentation	35
		7.8.4.1	assetManager	36
		7.8.4.2	castedWorkingDataSet	36
		7.8.4.3	currentCutOutImage	36
		7.8.4.4	currentRecognition	36
		7.8.4.5	nnModelInputHeight	36
		7.8.4.6	nnModelInputWidth	36
		7.8.4.7	threshold	37
		7.8.4.8	validRecognition	37

CONTENTS

7.9	FilterM	anagemer	atLibrary::PipeSystem::PipeFilter Class Reference	37
	7.9.1	Detailed	Description	38
	7.9.2	Construc	tor & Destructor Documentation	39
		7.9.2.1	PipeFilter()	39
		7.9.2.2	~PipeFilter()	39
	7.9.3	Member	Function Documentation	39
		7.9.3.1	getFilterID()	39
		7.9.3.2	getNextDesiredFilter()	39
		7.9.3.3	hasMarkedNextDesiredFilter()	40
		7.9.3.4	hasMarkedProcessingFinished()	40
		7.9.3.5	indicateProcessingFinished()	40
		7.9.3.6	initByPipeSetup()	41
		7.9.3.7	invokeNext()	41
		7.9.3.8	process()	41
		7.9.3.9	reset()	41
		7.9.3.10	setCredentials()	42
	7.9.4	Friends A	and Related Function Documentation	43
		7.9.4.1	ProcessingPipeline	43
	7.9.5	Member	Data Documentation	43
		7.9.5.1	filterID	43
		7.9.5.2	markedNextDesiredFilter	43
		7.9.5.3	markedProcessingFinished	43
		7.9.5.4	nextDesiredFilter	44
		7.9.5.5	pipeRegisteredFilters	44
		7.9.5.6	pipeWorkingDataSet	44
7.10	FilterM	anagemer	ntLibrary::PipeSystem::PipeRegisteredFilters Struct Reference	44
	7.10.1	Detailed	Description	45
7.11	FilterM	anagemer	atLibrary::PipeSystem::PipeWorkingDataSet Struct Reference	45
	7.11.1	Detailed	Description	45
7.12	FilterM	anagemer	atLibrary::PipeSystem::ProcessingPipeline Class Reference	45

vi

	7.12.1	Detailed Description	46
	7.12.2	Member Enumeration Documentation	47
		7.12.2.1 ErrorType	47
	7.12.3	Constructor & Destructor Documentation	47
		7.12.3.1 ProcessingPipeline()	48
		7.12.3.2 ~ProcessingPipeline()	48
	7.12.4	Member Function Documentation	48
		7.12.4.1 getLastError()	49
		7.12.4.2 processCurrentDataSet()	49
		7.12.4.3 registerFilter()	49
		7.12.4.4 setup()	50
	7.12.5	Member Data Documentation	50
		7.12.5.1 currentFilterID	50
		7.12.5.2 lastError	50
		7.12.5.3 manageExternalAllocatedRessources	50
		7.12.5.4 pipeRegisteredFiltersHeader	51
		7.12.5.5 processingFinishied	51
		7.12.5.6 registeredFilters	51
		7.12.5.7 workingDataSet	51
7.13	RoadS	gnAPI::RoadSignAPI Class Reference	51
	7.13.1	Detailed Description	53
	7.13.2	Constructor & Destructor Documentation	53
		7.13.2.1 RoadSignAPI() [1/2]	53
		7.13.2.2 RoadSignAPI() [2/2]	54
	7.13.3	Member Function Documentation	54
		7.13.3.1 feedImage()	54
		7.13.3.2 getClassifierApprovedDetectedSigns()	55
		7.13.3.3 getDetectedSignCombinations()	55
		7.13.3.4 getDetectedSigns()	56
		7.13.3.5 init()	56

CONTENTS vii

		7.13.3.6	staticFeedImage()	56
		7.13.3.7	staticGetClassifierApprovedDetectedSigns()	56
		7.13.3.8	staticGetDetectedSignCombinations()	57
		7.13.3.9	staticGetDetectedSigns()	57
		7.13.3.10	staticInit() [1/2]	58
		7.13.3.11	<b>staticInit()</b> [2/2]	58
	7.13.4	Member I	Data Documentation	58
		7.13.4.1	assetManager	58
		7.13.4.2	classifiedSignsGrouper	58
		7.13.4.3	detectionBasedImageSlicer	59
		7.13.4.4	instance	59
		7.13.4.5	mobilenetModelDescription	59
		7.13.4.6	pipeRegisteredFilters	59
		7.13.4.7	processingPipeline	59
		7.13.4.8	roadSignClassificator	59
		7.13.4.9	roadSignDetector	59
		7.13.4.10	roadSignDuplicationDeleter	60
		7.13.4.11	ssdLiteModelDescription	60
		7.13.4.12	workingDataSet	60
7.14	RoadS	ignAPI::Ro	padSignDuplicationDeleter Class Reference	60
	7.14.1	Detailed I	Description	61
	7.14.2	Member I	Function Documentation	61
		7.14.2.1	calculateBoxArea()	61
		7.14.2.2	calculateOverlapPercentage()	61
		7.14.2.3	deleteAllMarkedSignDescriptors()	62
		7.14.2.4	initByPipeSetup()	62
		7.14.2.5	isMarkedForDeletion()	62
		7.14.2.6	markForDeletion()	62
		7.14.2.7	process()	63
		7.14.2.8	signsDoOverlap()	63

viii CONTENTS

	7.14.3	Member Data Documentation	63
		7.14.3.1 castedWorkingDataSet	63
		7.14.3.2 minOverlapPercentage	64
		7.14.3.3 signsToDelete	64
7.15	RoadS	ignAPI::RSAPIPipeRegisteredFilters Struct Reference	64
	7.15.1	Detailed Description	64
	7.15.2	Member Data Documentation	64
		7.15.2.1 CLASSIFIED_SIGNS_GROUPER_FILTER	65
		7.15.2.2 DETECTION_BASED_IMAGE_SLICER_FILTER	65
		7.15.2.3 SIGN_DETECTION_FILTER	65
		7.15.2.4 SIGN_DUPLICATION_DELETER_FILTER	65
		7.15.2.5 SIGN_RECOGNITION_FILTER	65
7.16	RoadS	ignAPI::RSAPIWorkingDataSet Struct Reference	65
	7.16.1	Detailed Description	66
	7.16.2	Member Data Documentation	66
		7.16.2.1 classifierApprovedSigns	66
		7.16.2.2 cutOutImages	66
		7.16.2.3 detectedSignCombinations	66
		7.16.2.4 detectedSigns	66
		7.16.2.5 detectorScaledBGRImage	67
		7.16.2.6 originalBGRImage	67
		7.16.2.7 originalImageHeight	67
		7.16.2.8 originalImageWidth	67
7.17	RoadS	ignAPI::SSDLiteRoadSignDetector Class Reference	67
	7.17.1	Detailed Description	68
	7.17.2	Constructor & Destructor Documentation	68
		7.17.2.1 SSDLiteRoadSignDetector() [1/2]	68
		7.17.2.2 SSDLiteRoadSignDetector() [2/2]	69
	7.17.3	Member Function Documentation	69
		7.17.3.1 applyImageVectorFromOpenCVMat()	69

CONTENTS

7.17.3.2 initByPipeSetup()	70
7.17.3.3 onNNEvaluationFinished()	70
7.17.3.4 process()	71
7.17.4 Member Data Documentation	71
7.17.4.1 assetManager	71
7.17.4.2 castedWorkingDataSet	71
7.17.4.3 nnModelInputHeight	72
7.17.4.4 nnModelInputWidth	72
7.17.4.5 threshold	72
7.18 FilterManagementLibrary::TensorflowAndroidJNIUtils Class Reference	72
7.18.1 Member Function Documentation	72
7.18.1.1 IsAsset()	73
7.18.1.2 PortableReadFileToProto()	73
7.18.1.3 ReadFileToProto()	73
7.18.1.4 ReadFileToString()	73
7.18.1.5 ReadFileToVector()	73
7.18.1.6 WriteProtoToFile()	74
7.19 FilterManagementLibrary::TFIntegration::TensorflowNNInstance Class Reference	74
7.19.1 Detailed Description	75
7.19.2 Member Enumeration Documentation	75
7.19.2.1 ErrorType	75
7.19.3 Constructor & Destructor Documentation	76
7.19.3.1 TensorflowNNInstance()	76
7.19.3.2 ~TensorflowNNInstance()	77
7.19.4 Member Function Documentation	77
7.19.4.1 adjustModelFile()	77
7.19.4.2 applyImageInputVector()	78
7.19.4.3 applyInputVectorFromFloatData()	78
7.19.4.4 getInputTensor()	79
7.19.4.5 getLastError()	79

CONTENTS

		7.19.4.6	getModelDescription()	79
		7.19.4.7	getOutputTensors()	80
		7.19.4.8	getResultContainer()	80
		7.19.4.9	runInference()	80
		7.19.4.10	setupModelFromAssets()	81
		7.19.4.11	setupModelFromFile()	81
	7.19.5	Member	Data Documentation	81
		7.19.5.1	graphDef	81
		7.19.5.2	inputLayerName	82
		7.19.5.3	inputTensor	82
		7.19.5.4	lastError	82
		7.19.5.5	nnModelDescription	82
		7.19.5.6	numThreads	82
		7.19.5.7	outputTensors	82
		7.19.5.8	sessionOptions	82
		7.19.5.9	tensorflowSession	83
7.20	FilterM	anagemer	atLibrary::TFIntegration::TensorflowNNInstanceClassifier Class Reference	83
	7.20.1	Detailed	Description	84
	7.20.2	Construc	tor & Destructor Documentation	84
		7.20.2.1	TensorflowNNInstanceClassifier()	84
	7.20.3	Member	Function Documentation	84
		7.20.3.1	getTopNPredictions()	84
		7.20.3.2	getTopNPredictionsStringVector()	85
		7.20.3.3	readLabelsFile()	85
	7.20.4	Member	Data Documentation	86
		7.20.4.1	labelCount	86
		7.20.4.2	loadedLabels	86
7.21	FilterM	anagemer	atLibrary::TFIntegration::TensorflowNNModelDescription Struct Reference	86
	7.21.1	Member	Data Documentation	87
		7.21.1.1	channels	87

CONTENTS xi

7.21.1.2 input_floating	 87
7.21.1.3 input_mean	 87
7.21.1.4 input_std	 87
7.21.1.5 inputHeight	 88
7.21.1.6 inputLayerNameStr	 88
7.21.1.7 inputWidth	 88
7.21.1.8 modelFile	 88
7.21.1.9 outputLayerNames	 88
7.22 FilterManagementLibrary::TensorflowOpenCVUtils Class Reference	 88
7.22.1 Detailed Description	 89
7.22.2 Member Function Documentation	 89
7.22.2.1 fastApplyCVMatOnInputTensorFloat()	 89
7.22.2.2 fastApplyCVMatOnInputTensorUInt8()	 89
7.23 FilterManagementLibrary::TFIntegration::TensorflowResultContainer Class Reference	 90
7.23.1 Detailed Description	 91
7.23.2 Constructor & Destructor Documentation	 91
7.23.2.1 TensorflowResultContainer()	 91
7.23.3 Member Function Documentation	 91
7.23.3.1 getOutputTensorByID()	 91
7.23.3.2 getOutputTensorByLayerName()	 92
7.23.4 Member Data Documentation	 92
7.23.4.1 outputTensorNames	 92
7.23.4.2 outputTensors	 92
7.24 FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter Class Reference	 93
7.24.1 Detailed Description	 94
7.24.2 Constructor & Destructor Documentation	 94
7.24.2.1 TFNNBasedPipeFilter()	 94
7.24.2.2 ~TFNNBasedPipeFilter()	 95
7.24.3 Member Function Documentation	 95
7.24.3.1 adjustModelFile()	 95

xii CONTENTS

			7.24.3.2	applyNNImageInputVector()	95
			7.24.3.3	evaluateInputVectorByNN()	96
			7.24.3.4	getNeuralNetLastError()	96
			7.24.3.5	getNNInputTensor()	97
			7.24.3.6	getNNModelDescription()	97
			7.24.3.7	onNNEvaluationFinished()	97
			7.24.3.8	setupModelFromAssets()	97
			7.24.3.9	setupModelFromFile()	98
		7.24.4	Member	Data Documentation	98
			7.24.4.1	tfNNInstance	98
	7.25	FilterM	anagemer	ntLibrary::Utilities Class Reference	99
		7.25.1	Detailed	Description	99
		7.25.2	Member	Function Documentation	99
			7.25.2.1	fileExists()	99
8	File I	Docume	entation		101
	8.1			andalone_Tensorflow/_10_introduction.dox File Reference	
	8.2		_	andalone_Tensorflow/codesnippets.cpp File Reference	
		8.2.1	_		
			8.2.1.1	cameralmg()	101
			8.2.1.2	convertTo()	
			8.2.1.3	scaledImage()	
		8.2.2	Variable	Documentation	
			8.2.2.1	1	
			8.2.2.2	image	
			8.2.2.3	p	
			8.2.2.4	start time	
	8.3	RoadS		andalone_Tensorflow/header_files/FilterManagementLibrary/limiting_file_input_	102
	0.0		_	ference	102
	8.4		-	andalone_Tensorflow/header_files/FilterManagementLibrary/Logger.h File Refer-	103

CONTENTS xiii

8.5	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/Pipe ← Filter.h File Reference	103
8.6	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/Pipe RegisteredFilters.h File Reference	103
8.7	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/Pipe WorkingDataSet.h File Reference	104
8.8	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/← ProcessingPipeline.h File Reference	104
8.9	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/TFNN BasedPipeFilter.h File Reference	104
8.10	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowAndroid UNIUtils.h File Reference	105
8.11	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowIntegration/ TensorflowNNInstance.h File Reference	
8.12	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowIntegration/ TensorflowNNInstanceClassifier.h File Reference	
8.13	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowIntegration/ TensorflowNNModelDescription.h File Reference	
8.14	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowIntegration/ TensorflowResultContainer.h File Reference	
8.15	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowOpenC ← VUtils.h File Reference	106
8.16	RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/Utilities.h File Reference	107
8.17	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/DetectedSignCombination.h File Reference	107
8.18	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/DetectedSignDescriptor.h File Reference	107
8.19	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/ClassifiedSignsGrouper.h File Reference	107
8.20	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/DetectionBasedImage Slicer.h File Reference	108
8.21	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/MobilenetV2RoadSign Classificator.h File Reference	108
8.22	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/RoadSignDuplication Deleter.h File Reference	108
8.23	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/SSDLiteRoadSign Detector.h File Reference	108
8.24	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RoadSignAPI.h File Reference .	109

XIV

8.25	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RSAPIPipeRegisteredFilters/R← SAPIPipeRegisteredFilters.h File Reference				
8.26	RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RSAPIWorkingDataSet/RSAPI WorkingDataSet.h File Reference				
8.27	RoadSi	ignAPI_Sta	andalone_Tensorflow/main.cpp File Reference	109	
	8.27.1	Typedef [	Documentation	110	
		8.27.1.1	Pixel	110	
	8.27.2	Function	Documentation	110	
		8.27.2.1	applyFromCVMat()	110	
		8.27.2.2	get_us()	110	
		8.27.2.3	main()	110	
		8.27.2.4	openCVMatTouint8_t3DArray()	111	
		8.27.2.5	openCVMatTouint8_tArray()	111	
		8.27.2.6	readLabelsFile()	111	
		8.27.2.7	type2str()	111	
8.28	RoadSi	ignAPI_St	andalone_Tensorflow/mainold.cpp File Reference	111	
	8.28.1	Typedef [	Documentation	112	
		8.28.1.1	Pixel	112	
	8.28.2	Function	Documentation	112	
		8.28.2.1	get_us()	112	
		8.28.2.2	GetTopN()	112	
		8.28.2.3	main()	112	
		8.28.2.4	openCVMatTouint8_t3DArray()	113	
		8.28.2.5	openCVMatTouint8_tArray()	113	
		8.28.2.6	testNet()	113	
		8.28.2.7	type2str()	113	
8.29		_	andalone_Tensorflow/source_files/FilterManagementLibrary/Logger.cpp File Ref-	113	
8.30			andalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/Pipe↔ ference	113	
8.31			andalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/← ne.cpp File Reference	113	

CONTENTS xv

8.32	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/TFNN← BasedPipeFilter.cpp File Reference	113
8.33	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowAndroid UNIUtils.cpp File Reference	113
	8.33.1 Variable Documentation	114
	8.33.1.1 ASSET_PREFIX	114
8.34	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/-TensorflowNNInstance.cpp File Reference	
8.35	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/-TensorflowNNInstanceClassifier.cpp File Reference	
8.36	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/-TensorflowResultContainer.cpp File Reference	
8.37	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowOpenC \( \cdot \) VUtils.cpp File Reference	114
	8.37.1 Typedef Documentation	114
	8.37.1.1 Pixel	114
8.38	RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/Utilities.cpp File Reference	115
8.39	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/DetectedSignCombination.cpp File Reference	115
8.40	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/ClassifiedSignsGrouper.cppFile Reference	
8.41	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/DetectionBasedImage Slicer.cpp File Reference	115
8.42	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/MobilenetV2RoadSign Classificator.cpp File Reference	115
8.43	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/RoadSignDuplication Deleter.cpp File Reference	115
8.44	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/SSDLiteRoadSign Detector.cpp File Reference	115
8.45	RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/RoadSignAPI.cpp File Reference	115
Index		117

## **Einleitung**

Die Filterverwaltungs-Bibliothek ist nach dem Pipes-and-Filters-Prinzip aufgebaut. Dieses ist ein gängiges Architekturmuster für Systeme, die Datenströme verarbeiten. In unserem Fall bestehen diese aus (einer Folge von) Einzelbildern mit einem Header, in dem Zusatzinformationen Platz finden. Ein Filter stellt dabei einen Verarbeitungsschritt dar, wobei jeder Filter der Pipe ein Bild entgegennimmt und auch wieder ein Bild herausgibt. Darüber hinaus kann jeder Filter dem Datenpaket bestimmte Metainformationen beifügen, die später für eine genauere Analyse verwendet werden können. Wenn beispielsweise ein Filter ein Verkehrszeichen auf einem Bild an einer bestimmten Position detektiert und im darauffolgenden Filter das Verkehrszeichen klassifiziert wird, wird neben dem Zuschneiden des Bildes auf das erkannte Verkehrszeichen auch die Position des Verkehrszeichens im Original als Metainformation gespeichert. Dies ist eine Abwandlung der herkömmlichen Pipes-and-Filters-Architektur. Weiterhin ist es möglich, dass sich die Pipeline nach bestimmten Verarbeitungsschritten in mehrere untergeordnete Pipes aufteilt: Werden auf einem Bild mehrere Verkehrszeichen erkannt, wird der Filter für die Klassifikation für alle Kandidaten aufgerufen (ggf. parallel).

2 Einleitung

# Namespace Index

## 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

anonymous_namespace{TensorflowAndroidJNIUtils.cpp}	11
FilterManagementLibrary	
Everything the FilterManagementLibrary provides	11
FilterManagementLibrary::PipeSystem	
Everything that has to do with the pipes and filters part of the library	12
FilterManagementLibrary::TFIntegration	12
google	12
google::protobuf	13
RoadSignAPI	13
tensorflow	13
tensorflow::android	13

4 Namespace Index

# **Hierarchical Index**

## 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CopyingInputStream
anonymous_namespace{TensorflowAndroidJNIUtils.cpp}::lfstreamInputStream
tensorflow::android::LimitingFileInputStream
RoadSignAPI::DetectedSignCombination
RoadSignAPI::DetectedSignDescriptor
FilterManagementLibrary::Logger
FilterManagementLibrary::PipeSystem::PipeFilter
FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter
RoadSignAPI::MobilenetV2RoadSignClassificator
RoadSignAPI::SSDLiteRoadSignDetector
RoadSignAPI::ClassifiedSignsGrouper
RoadSignAPI::DetectionBasedImageSlicer
RoadSignAPI::RoadSignDuplicationDeleter
FilterManagementLibrary::PipeSystem::PipeRegisteredFilters
RoadSignAPI::RSAPIPipeRegisteredFilters
FilterManagementLibrary::PipeSystem::PipeWorkingDataSet
RoadSignAPI::RSAPIWorkingDataSet
FilterManagementLibrary::PipeSystem::ProcessingPipeline
RoadSignAPI::RoadSignAPI
FilterManagementLibrary::TensorflowAndroidJNIUtils
FilterManagementLibrary::TFIntegration::TensorflowNNInstance
FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier
FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription
FilterManagementLibrary::TensorflowOpenCVUtils
FilterManagementLibrary::TFIntegration::TensorflowResultContainer
FilterManagement ibrary: Itilities

6 Hierarchical Index

# **Class Index**

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

RoadSignAPI::ClassifiedSignsGrouper	15
RoadSignAPI::DetectedSignCombination	17
RoadSignAPI::DetectedSignDescriptor	20
RoadSignAPI::DetectionBasedImageSlicer	22
anonymous_namespace{TensorflowAndroidJNIUtils.cpp}::IfstreamInputStream	25
tensorflow::android::LimitingFileInputStream	26
FilterManagementLibrary::Logger	
Logger providing Desktop Linux <-> Android platform independent logging	28
RoadSignAPI::MobilenetV2RoadSignClassificator	31
FilterManagementLibrary::PipeSystem::PipeFilter	
Base class were all filters are derived from	37
FilterManagementLibrary::PipeSystem::PipeRegisteredFilters	
Template struct	44
FilterManagementLibrary::PipeSystem::PipeWorkingDataSet	
Template structfor PipeWorkingDataSets	45
FilterManagementLibrary::PipeSystem::ProcessingPipeline	
Pipeline of the pipes and filters architecture	45
RoadSignAPI::RoadSignAPI	51
RoadSignAPI::RoadSignDuplicationDeleter	60
RoadSignAPI::RSAPIPipeRegisteredFilters	64
RoadSignAPI::RSAPIWorkingDataSet	65
RoadSignAPI::SSDLiteRoadSignDetector	67
FilterManagementLibrary::TensorflowAndroidJNIUtils	72
FilterManagementLibrary::TFIntegration::TensorflowNNInstance	74
FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier	83
FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription	86
FilterManagementLibrary::TensorflowOpenCVUtils	88
FilterManagementLibrary::TFIntegration::TensorflowResultContainer	90
FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter	
Abstract template class to derive Tensorflow based filters from	93
FilterManagementLibrary::Utilities	
Providing some basic utilities, like checking if a file exists	99

8 Class Index

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

RoadSignAPI_Standalone_Tensorflow/codesnippets.cpp
RoadSignAPI_Standalone_Tensorflow/main.cpp
RoadSignAPI_Standalone_Tensorflow/mainold.cpp
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/limiting_file_input_
stream.h
RoadSignAPI Standalone Tensorflow/header files/FilterManagementLibrary/Logger.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowAndroidJNI
Utils.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowOpenCVUtils.h 106
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/Utilities.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/PipeFilter.h . 103
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/PipeRegistered
Filters.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/PipeWorking
DataSet.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/Processing
Pipeline.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/PipeSystem/TFNN←
BasedPipeFilter.h
RoadSignAPI_Standalone_Tensorflow/header_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow-
NNInstance.h
$Road Sign API\_Standal one\_Tensor flow/header\_files/Filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow-filter Management Library/Tensor flow-filter flow-filter flow-filter flow-fi$
NNInstanceClassifier.h
$Road Sign API\_Standal one\_Tensor flow/header\_files/Filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow-filter Management Library/Tensor flow-filter flow-filter flow-filter flow-fi$
NNModelDescription.h
$Road Sign API\_Standal one\_Tensor flow/header\_files/Filter Management Library/Tensor flow Integration/Tensor flow/flow/filter Management Library/Tensor flow Integration/Tensor flow/flow/filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow/flow-filter Management Library/Tensor flow Integration/Tensor flow flow flow flow flow flow flow flow$
ResultContainer.h
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/DetectedSignCombination.h 107
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/DetectedSignDescriptor.h 107
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RoadSignAPI.h
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/ClassifiedSignsGrouper.h 107
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/Filters/DetectionBasedImageSlicer.h . 108
$Road Sign API\_Standal one\_Tensor flow/header\_files/Road Sign API/Filters/Mobile net V2Road Sign Classificator. \leftarrow$
h 108
RoadSignAPI Standalone Tensorflow/header files/RoadSignAPI/Filters/RoadSignDuplicationDeleter.h . 108

10 File Index

${\sf RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/SSDLiteRoadSignDetector.h}$ . 1	108
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RSAPIPipeRegisteredFilters/RSAPI	
PipeRegisteredFilters.h	109
RoadSignAPI_Standalone_Tensorflow/header_files/RoadSignAPI/RSAPIWorkingDataSet/RSAPI	
WorkingDataSet.h	109
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/Logger.cpp	113
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowAndroidJNI	
Utils.cpp	113
$Road Sign API\_Standal one\_Tensor flow/source\_files/Filter Management Library/Tensor flow Open CVU tils. {\it } \\$	
	114
	115
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/PipeFilter.cpp 1	113
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/Processing←	
	113
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/PipeSystem/TFNNBased↔	
Programme and the second secon	113
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/ <mark>Tensorflo</mark>	→WC
NNInstance.cpp	114
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/ <mark>Tensorflo</mark>	→WC
	114
RoadSignAPI_Standalone_Tensorflow/source_files/FilterManagementLibrary/TensorflowIntegration/Tensorflo	→WC
ResultContainer.cpp	114
$Road Sign API\_Standal one\_Tensor flow/source\_files/Road Sign API/Detected Sign Combination.cpp ~~.~.~.~1$	115
RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/RoadSignAPI.cpp	115
RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/ClassifiedSignsGrouper.cpp 1	115
RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/DetectionBasedImageSlicer.cpp 1	115
$Road Sign API\_Standal one\_Tensor flow/source\_files/Road Sign API/Filters/Mobile net V2Road Sign Classificator.$	<i>ب</i>
cpp 1	115
RoadSignAPI_Standalone_Tensorflow/source_files/RoadSignAPI/Filters/RoadSignDuplicationDeleter.	
cpp	115
RoadSignAPI Standalone Tensorflow/source files/RoadSignAPI/Filters/SSDLiteRoadSignDetector.cop 1	115

## **Namespace Documentation**

6.1 anonymous\_namespace{TensorflowAndroidJNIUtils.cpp} Namespace Reference

#### Classes

• class IfstreamInputStream

### 6.2 FilterManagementLibrary Namespace Reference

Everything the FilterManagementLibrary provides.

#### **Namespaces**

PipeSystem

Everything that has to do with the pipes and filters part of the library.

• TFIntegration

#### Classes

· class Logger

Logger providing Desktop Linux <-> Android platform independent logging.

- class TensorflowAndroidJNIUtils
- class TensorflowOpenCVUtils
- · class Utilities

Providing some basic utilities, like checking if a file exists.

#### 6.2.1 Detailed Description

Everything the FilterManagementLibrary provides.

### 6.3 FilterManagementLibrary::PipeSystem Namespace Reference

Everything that has to do with the pipes and filters part of the library.

#### Classes

· class PipeFilter

Base class were all filters are derived from.

• struct PipeRegisteredFilters

Template struct.

struct PipeWorkingDataSet

Template structfor PipeWorkingDataSets.

· class ProcessingPipeline

Pipeline of the pipes and filters architecture.

· class TFNNBasedPipeFilter

Abstract template class to derive Tensorflow based filters from.

#### 6.3.1 Detailed Description

Everything that has to do with the pipes and filters part of the library.

### 6.4 FilterManagementLibrary::TFIntegration Namespace Reference

#### **Classes**

- · class TensorflowNNInstance
- · class TensorflowNNInstanceClassifier
- struct TensorflowNNModelDescription
- class TensorflowResultContainer

#### 6.4.1 Detailed Description

Provides functionalitys to load Tensorflow models and use them to do calculations / perform operations, interprete their output and so on.

### 6.5 google Namespace Reference

#### **Namespaces**

protobuf

### 6.6 google::protobuf Namespace Reference

### 6.7 RoadSignAPI Namespace Reference

#### **Classes**

- · class ClassifiedSignsGrouper
- class DetectedSignCombination
- struct DetectedSignDescriptor
- class DetectionBasedImageSlicer
- class MobilenetV2RoadSignClassificator
- class RoadSignAPI
- class RoadSignDuplicationDeleter
- struct RSAPIPipeRegisteredFilters
- struct RSAPIWorkingDataSet
- class SSDLiteRoadSignDetector

### 6.8 tensorflow Namespace Reference

#### **Namespaces**

· android

### 6.9 tensorflow::android Namespace Reference

#### Classes

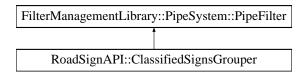
• class LimitingFileInputStream

## **Class Documentation**

### 7.1 RoadSignAPI::ClassifiedSignsGrouper Class Reference

#include <ClassifiedSignsGrouper.h>

Inheritance diagram for RoadSignAPI::ClassifiedSignsGrouper:



#### **Private Member Functions**

• bool initByPipeSetup ()

Initializes the filter.

• bool process ()

The process function of this filter.

#### **Private Attributes**

- RSAPIWorkingDataSet \* castedWorkingDataSet
- int imageWidth
- float horizontalRangePercentage = 0.025

#### **Additional Inherited Members**

#### 7.1.1 Detailed Description

For the user, it is nice to know which signs are mounted onto the same pole, so they are logically grouped. This has a direct effect on the scope of some signs (i.e. signs indicating danger can limit the validity of speed limits for the duration of a dangerous right turn etc.). This Filter aims to group those signs into one DetectedSignCombination}. For this, it estimates where the pole of a sign is located (for more, please refer to the process() function and to DetectedSignCombination}.

16 Class Documentation

#### 7.1.2 Member Function Documentation

#### 7.1.2.1 initByPipeSetup()

```
bool RoadSignAPI::ClassifiedSignsGrouper::initByPipeSetup ( ) [private], [virtual]
```

Initializes the filter.

As this filter does not need any specific environment variables, nothing really exciting is happenning here..

Returns

true, always

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References castedWorkingDataSet, FilterManagementLibrary::PipeSystem::PipeFilter::pipeWorkingDataSet, and FilterManagementLibrary::Logger::printfln().

#### 7.1.2.2 process()

```
bool RoadSignAPI::ClassifiedSignsGrouper::process ( ) [private], [virtual]
```

The process function of this filter.

Iterates through all signs that were successfully classified and whose IDs were thus added to the classifier ← Approved Signs list of the RSAPIWorking DataSet}. It calculates an estimated Position of the pole of a sign (by using the middle of the X coordinates of the edges of the box in which a sign is contained) and adding a threshold to it defined in the local member variable horizontal Range Percentage. which should not be to big! For the first sign in the list, a new Detected Sign Combination} will be created. For all other signs, they are checked against all existing sign Combinations (so for the second sign, there is exactly one!). If their X coordinates match the estimated pole area of a Detected Sign Combination), than they are added to it (one sign is only added to one combination!). If no matching combination were found, a new one is created.

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References RoadSignAPI::DetectedSignCombination::addDetectedSign(), castedWorkingDataSet, RoadSign API::RSAPIWorkingDataSet::classifierApprovedSigns, RoadSignAPI::RSAPIWorkingDataSet::detectedSigns, RoadSignAPI::DetectedSignCombination Combinations, RoadSignAPI::RSAPIWorkingDataSet::detectedSigns, RoadSignAPI::DetectedSignCombination Combination (), horizontalRangePercentage, imageWidth, FilterManagementLibrary::Pipe System::PipeFilter::indicateProcessingFinished(), RoadSignAPI::DetectedSignDescriptor::lowerRight, RoadSign API::RSAPIWorkingDataSet::originalImageWidth, and RoadSignAPI::DetectedSignDescriptor::upperLeft.

#### 7.1.3 Member Data Documentation

#### 7.1.3.1 castedWorkingDataSet

```
RSAPIWorkingDataSet* RoadSignAPI::ClassifiedSignsGrouper::castedWorkingDataSet [private]
```

Just a pointer casted from PipeWorkingDataSet\* to RSAPIWorkingDataSet\*, so we just don't have to do the casting every time we need it ;)

#### 7.1.3.2 horizontalRangePercentage

```
float RoadSignAPI::ClassifiedSignsGrouper::horizontalRangePercentage = 0.025 [private]
```

We try to find out which signs are on one pole. For this, we take the middle of the box of the detected sign and add a threshold in +X and -X direction, to guess where the pole is. This variable specifies how big this threshold is (i.e. how wide the area is in which the pole is expected). The percentage refers to the total width of the original BG $\leftarrow$  RImage. If the width for example is 1280, a horizontal Percentage Range of 10% would mean 128 pixels in EACH direction (which, of course would be a lot, so we would use something around 1-5%...). If we would exceed the image boundaries while applying this value, we of course would adjust it to stay within the boundaries (we assert that the range always starts where x > 0 and ends where x < original BGRImage Width).

#### 7.1.3.3 imageWidth

```
int RoadSignAPI::ClassifiedSignsGrouper::imageWidth [private]
```

We store a local copy to the originalImageWidth of the working data set (just for easier access and aesthetically more pleasing code ;) )

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/ClassifiedSignsGrouper.h
- RoadSignAPI Standalone Tensorflow/source files/RoadSignAPI/Filters/ClassifiedSignsGrouper.cpp

#### 7.2 RoadSignAPI::DetectedSignCombination Class Reference

#include <DetectedSignCombination.h>

#### **Public Member Functions**

void addDetectedSign (DetectedSignDescriptor detectedSign)

Adds a sign to the list of signs of this combination.

• int getDetectedSignsAmount () const

Returns the amount of signs contained by this combination.

• int getGestimatedPolePositionX () const

Returns the estimated pole position X coordinate of this sign combination.

• const std::vector< DetectedSignDescriptor > \* getSignsInCombination () const

Returns a vector containing all Signs of this combination.

DetectedSignCombination ()

Constructor of DetectedSignCombination.

∼DetectedSignCombination ()

Standard Destructor of DetectedSignCombination.

18 Class Documentation

#### **Private Member Functions**

void insertSorted (DetectedSignDescriptor \*detectedSign)

Inserts a detectedSign at the right position of this combination.

#### **Private Attributes**

- std::vector< DetectedSignDescriptor > signs
- int estimatedPolePositionX

#### 7.2.1 Detailed Description

A class combining multiple DetectedSignDescriptors into one object. The idea is that the RoadSignAPI not only detects and classifies the signs, but also tries to determine which signs form one combination, i.e. which signs are on one pole (this is done by ClassifiedSigsGrouper filter, NOT in this class!). All signs on one pole will be stored in a DetectedSignCombination, which also calculates the estimated X coordinate in the image, where the pole is expected to be by calculating the average of the center X coordinate of all signs in the combination. Furthermore, the signs will be sorted by their y coordinates in from top to bottom (top has the lowest y value, as the coordinate origin is at the top left), what insertion sort is used for (as the signs will be added to the combination one by one).

#### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 DetectedSignCombination()

```
{\tt RoadSignAPI::} {\tt DetectedSignCombination::} {\tt DetectedSignCombination} \ \ (\ \ )
```

Constructor of DetectedSignCombination.

Intializes all member variables and constructs a new object of type DetectedSignCombination. Does not do any other specific tasks.

References addDetectedSign(), and signs.

#### 7.2.2.2 ∼DetectedSignCombination()

```
{\tt RoadSignAPI::} Detected Sign Combination:: {\tt \sim} Detected Sign Combination \ (\ )
```

Standard Destructor of DetectedSignCombination.

Standard Destructor of DetectedSignCombination, does not do any specific tasks.

#### 7.2.3 Member Function Documentation

# 7.2.3.1 addDetectedSign()

Adds a sign to the list of signs of this combination.

Uses insertSorted(...) to add a detected sign to the list of signs in the this combination, so that the list is sorted by the y values of the signs (from top to bottom)

References estimatedPolePositionX, i, insertSorted(), RoadSignAPI::DetectedSignDescriptor::lowerRight, signs, and RoadSignAPI::DetectedSignDescriptor::upperLeft.

# 7.2.3.2 getDetectedSignsAmount()

```
\verb|int RoadSignAPI::DetectedSignCombination::getDetectedSignsAmount () const|\\
```

Returns the amount of signs contained by this combination.

### Returns

int Size of the list (= amount of entries in the list) containing all DetectedSignDescriptor}s of this combination.

References signs.

### 7.2.3.3 getGestimatedPolePositionX()

```
int RoadSignAPI::DetectedSignCombination::getGestimatedPolePositionX ( ) const
```

Returns the estimated pole position X coordinate of this sign combination.

The estimated pole position X is calculated by forming the average of the center X coordinate (middle of the box in which a sign is contained).

References estimatedPolePositionX, and getSignsInCombination().

### 7.2.3.4 getSignsInCombination()

```
\verb|const| std::vector<| RoadSignAPI::DetectedSignDescriptor| > * RoadSignAPI::DetectedSignCombination \\ \leftarrow ::getSignsInCombination ( ) const| \\
```

Returns a vector containing all Signs of this combination.

All DetectedSignCombination}s which were added to this combination will be added to a list, which can be retrieved using this function. This list is sorted by the y values of the signs from top to bottom.

References signs.

# 7.2.3.5 insertSorted()

```
\label{lem:condition} \begin{tabular}{ll} void & RoadSignAPI::DetectedSignCombination::insertSorted ( \\ & DetectedSignDescriptor * detectedSign ) & [private] \end{tabular}
```

Inserts a detectedSign at the right position of this combination.

Uses insertion sort to add the detected sign to the list of signs in the this combination, so that the list is sorted by the y values of the signs (from top to bottom).

References i, signs, and RoadSignAPI::DetectedSignDescriptor::upperLeft.

# 7.2.4 Member Data Documentation

#### 7.2.4.1 estimatedPolePositionX

```
int RoadSignAPI::DetectedSignCombination::estimatedPolePositionX [private]
```

To find out which sign belongs to a group of detected signs (a sign combination), i.e. signs on one pole, an external logic may be interested to know where the "middle" (in x direction) of the signs is. For this, we take all X positions of the signs in a combination, sum them up and calculate the average.

# 7.2.4.2 signs

```
std::vector<DetectedSignDescriptor> RoadSignAPI::DetectedSignCombination::signs [private]
```

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/DetectedSignCombination.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/DetectedSignCombination.cpp

# 7.3 RoadSignAPI::DetectedSignDescriptor Struct Reference

#include <DetectedSignDescriptor.h>

# **Public Attributes**

- cv::Point upperLeft
- cv::Point lowerRight
- int detectionPredictedClassID
- · float detectorConfidence
- int classifierApprovedClassID
- · float classifierConfidence

# 7.3.1 Detailed Description

Struct describing position and ID of a roadsign detected and classified in an image. It also contains the confidence values of the detector and the classificator. Mainly used in RSAPIWorkingDataSet.

# 7.3.2 Member Data Documentation

# 7.3.2.1 classifierApprovedClassID

int RoadSignAPI::DetectedSignDescriptor::classifierApprovedClassID

Class ID the classifier determined.

### 7.3.2.2 classifierConfidence

 $\verb|float RoadSignAPI::DetectedSignDescriptor::classifierConfidence|\\$ 

Confidence of the classifier for the classification of this detection. Will be filled when the DetectedSignDescriptor is passed to the MobilenetV2RoadSignClassificator.

### 7.3.2.3 detectionPredictedClassID

int RoadSignAPI::DetectedSignDescriptor::detectionPredictedClassID

Class ID the detection filter predicts.

# 7.3.2.4 detectorConfidence

float RoadSignAPI::DetectedSignDescriptor::detectorConfidence

Confidence of the detector for this detection.

# 7.3.2.5 lowerRight

cv::Point RoadSignAPI::DetectedSignDescriptor::lowerRight

Lower right position of the detection of the roadsign (mapped to the coordinates of the original image that was provied, not to the scaled image the detector uses).

### 7.3.2.6 upperLeft

```
cv::Point RoadSignAPI::DetectedSignDescriptor::upperLeft
```

Upper left position of the detection of the roadsign (mapped to the coordinates of the original image that was provied, not to the scaled image the detector uses).

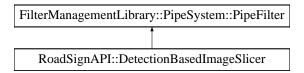
The documentation for this struct was generated from the following file:

RoadSignAPI Standalone Tensorflow/header files/RoadSignAPI/DetectedSignDescriptor.h

# 7.4 RoadSignAPI::DetectionBasedImageSlicer Class Reference

```
#include <DetectionBasedImageSlicer.h>
```

Inheritance diagram for RoadSignAPI::DetectionBasedImageSlicer:



### **Private Member Functions**

- void expandBox (cv::Point \*upperLeft, cv::Point \*lowerRight)
- bool initByPipeSetup ()

Initializes the filter.

• bool process ()

The process function of this filter.

# **Private Attributes**

- RSAPIWorkingDataSet \* castedWorkingDataSet
- float expandPercentage = 0.1
- int minExpandPixels = 5

# **Additional Inherited Members**

# 7.4.1 Detailed Description

This filter will use the detections the SSDLiteRoadSignDetector provides to cut the originalBGRImage into smaller pieces, which the MobilenetV2RoadSignClassificator can use to classify the detections.

The dataset which was used to train the network models consists of images where each road sign was labeled (using a box which enclosed the sign). These boxes were set in such a way that they perfectly enclose the signs. However to train the classificator, we expaneded each box in every direction (x, -x, y, -y) by 10% of the box width / height or at least 5 pixels (depending on which value is bigger), in order to achieve a greater independence from the background of the road sign and noise. Thus the network learned the road sign's class is independent from it's background. To achieve optimal results, we have to do the same expansion here instead of only crop the images from the given detection before feeding them into the classificator. We use the same values here, stored in expandPercentage (10%) and minExpandPixels (5).

# 7.4.2 Member Function Documentation

# 7.4.2.1 expandBox()

Expands the given coordinates of the given box (rectangle) by expandPercentage (see member variables) in each direction, if the amount of pixels to expand is bigger than minExpandPixels (see member variables), otherwise minExpandPixels will be used as amount of pixels to expand. If we exceed the border of the originalBGRImage (see RSAPIWorkingDataSet) at any direction (e.g. coordinates < 0 or > width or height), use set the highest / lowest possible value (so concerning the example, 0 or width / height).

### **Parameters**

cv::Point	*upperLeft Pointer to the upperLeft coordinates of the box which shall be expanded. We use copies from the corresponding values of the RSAPIWorkingDataSet, because we do not want to expand the original, close-fitting boxes.
cv::Point	*lowerRight Pointer to the lowerRight coordinates of the box which shall be expanded. We use copies from the corresponding values of the RSAPIWorkingDataSet, because we do not want to expand the original, close-fitting boxes.

References castedWorkingDataSet, expandPercentage, minExpandPixels, RoadSignAPI::RSAPIWorkingData  $\leftarrow$  Set::originalImageHeight, and RoadSignAPI::RSAPIWorkingDataSet::originalImageWidth.

# 7.4.2.2 initByPipeSetup()

```
bool RoadSignAPI::DetectionBasedImageSlicer::initByPipeSetup ( ) [private], [virtual]
```

Initializes the filter.

As this filter does not need any specific environment variables, nothing really exciting is happenning here..

# Returns

true, always

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References castedWorkingDataSet, FilterManagementLibrary::PipeSystem::PipeFilter::pipeWorkingDataSet, and FilterManagementLibrary::Logger::printfln().

# 7.4.2.3 process()

```
bool RoadSignAPI::DetectionBasedImageSlicer::process ( ) [private], [virtual]
```

The process function of this filter.

This function will be called by the ProcessingPipeline. Here the road signs detected by SSDRoadSignDetector will be cut out into smaller images (which only contain the sign), which will then be fed into the MobilenetV2RoadSign← Classificator. To be fast, we assume that the coordinates calculated by the previous filter are correct!

Returns

true, always

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References castedWorkingDataSet, RoadSignAPI::RSAPIWorkingDataSet::cutOutImages, RoadSignAPI::R SAPIWorkingDataSet::detectedSigns, expandBox(), FilterManagementLibrary::PipeSystem::PipeFilter::invoke Next(), RoadSignAPI::DetectedSignDescriptor::lowerRight, RoadSignAPI::RSAPIWorkingDataSet::originalBGR Image, FilterManagementLibrary::PipeSystem::PipeFilter::pipeRegisteredFilters, and RoadSignAPI::Detected SignDescriptor::upperLeft.

### 7.4.3 Member Data Documentation

### 7.4.3.1 castedWorkingDataSet

```
RSAPIWorkingDataSet* RoadSignAPI::DetectionBasedImageSlicer::castedWorkingDataSet [private]
```

Just a pointer casted from PipeWorkingDataSet\* to RSAPIWorkingDataSet\*, so we just don't have to do the casting every time we need it ;)

# 7.4.3.2 expandPercentage

```
float RoadSignAPI::DetectionBasedImageSlicer::expandPercentage = 0.1 [private]
```

This is the value we expand each box around a detected sign in any direction (x, -x, y, -x). Please refer to the class description for an explanation why we (have to) do this.

### 7.4.3.3 minExpandPixels

```
int RoadSignAPI::DetectionBasedImageSlicer::minExpandPixels = 5 [private]
```

This is the minimum amount of pixels we expand each box around a detected sign in any direction (x, -x, y, -x). It is used to expand the box, if the value calculated using expandPercentage is to small: We want to expand AT LEAST 5 pixels (or more, respectively). Please refer to the class description for an explanation why we (have to) do this.

The documentation for this class was generated from the following files:

- RoadSignAPI Standalone Tensorflow/header files/RoadSignAPI/Filters/DetectionBasedImageSlicer.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/DetectionBasedImageSlicer.cpp

# 7.5 anonymous\_namespace{TensorflowAndroidJNIUtils.cpp}::IfstreamInputStream Class Reference

 $Inheritance\ diagram\ for\ an onymous\_namespace \{TensorflowAndroidJNIUtils.cpp\} :: If streamInputStream:$ 



# **Public Member Functions**

- IfstreamInputStream (const std::string &file name)
- ∼IfstreamInputStream ()
- int Read (void \*buffer, int size)

# **Private Attributes**

std::ifstream ifs

# 7.5.1 Constructor & Destructor Documentation

# 7.5.1.1 IfstreamInputStream()

### 7.5.1.2 ~IfstreamInputStream()

 $anonymous\_namespace{TensorflowAndroidJNIUtils.cpp}:: If streamInputStream:: \sim If streamInputStream ( ) [inline]$ 

# 7.5.2 Member Function Documentation

# 7.5.2.1 Read()

```
int anonymous_namespace{TensorflowAndroidJNIUtils.cpp}::IfstreamInputStream::Read ( void * buffer, int size ) [inline]
```

# 7.5.3 Member Data Documentation

7.5.3.1 ifs

 $\verb|std::ifstream| anonymous_namespace{TensorflowAndroidJNIUtils.cpp}|::IfstreamInputStream::ifs\_\leftarrow [private]|$ 

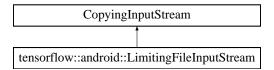
The documentation for this class was generated from the following file:

RoadSignAPI Standalone Tensorflow/source files/FilterManagementLibrary/TensorflowAndroidJNIUtils.cpp

# 7.6 tensorflow::android::LimitingFileInputStream Class Reference

#include <limiting\_file\_input\_stream.h>

Inheritance diagram for tensorflow::android::LimitingFileInputStream:



# **Public Member Functions**

- · LimitingFileInputStream (int fd, int limit)
- ∼LimitingFileInputStream ()
- int Read (void \*buffer, int size)
- int Skip (int count)

# **Private Attributes**

- · const int fd\_
- int bytes\_left\_
- int errno\_ = 0

# 7.6.1 Constructor & Destructor Documentation

# 7.6.1.1 LimitingFileInputStream()

# 7.6.1.2 ~LimitingFileInputStream()

```
tensorflow::android::LimitingFileInputStream::~LimitingFileInputStream ( ) [inline]
```

# 7.6.2 Member Function Documentation

# 7.6.2.1 Read()

References bytes\_left\_, errno\_, and fd\_.

# 7.6.2.2 Skip()

References bytes\_left\_, and fd\_.

# 7.6.3 Member Data Documentation

# 7.6.3.1 bytes\_left\_

```
int tensorflow::android::LimitingFileInputStream::bytes_left_ [private]
```

### 7.6.3.2 errno\_

```
int tensorflow::android::LimitingFileInputStream::errno_ = 0 [private]
```

### 7.6.3.3 fd

```
const int tensorflow::android::LimitingFileInputStream::fd_ [private]
```

The documentation for this class was generated from the following file:

• RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/limiting\_file\_input\_stream.h

# 7.7 FilterManagementLibrary::Logger Class Reference

Logger providing Desktop Linux <-> Android platform independent logging.

```
#include <Logger.h>
```

# **Static Public Member Functions**

- static void printfln (const char \*format,...)
   prints a new line to the console
- static void setLogTag (std::string logTag)

sets the logTag of the logger

# **Static Private Member Functions**

• static void getTimeString (std::string \*timeStr)

In-class helper function to get a string containing the current time.

# **Static Private Attributes**

• static std::string logTag = "FilterManagementLibrary"

# 7.7.1 Detailed Description

Logger providing Desktop Linux <-> Android platform independent logging.

While compiling, it's checked if the library is built for Android or any other architecture. The logger class provides platform indepent logging by using preprocessor directives to only compile those functions that are needed on the given architecture. On Linux for example, the logger prints to stdout. On Android however, the logging library is used and the logger prints to the logcat output console.

# 7.7.2 Member Function Documentation

# 7.7.2.1 getTimeString()

In-class helper function to get a string containing the current time.

Class intern helper function. It provides a string containing the current time as follows: hh:mm::ss hh being the hour mm being the minute ss being the second

# **Parameters**

std::string	*timeStr pointer to a string to wich the time string will be written to.

# Returns

void.

# 7.7.2.2 printfln()

prints a new line to the console

Takes a format c-string and a variable amount of parameters and formats an string accordingly. This string will be printed prefixed with the current time and the specified logTag.

### **Parameters**

const	char* format c-string containing the string which shall be printed. May contain conversion specifiers like d, f etc.
	variable of amount of parameters used to format the conversion specifiers.

References getTimeString(), and logTag.

# 7.7.2.3 setLogTag()

sets the logTag of the logger

The logTag is a prefix which will be printed when using printfln after the time string.

# **Parameters**

std::string logTag string containing the logTag
---

# Returns

void

References logTag.

# 7.7.3 Member Data Documentation

### 7.7.3.1 logTag

```
std::string FilterManagementLibrary::Logger::logTag = "FilterManagementLibrary" [static],
[private]
```

A tag to display in every print. By default it is set to "FilterManagementLibrary". With this, every use of printfln(...) would look like this: [00:38:12 - FilterManagementLibrary]: Logger print test. Notice that every print is prefixed with the current time for convenience.

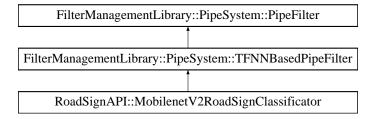
The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Logger.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/Logger.cpp

# 7.8 RoadSignAPI::MobilenetV2RoadSignClassificator Class Reference

#include <MobilenetV2RoadSignClassificator.h>

Inheritance diagram for RoadSignAPI::MobilenetV2RoadSignClassificator:



# **Public Member Functions**

 MobilenetV2RoadSignClassificator (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription, int numThreads, AAssetManager \*const assetManager)

Constructor of MobilenetV2RoadSignClassificator for Android environments. It basically just calls the TFNNBased PipeFilter super constructor. Refer to it for a more detailed description. Under Android, the RoadSignAPI expects the model files of our neuronal networks to be placed in the assets folder of the app. Thus, this constructor needs to be passed a pointer to an AssetManager (from java), which is MANDATORY for the RoadSignAPI under Android!

MobilenetV2RoadSignClassificator (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription, int numThreads)

Constructor of MobilenetV2RoadSignClassificator for non Android environments. It basically just calls the TFNN← BasedPipeFilter super constructor. Refer to it for a more detailed description.

### **Private Member Functions**

bool initByPipeSetup ()

Initializes the filter.

• bool process ()

Uses the neuronal network to classify the previous detected signs.

void onNNEvaluationFinished (const FilterManagementLibrary::TFIntegration::TensorflowResultContainer resultContainer)

Callback, will be called when the network finished it's prediction.

void applyImageVectorFromOpenCVMat (cv::Mat \*mat)

Uses TensorflowOpenCVUtils to apply a mat as input to the network.

bool isInUnwantedClasses (int classID) const

Checks if the given class ID is in the list of unwanted ID's.

### **Private Attributes**

- RSAPIWorkingDataSet \* castedWorkingDataSet
- · int nnModelInputHeight
- · int nnModelInputWidth
- float threshold = 0.95
- cv::Mat currentCutOutImage
- std::pair< float, int > currentRecognition
- bool validRecognition = false
- AAssetManager \*const assetManager

# **Additional Inherited Members**

# 7.8.1 Detailed Description

This filter is based on TFNNBasedPipeFilter and uses MobilenetV2 for (roadsign) classification.

# 7.8.2 Constructor & Destructor Documentation

# 7.8.2.1 MobilenetV2RoadSignClassificator() [1/2]

Constructor of MobilenetV2RoadSignClassificator for Android environments. It basically just calls the TFNNBased PipeFilter super constructor. Refer to it for a more detailed description. Under Android, the RoadSignAPI expects the model files of our neuronal networks to be placed in the assets folder of the app. Thus, this constructor needs to be passed a pointer to an AssetManager (from java), which is MANDATORY for the RoadSignAPI under Android!

All other class-member variables will be initialized to their default values.

### **Parameters**

TensorflowNNModelDescription	nnModelDescription a description of the Tensorflow model which will be used by the filter.
int	numThreads number of threads Tensorflow is allowed to use for computation.
AAssetManager*	const assetManager pointer to an Android Asset Manager which needs to be passed from the Java part of the Android App by any means. Manadatory to load the neuronal network models from the assets directory of the app.

# 7.8.2.2 MobilenetV2RoadSignClassificator() [2/2]

```
\label{lem:RoadSignAPI::MobilenetV2RoadSignClassificator::MobilenetV2RoadSignClassificator ( \\ FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription \\ nnModel \hookleftarrow \\ Description, \\ int \\ numThreads )
```

Constructor of MobilenetV2RoadSignClassificator for non Android environments. It basically just calls the TFNN← BasedPipeFilter super constructor. Refer to it for a more detailed description.

All class-member variables will be initialized to their default values.

# **Parameters**

TensorflowNNModelDescription	nnModelDescription a description of the Tensorflow model which will be used by the filter.
int	numThreads number of threads Tensorflow is allowed to use for computation.

# 7.8.3 Member Function Documentation

# 7.8.3.1 applyImageVectorFromOpenCVMat()

Uses TensorflowOpenCVUtils to apply a mat as input to the network.

# **Parameters**

cv::Mat	*mat pointer to an OpenCV Mat which shall be used as input.
---------	---

References FilterManagementLibrary::TensorflowOpenCVUtils::fastApplyCVMatOnInputTensorFloat(), Filter  $\leftarrow$  ManagementLibrary::PipeSystem::TFNNBasedPipeFilter::getNNInputTensor(), and FilterManagementLibrary:: $\leftarrow$ 

PipeSystem::TFNNBasedPipeFilter::getNNModelDescription().

### 7.8.3.2 initByPipeSetup()

```
bool RoadSignAPI::MobilenetV2RoadSignClassificator::initByPipeSetup ( ) [private], [virtual]
```

Initializes the filter.

Will be called by ProcessingPipeline. Here, basically just the neuronal network model will be loaded. For this, the setupModelFromFile() or setupModelFromAssets() function will be called accordingly, depending whether we are under an Android environment or not. Refer to TFNNBasedPipeFilter for a more detailed description.

# Returns

true, if the model could be loaded successfully, false otherwise

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References assetManager, castedWorkingDataSet, currentCutOutImage, FilterManagementLibrary::Pipe System::TFNNBasedPipeFilter::getNNModelDescription(), FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::inputHeight, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input Width, nnModelInputHeight, nnModelInputWidth, FilterManagementLibrary::PipeSystem::PipeFilter::pipeWorking DataSet, FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::setupModelFromAssets(), and FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::setupModel FromFile().

# 7.8.3.3 isInUnwantedClasses()

Checks if the given class ID is in the list of unwanted ID's.

### **Parameters**

int classID ID of the class which shall be checked.

### Returns

true if the given ID is in the list of unwanted class ID's (i.e. misc classes), false otherwise.

References on NNEvaluation Finished().

# 7.8.3.4 onNNEvaluationFinished()

```
\label{lem:container} \begin{tabular}{ll} void RoadSignAPI::MobilenetV2RoadSignClassificator::onNNEvaluationFinished ( \\ const FilterManagementLibrary::TFIntegration::TensorflowResultContainer \ result \leftarrow \\ Container \ ) \ [private], \ [virtual] \end{tabular}
```

Callback, will be called when the network finished it's prediction.

In process(), the TFNNBasedPipeFilter's (super class) evaluteInputVectorByNN() will be called. If it was successfull, this callback will be called and passed a TensorflowResultContainer with the result of the inference of the neuronal network model. Here we will check the prediction and find the highest confidence prediction, which exceeds a certain threshold. The prediction itself will be examined in process() itself (remember, this is a sync function! It will be called immediately after evaluteInputVectorByNN() was called and then will return to process() again).

#### **Parameters**

const	TensorflowResultContainer resultContainer contains the results (in Tensorflow tensors) of the network
	prediction.

Implements FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter.

References applyImageVectorFromOpenCVMat(), currentRecognition, i, threshold, and validRecognition.

### 7.8.3.5 process()

```
bool RoadSignAPI::MobilenetV2RoadSignClassificator::process ( ) [private], [virtual]
```

Uses the neuronal network to classify the previous detected signs.

Iterates over all the images the DetectionBasedImageSlicer generated from the originalBGRImage and uses the neuronal network model to classify them (uses evaluateInputVectorByNN() of TFNNBasedPipeFilter super class. Will set the class ID and the confidence accordingly. All classes which are unwanted (i.e. misc classes) will be filtered out, in other words they won't be added to classifierApprovedSigns of RSAPIWorkingDataSet's.

### Returns

true if evaluateInputVectorByNN() return true, false otherwise (does NOT return false if no signs could be classified!)

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References applyImageVectorFromOpenCVMat(), castedWorkingDataSet, RoadSignAPI::RSAPIWorkingDataSet::classifierApprovedSigns, currentCutOutImage, currentRecognition, RoadSignAPI::RSAPIWorkingDataSet::cutOutImages, FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::evaluateInputVectorByNN(), i, FilterManagementLibrary::PipeSystem::PipeFilter::indicateProcessingFinished(), FilterManagementLibrary::PipeSystem::PipeFilter::invokeNext(), isInUnwantedClasses(), FilterManagementLibrary::PipeSystem::PipeFilter::inpipeRegisteredFilters, and validRecognition.

# 7.8.4 Member Data Documentation

### 7.8.4.1 assetManager

AAssetManager\* const RoadSignAPI::MobilenetV2RoadSignClassificator::assetManager [private]

A pointer to an AssetManager which can be passed via the constructor. We need it to be able to load model files from the Android assets folder, which we want to do so the files do not have to be placed in the normal user space.

# 7.8.4.2 castedWorkingDataSet

RSAPIWorkingDataSet\* RoadSignAPI::MobilenetV2RoadSignClassificator::castedWorkingDataSet [private]

Just a pointer casted from PipeWorkingDataSet\* to RSAPIWorkingDataSet\*, so we just don't have to do the casting every time we need it ;)

# 7.8.4.3 currentCutOutImage

cv::Mat RoadSignAPI::MobilenetV2RoadSignClassificator::currentCutOutImage [private]

A cv::Mat were we will temporarily store a resized image matrix while we iterate over {

#### See also

RSAPIWorkingDataSet}::cutOutImages (we need to resize them to the size our network model expects).

# 7.8.4.4 currentRecognition

std::pair<float, int> RoadSignAPI::MobilenetV2RoadSignClassificator::currentRecognition [private]

Inter-class copy of top rated recognition we determined in on onNNEvaluationFinished(...). You may refer to the process() function of this class for a description of where this is used.

### 7.8.4.5 nnModelInputHeight

int RoadSignAPI::MobilenetV2RoadSignClassificator::nnModelInputHeight [private]

Just a copy from the input height provided in the TensorflowNNModelDescription of the underlaying TensorflowN← NInstance.

# 7.8.4.6 nnModelInputWidth

int RoadSignAPI::MobilenetV2RoadSignClassificator::nnModelInputWidth [private]

Just a copy from the input height provided in the TensorflowNNModelDescription of the underlying TensorflowNN← Instance.

#### 7.8.4.7 threshold

```
float RoadSignAPI::MobilenetV2RoadSignClassificator::threshold = 0.95 [private]
```

Threshold for the output confidence of the neuronal network model. Means a result needs to achieve a confidence >= threshold to be candidate for list of top N predictions.

# 7.8.4.8 validRecognition

```
bool RoadSignAPI::MobilenetV2RoadSignClassificator::validRecognition = false [private]
```

If none of the outputs of the network beats the threshold, we currentRecognition may store the result of a previous classification, which would lead to wrong results. This variable states if there was a valid recognition in the last evaluateImageByNN() call.

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/MobilenetV2RoadSignClassificator.
   h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/MobilenetV2RoadSignClassificator. ← cpp

# 7.9 FilterManagementLibrary::PipeSystem::PipeFilter Class Reference

Base class were all filters are derived from.

```
#include <PipeFilter.h>
```

Inheritance diagram for FilterManagementLibrary::PipeSystem::PipeFilter:



# **Public Member Functions**

• PipeFilter ()

Standard constructor of PipeFilter.

• int getFilterID () const

Returns the ID of the filter.

• int getNextDesiredFilter () const

Returns the ID of the filter which shall be invoked next.

• bool hasMarkedNextDesiredFilter () const

Returns whether a filter which shall be invoked next has been marked.

bool hasMarkedProcessingFinished () const

Returns whether the processing on the current DataSet is finished.

virtual ∼PipeFilter ()

Standard destructor of PipeFilter. Has no functionality yet.

### **Protected Member Functions**

- virtual bool initByPipeSetup ()=0
- virtual bool process ()=0
- · void indicateProcessingFinished ()

Function for derived filters to indicate all processing is done.

- void invokeNext (const int filterID)
- · void reset ()

Function to reset basic variables of the filter, called by the pipe.

# **Protected Attributes**

- PipeWorkingDataSet \* pipeWorkingDataSet
- PipeRegisteredFilters \* pipeRegisteredFilters

# **Private Member Functions**

void setCredentials (int filterID, PipeRegisteredFilters \*pipeRegisteredFilters, PipeWorkingDataSet \*pipe←
 WorkingDataSet)

Function which is used by ProcessingPipeline to hand in some variables.

# **Private Attributes**

- · int filterID
- int nextDesiredFilter = -1
- bool markedNextDesiredFilter = false
- bool markedProcessingFinished = false

# **Friends**

· class ProcessingPipeline

# 7.9.1 Detailed Description

Base class were all filters are derived from.

A filter in terms of the pipes and filters architecture is a single processing step. Originally, each filter has a data input and a data output. In our case, however, the data input and output is mapped to a single object (the data object, see PipeWorkingDataSet) allocated, on the heap, on which all operations are performed. In each processing step, the filter transforms the specific part of the data object that affects him. For this, each filter implements a function process() which defines the (conversion) functionality of the filter.

During each procession, parts of the data object may be added, edited or removed completely. The type of conversion is determined by each filter. As each filter knows the structure of the data object at compile time, compatibility between each filters of a pipe is guaranteed! This means, each filter get's at least those variables of the data object it expects.

# 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 PipeFilter()

```
FilterManagementLibrary::PipeSystem::PipeFilter::PipeFilter ( )
```

Standard constructor of PipeFilter.

Initialises all class member variables to their default values.

# 7.9.2.2 $\sim$ PipeFilter()

```
FilterManagementLibrary::PipeSystem::PipeFilter::~PipeFilter ( ) [virtual]
```

Standard destructor of PipeFilter. Has no functionality yet.

# 7.9.3 Member Function Documentation

# 7.9.3.1 getFilterID()

```
int FilterManagementLibrary::PipeSystem::PipeFilter::getFilterID ( ) const
```

Returns the ID of the filter.

### Returns

int filterID the ID of the filter.

References filterID, and getNextDesiredFilter().

# 7.9.3.2 getNextDesiredFilter()

```
\verb|intFilterManagementLibrary::PipeSystem::PipeFilter::getNextDesiredFilter () constants | Constants
```

Returns the ID of the filter which shall be invoked next.

Whenever a Filter returns true in it's process() function, it needs to inform the ProcessingPipeline about what to do next (is the processing finished, or is there a next filter which shall be invoked?). For the latter, the ID of the next filter will be set by invokeNextFilter(...) and can be derived by the ProcessingPipeline using this getter-function.

# Returns

int ID of the filter which shall be invoked next.

References hasMarkedNextDesiredFilter(), and nextDesiredFilter.

### 7.9.3.3 hasMarkedNextDesiredFilter()

```
bool FilterManagementLibrary::PipeSystem::PipeFilter::hasMarkedNextDesiredFilter ( ) const
```

Returns whether a filter which shall be invoked next has been marked.

Whenever a Filter returns true in it's process() function, it needs to inform the ProcessingPipeline about what to do next (is the processing finished, or is there a next filter which shall be invoked?). For the latter, the ProcessingPipe can check if a next filter is set using this function.

#### Returns

bool true if a next filter has been marked, else otherwise

References has Marked Processing Finished (), and marked Next Desired Filter.

### 7.9.3.4 hasMarkedProcessingFinished()

```
bool FilterManagementLibrary::PipeSystem::PipeFilter::hasMarkedProcessingFinished ( ) const
```

Returns whether the processing on the current DataSet is finished.

Whenever a Filter returns true in it's process() function, it needs to inform the ProcessingPipeline about what to do next (is the processing finished, or is there a next filter which shall be invoked?). Regarding the first, the ProcessingPipeline can check that using this function.

### Returns

bool true if the processing is marked finished, else otherwise

References markedProcessingFinished.

# 7.9.3.5 indicateProcessingFinished()

```
void FilterManagementLibrary::PipeSystem::PipeFilter::indicateProcessingFinished ( ) [protected]
```

Function for derived filters to indicate all processing is done.

Whenever a Filter returns true in it's process() function, it needs to inform the ProcessingPipeline about what to do next (e.g. which filter needs to be invoked next). However if a Filter decides that all processing is finsished and no following filter shall do further calculations on / modifications to the PipeWorkingDataSet, the filter shall indicate that the processing is finished (successfully, because it would return true!)

# Returns

void

References invokeNext(), and markedProcessingFinished.

# 7.9.3.6 initByPipeSetup()

```
virtual bool FilterManagementLibrary::PipeSystem::PipeFilter::initByPipeSetup ( ) [protected],
[pure virtual]
```

Get's called when the pipe's setup function is called after all the filters have been registered.

 $\label{lem:lemonted} \begin{tabular}{ll} Implemented in RoadSignAPI::MobilenetV2RoadSignClassificator, RoadSignAPI::DetectionBasedImageSlicer, RoadSignAPI::SSDLiteRoadSignDetector, RoadSignAPI::RoadSignDuplicationDeleter, and RoadSignAPI::ClassifiedSignsGrouper. \end{tabular}$ 

# 7.9.3.7 invokeNext()

References filterID, markedNextDesiredFilter, and nextDesiredFilter.

### 7.9.3.8 process()

```
virtual bool FilterManagementLibrary::PipeSystem::PipeFilter::process ( ) [protected], [pure
virtual]
```

 $\label{lem:lemonted} \begin{tabular}{ll} Implemented in RoadSignAPI::MobilenetV2RoadSignClassificator, RoadSignAPI::DetectionBasedImageSlicer, RoadSignAPI::SSDLiteRoadSignDetector, RoadSignAPI::RoadSignDuplicationDeleter, and RoadSignAPI::ClassifiedSignsGrouper. \end{tabular}$ 

# 7.9.3.9 reset()

```
void FilterManagementLibrary::PipeSystem::PipeFilter::reset ( ) [protected]
```

Function to reset basic variables of the filter, called by the pipe.

This function will be called by the pipe on the beginning of the processCurrentDataSet() function. It is intended to only reset thoe variables used for inter-pipe-information flow, so it only affects whether the processing is marked finished and the filter which is marked next. Reset of filter-specific variables will be done in the Filter's process function. This is NO virtual function and won't be inherited!

### Returns

void

References markedNextDesiredFilter, markedProcessingFinished, and nextDesiredFilter.

# 7.9.3.10 setCredentials()

```
void FilterManagementLibrary::PipeSystem::PipeFilter::setCredentials (
    int filterID,
    PipeRegisteredFilters * pipeRegisteredFilters,
    PipeWorkingDataSet * pipeWorkingDataSet ) [private]
```

Function which is used by ProcessingPipeline to hand in some variables.

When a filter is registered to a Pipeline, it needs to know on which objects the pipe is working (PipeWorkingDataSet) and which filters are registered to it (PipeRegisteredFilters). This is done through this function, which is called by ProcessingPipeline in it's registerFilter(...) function. It is not intended that this function is called anywhere outside of ProcessingPipeline.

### **Parameters**

int	filterID will be the ID used to identify the filter
PipeRegisteredFilters*	pipeRegisteredFilters pointer to the struct containing the ID's of all filters registed to the pipe
PipeWorkingDataSet*	pipeWorkingDataSet pointer to the dataSet the pipe is working on

# Returns

void

References filterID, indicateProcessingFinished(), pipeRegisteredFilters, and pipeWorkingDataSet.

# 7.9.4 Friends And Related Function Documentation

# 7.9.4.1 ProcessingPipeline

friend class ProcessingPipeline [friend]

# 7.9.5 Member Data Documentation

### 7.9.5.1 filterID

int FilterManagementLibrary::PipeSystem::PipeFilter::filterID [private]

# 7.9.5.2 markedNextDesiredFilter

bool FilterManagementLibrary::PipeSystem::PipeFilter::markedNextDesiredFilter = false [private]

Set to true when the invokeNext() function has been called. A getter function is provided so that the Processing Pipeline can query if a next filter has been marked.

# 7.9.5.3 markedProcessingFinished

bool FilterManagementLibrary::PipeSystem::PipeFilter::markedProcessingFinished = false [private]

This is a hint for the ProcessingPipeline; any filter can indicate that the processing of the pipeWorkingDataSet has is finished and no more other filters of the pipe shall be called. A getter function is provided so that the Processing← Pipeline can query if a next filter has been marked.

# 7.9.5.4 nextDesiredFilter

```
int FilterManagementLibrary::PipeSystem::PipeFilter::nextDesiredFilter = -1 [private]
```

This is a hint for the ProcessingPipeline; any filter can indicate which filter should be invoked next. For this, Pipe Filter provides invokeNext() function, whose result will be stored in this variable. A getter function is provided so that the ProcessingPipeline can query the ID of the next desired filter (if any).

### 7.9.5.5 pipeRegisteredFilters

```
PipeRegisteredFilters* FilterManagementLibrary::PipeSystem::PipeFilter::pipeRegisteredFilters [protected]
```

Pointer to a derivate of PipeRegisteredFilters, being a container for all ID's of the filters registered to the pipe, mapped to a unique name. Via this all filters know the ID's of every other filter registered to the pipe. The pointer will be passed to all filters using their setCredentials(...) function.

### 7.9.5.6 pipeWorkingDataSet

```
PipeWorkingDataSet* FilterManagementLibrary::PipeSystem::PipeFilter::pipeWorkingDataSet [protected]
```

Pointer to a derivate of PipeWorkingDataSet, being the dataSet the filters of the pipe will work on. Will be passed to all filters using their setCredentials(...) function.

The documentation for this class was generated from the following files:

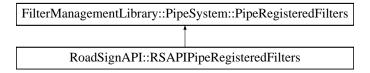
- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/PipeSystem/PipeFilter.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/PipeSystem/PipeFilter.cpp

# 7.10 FilterManagementLibrary::PipeSystem::PipeRegisteredFilters Struct Reference

Template struct.

```
#include <PipeRegisteredFilters.h>
```

Inheritance diagram for FilterManagementLibrary::PipeSystem::PipeRegisteredFilters:



# 7.10.1 Detailed Description

Template struct.

In from this derived structs, the user should organize it's Filters. We provide this to be able store the ID's of the registered Filters and get them from within any Filter which belongs to the corresponding pipe. Refer to Processing Pipeline for more context; when it's used and how.

The documentation for this struct was generated from the following file:

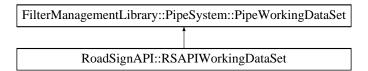
RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/PipeSystem/PipeRegistered
 —
 Filters.h

# 7.11 FilterManagementLibrary::PipeSystem::PipeWorkingDataSet Struct Reference

Template structfor PipeWorkingDataSets.

```
#include <PipeWorkingDataSet.h>
```

Inheritance diagram for FilterManagementLibrary::PipeSystem::PipeWorkingDataSet:



# 7.11.1 Detailed Description

Template structfor PipeWorkingDataSets.

In from this derived structs, the working data set for the specific pipe environment should be implemented. Refer to ProcessingPipeline for more context; when it's used and how.

The documentation for this struct was generated from the following file:

RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/PipeSystem/PipeWorking
 — DataSet.h

# 7.12 FilterManagementLibrary::PipeSystem::ProcessingPipeline Class Reference

Pipeline of the pipes and filters architecture.

#include <ProcessingPipeline.h>

# **Public Types**

enum ErrorType {
 ERROR\_NONE, ERROR\_LOGIC\_FAULT, ERROR\_FILTER\_INDICATED\_FAILURE, ERROR\_FILTER\_I
 D\_NOT\_FOUND,
 ERROR\_FILTER\_SETUP\_FAILED }

### **Public Member Functions**

ProcessingPipeline (PipeWorkingDataSet \*workingDataSet, PipeRegisteredFilters \*pipeRegisteredFilters \*pipeRegisteredFilters

Standard constructor of ProcessingPipeline.

• void registerFilter (PipeFilter \*filter, int \*filterID)

Registers a filter onto the ProcessingPipeline.

ErrorType getLastError () const

Returns an (enum) ID of the last error that happened.

• bool setup ()

Set up the pipe and it's filters.

• bool processCurrentDataSet ()

Process the PipeWorkingDataSet.

∼ProcessingPipeline ()

Destructor of ProcessingPipeline.

# **Protected Attributes**

- PipeWorkingDataSet \* workingDataSet
- · PipeRegisteredFilters \* pipeRegisteredFiltersHeader

# **Private Attributes**

- int currentFilterID = 0
- bool processingFinishied = false
- bool manageExternalAllocatedRessources = true
- std::vector< PipeFilter \* > registeredFilters
- ErrorType lastError

# 7.12.1 Detailed Description

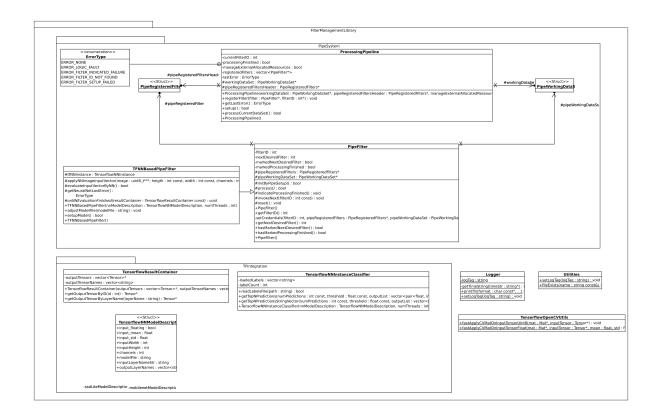
Pipeline of the pipes and filters architecture.

The Pipes-and-Filters-Architecture is a software architectural pattern usually used for systems processing data or data streams in a specific way. A pipe, which can be seen as a sequence of processing operations, represents the connection between the individual filters (processing steps). You may refer to PipeFilter for a more detailed explanation of the filters part of the architecture.

The output of a processing step (filter) can be seen as the input for the subsequent processing step, although in this implementation the input and output data is mapped to a single object allocated on the heap, the PipeWorking

DataSet.

As this implementation is based on the "Tee-and-Joins-Pipes-and-Filters-Architecture", the order in which the filters are executed can change dynamically. For now this is implemented as follows: Each filter provides a function process(). In this function a filter either has to indicate which filter shall be invoked next, or that it claims to be the last filter of the processing at least for the current iteration, which means the current data set has been fully processed. The filter may decide differently for other input data, however. Strictly speaking, this means every filter has to give a hint to the pipe to allow it to decide what to do next. For this, each filter knows the ID's of all registered filters on the pipe itself does belong to. These ID's are provided via a shared object, a derivate of the PipeRegisteredFilters struct.



# 7.12.2 Member Enumeration Documentation

# 7.12.2.1 ErrorType

enum FilterManagementLibrary::PipeSystem::ProcessingPipeline::ErrorType

# Enumerator

ERROR_NONE	Standard value set on initialization. Needed because getLastError() shall not return a random value (which could cause unexpected behavior)
ERROR_LOGIC_FAULT	E.g. if a filter neither marked a next filter nor indicated the workingDataSet has been fully processed.
ERROR_FILTER_INDICATED_FAILURE	If a filter returned false in process()
ERROR_FILTER_ID_NOT_FOUND	If a filter wants to invoke a next filter with an ID that is not assigned.
ERROR_FILTER_SETUP_FAILED	When we reun initByPipeSetup() on a filter and it returned false, which means the filter could not be set up and all filters depending on it may not be able to perform their tasks.

# 7.12.3 Constructor & Destructor Documentation

# 7.12.3.1 ProcessingPipeline()

Standard constructor of ProcessingPipeline.

The ProcessingPipeline needs to know the PipeWorkingDataSet on which all operations of the filters will be performed and the PipeRegisteredFilters struct which will contain the ID's of the filters registered to the pipe. Pointers to these coressponding objects need to be passed to this constructor. They will be passed to all registered filters. Also via manageExternalAllocatedRessources, it can be determined whether the ProcessingPipeline will "take ownership" of all those variables allocated on the heap (PipeWorkingDataSet, PipeRegisteredFilters and all registered Filters themselves), which means it will delete them on destruction. All other class members will be initialised to their defult values.

#### **Parameters**

PipeWorkingDataSet*	a pointer to an object of a struct derived from PipeWorkingDataSet. This is the data set the filters work on and will be passed to all registered filters.
PipeRegisteredFilters*	a pointer to an object of a struct derived from PipeRegisteredFilters. The struct should contain (integer) values to the ID's of all registered Filters. The filters need this for their invokeNext(const int filterID) function to know the ID of the filter they want to invoke next.
bool	manageExternalAllocatedRessources indicate whether the ProcessingPipeline shall take ownership of all variables passed to it which were allocated on the heap, therefore deleting them on destruction.

References registerFilter().

# 7.12.3.2 ∼ProcessingPipeline()

```
FilterManagementLibrary::PipeSystem::ProcessingPipeline::~ProcessingPipeline ( )
```

Destructor of ProcessingPipeline.

If manageExternalAllocatedRessources was set to true in constructor, all variables on the heap (PipeWorking DataSet, PipeRegisteredFilters, the filters itselves) will be deleted. Otherwise the destructor does do nothing.

# Returns

ProcessingPipeline::ErrorType the enum value of the last error that happened.

References manageExternalAllocatedRessources, pipeRegisteredFiltersHeader, registeredFilters, and working 

DataSet.

# 7.12.4 Member Function Documentation

# 7.12.4.1 getLastError()

```
\label{line::errorType} Filter \texttt{ManagementLibrary::} \\ \text{PipeSystem::ProcessingPipeline::getLastError () const}
```

Returns an (enum) ID of the last error that happened.

### Returns

ProcessingPipeline::ErrorType the enum value of the last error that happened.

References lastError.

# 7.12.4.2 processCurrentDataSet()

```
bool FilterManagementLibrary::PipeSystem::ProcessingPipeline::processCurrentDataSet ( )
```

Process the PipeWorkingDataSet.

Will iterate over all filters and call their process() functions, which means the DataSet will be processed. Each filter will (should!) indicate either if the processing has been finished, or if (and which) a filter shall be invoked next, as long as their process function returns true. Otherwise, if a filter returns false, this means the current dataset couldn't be processed. Either way, in any case of failure, lastError will be set accordingly.

### Returns

bool true if all filters return true in their process() functions AND either indicate which filter to invoke next or that the processing is finished, false otherwise.

References currentFilterID, FilterManagementLibrary::PipeSystem::PipeFilter::getNextDesiredFilter(), Filter ManagementLibrary::PipeSystem::PipeFilter::hasMarkedNextDesiredFilter(), FilterManagementLibrary::Pipe System::PipeFilter::hasMarkedProcessingFinished(), lastError, FilterManagementLibrary::Logger::printfln(), Filter ManagementLibrary::PipeSystem::PipeFilter::process(), processingFinishied, registeredFilters, and Filter ManagementLibrary::PipeSystem::PipeFilter::reset().

# 7.12.4.3 registerFilter()

Registers a filter onto the ProcessingPipeline.

Each filter which shall be included into the ProcessingPipeline needs to be registered to it using this function.

### **Parameters**

PipeFilter*   filter pointer to the filter to register	
int	filterID a pointer to an integer whose value will be changed in this function. It will contain the ID of
Generated by Doxygene newly registered filter afterwards.	

#### Returns

void

References pipeRegisteredFiltersHeader, registeredFilters, FilterManagementLibrary::PipeSystem::PipeFilter :: setCredentials(), and workingDataSet.

### 7.12.4.4 setup()

```
bool FilterManagementLibrary::PipeSystem::ProcessingPipeline::setup ( )
```

Set up the pipe and it's filters.

oracion Will iterate over all filters and call initByPipeSetup() on them. On failure, lastError will be set accordingly.

# Returns

bool true if all filters return true in their initByPipeSetup() functions, false otherwise.

References FilterManagementLibrary::PipeSystem::PipeFilter::getFilterID(), FilterManagementLibrary::Pipe System::PipeFilter::initByPipeSetup(), lastError, FilterManagementLibrary::Logger::printfln(), processCurrent DataSet(), and registeredFilters.

# 7.12.5 Member Data Documentation

### 7.12.5.1 currentFilterID

```
int FilterManagementLibrary::PipeSystem::ProcessingPipeline::currentFilterID = 0 [private]
```

ID of the currently active filter. Each filter ID is unique and has to be positive. Uniqueness is assured by the pipe itself due to continuous numerating in registerFilter()

# 7.12.5.2 lastError

```
ErrorType FilterManagementLibrary::PipeSystem::ProcessingPipeline::lastError [private]
```

Error code of the last error that happened. See ProcessingPipeline::ErrorType

# 7.12.5.3 manageExternalAllocatedRessources

```
bool FilterManagementLibrary::PipeSystem::ProcessingPipeline::manageExternalAllocatedRessources
= true [private]
```

Indicate whether we shall manage the heap allocated memory of workingDataSet, pipeRegisteredFiltersHeader and the themselves, i.e. explicitly delete those externally allocated objects in our destructor.

# 7.12.5.4 pipeRegisteredFiltersHeader

PipeRegisteredFilters\* FilterManagementLibrary::PipeSystem::ProcessingPipeline::pipeRegistered← FiltersHeader [protected]

Pointer to a derivate of PipeRegisteredFilters, being a container for all ID's of the filters registered to the pipe, mapped to a unique name. Via this all filters know the ID's of every other filter registered to the pipe. The pointer will be passed to all filters using their setCredentials(...) function.

### 7.12.5.5 processingFinishied

bool FilterManagementLibrary::PipeSystem::ProcessingPipeline::processingFinishied = false
[private]

Stores whether the last filter indicated the current data set has been fully processed.

# 7.12.5.6 registeredFilters

std::vector<PipeFilter\*> FilterManagementLibrary::PipeSystem::ProcessingPipeline::registered← Filters [private]

Just a list containing pointers to all filters registered on this instance of the pipe.

# 7.12.5.7 workingDataSet

PipeWorkingDataSet\* FilterManagementLibrary::PipeSystem::ProcessingPipeline::workingDataSet
[protected]

Pointer to a derivate of PipeWorkingDataSet, being the dataSet the filters of the pipe will work on. Will be passed to all filters using their setCredentials(...) function.

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/PipeSystem/Processing
   — Pipeline.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/PipeSystem/Processing
   — Pipeline.cpp

# 7.13 RoadSignAPI::RoadSignAPI Class Reference

#include <RoadSignAPI.h>

### **Public Member Functions**

· bool init ()

Initializes the RoadSignAPI's static interface.

bool feedImage (cv::Mat iamge)

Takes an OpenCV Mat uses the filter to examine it for road signs.

const std::vector< DetectedSignDescriptor > \* getDetectedSigns ()

Returns a vector of all detected (not classified!) signs.

void getClassifierApprovedDetectedSigns (std::vector < const DetectedSignDescriptor \*> \*detectedSigns)
 Returns a vector of all classified signs.

 $\bullet \ \ const \ std:: vector < \ \ Detected Sign Combination > * const \ get Detected Sign Combinations \ () \ const$ 

Returns a vector of all DetectedSignCombination}s.

RoadSignAPI (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription detectorModel
 — Description, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription classificatorModel
 — Description, const int numThreads, AAssetManager \*const assetManager)

Constructor of the RoadSignAPI for Android environments.

RoadSignAPI (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription detectorModel
 — Description, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription classificatorModel
 — Description, const int numThreads)

Constructor of the RoadSignAPI for non Android environments.

# **Static Public Member Functions**

static bool staticInit (const int numThreads, AAssetManager \*const assetManager)

Initializes the RoadSignAPI's static interface.

· static bool staticInit (const int numThreads)

Initializes the RoadSignAPI's static interface.

static bool staticFeedImage (cv::Mat image)

Takes an OpenCV Mat uses the filter to examine it for road signs.

static const std::vector< DetectedSignDescriptor > \* staticGetDetectedSigns ()

Returns a vector of all detected (not classified!) signs.

static void staticGetClassifierApprovedDetectedSigns (std::vector< const DetectedSignDescriptor \*>
 \*detectedSigns)

Returns a vector of all classified signs.

static const std::vector< DetectedSignCombination > \*const staticGetDetectedSignCombinations ()

Returns a vector of all DetectedSignCombination}s.

# **Private Attributes**

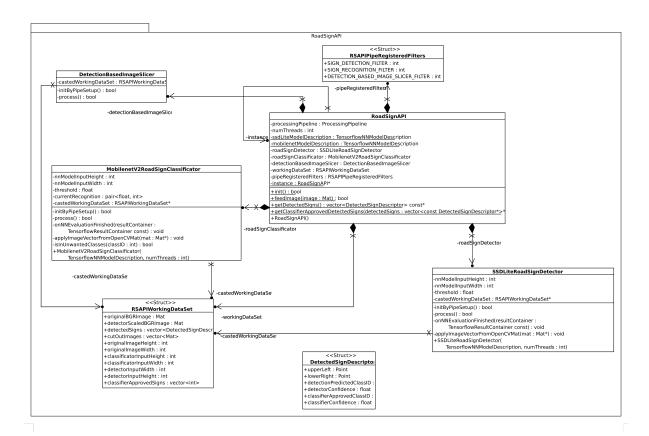
- FilterManagementLibrary::PipeSystem::ProcessingPipeline processingPipeline
- SSDLiteRoadSignDetector roadSignDetector
- MobilenetV2RoadSignClassificator roadSignClassificator
- DetectionBasedImageSlicer detectionBasedImageSlicer
- ClassifiedSignsGrouper classifiedSignsGrouper
- · RoadSignDuplicationDeleter roadSignDuplicationDeleter
- RSAPIWorkingDataSet workingDataSet
- RSAPIPipeRegisteredFilters pipeRegisteredFilters
- AAssetManager \*const assetManager

# **Static Private Attributes**

- static FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription ssdLiteModelDescription
- static FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription mobilenetModelDescription
- static RoadSignAPI \* instance

# 7.13.1 Detailed Description

Implemenation of the RoadSignAPI. Here all filters used for roadsign detection and classification will be set up and managed. It provides a static interface using a static instance of RoadSignAPI to easily feed an image and analyze it from static environments (i.e. Java Native Interface)



# 7.13.2 Constructor & Destructor Documentation

### 7.13.2.1 RoadSignAPI() [1/2]

Constructor of the RoadSignAPI for Android environments.

Under Android, we need a slightly different constructor for the RoadSignAPI: As we want to store the neuronal network model files in the assets folder of the app, we need a Pointer to an AssetManager passed from the Java part of the app (by any means). The Android-platform dependent code of the RoadSignAPI expects this (also refer to SSDLiteRoadSignDetector and MobilenetV2RoadSignClassificator for more details). Of course this constructor is only available under Android environments. For other platforms, please use the corresponding constructor. Calls the constructor of the ProcessingPipeline and all filters. Afterwards the filters are registered to the pipe.

### **Parameters**

TensorflowNNModelDescription	The model description for the detector network.
TensorflowNNModelDescription	The model description for the classificator network.
const	int numThreads Number of threads Tensorflow shall be allowed to use at
	max.
AAssetManager*	const assetManager pointer to an AssetManager, which we will use to load neuronal network model files from the assets folder of the Android app.  Needs to be passed from the Java part of the app by any means.

# 7.13.2.2 RoadSignAPI() [2/2]

Constructor of the RoadSignAPI for non Android environments.

Calls the constructor of the ProcessingPipeline and all filters. Afterwards the filters are registered to the pipe.

### **Parameters**

TensorflowNNModelDescription	The model description for the detector network.
TensorflowNNModelDescription	The model description for the classificator network.
const	int numThreads Number of threads Tensorflow shall be allowed to use at max.

# 7.13.3 Member Function Documentation

# 7.13.3.1 feedImage()

Takes an OpenCV Mat uses the filter to examine it for road signs.

References image.

## 7.13.3.2 getClassifierApprovedDetectedSigns()

Returns a vector of all classified signs.

Should only be called after the feedImage(...) function was used. The vector contains all signs which were detected AND classified on the previous image. All images which were detected but could not be classified or are classified as a part of the unwanted classes won't be contained in this list!

#### **Parameters**

std::vector <const< th=""><th colspan="2">DetectedSignDescriptor*&gt;* a pointer to a vector of pointers to a DetectedSignDescriptor.</th></const<>	DetectedSignDescriptor*>* a pointer to a vector of pointers to a DetectedSignDescriptor.	
	The vector will be filled with the DetectedSignDescriptors of the RSAPIWorkingDataSet	
	which were successfully classified as relevant roadsigns.	

Returns

void

References i.

# 7.13.3.3 getDetectedSignCombinations()

```
\verb|const| std::vector<| RoadSignAPI::DetectedSignCombination| > *const| RoadSignAPI::RoadSignAPI::DetectedSignCombination| > *const| RoadSignAPI::RoadSignAPI::DetectedSignCombination| > *const| RoadSignAPI::RoadSignAPI::DetectedSignCombination| > *const| RoadSignAPI::RoadSignAPI::DetectedSignCombination| > *const| RoadSignAPI::DetectedSignCombination| > *const| R
```

Returns a vector of all DetectedSignCombination}s.

Should only be called after the feedImage(...) function was used. The vector contains all signs which were detected AND classified on the previous image grouped into <a href="DetectedSignCombination">DetectedSignCombination</a>}s. All images which were detected but could not be classified or are classified as a part of the unwanted classes won't be contained in this list!

#### Returns

const std::vector<DetectedSignCombination>\* const pointer to a vector containing all DetectedSign← Combination}s, that were found in the image that was lastly provided to feedImage(...).

## 7.13.3.4 getDetectedSigns()

```
\verb|const| std::vector<| RoadSignAPI::DetectedSignDescriptor| > * RoadSignAPI::RoadSignAPI::get \leftarrow \\ \texttt|DetectedSignS| ( )
```

Returns a vector of all detected (not classified!) signs.

Should only be called after the feedImage(...) function was used. The vector contains all signs which were detected on the previous image.

#### Returns

const std::vector<DetectedSignDescriptor> a vector containing all detected signs.

#### 7.13.3.5 init()

```
bool RoadSignAPI::RoadSignAPI::init ( )
```

Initializes the RoadSignAPI's static interface.

We use a Singleton to provide a static interface to the RoadSignAPI. This Singleton will be initialized in this function: The model descriptions for the SSDLite and MobilenetV2 neuronal networks will be created, the Road← SignAPI will be instantiated and the ProcessingPipeline's setUp function is called.

#### 7.13.3.6 staticFeedImage()

Takes an OpenCV Mat uses the filter to examine it for road signs.

References feedImage().

#### 7.13.3.7 staticGetClassifierApprovedDetectedSigns()

```
\label{lem:proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_proved_pro
```

Returns a vector of all classified signs.

Should only be called after the staticFeedImage(...) function was used. The vector contains all signs which were detected AND classified on the previous image. All images which were detected but could not be classified or are classified as a part of the unwanted classes won't be contained in this list!

#### **Parameters**

std::vector <const< th=""><th>DetectedSignDescriptor*&gt;* a pointer to a vector of pointers to a DetectedSignDescriptor.</th></const<>	DetectedSignDescriptor*>* a pointer to a vector of pointers to a DetectedSignDescriptor.
	The vector will be filled with the DetectedSignDescriptors of the RSAPIWorkingDataSet
	which were successfully classified as relevant roadsigns.

#### Returns

void

# 7.13.3.8 staticGetDetectedSignCombinations()

```
\label{lem:const_static} $$\operatorname{const_RoadSignAPI::DetectedSignCombination} > *\operatorname{const_RoadSignAPI::RoadSignAPI} \leftrightarrow ::\operatorname{staticGetDetectedSignCombinations} () [static]
```

Returns a vector of all DetectedSignCombination}s.

Should only be called after the staticFeedImage(...) function was used. The vector contains all signs which were detected AND classified on the previous image grouped into <a href="DetectedSignCombination">DetectedSignCombination</a>}s. All images which were detected but could not be classified or are classified as a part of the unwanted classes won't be contained in this list!

## Returns

const std::vector<DetectedSignCombination>∗ const pointer to a vector containing all DetectedSign← Combination}s, that were found in the image that was lastly provided to staticFeedImage(...).

References getDetectedSignCombinations().

#### 7.13.3.9 staticGetDetectedSigns()

Returns a vector of all detected (not classified!) signs.

Should only be called after the feedImage(...) function was used. The vector contains all signs which were detected on the previous image.

#### Returns

const std::vector<DetectedSignDescriptor> a vector containing all detected signs.

References getDetectedSigns().

```
7.13.3.10 staticInit() [1/2]
```

Initializes the RoadSignAPI's static interface.

This is the staticInit function for Android environments. As under Android, we want the neuronal network model files to be stored inside the assets folder of the app, we need an AssetManager passed from the Java part of the app to be able to do so. On non Android environments, this function just lacks it's last parameter.

We use a Singleton to provide a static interface to the RoadSignAPI. This Singleton will be initialized in this function: The model descriptions for the SSDLite and MobilenetV2 neuronal networks will be created, the Road← SignAPI will be instantiated and the ProcessingPipeline's setUp function is called.

References init().

```
7.13.3.11 staticInit() [2/2]
```

Initializes the RoadSignAPI's static interface.

This is the staticInit function for non-Android environments. We use a Singleton to provide a static interface to the RoadSignAPI. This Singleton will be initialized in this function: The model descriptions for the SSDLite and MobilenetV2 neuronal networks will be created, the RoadSignAPI will be instantiated and the ProcessingPipeline's setUp function is called.

References init().

# 7.13.4 Member Data Documentation

# 7.13.4.1 assetManager

```
AAssetManager* const RoadSignAPI::RoadSignAPI::assetManager [private]
```

A pointer to an AssetManager which can be passed via the constructor. We need it to be able to load model files from the Android assets folder, which we want to do so the files do not have to be placed in the normal user space. Will be passed to SSDLiteRoadSignDetector and MobilenetV2RoadSignClassificator.

#### 7.13.4.2 classifiedSignsGrouper

```
ClassifiedSignsGrouper RoadSignAPI::RoadSignAPI::classifiedSignsGrouper [private]
```

Instance of ClassifiedSignsGrouper which groups all signs that where detected AND classified into DetectedSign← Combination}s

## 7.13.4.3 detectionBasedImageSlicer

```
DetectionBasedImageSlicer RoadSignAPI::RoadSignAPI::detectionBasedImageSlicer [private]
```

Instance of DetectionBasedImageSlicer used to crop the boxes in which the roadsigns are contained, predicted by the SSDLiteRoadSignDetector.

#### 7.13.4.4 instance

```
RoadSignAPI::RoadSignAPI * RoadSignAPI::nostance [static], [private]
```

Static instance of RoadSignAPI for static interface.

#### 7.13.4.5 mobilenetModelDescription

```
FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription RoadSignAPI::RoadSignAP← I::mobilenetModelDescription [static], [private]
```

Static description for the mobilenetv2 network, used for static interface.

#### 7.13.4.6 pipeRegisteredFilters

```
RSAPIPipeRegisteredFilters RoadSignAPI::RoadSignAPI::pipeRegisteredFilters [private]
```

Instance of RSAPIPipeRegisteredFilters, which is a struct in which all IDs of the filters which are added to the ProcessingPipeline of this class are stored. It is passed to all filters so they know the IDs of all other filters.

# 7.13.4.7 processingPipeline

```
FilterManagementLibrary:: PipeSystem::ProcessingPipeline RoadSignAPI::RoadSignAPI::processing←Pipeline [private]
```

Pipeline which is used for sequencial processing of filters used for roadsign detection, classification, grouping and filtering.

## 7.13.4.8 roadSignClassificator

```
{\tt MobilenetV2RoadSignClassificator~RoadSignAPI::RoadSignAPI::roadSignClassificator~[private]}
```

Instance of the MobilenetV2RoadSignClassificator used for roadsign classification, which is registered to the pipe of an instance of RoadSignAPI.

# 7.13.4.9 roadSignDetector

```
SSDLiteRoadSignDetector RoadSignAPI::RoadSignAPI::roadSignDetector [private]
```

Instance of the SSDLiteRoadSignDetector used for roadsign detection, which is registered to the pipe of an instance of RoadSignAPI.

#### 7.13.4.10 roadSignDuplicationDeleter

```
RoadSignDuplicationDeleter RoadSignAPI::RoadSignAPI::roadSignDuplicationDeleter [private]
```

Instance of RoadSignDuplicationDeleter which is used to delete signs that were detected multiple times (with a little offset) by the SSDLiteRoadSignDetector.

#### 7.13.4.11 ssdLiteModelDescription

```
\label{lem:reduced} Filter \texttt{ManagementLibrary::TFIntegration::TensorflowNNModelDescription} \ \ RoadSignAPI::RoadSignAP \leftrightarrow I::ssdLiteModelDescription \ \ [static], \ [private]
```

Static description for the ssdLite network, used for static interface.

#### 7.13.4.12 workingDataSet

```
RSAPIWorkingDataSet RoadSignAPI::RoadSignAPI::workingDataSet [private]
```

Instance of RSAPIWorkingDataSet, which is the shared resource in which all results of the filters used in this class are stored.

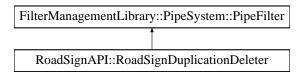
The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/RoadSignAPI.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/RoadSignAPI.cpp

# 7.14 RoadSignAPI::RoadSignDuplicationDeleter Class Reference

#include <RoadSignDuplicationDeleter.h>

Inheritance diagram for RoadSignAPI::RoadSignDuplicationDeleter:



## **Private Member Functions**

- void markForDeletion (int ID)
- void deleteAllMarkedSignDescriptors ()
- bool isMarkedForDeletion (int ID) const
- bool signsDoOverlap (DetectedSignDescriptor \*first, DetectedSignDescriptor \*second) const
- float calculateOverlapPercentage (DetectedSignDescriptor \*first, DetectedSignDescriptor \*second) const
- int calculateBoxArea (DetectedSignDescriptor \*signDescriptor) const
- bool initByPipeSetup ()

Initializes the filter.

• bool process ()

The process function of this filter.

#### **Private Attributes**

- RSAPIWorkingDataSet \* castedWorkingDataSet
- float minOverlapPercentage = 0.60
- std::vector< int > signsToDelete

#### **Additional Inherited Members**

#### 7.14.1 Detailed Description

As it is possible that the detector detects one RoadSign at about the same position (with a small offset), it is neccessary to filter those duplicates out, because they would be classified and added to a DetectedSignCombination (by ClassifiedSignsGrouper) twice. This Filter tries to the detect duplicates by checking the congruency of the rectangles of the detected signs. If they overlap by a certain percentage (specified by minOverLapPercentage), the bigger sign will be dropped. Why the bigger sign? Because it is possible that the detector detects two signs three times: two times as single signs, and the third time combined into one sign, which is - obviously - not wanted.

#### 7.14.2 Member Function Documentation

#### 7.14.2.1 calculateBoxArea()

Calculates the area of the box of a DetectedSignDescriptor}

#### Returns

int width \* height as the area of the box of the signDescriptor.

# 7.14.2.2 calculateOverlapPercentage()

Calculates the percentage of which the box (rectangle) of the first DetectedSignDescriptor} does overlap with the second (the order of first and second is relevant! The percentage is calculated relative of the area of the box of first).

#### Returns

float Percentage of the overlapping area of the boxes of the two signs relative to the area of the box of first.

 $References\ calculate Box Area(),\ Road Sign API:: Detected Sign Descriptor:: lower Right,\ and\ Road Sign API:: Detected \cite{Continuous Sign Descriptor:: upper Left}.$ 

#### 7.14.2.3 deleteAllMarkedSignDescriptors()

```
\verb|void RoadSignAPI::RoadSignDuplicationDeleter::deleteAllMarkedSignDescriptors () | [private]| \\
```

Deletes all signs in castedWorkingDataSet->detectedSigns, which were marked for deletion during the process() function. This may change the order of the detectedSigns list, because we want to avoid bigger copy and / or move operations. So the element which shall be deleted is overriden by the last element of the list, and afterwards the list is reduced by one.

References castedWorkingDataSet, RoadSignAPI::RSAPIWorkingDataSet::detectedSigns, i, and signsToDelete.

# 7.14.2.4 initByPipeSetup()

```
bool RoadSignAPI::RoadSignDuplicationDeleter::initByPipeSetup ( ) [private], [virtual]
```

Initializes the filter.

As this filter does not need any specific environment variables, nothing really exciting is happenning here..

Returns

true, always

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References castedWorkingDataSet, FilterManagementLibrary::PipeSystem::PipeFilter::pipeWorkingDataSet, and FilterManagementLibrary::Logger::printfln().

#### 7.14.2.5 isMarkedForDeletion()

References signsToDelete.

#### 7.14.2.6 markForDeletion()

Marks the ID of the sign for deletion, thus adding it to the signsToDelete (member variable) list.

References signsToDelete.

#### 7.14.2.7 process()

```
bool RoadSignAPI::RoadSignDuplicationDeleter::process ( ) [private], [virtual]
```

The process function of this filter.

Filters duplicates of signs (that were detected multiple times with an offset). For this it iterates over the list of all signs and checks each sign against each other sign and calculates the percentage of the overlap of both rectangles (of the signs). If they overlap by at least minOverlapPercentage (member variable), than the bigger sign is dropped. Why the bigger sign, you may ask? Because it is likely that, if we have one big detected box and one smaller box in it, there are actually two signs inside the box. If that's the case it is likely that also the second smaller sign is detected another time (so we have three detections: two smaller ones, each representing one sign, and one bigger which contains the two signs combined), so those two smaller detections shall be keeped. And if the second smaller sign is not detected, the bigger box would probably be false anyways.

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References calculateBoxArea(), calculateOverlapPercentage(), castedWorkingDataSet, deleteAllMarkedSign Descriptors(), RoadSignAPI::RSAPIWorkingDataSet::detectedSigns, i, FilterManagementLibrary::PipeSystem:: PipeFilter::invokeNext(), isMarkedForDeletion(), markForDeletion(), minOverlapPercentage, FilterManagement Library::PipeSystem::PipeFilter::pipeRegisteredFilters, signsDoOverlap(), and signsToDelete.

#### 7.14.2.8 signsDoOverlap()

Checks if the boxes (rectangles) of two DetectedSignDescriptor}s do overlap by doing some basic mathematical operations using the position and dimension of the boxes.

# Returns

true, if they do overlap at least by one pixel, false otherwise

References RoadSignAPI::DetectedSignDescriptor::lowerRight, and RoadSignAPI::DetectedSignDescriptor  $\leftarrow$  ::upperLeft.

#### 7.14.3 Member Data Documentation

#### 7.14.3.1 castedWorkingDataSet

```
RSAPIWorkingDataSet* RoadSignAPI::RoadSignDuplicationDeleter::castedWorkingDataSet [private]
```

Just a pointer casted from PipeWorkingDataSet\* to RSAPIWorkingDataSet\*, so we just don't have to do the casting every time we need it ;)

## 7.14.3.2 minOverlapPercentage

```
float RoadSignAPI::RoadSignDuplicationDeleter::minOverlapPercentage = 0.60 [private]
```

The minimum percentage by which two signs need to overlap to be considered equal.

# 7.14.3.3 signsToDelete

```
std::vector<int> RoadSignAPI::RoadSignDuplicationDeleter::signsToDelete [private]
```

A local list which is used by markForDeletion(...), deleteAllMarkedSignDescriptors() and isMarkedForDeletion(...). It is used to remember which of the signs which were evaluated being equal in the process() function shall be deleted, because they are duplicates.

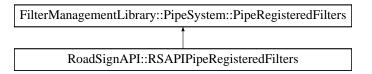
The documentation for this class was generated from the following files:

- $\bullet \ \ Road Sign API\_Standal one\_Tensor flow/header\_files/Road Sign API/Filters/Road Sign Duplication Deleter.h$
- RoadSignAPI Standalone Tensorflow/source files/RoadSignAPI/Filters/RoadSignDuplicationDeleter.cpp

# 7.15 RoadSignAPI::RSAPIPipeRegisteredFilters Struct Reference

```
#include <RSAPIPipeRegisteredFilters.h>
```

Inheritance diagram for RoadSignAPI::RSAPIPipeRegisteredFilters:



## **Public Attributes**

- int SIGN\_DETECTION\_FILTER
- int SIGN RECOGNITION FILTER
- int SIGN DUPLICATION DELETER FILTER
- int DETECTION\_BASED\_IMAGE\_SLICER\_FILTER
- int CLASSIFIED\_SIGNS\_GROUPER\_FILTER

# 7.15.1 Detailed Description

Contains variables which will hold the ID's of the filters of the API. See PipeWorkingDataSet for a detailed description of the the intention of this struct.

# 7.15.2 Member Data Documentation

# 7.15.2.1 CLASSIFIED\_SIGNS\_GROUPER\_FILTER

int RoadSignAPI::RSAPIPipeRegisteredFilters::CLASSIFIED\_SIGNS\_GROUPER\_FILTER

#### 7.15.2.2 DETECTION\_BASED\_IMAGE\_SLICER\_FILTER

 $\verb|int RoadSignAPI::RSAPIPipeRegisteredFilters::DETECTION_BASED_IMAGE\_SLICER\_FILTER| \\$ 

#### 7.15.2.3 SIGN\_DETECTION\_FILTER

int RoadSignAPI::RSAPIPipeRegisteredFilters::SIGN\_DETECTION\_FILTER

# 7.15.2.4 SIGN\_DUPLICATION\_DELETER\_FILTER

int RoadSignAPI::RSAPIPipeRegisteredFilters::SIGN\_DUPLICATION\_DELETER\_FILTER

# 7.15.2.5 SIGN\_RECOGNITION\_FILTER

int RoadSignAPI::RSAPIPipeRegisteredFilters::SIGN\_RECOGNITION\_FILTER

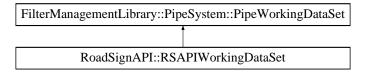
The documentation for this struct was generated from the following file:

RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/RSAPIPipeRegisteredFilters/RSAPI
 PipeRegisteredFilters.h

# 7.16 RoadSignAPI::RSAPIWorkingDataSet Struct Reference

#include <RSAPIWorkingDataSet.h>

Inheritance diagram for RoadSignAPI::RSAPIWorkingDataSet:



#### **Public Attributes**

- cv::Mat originalBGRImage
- cv::Mat detectorScaledBGRImage
- std::vector< DetectedSignDescriptor > detectedSigns
- std::vector< cv::Mat > cutOutImages
- · int originalImageHeight
- int originalImageWidth
- std::vector< int > classifierApprovedSigns
- std::vector< DetectedSignCombination > detectedSignCombinations

# 7.16.1 Detailed Description

Contains all variables which are needed to (efficiently!) detect and classifiy roadsigns from an OpenCV image Matrix. The detected and classified signs will also be stored here as a result.

#### 7.16.2 Member Data Documentation

# 7.16.2.1 classifierApprovedSigns

std::vector<int> RoadSignAPI::RSAPIWorkingDataSet::classifierApprovedSigns

Contains the ID of entries in detectedSigns which were successfully classified by the classifier Filter and are not in the class of unimportant signs.

#### 7.16.2.2 cutOutImages

 $\verb|std::vector| < cv::Mat| > RoadSignAPI::RSAPIWorkingDataSet::cutOutImages| \\$ 

This vector will be filled by the DetectionBasedImageSlicer. It uses the detectedSigns vector to crop the corresponding signs from the originalBgrImage. MobilenetV2RoadSignClassificator will scale them to the size it needs using bicubic interpolation.

#### 7.16.2.3 detectedSignCombinations

std::vector<DetectedSignCombination> RoadSignAPI::RSAPIWorkingDataSet::detectedSignCombinations

The ClassifiedSignsGrouper tries to group all successfully classified signs on an (estimated) pole (which means they are located vertically above each other) into an DetectedSignCombination. The result is stored in this vector.

## 7.16.2.4 detectedSigns

std::vector<DetectedSignDescriptor> RoadSignAPI::RSAPIWorkingDataSet::detectedSigns

Describes the signs that were detected by the SSDLiteRoadSignDetector. See DetectedSignDescriptor for a more detailed description.

## 7.16.2.5 detectorScaledBGRImage

```
cv::Mat RoadSignAPI::RSAPIWorkingDataSet::detectorScaledBGRImage
```

This will be initialized by SSDLiteRoadSignDetector. It creates a scaled cv::Mat from the originalBGRImage (in 8U ← C3 format, too) which matches the input size of the neuronal network model used for the SSDLiteRoadSignDetector using bicubic interpolation.

# 7.16.2.6 originalBGRImage

```
cv::Mat RoadSignAPI::RSAPIWorkingDataSet::originalBGRImage
```

This is the image which was feed into the API using {see RoadSignAPI}'s feedImage(...) function Needs to be 8UC3 BGR Image because {link TensorflowOpenCVUtils} expects that.

#### 7.16.2.7 originallmageHeight

```
int RoadSignAPI::RSAPIWorkingDataSet::originalImageHeight
```

Height of the originalBGRImage.

#### 7.16.2.8 originallmageWidth

```
int RoadSignAPI::RSAPIWorkingDataSet::originalImageWidth
```

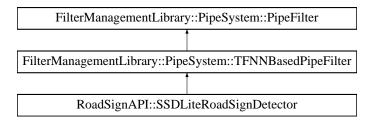
#### Width of the originalBGRImage

The documentation for this struct was generated from the following file:

# 7.17 RoadSignAPI::SSDLiteRoadSignDetector Class Reference

```
#include <SSDLiteRoadSignDetector.h>
```

Inheritance diagram for RoadSignAPI::SSDLiteRoadSignDetector:



#### **Public Member Functions**

 SSDLiteRoadSignDetector (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription, int numThreads, AAssetManager \*const assetManager)

Constructor of SSDLiteRoadSignDetector for Android environments. It basically just calls the TFNNBasedPipeFilter super constructor. Refer to it for a more detailed description. Under Android, the RoadSignAPI expects the model files of our neuronal networks to be placed in the assets folder of the app. Thus, this constructor needs to be passed a pointer to an AssetManager (from java), which is MANDATORY for the RoadSignAPI under Android!

SSDLiteRoadSignDetector (FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription, int numThreads)

Constructor of SSDLiteRoadSignDetector It basically just calls the TFNNBasedPipeFilter super constructor. Refer to it for a more detailed description.

#### **Private Member Functions**

bool initByPipeSetup ()

Initializes the filter.

· bool process ()

Uses the neuronal network to detect signs on the current image.

void onNNEvaluationFinished (const FilterManagementLibrary::TFIntegration::TensorflowResultContainer resultContainer)

Callback, will be called when the network finished it's prediction.

void applyImageVectorFromOpenCVMat (cv::Mat \*mat)

Uses TensorflowOpenCVUtils to apply a mat as input to the network.

#### **Private Attributes**

- · int nnModelInputHeight
- int nnModelInputWidth
- float threshold = 0.28f
- RSAPIWorkingDataSet \* castedWorkingDataSet
- AAssetManager \*const assetManager

#### **Additional Inherited Members**

# 7.17.1 Detailed Description

This filter is based on TFNNBasedPipeFilter and uses SSDLite for (roadsign) detection.

# 7.17.2 Constructor & Destructor Documentation

# 7.17.2.1 SSDLiteRoadSignDetector() [1/2]

```
RoadSignAPI::SSDLiteRoadSignDetector::SSDLiteRoadSignDetector (  FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription \ nnModel \leftarrow Description, \\ int \ numThreads, \\ AAssetManager *const \ assetManager )
```

Constructor of SSDLiteRoadSignDetector for Android environments. It basically just calls the TFNNBasedPipeFilter super constructor. Refer to it for a more detailed description. Under Android, the RoadSignAPI expects the model files of our neuronal networks to be placed in the assets folder of the app. Thus, this constructor needs to be passed a pointer to an AssetManager (from java), which is MANDATORY for the RoadSignAPI under Android!

All other class-member variables will be initialized to their default values.

#### **Parameters**

TensorflowNNModelDescription	nnModelDescription a description of the Tensorflow model which will be used by the filter.
int	numThreads number of threads Tensorflow is allowed to use for computation.
AAssetManager*	const assetManager pointer to an Android Asset Manager which needs to be passed from the Java part of the Android App by any means. Manadatory to load the neuronal network models from the assets directory of the app.

# 7.17.2.2 SSDLiteRoadSignDetector() [2/2]

```
\label{lem:readSignAPI::SSDLiteRoadSignDetector::SSDLiteRoadSignDetector ( \\ FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription \ nnModel \hookleftarrow \\ Description, \\ int \ numThreads \ )
```

Constructor of SSDLiteRoadSignDetector It basically just calls the TFNNBasedPipeFilter super constructor. Refer to it for a more detailed description.

All class-member variables will be initialized to their default values.

#### **Parameters**

TensorflowNNModelDescription	nnModelDescription a description of the Tensorflow model which will be used by the filter.
int	numThreads number of threads Tensorflow is allowed to use for computation.

# 7.17.3 Member Function Documentation

# 7.17.3.1 applyImageVectorFromOpenCVMat()

Uses TensorflowOpenCVUtils to apply a mat as input to the network.

## **Parameters**

#### Returns

void

References FilterManagementLibrary::TensorflowOpenCVUtils::fastApplyCVMatOnInputTensorUInt8(), and Filter ManagementLibrary::PipeSystem::TFNNBasedPipeFilter::getNNInputTensor().

#### 7.17.3.2 initByPipeSetup()

```
bool RoadSignAPI::SSDLiteRoadSignDetector::initByPipeSetup ( ) [private], [virtual]
```

Initializes the filter.

Will be called by ProcessingPipeline. Here, basically just the neuronal network model will be loaded. For this, the setupModel() function will be called, refer to TFNNBasedPipeFilter for a more detailed description. Also the RSAPIWorkingDataSet's detectorSgaledBGRImage will be initialized (with the corresponding size and type).

#### Returns

true, if the model could be loaded successfully, flase otherwise

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References assetManager, castedWorkingDataSet, RoadSignAPI::RSAPIWorkingDataSet::detectorScaled BGRImage, FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::getNNModelDescription(), Filter ManagementLibrary::TFIntegration::TensorflowNNModelDescription::inputHeight, FilterManagementLibrary:: TFIntegration::TensorflowNNModelDescription::inputWidth, nnModelInputHeight, nnModelInputWidth, Filter ManagementLibrary::PipeSystem::PipeFilter::pipeWorkingDataSet, FilterManagementLibrary::Logger::printfln(), FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::setupModelFromAssets(), and FilterManagement Library::PipeSystem::TFNNBasedPipeFilter::setupModelFromFile().

## 7.17.3.3 onNNEvaluationFinished()

Callback, will be called when the network finished it's prediction.

In process(), the TFNNBasedPipeFilter's (super class) evaluteInputVectorByNN() will be called. If it was successfull, this callback will be called and passed a TensorflowResultContainer with the result of the inference of the neuronal network model. Here, we will interpete the output of the network. We iterate over all it's detections and add them to a DetectedSignDescriptor, if the cofidence exceeds a certain threshold value. This descriptor is then added to RSAPIWorkingDataSet.

#### **Parameters**

const	TensorflowResultContainer resultContainer contains the results (in Tensorflow tensors) of the network	
	prediction.	

Implements FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter.

References applyImageVectorFromOpenCVMat(), castedWorkingDataSet, RoadSignAPI::RSAPIWorkingData Set::detectedSigns, RoadSignAPI::DetectedSignDescriptor::detectionPredictedClassID, RoadSignAPI::Detected SignDescriptor::detectorConfidence, i, FilterManagementLibrary::PipeSystem::PipeFilter::indicateProcessing Finished(), RoadSignAPI::DetectedSignDescriptor::lowerRight, threshold, and RoadSignAPI::DetectedSign Descriptor::upperLeft.

#### 7.17.3.4 process()

```
bool RoadSignAPI::SSDLiteRoadSignDetector::process ( ) [private], [virtual]
```

Uses the neuronal network to detect signs on the current image.

The originalBGR image will be scaled to the size the network model expects. Afterfards, it uses the TFNNBased← PipeFilter super class's evaluateInputVectorByNN() function to detect the road signs.

#### Returns

true if evaluateInputVectorByNN() return true, false otherwise (does NOT return false if no signs were detected!)

Implements FilterManagementLibrary::PipeSystem::PipeFilter.

References applyImageVectorFromOpenCVMat(), castedWorkingDataSet, RoadSignAPI::RSAPIWorkingData&Set::detectedSignCombinations, RoadSignAPI::RSAPIWorkingDataSet::detectedSigns, RoadSignAPI::RSA&PIWorkingDataSet::detectedSigns, RoadSignAPI::RSA&PIWorkingDataSet::detectedSigns, RoadSignAPI::RSA&PIWorkingDataSet::detectedSigns, RoadSignAPI::RSA&PIWorkingDataSet::detectedSigns, RoadSignAPI::RSAPIWorkingDataSet::ipipeSystem::PipeSystem::PipeSystem::TFNNBasedPipeFilter&PipeFilter::invokeNext(), onNNEvaluation&Pinished(), RoadSignAPI::RSAPIWorkingDataSet::originalBGRImage, FilterManagementLibrary::PipeSystem::&PipeFilter::pipeRegisteredFilters, and FilterManagementLibrary::Logger::printfln().

# 7.17.4 Member Data Documentation

#### 7.17.4.1 assetManager

```
AAssetManager* const RoadSignAPI::SSDLiteRoadSignDetector::assetManager [private]
```

A pointer to an AssetManager which can be passed via the constructor (only available under Android environments). We need it to be able to load model files from the Android assets folder, which we want to do so the files do not have to be placed in the normal user space.

## 7.17.4.2 castedWorkingDataSet

```
RSAPIWorkingDataSet* RoadSignAPI::SSDLiteRoadSignDetector::castedWorkingDataSet [private]
```

Just a pointer casted from PipeWorkingDataSet\* to RSAPIWorkingDataSet\*, so we just don't have to do the casting every time we need it ;)

## 7.17.4.3 nnModelInputHeight

```
int RoadSignAPI::SSDLiteRoadSignDetector::nnModelInputHeight [private]
```

Just a copy from the input height provided in the TensorflowNNModelDescription of the underlaying TensorflowN← NInstance.

#### 7.17.4.4 nnModelInputWidth

```
int RoadSignAPI::SSDLiteRoadSignDetector::nnModelInputWidth [private]
```

Just a copy from the input height provided in the TensorflowNNModelDescription of the underlaying TensorflowN← NInstance.

#### 7.17.4.5 threshold

```
float RoadSignAPI::SSDLiteRoadSignDetector::threshold = 0.28f [private]
```

Threshold the confidence of a prediction from our underlying neuronal network model needs to exceed in order to be included to RSAPIWorkingDataSet::detectedSigns

The documentation for this class was generated from the following files:

- $\bullet \ \ Road Sign API\_Standal one\_Tensor flow/header\_files/Road Sign API/Filters/SSDLiteRoad Sign Detector.h$
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/SSDLiteRoadSignDetector.cpp

# 7.18 FilterManagementLibrary::TensorflowAndroidJNIUtils Class Reference

#include <TensorflowAndroidJNIUtils.h>

# **Static Public Member Functions**

- static bool PortableReadFileToProto (const std::string &file\_name, ::google::protobuf::MessageLite \*proto)
- static bool IsAsset (std::string filename)
- static void ReadFileToString (AAssetManager \*const asset\_manager, std::string filename, std::string \*str)
- static void WriteProtoToFile (std::string filename, const google::protobuf::MessageLite &message)

#### 7.18.1 Member Function Documentation

#### 7.18.1.1 IsAsset()

References ASSET\_PREFIX.

## 7.18.1.2 PortableReadFileToProto()

# 7.18.1.3 ReadFileToProto()

References ASSET\_PREFIX, IsAsset(), and FilterManagementLibrary::Logger::printfln().

# 7.18.1.4 ReadFileToString()

References ASSET\_PREFIX, IsAsset(), and FilterManagementLibrary::Logger::printfln().

# 7.18.1.5 ReadFileToVector()

References FilterManagementLibrary::Logger::printfln().

#### 7.18.1.6 WriteProtoToFile()

References FilterManagementLibrary::Logger::printfln().

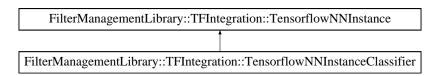
The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowAndroidJNIUtils.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/TensorflowAndroidJNIUtils.cpp

# 7.19 FilterManagementLibrary::TFIntegration::TensorflowNNInstance Class Reference

```
#include <TensorflowNNInstance.h>
```

Inheritance diagram for FilterManagementLibrary::TFIntegration::TensorflowNNInstance:



# **Public Types**

enum ErrorType {
 ErrorType::ERROR\_NONE, ErrorType::ERROR\_INVALID\_MODEL\_FILE, ErrorType::ERROR\_FAILED\_T
 O\_CONSTRUCT\_NEW\_SESSION, ErrorType::ERROR\_COULD\_NOT\_LOAD\_MODEL,
 ErrorType::ERROR\_COULD\_NOT\_ADD\_GRAPH\_TO\_SESSION, ErrorType::ERROR\_SESSION\_RUN\_
 FAILED, ErrorType::ERROR\_INPUT\_SIZE\_MISMATCH, ErrorType::READ\_FILE\_TO\_PROTO\_FROM\_A
 SSETS\_FAILED,
 ErrorType::ERROR\_TFNN\_CLASSIFIER\_INVALID\_LABELS\_FILE }

# **Public Member Functions**

• bool setupModelFromFile ()

Sets up the Tensorflow model using the provieded model description.

• bool setupModelFromAssets (AAssetManager \*const assetManager)

Sets up the Tensorflow model using the provieded model description.

• bool runInference ()

Let's the neuronal network examine the applied input.

• bool applyImageInputVector (uint8\_t \*\*\*inputVector, int imageHeight, int imageWidth, int channels)

Applies an image vector to the instance of the Tensorflow model.

- bool applyInputVectorFromFloatData (float \*floatData, const int heigth, const int width, const int channels)
  - Applies an already float input vector to the neuronal network model.
- void adjustModelFile (std::string modelFile)

Allows to change ONLY the model file path set in the model description.

tensorflow::Tensor \* getInputTensor ()

Returns the input Tensor of this instance.

std::vector< tensorflow::Tensor > \* getOutputTensors ()

Returns the tensors containg the output of the previous inference.

TensorflowResultContainer getResultContainer ()

Returns a container containing the result of the last inference.

const TensorflowNNModelDescription \* getModelDescription () const

Returns the model description which was passed to this instance.

ErrorType getLastError () const

Returns the last error that happened.

TensorflowNNInstance (TensorflowNNModelDescription nnModelDescription, int numThreads=1)

Constructor of TensorflowNNInstance.

∼TensorflowNNInstance ()

Destructor of the TensorflowNNInstance class.

# **Protected Attributes**

- tensorflow::Tensor inputTensor
- std::vector< tensorflow::Tensor > outputTensors
- std::unique\_ptr< tensorflow::Session > tensorflowSession
- tensorflow::SessionOptions sessionOptions
- tensorflow::GraphDef graphDef
- std::unique ptr< std::string > inputLayerName
- ErrorType lastError

## **Private Attributes**

- TensorflowNNModelDescription nnModelDescription
- int numThreads = 1

# 7.19.1 Detailed Description

Provides all the functionality needed to load tensorflow models, add them to a tensorflow session, provide an input vector, functions to run inferences and methods to process the output.

# 7.19.2 Member Enumeration Documentation

#### 7.19.2.1 ErrorType

enum FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ErrorType [strong]

#### Enumerator

ERROR_NONE	Standard value set on initialization. Needed because getLastError() shall not return a random value (which could cause unexpected behavior)
ERROR_INVALID_MODEL_FILE	If the model file does not exist at the given path.
ERROR_FAILED_TO_CONSTRUCT_NEW_SESS↔ ION	Tensorflow related error. Thrown, if the NewSession() function in setupModel() fails.
ERROR_COULD_NOT_LOAD_MODEL	If the given model file could not be loaded because it is in wrong format or we do not have sufficient permission to read it.
ERROR_COULD_NOT_ADD_GRAPH_TO_SESSI↔ ON	Tensorflow related error. Thrown, if the function Create() of the Tensorflow Session fails with the graph definition build from the model file.
ERROR_SESSION_RUN_FAILED	Tensorflow related error. Thrown, if we could not run the session (maybe incorrect model file, missing ops in the Tensorflow library or any other configuration error).
ERROR_INPUT_SIZE_MISMATCH	If the dimensions of the given input vector do not match with the input sized in the provided TensorflowNNModelDescription.
READ_FILE_TO_PROTO_FROM_ASSETS_FAILED	Only available under Android environments. If using the setupModelFromAssets() function, but the modelFile provided in TensorflowNNModelDescription could not be loaded (for example is not found in the assets folder).
ERROR_TFNN_CLASSIFIER_INVALID_LABELS_↔ FILE	Only relevant for TensorflowNNInstanceClassifier if the specified labels file does not exist and thus could not be loaded.

# 7.19.3 Constructor & Destructor Documentation

# 7.19.3.1 TensorflowNNInstance()

```
\label{thm:continuous} Filter {\tt ManagementLibrary::TFIntegration::TensorflowNNInstance::TensorflowNNInstance:} \\ It {\tt TensorflowNNModelDescription:nnModelDescription:} \\ int {\tt numThreads:=1:} \\ )
```

#### Constructor of TensorflowNNInstance.

This is the only constructor the TensorflowNNInstance has, and the only one that should be allowed! As an object of type TensorflowNNInstance is useless without a TensorflowNNModelDescription, it is MANDATORY to have a constructor to which an object of that type is passed. Additionally, it needs to be specified how many threads Tensorflow is allowed to use for computation! An Tensor (of type tensorflow::Tensor), which is used as input Tensor, of type specified in the model description will be created (for now, only uint8\_t and float are supported!) All other class members will be initialised to their default values.

#### **Parameters**

TensorflowNNModelDescription	nnModelDescription the model description containing all information needed
	to load and use the Tensorflow model.
int	numThreads number of threads Tensorflow is allowed to use.

References FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::channels, FilterManagement Library::TFIntegration::TensorflowNNModelDescription::input\_floating, FilterManagementLibrary::TFIntegration::

TensorflowNNModelDescription::inputHeight, inputTensor, FilterManagementLibrary::TFIntegration::TensorflowN NModelDescription::inputWidth, and setupModelFromFile().

#### 7.19.3.2 ∼TensorflowNNInstance()

```
\label{thm:prop:managementLibrary::TFIntegration::TensorflowNNInstance:: $$\sim$TensorflowNNInstance ( )$
```

Destructor of the TensorflowNNInstance class.

Destructor whose only task is to close the Tensorflow Session which was used for this instance.

References tensorflowSession.

#### 7.19.4 Member Function Documentation

#### 7.19.4.1 adjustModelFile()

```
\label{thm:cond} \begin{tabular}{ll} void Filter Management Library:: TFIntegration:: Tensor flow NNInstance:: adjust Model File ( std:: string $model File ). \end{tabular}
```

Allows to change ONLY the model file path set in the model description.

Can be used to adjust the path of the model file of the TensorflowNNModelDescription at run-time.

# **Parameters**

	std::string	new path to the modelFile to use.
--	-------------	-----------------------------------

## Returns

void

References FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::modelFile, and nnModel ← Description.

#### 7.19.4.2 applyImageInputVector()

Applies an image vector to the instance of the Tensorflow model.

This takes a 3D uint8\_t vector and will set the data on the input Tensor of the model. It is intended to be used for 3 Channel RGB images. If the model expects float as input type instead, the uint8\_t values will be converted accordingly using the mean and std value provided with the model description.

#### **Parameters**

uint8_t***	image 3D array of uint8_t values which will be applied to the model	
int	height height of the image (columns of the vector)	
int	width width of the image (rows of the vector)	
int	channels channels of the image (depth of the vector)	

#### Returns

bool true if the vector was successfully applied to the model, false otherwise

References applyInputVectorFromFloatData(), ERROR\_INPUT\_SIZE\_MISMATCH, i, FilterManagement  $\leftarrow$  Library::TFIntegration::TensorflowNNModelDescription::input\_floating, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input\_mean, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input\_std, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::inputHeight, inputTensor, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::inputWidth, lastError, and nnModelDescription.

#### 7.19.4.3 applyInputVectorFromFloatData()

Applies an already float input vector to the neuronal network model.

Used std::copy\_n to directly copy the flat float vector to the input tensor May not always work (or at all).

References getLastError(), and inputTensor.

#### 7.19.4.4 getInputTensor()

```
\texttt{tensorflow::} Tensor * Filter \texttt{ManagementLibrary::} TFIntegration::} Tensorflow \texttt{NNInstance::} getInput \leftarrow \texttt{Tensor} \ ( )
```

Returns the input Tensor of this instance.

Normally the Tensorflow input Tensor for the corresponding neuronal network model should'nt be visible outside this class. However it can be useful to provide a direct access to this Tensor, because specific tasks may need a specific way to apply an input to the network. TensorflowOpenCVUtils, for example, use this function for the fastApplyCVMatOnInputTensor functions. This, only use this function if you know what you're doing!

References inputTensor, and  $\sim$ TensorflowNNInstance().

#### 7.19.4.5 getLastError()

```
FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ErrorType FilterManagement ← Library::TFIntegration::TensorflowNNInstance::getLastError ( ) const
```

Returns the last error that happened.

## Returns

TensorflowNNInstance::ErrorType the enum value of the last error that happened when executing any previous function of TensorflowNNInstance

References getOutputTensors(), and lastError.

# 7.19.4.6 getModelDescription()

```
const FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription * FilterManagement ←
Library::TFIntegration::TensorflowNNInstance::getModelDescription ( ) const
```

Returns the model description which was passed to this instance.

#### Returns

TensorflowNNModelDescription the model description which was passed to this instance in constructor.

References getInputTensor(), and nnModelDescription.

## 7.19.4.7 getOutputTensors()

```
\verb|std::vector| < tensorflow::Tensor| > * FilterManagementLibrary::TFIntegration::TensorflowNN \leftarrow Instance::getOutputTensors ( )
```

Returns the tensors containg the output of the previous inference.

Returns the output tensors of this instance. If an inference was successfully executed before, those tensors will hold the output of the neuronal network model used. However, to examine the output, getResultContainer() should be used.

#### Returns

std::vector<stensorflow::tensor>\* vector containg the output Tensors

References outputTensors, and runInference().

#### 7.19.4.8 getResultContainer()

Returns a container containing the result of the last inference.

Packs Tensorflow output tensors of this instance to a TensorflowNNResultContainer which can be used to process the result of the last inference.

#### Returns

TensorflowResultContainer container containing the result of the last inference.

References getModelDescription(), nnModelDescription, FilterManagementLibrary::TFIntegration::TensorflowNN← ModelDescription::outputLayerNames, and outputTensors.

# 7.19.4.9 runInference()

```
bool FilterManagementLibrary::TFIntegration::TensorflowNNInstance::runInference ( )
```

Let's the neuronal network examine the applied input.

An input vector should have been applied prior to calling this function. It will run the Tensorflow Session's run function which calculates the output to the given input. On failure, lastError will be set accordingly.

# Returns

true if the inference was successfull, false otherwise

References adjustModelFile(), ERROR\_SESSION\_RUN\_FAILED, inputLayerName, inputTensor, lastError, nn ModelDescription, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::outputLayerNames, outputTensors, FilterManagementLibrary::Logger::printfln(), and tensorflowSession.

## 7.19.4.10 setupModelFromAssets()

```
\label{thm:constraint} bool \ \mbox{FilterManagementLibrary::TFIntegration::TensorflowNNInstance::setupModelFromAssets} \ \ ( \ \mbox{AAssetManager } * \mbox{const} \ \ \mbox{assetManager} )
```

Sets up the Tensorflow model using the provieded model description.

In contrast to setupModelFromFile(), this functions interpretes the modelFile specified in the TensorflowNNModel ← Description as an Android Assets file and thus tries to load the model from the Assets belonging to the App. Of course, this is only available under Android environments. If the file could be loaded, a graph is built from it and a new Tensorflow Session will be created. On failure, lastError will be set accordingly. If the model was loaded and setup successfully, false otherwise

References applyImageInputVector(), ERROR\_COULD\_NOT\_ADD\_GRAPH\_TO\_SESSION, ERROR\_FAILE 
D\_TO\_CONSTRUCT\_NEW\_SESSION, graphDef, inputLayerName, FilterManagementLibrary::TFIntegration 
::TensorflowNNModelDescription::inputLayerNameStr, lastError, FilterManagementLibrary::TFIntegration::

TensorflowNNModelDescription::modelFile, nnModelDescription, FilterManagementLibrary::Logger::printfln(), 
FilterManagementLibrary::TensorflowAndroidJNIUtils::ReadFileToProto(), sessionOptions, and tensorflowSession.

#### 7.19.4.11 setupModelFromFile()

```
bool FilterManagementLibrary::TFIntegration::TensorflowNNInstance::setupModelFromFile ( )
```

Sets up the Tensorflow model using the provieded model description.

Tries to load the Tensorflow model from the file specified in the TensorflowNNModelDescription which is passed to the constructor. If the file could be loaded, a graph is built from it and a new Tensorflow Session will be created. On failure, lastError will be set accordingly. if the model was loaded and setup successfully, false otherwise

References ERROR\_COULD\_NOT\_ADD\_GRAPH\_TO\_SESSION, ERROR\_COULD\_NOT\_LOAD\_MODEL, E  $\leftarrow$  RROR\_FAILED\_TO\_CONSTRUCT\_NEW\_SESSION, ERROR\_INVALID\_MODEL\_FILE, FilterManagement  $\leftarrow$  Library::Utilities::fileExists(), graphDef, inputLayerName, FilterManagementLibrary::TFIntegration::Tensorflow  $\leftarrow$  NNModelDescription::inputLayerNameStr, lastError, FilterManagementLibrary::TFIntegration::TensorflowNN  $\leftarrow$  ModelDescription::modelFile, nnModelDescription, FilterManagementLibrary::Logger::printfln(), sessionOptions, setupModelFromAssets(), and tensorflowSession.

#### 7.19.5 Member Data Documentation

## 7.19.5.1 graphDef

tensorflow::GraphDef FilterManagementLibrary::TFIntegration::TensorflowNNInstance::graphDef
[protected]

Tensorflow graph (built from the model)

#### 7.19.5.2 inputLayerName

std::unique\_ptr<std::string> FilterManagementLibrary::TFIntegration::TensorflowNNInstance←::inputLayerName [protected]

Name of the input layer built from NeuronalNetwork

#### 7.19.5.3 inputTensor

tensorflow::Tensor FilterManagementLibrary::TFIntegration::TensorflowNNInstance::inputTensor
[protected]

Tensor where our input is copied to.

#### 7.19.5.4 lastError

ErrorType FilterManagementLibrary::TFIntegration::TensorflowNNInstance::lastError [protected]

Error code of the last error that happened. See TensorflowNNInstance::ErrorType

#### 7.19.5.5 nnModelDescription

TensorflowNNModelDescription FilterManagementLibrary::TFIntegration::TensorflowNNInstance←
::nnModelDescription [private]

Everything we need to load, build and run the model. See TensorflowNNModelDescription for a detailed description.

# 7.19.5.6 numThreads

int FilterManagementLibrary::TFIntegration::TensorflowNNInstance::numThreads = 1 [private]

Number of threads Tensorflow is allowed to use. Standard value is 1, but can be set via constructor.

# 7.19.5.7 outputTensors

std::vector<tensorflow::Tensor> FilterManagementLibrary::TFIntegration::TensorflowNNInstance← ::outputTensors [protected]

Stores the result of the inference

#### 7.19.5.8 sessionOptions

 $tensorflow:: SessionOptions \ Filter Management Library:: TFIntegration:: TensorflowNNInstance \\ \\ :: sessionOptions \ [protected]$ 

Options for the session holding the model

#### 7.19.5.9 tensorflowSession

std::unique\_ptr<tensorflow::Session> FilterManagementLibrary::TFIntegration::TensorflowNN← Instance::tensorflowSession [protected]

Stores and holds the instance of the used model

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
   NNInstance.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow NNInstance.cpp

# 7.20 FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier Class Reference

#include <TensorflowNNInstanceClassifier.h>

Inheritance diagram for FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier:

FilterManagementLibrary::TFIntegration::TensorflowNNInstance

FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier

# **Public Member Functions**

• bool readLabelsFile (std::string path)

Reads labels from a file.

bool getTopNPredictions (const int numPredictions, const float threshold, std::vector< std::pair< float, int > \*outputList)

Return the top N predictions of the network and their confidences.

bool getTopNPredictionsStringVector (const int numPredictions, const float threshold, std::vector< std::string > \*outputList)

Return the top N labels of the network and their confidences.

• TensorflowNNInstanceClassifier (TensorflowNNModelDescription nnModelDescription, int numThreads=1)

Constructor of TensorflowNNInstanceClassifier.

# **Private Attributes**

- std::vector< std::string > loadedLabels
- · int labelCount

#### **Additional Inherited Members**

# 7.20.1 Detailed Description

This is just a very basic class to fastly implement classification models like MobilenetV1/2 or Inception(V3) as they use the same method for output encoding (check getTopNPredictions(...) for more). It is not used in any of the productive PipeSystem Code, so see it more as an example or class for debugging and testing.

It only needs to be provided a correct label file for the corresponding trained model and a correct model description. Afterwards, getTopNPredictions(...) and getTopNPredictionsStringVector(...) can be used for easy output interpretation (don't forget to call runInference() first though).

# 7.20.2 Constructor & Destructor Documentation

## 7.20.2.1 TensorflowNNInstanceClassifier()

Constructor of TensorflowNNInstanceClassifier.

Basically just calls the TensorflowNNInstance super constructor, which you may refer to for a more detailed description.

See also

# **Parameters**

TensorflowNNModelDescription	nnModelDescription the model description containing all information needed to load and use the Tensorflow model.
int	numThreads number of threads Tensorflow is allowed to use.

References readLabelsFile().

#### 7.20.3 Member Function Documentation

# 7.20.3.1 getTopNPredictions()

```
\verb|bool FilterManagementLibrary:: TFIntegration:: TensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflowNNInstanceClassifier:: getTopNPredictions (|continuous tensorflow) (|continuous tensorflow) (|continuous tensorflow) (|continuous tensorflow) (|c
```

```
const int numPredictions,
const float threshold,
std::vector< std::pair< float, int > > * outputList )
```

Return the top N predictions of the network and their confidences.

Returns the top N confidence values over threshold in the provided vector, sorted by confidence in descending order.

#### **Parameters**

const	int numPredictions number of predictions that the outputList shall contain (the 'N' in getTopNPredictions())
const	float threshold min confidence value a prediction needs to have to be included in the outputList
std::vector <std::pair<float,int></std::pair<float,int>	>* outputList pointer to a vector where the results will be written to.

 $References \ get Top NP redictions String Vector(), \ i, \ and \ Filter Management Library :: TFIntegration :: Tensor flow NN \leftarrow Instance :: output Tensors.$ 

#### 7.20.3.2 getTopNPredictionsStringVector()

Return the top N labels of the network and their confidences.

Uses getTopNPredictions(...) to get a list of the top N confidence predictions and their confidence values, and then uses the ID's to get the corresponding label tag from the labels file which should have been loaded prior to calling this function.

# Parameters

const	int numPredictions number of predictions that the outputList shall contain (the 'N' in getTopNPredictions())
const	float threshold min confidence value a prediction needs to have to be included in the outputList
std::vector <std::string>*</std::string>	outputList pointer to a vector where the labels will be written to.

References getTopNPredictions(), labelCount, and loadedLabels.

# 7.20.3.3 readLabelsFile()

```
\label{thm:continuous} bool \ \mbox{FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier::readLabelsFile ( \mbox{ std::string } path \mbox{ )}
```

Reads labels from a file.

The specified file will be read line by line whereas each line represents an entry in the loadedLabels vector.

#### **Parameters**

std::string	path path to the labels file.
-------------	-------------------------------

#### Returns

bool true if the labels file exists and could be read, false otherwise

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_TFNN\_CLASSIFIER\_INV 
ALID\_LABELS\_FILE, getTopNPredictions(), labelCount, FilterManagementLibrary::TFIntegration::TensorflowNN 
Instance::lastError, and loadedLabels.

#### 7.20.4 Member Data Documentation

#### 7.20.4.1 labelCount

int FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier::labelCount [private]

Number of labels that were read. Assume these are equal to the output size of the used model!

## 7.20.4.2 loadedLabels

std::vector<std::string> FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier←::loadedLabels [private]

List of labels loaded from file

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
   NNInstanceClassifier.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
   NNInstanceClassifier.cpp

# 7.21 FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription Struct Reference

#include <TensorflowNNModelDescription.h>

#### **Public Attributes**

- bool input\_floating = false
- float input mean = 127.5f
- float input std = 127.5f
- · int inputWidth
- · int inputHeight
- int channels = 3
- std::string modelFile
- std::string inputLayerNameStr
- std::vector< std::string > outputLayerNames

#### 7.21.1 Member Data Documentation

#### 7.21.1.1 channels

int FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::channels = 3

Amount of channels of the input layer (channels(RGB) = 3) Although if it may be flattened, Tensorflow provides functions to map tensors to access the data using x, y and z coordinates, which is what is used in  $TensorflowNN \leftarrow Instance::applyImageInputVector(...)$ 

#### 7.21.1.2 input\_floating

bool FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input\_floating =
false

Specifys whether the input type of the neuronal network model is tensorflow::DT\_UINT8 or tensorflow::DT\_FLOAT (true means float, false means uint8). Others are not supported for now.

## 7.21.1.3 input\_mean

float FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input\_mean =  $127. \leftarrow 5.5$ 

Only used if input\_floating = true. As the default operations of the FilterManagementLibrary are implemented to use uint8\_t, we need a way to convert those values to their float counterparts. For this, we need a mean and a std value for normalization. Formula is: float\_value = (uint8\_t\_value - input\_mean) / input\_std;

#### 7.21.1.4 input\_std

float FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input\_std = 127.5f

Only used if input\_floating = true. As the default operations of the FilterManagementLibrary are implemented to use uint8\_t, we need a way to convert those values to their float counterparts. For this, we need a mean and a std value for normalization. Formula is: float\_value = (uint8\_t\_value - input\_mean) / input\_std;

#### 7.21.1.5 inputHeight

int FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::inputHeight

Height of the input layer. Although if it may be flattened, Tensorflow provides functions to map tensors to access the data using x, y and z coordinates, which is what is used in TensorflowNNInstance::applyImageInputVector(...)

## 7.21.1.6 inputLayerNameStr

 $\verb|std::string| Filter Management Library:: TFIntegration:: Tensor flow NNM odel Description:: input Layer \leftarrow Name Str|$ 

Name of the input layer of the Neuronal Network model.

#### 7.21.1.7 inputWidth

Width of the input layer. Although if it may be flattened, Tensorflow provides functions to map tensors to access the data using x, y and z coordinates, which is what is used in TensorflowNNInstance::applyImageInputVector(...)

#### 7.21.1.8 modelFile

std::string FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::modelFile

Path to the protobuf model file which contains the model and where the Tensorflow graph is built from.

# 7.21.1.9 outputLayerNames

 $\verb|std::vector| < \verb|std::string| > Filter Management Library:: TFIntegration:: Tensor flow NNM odel Description \\ \leftarrow :: output Layer Names$ 

List of names of the output layers that shall be used during inference. For output interpretation, you can also use these names to get the corresponding output tensor from a TensorflowResultContainer. But if your network model has really silly or annoying 1042 character long names, be aware that you could also use an ID tou get the corresponding tensor form TensorflowResultContainer. The ID corresponds to the position of the layer in this vector, so be sure to remember it!

The documentation for this struct was generated from the following file:

RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
 NNModelDescription.h

# 7.22 FilterManagementLibrary::TensorflowOpenCVUtils Class Reference

#include <TensorflowOpenCVUtils.h>

#### **Static Public Member Functions**

- static void fastApplyCVMatOnInputTensorUInt8 (cv::Mat \*mat, tensorflow::Tensor \*inputTensor)

  Applies the raw data of a openCV mat to an Tensorflow input tensor.
- static void fastApplyCVMatOnInputTensorFloat (cv::Mat \*mat, tensorflow::Tensor \*inputTensor, float mean, float std)

Applies the raw data of a openCV mat to an Tensorflow input tensor.

# 7.22.1 Detailed Description

Some utilities to efficiently use OpenCV with the Filter management library. Remember YOU DON'T NEED THIS TO USE THE FILTER MANAGEMENT LIBRARY!!

#### 7.22.2 Member Function Documentation

#### 7.22.2.1 fastApplyCVMatOnInputTensorFloat()

Applies the raw data of a openCV mat to an Tensorflow input tensor.

Does the same as fastApplyCVMatOnInputTensorUInt8, but additionally converts the uint8\_t values to their corresponding float values using a mean and std value.

If you use this, make sure your CV Mat is in the right format, e.g. 8UC3. We could use a 32FC3 CV Float Mat for this, but we may wan't to apply our own std and mean values (don't know maybe we don't want to use 127.5f or vice versa), that's why we expect a normal uint based BGR Matrix). Also make sure mat and inputTensor have the same dimension!! We don't do any safety checks here, because we want to be *fast* 

You may check TFNNBasedPipeFilter class description for how to use this function easily in a TFNNBasedPipeFilter (it's a one liner basically!).

#### **Parameters**

cv::Mat	*mat pointer to the OpenCV matrix from which the data will be copied from
tensorflow::Tensor	*inputTensor Tensorflow Tensor where the data will be copied to.

# 7.22.2.2 fastApplyCVMatOnInputTensorUInt8()

Applies the raw data of a openCV mat to an Tensorflow input tensor.

Uses OpenCV's forEach<...>() function to *fastly* iterate over all pixels of the OpenCV Mat and copy them to the input tensor. As forEach<...>() uses multi threading, this method is *WAY* faster (5 - 10 times in my tests) than naive per pixel copying.

If you use this, make sure your CV Mat is in the right format, e.g. 8UC3. Also make sure mat and inputTensor have the same dimension!! We don't do any safety checks here, because we want to be *fast* (although it's not called "fast" because it lacks safety, I want to notice).

You may check TFNNBasedPipeFilter class description for how to use this function easily in a TFNNBasedPipeFilter (it's a one liner basically!).

#### **Parameters**

cv::Mat	*mat pointer to the OpenCV matrix from which the data will be copied from
tensorflow::Tensor	*inputTensor Tensorflow Tensor where the data will be copied to.

References fastApplyCVMatOnInputTensorFloat().

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowOpenCVUtils.h
- RoadSignAPI Standalone Tensorflow/source files/FilterManagementLibrary/TensorflowOpenCVUtils.cpp

# 7.23 FilterManagementLibrary::TFIntegration::TensorflowResultContainer Class Reference

#include <TensorflowResultContainer.h>

#### **Public Member Functions**

TensorflowResultContainer (std::vector< tensorflow::Tensor > \*outputTensors, std::vector< std::string > outputTensorNames)

Constructor of TensorflowResultContainer All information / data contained in an object of type TensorflowResultContainer will be set using this constructor. It does not contain any other class member variables!

tensorflow::Tensor \* getOutputTensorByID (int id) const

Returns a pointer to the output Tensor with the given ID.

tensorflow::Tensor \* getOutputTensorByLayerName (std::string layerName) const

Returns a pointer to the output Tensor at the given layer name.

# **Private Attributes**

- std::vector< tensorflow::Tensor > \* outputTensors
- std::vector< std::string > outputTensorNames

### 7.23.1 Detailed Description

This class is a container for the results of a TensorflowNNInstance. It wraps the output tensors of an inference and their corresponding names so the internal structurization of those tensors can be hidden. It allows to get the Tensor of interest by name, so one does not have to remember the ID (which is the order of the output layer names provided in the TensorflowNNModelDescription used to create the TensorflowNNInstance.

#### 7.23.2 Constructor & Destructor Documentation

#### 7.23.2.1 TensorflowResultContainer()

Constructor of TensorflowResultContainer All information / data contained in an object of type TensorflowResult← Container will be set using this constructor. It does not contain any other class member variables!

#### **Parameters**

std::vector <tensorflow::tensor>*</tensorflow::tensor>	outputTensors the list of output tensors of the corresponding inference (the Tensors containing the result)
std::vector <std::string></std::string>	outputTensorNames layer names of the output tensors. The names <i>HAVE</i> to be in the same order as the outputTensors vector! Otherwise a wrong tensor would be returned when using the getOutTensorByLayerName() function.

References getOutputTensorByID().

#### 7.23.3 Member Function Documentation

#### 7.23.3.1 getOutputTensorByID()

```
\label{tensorflow::Tensor} tensorflow::TensorflowResultContainer::get \leftarrow \\ Output TensorBy ID \ ( \\ int \ id \ ) \ const
```

Returns a pointer to the output Tensor with the given ID.

#### Returns

tensorflow::Tensor the output Tensor with the given ID.

References getOutputTensorByLayerName(), and outputTensors.

92 Class Documentation

#### 7.23.3.2 getOutputTensorByLayerName()

```
\label{tensorflow::Tensor} tensorflow:: TensorflowResultContainer:: get \\ \\ Output Tensor By Layer Name ( \\ std:: string \textit{layerName} ) const
```

Returns a pointer to the output Tensor at the given layer name.

#### Returns

tensorflow::Tensor the output Tensor at the given layer name.

References outputTensorNames, and outputTensors.

#### 7.23.4 Member Data Documentation

#### 7.23.4.1 outputTensorNames

```
std::vector<std::string> FilterManagementLibrary::TFIntegration::TensorflowResultContainer←::outputTensorNames [private]
```

List of output tensors provided by the TensorflowNNModelDescription of the corresponding TensorflowNNInstancce. The order of the tensor names here matches the order of the tensor names in the model description! So the ID of the first output layer name in the model description is also the ID of the corresponding output tensor in this vector.

### 7.23.4.2 outputTensors

```
std::vector<tensorflow::Tensor>* FilterManagementLibrary::TFIntegration::TensorflowResult← Container::outputTensors [private]
```

List of output tensors created by TensorflowNNInstance according to the output layer names wich were provided in TensorflowNNModelDescription. TensorflowNNInstance assures that the order of output layer names provided matches the order of the output tensors. So the ID of the first output layer name in the model description is also the ID of the corresponding output tensor in this vector.

The documentation for this class was generated from the following files:

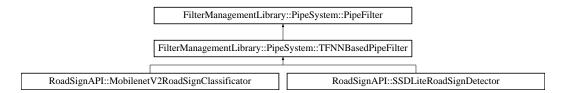
- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
   — ResultContainer.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/TensorflowIntegration/Tensorflow
   — ResultContainer.cpp

### 7.24 FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter Class Reference

Abstract template class to derive Tensorflow based filters from.

#include <TFNNBasedPipeFilter.h>

Inheritance diagram for FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter:



#### **Public Member Functions**

- TFNNBasedPipeFilter (TFIntegration::TensorflowNNModelDescription nnModelDescription, int numThreads) Standard constructor of TFNNBasedPipeFilter.
- void adjustModelFile (std::string modelFile)

Allows to change ONLY the model file path set in the model description.

bool setupModelFromFile ()

Set's up the Tensorflow model of the TensorflowNNInstance.

• bool setupModelFromAssets (AAssetManager \*const assetManager)

Set's up the Tensorflow model of the TensorflowNNInstance.

virtual ~TFNNBasedPipeFilter ()

Destructor of TFNNBasedPipeFilter. Does nothing by now.

#### **Protected Member Functions**

- bool applyNNImageInputVector (uint8\_t \*\*\*image, const int height, const int width, const int channels)
   Applies an image vector to the TensorflowNNInstance.
- bool evaluateInputVectorByNN ()

Let's the neuronal network evaluate the given input.

tensorflow::Tensor \* getNNInputTensor ()

Returns the input tensor of the underlaying neuronal network.

const FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription \* getNNModelDescription ()
 const

Returns the model description of the underlaying neuronal network.

• TFIntegration::TensorflowNNInstance::ErrorType getNeuralNetLastError () const

Returns the last error that happened concerning the neuronal instance.

virtual void onNNEvaluationFinished (const TFIntegration::TensorflowResultContainer resultContainer)=0

#### **Private Attributes**

TFIntegration::TensorflowNNInstance tfNNInstance

94 Class Documentation

#### **Additional Inherited Members**

### 7.24.1 Detailed Description

Abstract template class to derive Tensorflow based filters from.

A TFNNBasedPipeFilter is a special type of filter (thus derived from PipeFilter) already providing some functions to easily implemented filters based on Tensorflow Neuronal Networks. Strictly speaking the main processing of these filters is done by a preloaded Tensorflow model. The filter may take a part of the working data set and feed it into the model using the output of it's inference to generate the result of this filter. If you use it, make sure to call evaluateInputVectorByNN() in order to let the neuronal network do it's predictions.

For this, the TFNNBasedPipeFilter abstract class uses a TensorflowNNInstance (TensorflowNNInstanceClassifier is not used, as this is just a test class for basic classification models with a predefined way of interpreting their outputs, which is not assured to be a general way to interprete the output other classificator networks may use).

It provides a predefined easy way to apply input vectors from a 3 dimensional array of uint8\_t's (applyNNImage InputVector). If the underlying model expects floats however, the conversion will be done automatically (see TensorflowNNInstance for more details). If a more specific way to apply an input is needed, the function get InputTensor() is provided which provides a direct access to the input tensor of the Neural Network Instance. Also keep in mind we provide some utility functions for easy OpenCV integration! In detail, offering some functions to directly apply a cv::Mat to an input tensor (see TensorflowOpenCVUtils).

A basic example to performantely effective apply a matrix to the underlying neuronal network, expecting uint8\_t as input vector, from within a derivate of TFNNBasedPipeFilter would be: TensorflowOpenCVUtils::fastApplyCVMat OnInputTensorUInt8(&yourMat, this->getNNInputTensor()); simple as that!

However, keep in mind TensorflowOpenCVUtils do NOT have to be compiled with and linked to a exectuable or libary using the FilterManagementLibrary, if for example you don't use OpenCV (Mat's as input methods).

### 7.24.2 Constructor & Destructor Documentation

#### 7.24.2.1 TFNNBasedPipeFilter()

```
\label{thm:continuous} Filter \texttt{ManagementLibrary::PipeSystem::TFNNBasedPipeFilter::TFNNBasedPipeFilter::TFNNBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBasedPipeFilter::TFNnBas
```

Standard constructor of TFNNBasedPipeFilter.

The TFNNBasedPipeFilter will use a TensorflowNNInstance to load and instantiate Tensorflow model (and a corresponding session). For this, it needs to tell the TensorflowNNInstance a description of a model to use (Tensorflow←NNModelDescription) and how many threads Tensorflow is allowed to use for computation.

#### **Parameters**

TensorflowNNModelDescription	nnModelDescription a description of the Tensorflow model which will be used by the filter.
int	numThreads number of threads Tensorflow is allowed to use for computation.

References applyNNImageInputVector().

#### 7.24.2.2 ~TFNNBasedPipeFilter()

```
FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::~TFNNBasedPipeFilter ( ) [virtual]
```

Destructor of TFNNBasedPipeFilter. Does nothing by now.

#### 7.24.3 Member Function Documentation

#### 7.24.3.1 adjustModelFile()

```
\label{thm:cond} \mbox{void FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::adjustModelFile (} \\ std::string \mbox{ modelFile )}
```

Allows to change ONLY the model file path set in the model description.

Can be used to adjust the path of the model file of the TensorflowNNModelDescription at run-time. TensorflowN $\leftarrow$  NInstance's adjustModelFile(...) function will be called accordingly.

#### **Parameters**

```
std::string new path to the modelFile to use.
```

#### Returns

void

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::adjustModelFile(), getNeuralNet  $\leftarrow$  LastError(), and tfNNInstance.

#### 7.24.3.2 applyNNImageInputVector()

Applies an image vector to the TensorflowNNInstance.

This takes a 3D uint8\_t vector and will call the TensorflowNNInstance's applyImageInputVector(...)function accordingly, which will set the data on the input Tensor of the model. It originally was intended to be used for 3 Channel RGB images, as the TensorflowNNInstance's function may scale the input as needed (i.e. normalization using floats and a mean and std value).

96 Class Documentation

#### **Parameters**

uint8_t***	image 3D array of uint8_t values which will be applied to the model
int	height height of the image (columns of the vector)
int	width width of the image (rows of the vector)
int	channels channels of the image (depth of the vector)

#### Returns

bool true if the vector was successfully applied to the model, false otherwise

References evaluateInputVectorByNN(), and tfNNInstance.

### 7.24.3.3 evaluateInputVectorByNN()

bool FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::evaluateInputVectorByNN ( )
[protected]

Let's the neuronal network evaluate the given input.

An input vector should have been applied to the neuronal network (the TensorflowNNInstance) beforehand, by any means. This will call the TensorflowNNInstance's runInference() function accordingly and will call the onNN EvaluationFinished() callback function on success (synchronously).

#### Returns

bool true if the inference (TensorflowNNInstance) was successfull, false otherwise.

References getNNInputTensor(), FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getResult  $\leftarrow$  Container(), onNNEvaluationFinished(), FilterManagementLibrary::Logger::printfln(), FilterManagementLibrary:: $\leftarrow$  TFIntegration::TensorflowNNInstance::runInference(), and tfNNInstance.

#### 7.24.3.4 getNeuralNetLastError()

FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ErrorType FilterManagement ← Library::PipeSystem::TFNNBasedPipeFilter::getNeuralNetLastError () const [protected]

Returns the last error that happened concerning the neuronal instance.

#### Returns

TensorflowNNInstance::ErrorType the enum value of the last error that happened concering the Tensorflow← NNInstance

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getLastError(), tfNNInstance, and  $\sim$ TFNNBasedPipeFilter().

### 7.24.3.5 getNNInputTensor()

```
tensorflow::Tensor * FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::getNNInput←
Tensor () [protected]
```

Returns the input tensor of the underlaying neuronal network.

As we want to hide the object of TensorflowNNInstance from a derived filter, we provide this function to directly access the input tensor of the underlaying tensorflow neuronal network.

#### Returns

tensorflow::Tensor\* pointer to the input tensor

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getInputTensor(), getNNModel ← Description(), and tfNNInstance.

#### 7.24.3.6 getNNModelDescription()

```
const FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription * FilterManagement ←
Library::PipeSystem::TFNNBasedPipeFilter::getNNModelDescription ( ) const [protected]
```

Returns the model description of the underlaying neuronal network.

As we want to hide the object of TensorflowNNInstance from a derived filter, we provide this function to a const pointer to the model description of the underlaying neuronal network.

#### Returns

FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription\* pointer to the model description.

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getModelDescription(), setupModel $\leftarrow$  FromFile(), and tfNNInstance.

### 7.24.3.7 onNNEvaluationFinished()

 $Implemented\ in\ RoadSignAPI::Mobilenet V2RoadSignClassificator,\ and\ RoadSignAPI::SSDLiteRoadSignDetector.$ 

#### 7.24.3.8 setupModelFromAssets()

```
\label{thm:poster} bool \ \mbox{FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::setupModelFromAssets} \ \ ( \\ \mbox{AAssetManager *const } assetManager \ )
```

Set's up the Tensorflow model of the TensorflowNNInstance.

CAUTION: THIS FUNCTION IS ONLY AVAILABLE UNDER ANDROID ENVIRONMENTS! This will call TensorflowNNInstance's setupModelFromAssets() function accordingly, which will try to load and build the Tensorflow model from the given TensorflowNNModelDescription (passed to constructor). In contrast to setupModel FromFile(), the modelFile given in the TensorflowNNModelDescription will be threated as an android assets file and loaded via an AssetsManager which needs to be passed to this function.

98 Class Documentation

#### **Parameters**

const

AAssetManager\* assetManager Pointer to an AssetManager which, by any means, needs to to be passed from the Java side of the code.

#### Returns

bool true if network model was setup successfully, false otherwise

References adjustModelFile(), FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_COUL  $\leftarrow$  D\_NOT\_ADD\_GRAPH\_TO\_SESSION, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERRO $\leftarrow$  R\_COULD\_NOT\_LOAD\_MODEL, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_F  $\leftarrow$  Alled\_TO\_CONSTRUCT\_NEW\_SESSION, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::E  $\leftarrow$  RROR\_INVALID\_MODEL\_FILE, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getLastError(), FilterManagementLibrary::TFIntegration::TensorflowNNInstance  $\leftarrow$  ::setupModelFromAssets(), and tfNNInstance.

#### 7.24.3.9 setupModelFromFile()

```
\verb|bool FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter::setupModelFromFile ( ) \\
```

Set's up the Tensorflow model of the TensorflowNNInstance.

This will call TensorflowNNInstance's setupModelFromFile() function accordingly, which will try to load and build the Tensorflow model from the given TensorflowNNModelDescription (passed to constructor)

#### Returns

bool true if network model was setup successfully, false otherwise

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_COULD\_NOT\_ADD\_G RAPH\_TO\_SESSION, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_COULD\_N OT\_LOAD\_MODEL, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_FAILED\_TO\_ CONSTRUCT\_NEW\_SESSION, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR\_IN VALID\_MODEL\_FILE, FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getLastError(), Filter ManagementLibrary::Logger::printfln(), setupModelFromAssets(), FilterManagementLibrary::TFIntegration:: TensorflowNNInstance::setupModelFromFile(), and tfNNInstance.

#### 7.24.4 Member Data Documentation

#### 7.24.4.1 tfNNInstance

TFIntegration::TensorflowNNInstance FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter← ::tfNNInstance [private]

The filter local instance of the underlying tensorflow neuronal network model.

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/PipeSystem/TFNNBased←
   PipeFilter.cpp

### 7.25 FilterManagementLibrary::Utilities Class Reference

Providing some basic utilities, like checking if a file exists.

```
#include <Utilities.h>
```

#### **Static Public Member Functions**

• static bool fileExists (const std::string &name)

Check if a file with the given path exists.

### 7.25.1 Detailed Description

Providing some basic utilities, like checking if a file exists.

### 7.25.2 Member Function Documentation

#### 7.25.2.1 fileExists()

Check if a file with the given path exists.

### **Parameters**

const std::string& name string containing the path of the file

### Returns

true if the file exists, false if the file does not exist or if we have insufficient permission to read it.

The documentation for this class was generated from the following files:

- RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Utilities.h
- RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/Utilities.cpp

100 Class Documentation

## **Chapter 8**

## **File Documentation**

- 8.1 RoadSignAPI\_Standalone\_Tensorflow/\_10\_introduction.dox File Reference
- 8.2 RoadSignAPI\_Standalone\_Tensorflow/codesnippets.cpp File Reference

#### **Functions**

- Mat scaledImage (settings.inputHeigth, settings.inputWidth, CV\_32FC3)
- cv::Mat cameralmg (settings.inputHeigth, settings.inputWidth, CV\_32FC3, p)
- scaledImage convertTo (cameralmg, CV\_32FC3)

### **Variables**

- OpenCV mat to tensorflow with CV FakeMat cv::Mat image = cv::imread("grace\_hopper.bmp")
- gettimeofday & start\_time
- int i = 0
- float \* p = tflnstance.getInputTensor()->flat<float>().data()

#### 8.2.1 Function Documentation

### 8.2.1.1 cameralmg()

### 8.2.1.2 convertTo()

### 8.2.1.3 scaledImage()

### 8.2.2 Variable Documentation

```
8.2.2.1 i
```

int i = 0

### 8.2.2.2 image

```
OpenCV mat to tensorflow with CV FakeMat cv::Mat image = cv::imread("grace_hopper.bmp")
```

#### 8.2.2.3 p

```
float* p = tfInstance.getInputTensor()->flat<float>().data()
```

### 8.2.2.4 start\_time

gettimeofday& start\_time

### Classes

 $\bullet \ class\ tensorflow:: and roid:: Limiting File Input Stream$ 

### **Namespaces**

- · tensorflow
- · tensorflow::android
- 8.4 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

  Logger.h File Reference

### Classes

· class FilterManagementLibrary::Logger

Logger providing Desktop Linux <-> Android platform independent logging.

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

8.5 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Pipe

System/PipeFilter.h File Reference

#### **Classes**

class FilterManagementLibrary::PipeSystem::PipeFilter

Base class were all filters are derived from.

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

• FilterManagementLibrary::PipeSystem

Everything that has to do with the pipes and filters part of the library.

8.6 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Pipe

System/PipeRegisteredFilters.h File Reference

### Classes

struct FilterManagementLibrary::PipeSystem::PipeRegisteredFilters
 Template struct.

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

• FilterManagementLibrary::PipeSystem

Everything that has to do with the pipes and filters part of the library.

8.7 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Pipe

System/PipeWorkingDataSet.h File Reference

#### Classes

• struct FilterManagementLibrary::PipeSystem::PipeWorkingDataSet

Template structfor PipeWorkingDataSets.

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

FilterManagementLibrary::PipeSystem

Everything that has to do with the pipes and filters part of the library.

8.8 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Pipe

System/ProcessingPipeline.h File Reference

### Classes

· class FilterManagementLibrary::PipeSystem::ProcessingPipeline

Pipeline of the pipes and filters architecture.

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

 $\bullet \ \ Filter Management Library :: Pipe System$ 

Everything that has to do with the pipes and filters part of the library.

8.9 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/Pipe

System/TFNNBasedPipeFilter.h File Reference

### Classes

· class FilterManagementLibrary::PipeSystem::TFNNBasedPipeFilter

Abstract template class to derive Tensorflow based filters from.

Namespaces

· FilterManagementLibrary

Everything the FilterManagementLibrary provides.

• FilterManagementLibrary::PipeSystem

Everything that has to do with the pipes and filters part of the library.

8.10 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

TensorflowAndroidJNIUtils.h File Reference

#### **Classes**

· class FilterManagementLibrary::TensorflowAndroidJNIUtils

### **Namespaces**

- · google
- · google::protobuf
- FilterManagementLibrary

Everything the FilterManagementLibrary provides.

8.11 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/
TensorflowIntegration/TensorflowNNInstance.h File Reference

### Classes

· class FilterManagementLibrary::TFIntegration::TensorflowNNInstance

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

- FilterManagementLibrary::TFIntegration
- 8.12 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

  TensorflowIntegration/TensorflowNNInstanceClassifier.h File Reference

#### **Classes**

class FilterManagementLibrary::TFIntegration::TensorflowNNInstanceClassifier

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

· FilterManagementLibrary::TFIntegration

8.13 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

TensorflowIntegration/TensorflowNNModelDescription.h File Reference

#### Classes

• struct FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription

#### **Namespaces**

- FilterManagementLibrary
  - Everything the FilterManagementLibrary provides.
- · FilterManagementLibrary::TFIntegration
- 8.14 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

  TensorflowIntegration/TensorflowResultContainer.h File Reference

### Classes

· class FilterManagementLibrary::TFIntegration::TensorflowResultContainer

### **Namespaces**

- FilterManagementLibrary
  - Everything the FilterManagementLibrary provides.
- · FilterManagementLibrary::TFIntegration
- 8.15 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

  TensorflowOpenCVUtils.h File Reference

### **Classes**

· class FilterManagementLibrary::TensorflowOpenCVUtils

### **Namespaces**

FilterManagementLibrary

Everything the FilterManagementLibrary provides.

8.16 RoadSignAPI\_Standalone\_Tensorflow/header\_files/FilterManagementLibrary/

Utilities.h File Reference

#### **Classes**

· class FilterManagementLibrary::Utilities

Providing some basic utilities, like checking if a file exists.

### **Namespaces**

• FilterManagementLibrary

Everything the FilterManagementLibrary provides.

8.17 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/DetectedSign← Combination.h File Reference

### **Classes**

· class RoadSignAPI::DetectedSignCombination

### **Namespaces**

- RoadSignAPI
- 8.18 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/DetectedSign 

  Descriptor.h File Reference

### **Classes**

struct RoadSignAPI::DetectedSignDescriptor

### **Namespaces**

- RoadSignAPI
- 8.19 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/Classified 

  SignsGrouper.h File Reference

### Classes

• class RoadSignAPI::ClassifiedSignsGrouper

### **Namespaces**

- RoadSignAPI
- 8.20 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/Detection ← BasedImageSlicer.h File Reference

#### **Classes**

· class RoadSignAPI::DetectionBasedImageSlicer

### **Namespaces**

- RoadSignAPI
- 8.21 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/Mobilenet 

  V2RoadSignClassificator.h File Reference

#### Classes

· class RoadSignAPI::MobilenetV2RoadSignClassificator

### **Namespaces**

- RoadSignAPI
- 8.22 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/Road

  SignDuplicationDeleter.h File Reference

### Classes

· class RoadSignAPI::RoadSignDuplicationDeleter

### **Namespaces**

- RoadSignAPI
- 8.23 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/Filters/SSDLite

  RoadSignDetector.h File Reference

### Classes

class RoadSignAPI::SSDLiteRoadSignDetector

- RoadSignAPI
- 8.24 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/RoadSignAPI.h File Reference

#### Classes

• class RoadSignAPI::RoadSignAPI

### **Namespaces**

- RoadSignAPI
- 8.25 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/RSAPIPipe

  RegisteredFilters/RSAPIPipeRegisteredFilters.h File Reference

#### Classes

• struct RoadSignAPI::RSAPIPipeRegisteredFilters

### **Namespaces**

- RoadSignAPI
- 8.26 RoadSignAPI\_Standalone\_Tensorflow/header\_files/RoadSignAPI/RSAPIWorking DataSet/RSAPIWorkingDataSet.h File Reference

### Classes

• struct RoadSignAPI::RSAPIWorkingDataSet

### **Namespaces**

- RoadSignAPI
- 8.27 RoadSignAPI\_Standalone\_Tensorflow/main.cpp File Reference

### **Typedefs**

typedef Point3\_< uint8\_t > Pixel

### **Functions**

- double get us (struct timeval t)
- bool readLabelsFile (std::string path, std::vector< std::string > \*output)
- bool openCVMatTouint8\_tArray (cv::Mat \*mat, uint8\_t \*output, int \*width, int \*heigth)
- bool openCVMatTouint8\_t3DArray (cv::Mat \*mat, uint8\_t \*\*\*output, int \*width, int \*heigth)
- bool applyFromCVMat (cv::Mat \*mat, tensorflow::Tensor \*inputTensor)
- std::string type2str (int type)
- int main ()

### 8.27.1 Typedef Documentation

```
8.27.1.1 Pixel

typedef Point3_<uint8_t> Pixel
```

### 8.27.2 Function Documentation

```
8.27.2.1 applyFromCVMat()
```

struct timeval t )

References get\_us().

8.27.2.2 get\_us()

```
double get_us (
```

### 8.27.2.3 main()

```
int main ( )
```

References RoadSignAPI::DetectedSignCombination::addDetectedSign(), RoadSignAPI::DetectedSignDescriptor::classifierApprovedClassID, RoadSignAPI::DetectedSignDescriptor::detectorConfidence, RoadSignAPI::DetectedSignCombination::getDetectedSignsAmount(), RoadSignAPI::DetectedSignCombination::getGestimatedPolePositionX(), RoadSignAPI::DetectedSignCombination::getSignsInCombination(), i, RoadSignAPI::DetectedSignDescriptor::lowerRight, readLabelsFile(), scaledImage(), RoadSignAPI::RoadSignAPI::staticFeedImage(),
RoadSignAPI::RoadSignAPI::staticGetDetectedSignCombinations(), RoadSignAPI::RoadSignAPI::staticInit(),
type2str(), and RoadSignAPI::DetectedSignDescriptor::upperLeft.

### 8.27.2.4 openCVMatTouint8\_t3DArray()

### 8.27.2.5 openCVMatTouint8\_tArray()

### 8.27.2.6 readLabelsFile()

#### 8.27.2.7 type2str()

### 8.28 RoadSignAPI\_Standalone\_Tensorflow/mainold.cpp File Reference

### **Typedefs**

typedef Point3\_< uint8\_t > Pixel

### **Functions**

- double get\_us (struct timeval t)
- bool openCVMatTouint8\_tArray (cv::Mat \*mat, uint8\_t \*output, int \*width, int \*heigth)
- bool openCVMatTouint8\_t3DArray (cv::Mat \*mat, uint8\_t \*\*\*output, int \*width, int \*heigth)
- std::string type2str (int type)
- static void GetTopN (const Eigen::TensorMap< Eigen::Tensor< float, 1, Eigen::RowMajor >, Eigen::Aligned > &prediction, const int num\_results, const float threshold, std::vector< std::pair< float, int > > \*top\_results)
- void testNet ()
- int main ()

### 8.28.1 Typedef Documentation

```
8.28.1.1 Pixel

typedef Point3_<uint8_t> Pixel
```

#### 8.28.2 Function Documentation

### 8.28.2.3 main()

```
int main ( )
```

References FilterManagementLibrary::TFIntegration::TensorflowNNInstance::applyImageInputVector(), ManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR COULD NOT ADD GRAPH TO SES← FilterManagementLibrary::TFIntegration::TensorflowNNInstance::ERROR COULD NOT LOAD MO← SION.  $Filter Management Library:: TF Integration:: Tensorflow NN Instance:: ERROR\_FAILED\_TO\_CONSTRUCT \leftarrow TABLE TO_CONSTRUCT TO TABLE TO THE TABLE TO THE$ DEL, \_NEW\_SESSION,  $Filter Management Library :: TFIntegration :: Tensor flow NN Instance :: ERROR\_INVALID\_MOD {\leftarrow}$ EL\_FILE, get\_us(), FilterManagementLibrary::TFIntegration::TensorflowNNInstance::getLastError(), i, image, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input floating, FilterManagement← Library::TFIntegration::TensorflowNNModelDescription::input\_mean,  $Filter Management Library :: TFIntegration \leftarrow$ ::TensorflowNNModelDescription::input std,  $Filter Management Library :: TFIntegration :: Tensor flow NNM odel \leftarrow$ Description::inputLayerNameStr, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::input← Width, FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::modelFile, openCVMatTouint8← \_t3DArray(), FilterManagementLibrary::TFIntegration::TensorflowNNModelDescription::outputLayerNames, Filter $\hookleftarrow$ ManagementLibrary::Logger::printfln(), FilterManagementLibrary::TFIntegration::TensorflowNNInstance::run← Inference(), scaledImage(), and type2str().

#### 8.28.2.4 openCVMatTouint8\_t3DArray()

#### 8.28.2.5 openCVMatTouint8\_tArray()

#### 8.28.2.6 testNet()

```
void testNet ( )
```

References GetTopN(), i, image, openCVMatTouint8\_t3DArray(), scaledImage(), and start\_time.

#### 8.28.2.7 type2str()

- 8.29 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/

  Logger.cpp File Reference
- 8.30 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/Pipe System/PipeFilter.cpp File Reference
- 8.31 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/Pipe

  System/ProcessingPipeline.cpp File Reference
- 8.32 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/Pipe

  System/TFNNBasedPipeFilter.cpp File Reference
- 8.33 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/

  TensorflowAndroidJNIUtils.cpp File Reference

### Classes

• class anonymous\_namespace{TensorflowAndroidJNIUtils.cpp}::IfstreamInputStream

### **Namespaces**

anonymous\_namespace{TensorflowAndroidJNIUtils.cpp}

#### **Variables**

• static const char \*const ASSET\_PREFIX = "file:///android\_asset/"

#### 8.33.1 Variable Documentation

```
8.33.1.1 ASSET PREFIX
```

```
const char* const ASSET_PREFIX = "file:///android_asset/" [static]
```

- 8.34 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/

  TensorflowIntegration/TensorflowNNInstance.cpp File Reference
- 8.35 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/
  TensorflowIntegration/TensorflowNNInstanceClassifier.cpp File Reference
- 8.36 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/

  TensorflowIntegration/TensorflowResultContainer.cpp File Reference
- 8.37 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/
  TensorflowOpenCVUtils.cpp File Reference

### **Typedefs**

typedef cv::Point3\_< uint8\_t > Pixel

### 8.37.1 Typedef Documentation

### 8.37.1.1 Pixel

typedef cv::Point3\_<uint8\_t> Pixel

- 8.38 RoadSignAPI\_Standalone\_Tensorflow/source\_files/FilterManagementLibrary/
  Utilities.cpp File Reference
- 8.39 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/DetectedSign← Combination.cpp File Reference
- 8.40 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/Classified 
  SignsGrouper.cpp File Reference
- 8.41 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/Detection 

  BasedImageSlicer.cpp File Reference
- 8.42 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/Mobilenet 

  V2RoadSignClassificator.cpp File Reference
- 8.43 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/Road SignDuplicationDeleter.cpp File Reference
- 8.44 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/Filters/SSDLite RoadSignDetector.cpp File Reference
- 8.45 RoadSignAPI\_Standalone\_Tensorflow/source\_files/RoadSignAPI/RoadSignAPI.cpp File Reference

# Index

~DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18	FilterManagementLibrary::PipeSystem::TFNN↔ BasedPipeFilter, 95
$\sim$ lfstreamInputStream	assetManager
anonymous_namespace{TensorflowAndroidJNI← Utils.cpp}::IfstreamInputStream, 25	RoadSignAPI::MobilenetV2RoadSignClassificator, 35
$\sim$ LimitingFileInputStream	RoadSignAPI::RoadSignAPI, 58
tensorflow::android::LimitingFileInputStream, 27	RoadSignAPI::SSDLiteRoadSignDetector, 71
$\sim$ PipeFilter	
FilterManagementLibrary::PipeSystem::PipeFilter,	bytes_left_
39	tensorflow::android::LimitingFileInputStream, 27
$\sim$ ProcessingPipeline	- ,
FilterManagementLibrary::PipeSystem::Processing	CLASSIFIED_SIGNS_GROUPER_FILTER
Pipeline, 48	RoadSignAPI::RSAPIPipeRegisteredFilters, 64
~TFNNBasedPipeFilter	calculateBoxArea
FilterManagementLibrary::PipeSystem::TFNN↔	RoadSignAPI::RoadSignDuplicationDeleter, 61
BasedPipeFilter, 95	calculateOverlapPercentage
~TensorflowNNInstance	RoadSignAPI::RoadSignDuplicationDeleter, 61
FilterManagementLibrary::TFIntegration::Tensorflow	- ,
NNInstance, 77	codesnippets.cpp, 101
Ministance, 77	castedWorkingDataSet
ASSET_PREFIX	RoadSignAPI::ClassifiedSignsGrouper, 16
TensorflowAndroidJNIUtils.cpp, 114	RoadSignAPI::DetectionBasedImageSlicer, 24
addDetectedSign	RoadSignAPI::MobilenetV2RoadSignClassificator,
RoadSignAPI::DetectedSignCombination, 18	36
adjustModelFile	RoadSignAPI::RoadSignDuplicationDeleter, 63
FilterManagementLibrary::PipeSystem::TFNN↔	RoadSignAPI::SSDLiteRoadSignDetector, 71
BasedPipeFilter, 95	channels
FilterManagementLibrary::TFIntegration::Tensorflow-NNInstance, 77	NNModelDescription, 87
anonymous_namespace{TensorflowAndroidJNIUtils.↔	classifiedSignsGrouper
cpp}, 11	RoadSignAPI::RoadSignAPI, 58
anonymous_namespace{TensorflowAndroidJNIUtils.←	classifierApprovedClassID
cpp}::IfstreamInputStream, 25	RoadSignAPI::DetectedSignDescriptor, 21
$\sim$ lfstreamInputStream, 25	classifierApprovedSigns
ifs_, 26	RoadSignAPI::RSAPIWorkingDataSet, 66
IfstreamInputStream, 25	classifierConfidence
Read, 25	RoadSignAPI::DetectedSignDescriptor, 21
applyFromCVMat	codesnippets.cpp
main.cpp, 110	cameralmg, 101
applyImageInputVector	convertTo, 101
FilterManagementLibrary::TFIntegration::Tensorflow	→ i, 102
NNInstance, 77	image, 102
applyImageVectorFromOpenCVMat	p, 102
RoadSignAPI::MobilenetV2RoadSignClassificator,	scaledImage, 102
33	start time, 102
RoadSignAPI::SSDLiteRoadSignDetector, 69	convertTo
applyInputVectorFromFloatData	codesnippets.cpp, 101
FilterManagementLibrary::TFIntegration::Tensorflow	· · · · · · · · · · · · · · · · · · ·
NNInstance, 78	RoadSignAPI::MobilenetV2RoadSignClassificator,
applyNNImageInputVector	36

FilterManagementLibrary:-PipeSystem:-Processing-Pipeline, 50 currentRecognition RoadSignAPI::MobilenetV2RoadSignClassificator, 68 cutOutmages RoadSignAPI::BAAPIWorkingDataSet, 66 cutOutmages RoadSignAPI::RSAPIWorkingDataSet, 66 cutOutmages RoadSignAPI::RSAPIWorkingDataSet, 66 cutoUtmages RoadSignAPI::RoadSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::BAAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RoadSignAPI: 58 detectionPredictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorCoateBaEGRimage RoadSignAPI::DetectedSignDescriptor, 21 felterManagementLibrary::PipeSystem::Processing-Pipeline, 43 reset, 41 processingPipeline, 43 reset, 44 pipeWorkingDataSet, 44 processingFinished, 40 initByPipeSetup, 40 initByPipeSet	currentFilterID	FilterManagementLibrary::PipeSystem::PipeFilter,
curontRecognition RoadSignAPI::MobilenetV2RoadSignClassificator, 36 CutOutmages RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIWorkingDataSet, 66 CeleteAllMarkedSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombination RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSiticer RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSiticer RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSiticer RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignCombination RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRimage RoadSignAPI::DetectedSignCombination RoadSignAPI::DetectedSignCombination RoadSignAPI::DetectedSignDescriptor, 21 descortionPrediction RoadSignAPI::DetectedSignDescriptor, 21 descortionPrediction RoadSignAPI::DetectedSignDescriptor, 21 descortionPrediction RoadSignAPI::DetectedSignCombination RoadSignAPI::DetectedSignCom		
RoadSignAPI::MobilenetV2RoadSignClassificator, 36 cutOutImages RoadSignAPI::RSAPIWorkingDataSet, 66 FilterManagementLibrary::PipeSystem. 12 FITECTION_BASED_IMAGE_SLICER_FILTER RoadSignAPI::RSAPIPipeRegisteredFilters, 65 deletetAllMarkedSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61 detectedSignCombination RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSignCombination RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSignAPI::RSAPIWorkingDataSet, 66 detectedSignAPI::RSAPIWorkingDataSet, 66 detectedSignAPI::DetectedSignOpt. 58 RoadSignAPI::DetectedSignOpt. 51 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignCombination, 20 evaluateInput/vectorByNN FilterManagementLibrary::PipeSystem::Processing-FilterManagementLibrary::PipeSystem::Pipe	•	•
36 cutOutImages RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIPipeRegisteredFilters, 65 deleteAllMarkedSignDescriptors RoadSignAPI::RSAPIPipeRegisteredFilters, 65 deleteAllMarkedSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::RoadSignAPI::RoadSignAPI:RoadS	<del>-</del>	
eufoutmages RoadSignAPI::RSAPIWorkingDataSet, 66  DETECTION_BASED_IMAGE_SLICER_FILTER RoadSignAPI::RSAPIPpeRegisteredFilters, 65  deletaMilmarkedSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61  DetectedSignCombination RoadSignAPI::RoadSignDuplicationDeleter, 61  DetectedSignCombination RoadSignAPI::RoadSignDuplicationDeleter, 61  DetectedSignCombination RoadSignAPI::RSAPIWorkingDataSet, 66  detectionDescriptors RoadSignAPI::RoadSignAPI, 58  detectionPredictedClassID RoadSignAPI::RoadSignAPI-Sold (indicateProcessingFinished, 40  indicateProcessingFinished,	-	
RoadSignAPI::RSAPIWorkingDataSet, 66  DETECTION BASED_IMAGE_SLICER_FILTER RoadSignAPI::RSAPIPipeRegisteredFilters, 65 deletaMImkredSignDescriptors RoadSignAPI::RoadSignDuplicationDeleter, 61  DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::RSAPIWorkingDataSet, 66 detectodSignSeptDuplicationDeleter, 61  DetectedSignSpAPI::RSAPIWorkingDataSet, 66 detectodSignSpAPI::RSAPIWorkingDataSet, 66 detectodSignSeptDuplicationSpapp		
FilterManagementLibrary::PipeSystem. 12		•
DETECTION_BASED_IMAGE_SLICER_FILTER RoadSignAPI:RSAPIPipeRegisteredFilters, 65 deleteAllMarkedSignDescriptors RoadSignAPI:RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI:DetectedSignCombination, 18 detectedSignAPI:RSAPIWorkingDataSet, 66 detectedSignAPI:RSAPIWorkingDataSet, 66 detectedSignSpAPI:RSAPIWorkingDataSet, 66 detectedSignSpAPI:RSAPIWorkingDataSet, 66 detectedSignAPI:RSAPIWorkingDataSet, 66 detectionPadasedImageSlicer RoadSignAPI:RoadSignAPI, 58 detectionPredictedClassID RoadSignAPI:DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI:DetectedSignDescriptor, 21 detectorScaledBGRimage RoadSignAPI:DetectedSignDescriptor, 21 detectorScaledBGRimage RoadSignAPI:DetectedSignDescriptor, 21 detectorScaledRGRimage RoadSignAPI:DetectedSignDescriptor, 21 detectorScaledRGRimage RoadSignAPI:DetectedSignCombination, 20 expandPow RoadSignAPI:DetectedSignCombination, 20 expandBox RoadSignAPI:DetectionBasedImageSlicer, 24 processingPipeline, 47 FilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary:PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::ProcessingFilt	RoadSignAPI::RSAPIWorkingDataSet, 66	
RoadSignAPI::RSAPIPipoRegisteredFilters, 65 RoadSignAPI::RoadSignDuplicationDeleter, 61 DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::MsAPIWorkingDataSet, 66 detectedSign RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSignS RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBaseadImageSilcer RoadSignAPI::RoadSignDescriptor, 21 detectorConfidence RoadSignAPI::BetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::BetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::BetectedSignDescriptor, 21 detectorSoaledBGRimage RoadSignAPI::DetectedSignDescriptor, 21 detectorSoaledBGRimage RoadSignAPI::DetectedSignDescriptor, 21 detectorSoaledBGRimage RoadSignAPI::DetectedSignDescriptor, 21 detectorSoaledBGRimage RoadSignAPI::DetectedSignDescriptor, 21 detectorSoslaedBGRimage RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN RoadSignAPI::DetectionBasedImageSilcer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSilcer, 24 fliterManagementLibrary::PipeSystem::TFNNP BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSilcer, 24 fliterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorfloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensordid::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 ft_ ft_ ft_ ft_ ft_ ft_ ft_ ft_ ft_		
delete AllMarkedSignDescriptors RoadSignAPI::DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSlicer RoadSignAPI::DetectedSignDescriptor, 21 detectionCassedImageSlicer RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNNL- BasedPipeFilter, 95 dastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ t_essorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 getNNIModelDescription, 97		
RoadSignAPI::RoadSignCombination RoadSignAPI::RoadSignCombination RoadSignAPI::RoadSignCombinations RoadSignAPI::RoadSign		·
DetectedSignCombination RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSilicer RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSilicer RoadSignAPI::DetectedSignDescriptor, 21 detectorOndidence RoadSignAPI::DetectedSignDescriptor, 21 detectorOndidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorOndidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorOndidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorSignApication RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByMN FilterManagementLibrary::PipeSystem::TFNN- BasedPipeFiliter, 95 detastError, 48 lastError, 48 lastError, 49 lastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::PipeSystem::TFNNBasedPipeFilter, 95 applyNNImageInputVector, 95 evaluateInputVectorByNN, 96 getNNINdelDescription, 97	- · · · · · · · · · · · · · · · · · · ·	
RoadSignAPI::DetectedSignCombination, 18 detectedSignCombinations RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSlicer RoadSignAPI::RoadSignAPI, 58 detectionPredictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorOnflidence RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::DetectedSignDescriptor, 21 detectorConflidence RoadSignAPI::BsAPIWorkingDataSet, 66 RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21 filterManagementLibrary::PipeSystem:Processing—FilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::		
detectedSignCombinations RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSlicer RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSlicer RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBagRimage RoadSignAPI::DetectionBasedImageSlicer, 24 errno_ selfiterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::PipeRegistered—FilterManagementLibrary::PipeSystem::PipeWorking FilterManagementLibrary::PipeSystem::Processing—Pipeline, 45 rorrorspace RoadSignAPI::DetectionBasedImageSlicer, 24 expandBox RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 25 RoadSignAPI::DetectionBasedImageSlicer, 26 RoadSignAPI::DetectionBasedImageSlicer, 27 RoadSignAPI::DetectionBasedImageSlicer, 28 RoadSignAPI::DetectionBasedImageSlicer, 29 RoadSignAPI::DetectionBasedImageSlicer, 29 RoadSignAPI::DetectionBasedImageSlicer, 29 RoadSignAPI::DetectionBasedImageSlicer, 29 RoadSignAPI::DetectionBasedImageSlicer, 29		<del>-</del>
RoadSignAPI::RSAPIWorkingDataSet, 66 detectedSigns RoadSignAPI::RSAPIWorkingDataSet, 66 detectionBasedImageSlicer RoadSignAPI::RoadSignAPI, 58 detectionPredictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::DetectedSignCombination Pipeline, 47 FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::PipeSystem::TFNN→ BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::FinsorflowOpenCVUtils, 89 flastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 flastApplyCVMatOnInputTensorVInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 flastApplyCVMatOnInputTensorVInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 flastSignAPI::RoadSignAPI, 54 filleExists FilterManagementLibrary::Utilitites, 99 feellmage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilitites, 99 feellmage getNNImputTensor, 95 evaluateInputVectorByNN, 96 getNNImputTensor, 97 evalua		
detectionBasedImageSlicer RoadSignAPI::RoadSignAPI, 58 detectionBasedImageSlicer RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignCombination; 20 evaluateInput/VectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 49 registerFilter, 49 registerFilter, 49 registeredFilters, 44 pipeWorkingDataSet, 41 processingPipeline, 47 registerFilter, 47 registerFilter, 49 registeredFilters, 44 pipeWorkingDataSet, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilter,		
RoadSignAPI::RSAPIWorkingDataSet, 66  detectionBasedImageSlicer RoadSignAPI::DetectedSignDescriptor, 21  detectionCridience RoadSignAPI::DetectedSignDescriptor, 21  detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21  detectorScaledBGRImage RoadSignAPI::DetectedSignDescriptor, 21  detectorScaledBGRImage RoadSignAPI::RSAPIWorkingDataSet, 66  errno_ tensorflow::android::LimittingFileInputStream, 27  ErrorType FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::ProcessingPipeline, 45  estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20  evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN→ BasedPipeFilter, 39  expandBox RoadSignAPI::DetectionBasedImageSlicer, 23  expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  processingPipeline, 47  festApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUlnt8 FilterManagementLibrary::PipeSystem::TFNNBased→PipeFilter, 95  atensorflow::android::LimitingFileInputStream, 28  feedImage RoadSignAPI::RoadSignAPI, 54  filleExists  getNNIModelDescription, 97		•
detectionBasedImageSlicer RoadSignAPI::RoadSignAPI, 58 detectionPedictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::BetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::BetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::BradPilworkingDataSet, 66 errno_ tensorflow::android::LimitingFileInputStream, 27 FilterManagementLibrary::PipeSystem::Processing- FilterManagementLibrary::PipeSystem::Processing- NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilities, 99  markedNextDesiredFilter, 43 markedProcessingFinished, 43 pipeFilter, 39 pipeRegisteredFilters, 44 pipeWorkingDataset, 41 processingPipeline, 43 reset, 41 processingPipeline, 45 ~ProcessingPipeline, 45 ~ProcessingPipeline, 45 ~ProcessingPipeline, 45 ~ProcessingPipeline, 45 pipeRegisteredFilters-Hade pipeRegisteredFilters-Hade pipeRegisteredFilters-Hade pipeRegisteredFilters-Hade processingPipeline, 43 reset, 41 processingPipeline, 43 reset, 41 processingPipeline, 45 reset, 41 processingPipeline, 45 rest, 41 processingPipeline, 49 letterManagementLibrary::PipeSystem::Processing- pipeRegisteredFilters-Hade processingPipeRider, 43 letterManagementLibrary::Pipe	_	
RoadSignAPI::RoadSignAPI, 58 detectionPredictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::BaPIWorkingDataSet, 66  errno_ tensorflow::android::LimitingFileInputStream, 27 ErrorType FilterManagementLibrary::PipeSystem::ProcessingFilterManagementLibrary::PipeSystem::PipeRegisteredFilters, 44 FilterManagementLibrary::PipeSystem::Processing→FilterManagementLibrary::PipeSystem::PipeWorking→ Pipeline, 47 FilterManagementLibrary::Tentegration::Tensorflow-FilterManagementLibrary::PipeSystem::Processing→Pipeline, 45 FilterManagementLibrary::PipeSystem::Processing→Pipeline, 45 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN→BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::TensorflowOpe		
detectionPredictedClassID RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIWorkingDataSet, 66  errno_ tensorflow::android::LimitingFiteInputStream, 27 ErrorType FilterManagementLibrary::PipeSystem::Processing← Filters, 44 pipeWorkingDataSet, 44 process, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 pipeWorkingDataSet, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 pipeWorkingDataSet, 44 processingPipeline, 43 reset, 41 setCredentials, 41 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 filterManagementLibrary::UpipeSystem::TFNNBasedFipeFilter, 95 adjustModelFile, 95 adjustModelPiseo, 96 getNNIModelDescription, 97		
RoadSignAPI::DetectedSignDescriptor, 21 detectorConfidence RoadSignAPI::DetectedSignDescriptor, 21 pipeRegisteredFilters, 44 pipeWorkingDataSet, 44 detectorScaledBGRImage RoadSignAPI::RSAPIWorkingDataSet, 66 ProcessingPipeline, 43 reset, 41 setCredentials, 41 remort/livers and process. 41 setCredentials, 41 remort/livers and process. 45 FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::PipeRegistered—Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking—DataSet, 45 FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—FilterManagementLibrary::PipeSystem::Processing—Pipeline, 47 FilterManagementLibrary::PipeSystem::Processing—Pipeline, 47 FilterManagementLibrary::PipeSystem::Processing—Pipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingPipeline, 47 registerFilter, 49 registe		•
detectorConfidence RoadSignAPl::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPl::RSAPIWorkingDataSet, 66  errno_ tensorflow::android::LimitingFileInputStream, 27 ErrorType FilterManagementLibrary::PipeSystem::Processing→ FilterManagementLibrary::PipeSystem::PipeRegistered→ FilterManagementLibrary::PipeSystem::PipeWorkingDataSet, 45 FilterManagementLibrary::TFIntegration::Tensorflow, FilterManagementLibrary::PipeSystem::Processing→ DataSet, 45 FilterManagementLibrary::TFIntegration::Tensorflow, FilterManagementLibrary::PipeSystem::Processing→ NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN→ BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensor.96 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99		
RoadSignAPI::DetectedSignDescriptor, 21 detectorScaledBGRImage RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIWorkingDataSet, 66 RoadSignAPI::RSAPIWorkingDataSet, 66 Reset, 41 errno		
detectorScaledBGRImage RoadSignAPI::RSAPIWorkingDataSet, 66 errno_ tensorflow::android::LimitingFileInputStream, 27 ErrorType FilterManagementLibrary::PipeSystem::Processing→ FilterManagementLibrary::PipeSystem::TFNNBased→ PipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95 evaluateInputVector, 95 evaluat		
RoadSignAPI::RSAPIWorkingDataSet, 66  errno_ tensorflow::android::LimitingFileInputStream, 27  ErrorType FilterManagementLibrary::PipeSystem::Processing ← FilterManagementLibrary::PipeSystem::PipeRegistered ← FilterManagementLibrary::PipeSystem:	- · · · · · · · · · · · · · · · · · · ·	
reset, 41  tensorflow::android::LimitingFileInputStream, 27  ErrorType  FilterManagementLibrary::PipeSystem::Processing — FilterManagementLibrary::PipeSystem::PipeWorking — Pipeline, 47  FilterManagementLibrary::TFIntegration::Tensorflow — FilterManagementLibrary::PipeSystem::Processing — FilterManagementLibrary::PipeSystem::Processing — FilterManagementLibrary::PipeSystem::Processing — DataSet, 45  estimatedPolePositionX  RoadSignAPI::DetectedSignCombination, 20  evaluateInputVectorByNN  FilterManagementLibrary::PipeSystem::TFNN— BasedPipeFilter, 96  expandBox  RoadSignAPI::DetectionBasedImageSlicer, 23  expandPercentage  RoadSignAPI::DetectionBasedImageSlicer, 24  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUnt8  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUnt8  FilterManagementLibrary::PipeSystem::TFNNBased— PipeFilter, 95  adjustModelFile, 95  adjustModelFile, 95  adjustModelPescription, 97  filterManagementLibrary::Utilities, 99		•
errno_ tensorflow::android::LimitingFileInputStream, 27 ErrorType FilterManagementLibrary::PipeSystem::Processing ← FilterManagementLibrary::PipeSystem::PipeMorking ← Pipeline, 47 FilterManagementLibrary::PipeSystem::Processing ← FilterManagementLibrary::PipeSystem::PipeWorking ← DataSet, 45 FilterManagementLibrary::PipeSystem::Processing ← FilterManagementLibrary::PipeSystem::Processing ← Pipeline, 45 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorIUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilities, 99 ffleExists FilterManagementLibrary::Utilities, 99 formore ScurrentDataSet, 49 processingPipeline, 47 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::Processing ← PipeFilter, 93 adjustModelFile, 95 adjustModelDescription, 97	RoadSignAPI::RSAPIWorkingDataSet, 66	
tensorflow::android::LimitingFileInputStream, 27 ErrorType Filters, 44 FilterManagementLibrary::PipeSystem::Processing←		
ErrorType FilterManagementLibrary::PipeSystem::Processing → FilterManagementLibrary::PipeSystem::PipeWorking → DataSet, 45 FilterManagementLibrary::TFIntegration::Tensorflow ← FilterManagementLibrary::PipeSystem::Processing ← Pipeline, 45  estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN ← BasedPipeFilter, 96  expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorFlowOpenCVUtils, 89  feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilitites, 99  FilterManagementLibrary::Utilitites, 99  FilterManagementLibrary::Utilitites, 99  FilterManagementLibrary::Otilities, 99  FilterManagementLibrary::Otilities, 99  FilterManagementLibrary::Otilities, 99  FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Otilities, 99  FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Otilities, 99	errno_	setGredentials, 41
FilterManagementLibrary::PipeSystem::Processing FilterManagementLibrary::PipeSystem::PipeWorking Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow FilterManagementLibrary::PipeSystem::Processing NNInstance, 75 estimatedPolePositionX ProcessingPipeline, 45 estimatedPolePositionX ProcessingPipeline, 45 estimatedPolePositionX ProcessingPipeline, 45 estimatedPolePositionX ProcessingPipeline, 45 estimatedPolePositionX ProcessingPipeline, 48 estadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN ErrorType, 47 FilterManagementLibrary::PipeSystem::TFNN getLastError, 50 expandBox manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingPipeline, 47 fastApplyCVMatOnInputTensorFloat registerFilter, 49 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 registeredFilters, 51 setup, 50 sextup, 50 sextup, 50 fastApplyCVMatOnInputTensorUInt8 pitterManagementLibrary::PipeSystem::TFNNBased PipeFilter, 93 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filteExists getNNInputTensor, 96 getNNModelDescription, 97		
Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow-FilterManagementLibrary::PipeSystem::Processing → NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN→ BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 feedImage RoadSignAPI::RoadSignAPI, 54 filteExists FilterManagementLibrary::Uililities, 99  Pathon And State (1) FilterManagementLibrary::PipeSystem::TFNNBased→ pipeFilter, 95 adjustModelFile, 95 adjustModelDescription, 97	- · · · · · · · · · · · · · · · · · · ·	$Filter Management Library :: Pipe System :: Pipe Registered \leftarrow$
FilterManagementLibrary::TFIntegration::Tensorflow-FilterManagementLibrary::PipeSystem::Processing-NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN- BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 fgtNNInputTensor, 96 getNNInputTensor, 97	ErrorType	$\label{likelihood} Filter Management Library :: Pipe System :: Pipe Registered \leftarrow Filters, \ 44$
NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN→ BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99  FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99 FilterManagementLibrary::Utilities, 99	ErrorType FilterManagementLibrary::PipeSystem::Processing	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 -FilterManagementLibrary::PipeSystem::PipeWorking←
estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 fileExists FilterManagementLibrary::Utilities, 99 forecassingPipeline, 48 currentFilterID, 50 extarct FilterID, 50 extarct Filter JB extarct Filter JB extarct Filter JB extarct Filter JB extarct AB lastError, 48 lastError, 49 recesscurrentDatacet, 50 processCurrentDatacet, 50 processCurrentDatederenein salesteredFiltersHeader, 50 processCurrentDatederenein salesteredFiltersHeader, 50 processCurrentDatederenein salesteredFiltersHeader, 50 processCurrentDatederenein salesteredFiltersHeader, 50 processCurrentDatederenein salesteredFiltersHeader salesteredFiltersHeader salesteredFiltersHeader salesteredFiltersHeader salesteredFiltersHeader salesteredFilte	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45
RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ Lensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingFinishied, 51 processingFinishied, 51 processingFilter, 49 registeredFilters, 51 setup, 50 setup, 50 vorkingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 feedImage RoadSignAPI::RoadSignAPI, 54 filterManagementLibrary::Utilities, 99 getNNInputTensor, 96 getNNInputTensor, 96 getNNInputTensor, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 ←FilterManagementLibrary::PipeSystem::Processing←
evaluateInputVectorByNN    FilterManagementLibrary::PipeSystem::TFNN←    BasedPipeFilter, 96  expandBox    RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage    RoadSignAPI::DetectionBasedImageSlicer, 24    RoadSignAPI::DetectionBasedImageSlicer, 24    RoadSignAPI::DetectionBasedImageSlicer, 24    RoadSignAPI::DetectionBasedImageSlicer, 24    RoadSignAPI::DetectionBasedImageSlicer, 24    RoadSignAPI::DetectionBasedImageSlicer, 24    FilterManagementLibrary::TensorflowOpenCVUtils, 89    fastApplyCVMatOnInputTensorUnt8    FilterManagementLibrary::TensorflowOpenCVUtils, 89    fastApplyCVMatOnInputTensorUnt8    FilterManagementLibrary::TensorflowOpenCVUtils, 89    fd_	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45
FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  RoadSignAPI::DetectionBasedImageSlicer, 24  RoadSignAPI::DetectionBasedImageSlicer, 24  RoadSignAPI::DetectionBasedImageSlicer, 24  RoadSignAPI::DetectionBasedImageSlicer, 24  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ filterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ feedImage RoadSignAPI::RoadSignAPI, 54  filleExists FilterManagementLibrary::Utilities, 99  getLastError, 48 lastError, 50  manageExternalAllocatedRessources, 50  pipeRegisteredFiltersHeader, 50  processingFinishied, 51  ProcessingPipeline, 47  registeredFilters, 51  setup, 50  workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased←  PipeFilter, 93  ∼TFNNBasedPipeFilter, 95  adjustModelFile, 95  adjustModelFile, 95  evaluateInputVectorByNN, 96  getNNInputTensor, 96  getNNInputTensor, 96  getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48
BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 ProcessingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 setup, 50 workingDataSet, 51 FilterManagementLibrary::TensorflowOpenCVUtils, 89 ripeFilter, 93 refilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 adjustModelFile, 95 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilities, 99 getNNInputTensor, 96 getNNInputTensor, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ProcessingPipeline, 48 currentFilterID, 50
expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47
RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 ProcessingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← RoadSignAPI::RoadSignAPI::TensorflowOpenCVUtils, tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilities, 99 getNNInputTensor, 96 getNNInputTensor, 96 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN←	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48
expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 RoadSignAPI::DetectionBasedImageSlicer, 24 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ 89 fd_ 89 fd_ 89 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists FilterManagementLibrary::Utilities, 99 fileExists FilterManagementLibrary::Utilities, 99 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registeredFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← vortice A7 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← adjustModelFile, 95 adjustModelFile, 95 evaluateInputVector, 95 evaluateInputVectorByNN, 96 getNNInputTensor, 96 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50
RoadSignAPI::DetectionBasedImageSlicer, 24  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ 89  fd_ 70  tensorflow::android::LimitingFileInputStream, 28  feedImage RoadSignAPI::RoadSignAPI, 54  filleExists getNNInputTensor, 96  FilterManagementLibrary::Utilities, 99  processingFinishied, 51  ProcesingFinishied, 51  ProcesingFinishied, 51  ProcesingFinishied, 51  ProcesingFilter, 49  registeredFilters, 51  setup, 50  workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased →  PipeFilter, 93  ∼TFNNBasedPipeFilter, 95  adjustModelFile, 95  adjustModelFile, 95  evaluateInputVector, 95  evaluateInputVectorByNN, 96  getNNInputTensor, 96  getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45  ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50
ProcessingPipeline, 47  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ 7	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45  ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50
fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUnt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fol_ 100 100 100 100 100 100 100 100 100 10	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45  ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49
FilterManagementLibrary::TensorflowOpenCVUtils, 89  fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54  filterManagementLibrary::Utilities, 99  registeredFilters, 51 setup, 50 workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased↔ PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95 evaluateInputVectorByNN, 96 getNNInputTensor, 96 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51
89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingPipeline, 47
fastApplyCVMatOnInputTensorUInt8 workingDataSet, 51  FilterManagementLibrary::TensorflowOpenCVUtils, 89  fd_ PipeFilter, 93  ∼TFNNBasedPipeFilter, 95  tensorflow::android::LimitingFileInputStream, 28  feedImage applyNNImageInputVector, 95  RoadSignAPI::RoadSignAPI, 54  fileExists getNNInputTensor, 96  FilterManagementLibrary::Utilities, 99  workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased↔  ApplyFilter, 95  adjustModelFile, 95  applyNNImageInputVector, 95  evaluateInputVectorByNN, 96  getNNInputTensor, 96  getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24 fastApplyCVMatOnInputTensorFloat	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49
FilterManagementLibrary::TensorflowOpenCVUtils, 89 FilterManagementLibrary::PipeSystem::TFNNBased↔ PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95 evaluateInputVectorByNN, 96 fileExists getNNInputTensor, 96 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils,	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingPipeline, 47 registerFilter, 49 registeredFilters, 51
89 PipeFilter, 93  fd_ ~TFNNBasedPipeFilter, 95  tensorflow::android::LimitingFileInputStream, 28 feedImage applyNNImageInputVector, 95 RoadSignAPI::RoadSignAPI, 54 evaluateInputVectorByNN, 96 fileExists getNNInputTensor, 96 FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45  ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerEilter, 49 registeredFilters, 51 setup, 50
fd_	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8	FilterManagementLibrary::PipeSystem::PipeRegistered Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking DataSet, 45 FilterManagementLibrary::PipeSystem::Processing Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registeredFilters, 49 registeredFilters, 51 setup, 50 workingDataSet, 51
tensorflow::android::LimitingFileInputStream, 28 feedImage applyNNImageInputVector, 95 RoadSignAPI::RoadSignAPI, 54 evaluateInputVectorByNN, 96 fileExists getNNInputTensor, 96 FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils,	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased←
feedImage applyNNImageInputVector, 95 RoadSignAPI::RoadSignAPI, 54 evaluateInputVectorByNN, 96 fileExists getNNInputTensor, 96 FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93
RoadSignAPI::RoadSignAPI, 54 evaluateInputVectorByNN, 96 fileExists getNNInputTensor, 96 FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95
fileExists getNNInputTensor, 96 FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registeredFilters, 49 registeredFilters, 51 setup, 50 workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95
FilterManagementLibrary::Utilities, 99 getNNModelDescription, 97	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 ~ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ~TFNNBasedPipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95
	ErrorType FilterManagementLibrary::PipeSystem::Processing← Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN← BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44 FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45 ←FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45 ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51 FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95 evaluateInputVectorByNN, 96
Motifolial following the second secon	ErrorType FilterManagementLibrary::PipeSystem::Processing Pipeline, 47 FilterManagementLibrary::TFIntegration::Tensorflow NNInstance, 75 estimatedPolePositionX RoadSignAPI::DetectedSignCombination, 20 evaluateInputVectorByNN FilterManagementLibrary::PipeSystem::TFNN BasedPipeFilter, 96 expandBox RoadSignAPI::DetectionBasedImageSlicer, 23 expandPercentage RoadSignAPI::DetectionBasedImageSlicer, 24  fastApplyCVMatOnInputTensorFloat FilterManagementLibrary::TensorflowOpenCVUtils, 89 fastApplyCVMatOnInputTensorUInt8 FilterManagementLibrary::TensorflowOpenCVUtils, 89 fd_ tensorflow::android::LimitingFileInputStream, 28 feedImage RoadSignAPI::RoadSignAPI, 54 fileExists	FilterManagementLibrary::PipeSystem::PipeRegistered← Filters, 44  FilterManagementLibrary::PipeSystem::PipeWorking← DataSet, 45  FilterManagementLibrary::PipeSystem::Processing← Pipeline, 45  ∼ProcessingPipeline, 48 currentFilterID, 50 ErrorType, 47 getLastError, 48 lastError, 50 manageExternalAllocatedRessources, 50 pipeRegisteredFiltersHeader, 50 processCurrentDataSet, 49 processingFinishied, 51 ProcessingPipeline, 47 registerFilter, 49 registeredFilters, 51 setup, 50 workingDataSet, 51  FilterManagementLibrary::PipeSystem::TFNNBased← PipeFilter, 93 ∼TFNNBasedPipeFilter, 95 adjustModelFile, 95 applyNNImageInputVector, 95 evaluateInputVectorByNN, 96 getNNInputTensor, 96

onNNEvaluationFinished, 97 setupModelFromAssets, 97	FilterManagementLibrary::TensorflowAndroidJNIUtils,
setupModelFromFile, 98	IsAsset, 72
TFNNBasedPipeFilter, 94	PortableReadFileToProto, 73
tfNNInstance, 98	ReadFileToProto, 73
FilterManagementLibrary::TFIntegration, 12	ReadFileToString, 73
FilterManagementLibrary::TFIntegration::TensorflowN←	ReadFileToVector, 73
NInstance, 74	WriteProtoToFile, 73
~TensorflowNNInstance, 77	FilterManagementLibrary::TensorflowOpenCVUtils, 88
	fastApplyCVMatOnInputTensorFloat, 89
adjustModelFile, 77	fastApplyCVMatOnInputTensorUInt8, 89
applyImageInputVector, 77	FilterManagementLibrary::Utilities, 99
applyInputVectorFromFloatData, 78	fileExists, 99
ErrorType, 75	
getInputTensor, 78	get_us
getLastError, 79	main.cpp, 110
getModelDescription, 79	mainold.cpp, 112
getOutputTensors, 79	getClassifierApprovedDetectedSigns
getResultContainer, 80	RoadSignAPI::RoadSignAPI, 55
graphDef, 81	getDetectedSignCombinations
inputLayerName, 81	RoadSignAPI::RoadSignAPI, 55
inputTensor, 82	getDetectedSigns
lastError, 82	RoadSignAPI::RoadSignAPI, 55
nnModelDescription, 82	getDetectedSignsAmount
numThreads, 82	RoadSignAPI::DetectedSignCombination, 19
outputTensors, 82	getFilterID
runInference, 80	FilterManagementLibrary::PipeSystem::PipeFilter,
sessionOptions, 82	39
setupModelFromAssets, 80	getGestimatedPolePositionX
setupModelFromFile, 81	RoadSignAPI::DetectedSignCombination, 19
TensorflowNNInstance, 76	getInputTensor
tensorflowSession, 82	FilterManagementLibrary::TFIntegration::Tensorflow
$Filter Management Library :: TFIntegration :: Tensor flow N {\leftarrow}$	NNInstance, 78
NInstanceClassifier, 83	getLastError
getTopNPredictions, 84	FilterManagementLibrary::PipeSystem::Processing ← Pipeline, 48
getTopNPredictionsStringVector, 85	•
labelCount, 86	FilterManagementLibrary::TFIntegration::Tensorflow← NNInstance, 79
loadedLabels, 86	getModelDescription
readLabelsFile, 85	FilterManagementLibrary::TFIntegration::Tensorflow←
TensorflowNNInstanceClassifier, 84	NNInstance, 79
$Filter Management Library :: TFIntegration :: Tensorflow N {\leftarrow}$	getNNInputTensor
NModelDescription, 86	FilterManagementLibrary::PipeSystem::TFNN⊷
channels, 87	BasedPipeFilter, 96
input_floating, 87	getNNModelDescription
input_mean, 87	FilterManagementLibrary::PipeSystem::TFNN⊷
input_std, 87	BasedPipeFilter, 97
inputHeight, 87	getNeuralNetLastError
inputLayerNameStr, 88	FilterManagementLibrary::PipeSystem::TFNN↔
inputWidth, 88	BasedPipeFilter, 96
modelFile, 88	getNextDesiredFilter
outputLayerNames, 88	FilterManagementLibrary::PipeSystem::PipeFilter,
FilterManagementLibrary::TFIntegration::Tensorflow ←	39
ResultContainer, 90	getOutputTensorByID
getOutputTensorByID, 91	FilterManagementLibrary::TFIntegration::Tensorflow
getOutputTensorByLayerName, 91	ResultContainer, 91
outputTensorNames, 92	getOutputTensorByLayerName
outputTensors, 92	FilterManagementLibrary::TFIntegration::Tensorflow-
TensorflowResultContainer, 91	ResultContainer, 91

getOutputTensors	input_floating
FilterManagementLibrary::TFIntegration::Tensorflow-	→ FilterManagementLibrary::TFIntegration::Tensorflow ←
NNInstance, 79	NNModelDescription, 87
getResultContainer	input_mean
FilterManagementLibrary::TFIntegration::Tensorflow	• —
NNInstance, 80	NNModelDescription, 87
getSignsInCombination	input_std
RoadSignAPI::DetectedSignCombination, 19	FilterManagementLibrary::TFIntegration::Tensorflow ←
getTimeString	NNModelDescription, 87
FilterManagementLibrary::Logger, 29	inputHeight
getTopNPredictions	FilterManagementLibrary::TFIntegration::Tensorflow ←
FilterManagementLibrary::TFIntegration::Tensorflow	
NNInstanceClassifier, 84	inputLayerName
getTopNPredictionsStringVector	FilterManagementLibrary::TFIntegration::Tensorflow←
FilterManagementLibrary::TFIntegration::Tensorflow	
NNInstanceClassifier, 85	inputLayerNameStr
GetTopN	FilterManagementLibrary::TFIntegration::Tensorflow ←
mainold.cpp, 112	NNModelDescription, 88
google, 12	inputTensor
google::protobuf, 13	FilterManagementLibrary::TFIntegration::Tensorflow ←
graphDef	NNInstance, 82
FilterManagementLibrary::TFIntegration::Tensorflow	
NNInstance, 81	FilterManagementLibrary::TFIntegration::Tensorflow←
WWW.starioc, or	NNModelDescription, 88
hasMarkedNextDesiredFilter	·
	insertSorted
FilterManagementLibrary::PipeSystem::PipeFilter,	RoadSignAPI::DetectedSignCombination, 19
39	instance
hasMarkedProcessingFinished	RoadSignAPI::RoadSignAPI, 59
FilterManagementLibrary::PipeSystem::PipeFilter,	invokeNext
40	FilterManagementLibrary::PipeSystem::PipeFilter,
horizontalRangePercentage	41
RoadSignAPI::ClassifiedSignsGrouper, 17	IsAsset
	FilterManagementLibrary::TensorflowAndroidJNI↔
i	Utils, 72
codesnippets.cpp, 102	isInUnwantedClasses
ifs_	
anonymous_namespace{TensorflowAndroidJNI↔	RoadSignAPI::MobilenetV2RoadSignClassificator,
Utils.cpp}::lfstreamInputStream, 26	34
	isMarkedForDeletion
IfstreamInputStream	RoadSignAPI::RoadSignDuplicationDeleter, 62
anonymous_namespace{TensorflowAndroidJNI↔	
Utils.cpp}::IfstreamInputStream, 25	labelCount
image	FilterManagementLibrary::TFIntegration::Tensorflow ←
codesnippets.cpp, 102	NNInstanceClassifier, 86
imageWidth	lastError
RoadSignAPI::ClassifiedSignsGrouper, 17	FilterManagementLibrary::PipeSystem::Processing
indicateProcessingFinished	Pipeline, 50
FilterManagementLibrary::PipeSystem::PipeFilter,	FilterManagementLibrary::TFIntegration::Tensorflow
40	
init	NNInstance, 82
	LimitingFileInputStream
RoadSignAPI::RoadSignAPI, 56	tensorflow::android::LimitingFileInputStream, 26
initByPipeSetup	loadedLabels
FilterManagementLibrary::PipeSystem::PipeFilter,	FilterManagementLibrary::TFIntegration::Tensorflow ←
40	NNInstanceClassifier, 86
RoadSignAPI::ClassifiedSignsGrouper, 16	logTag
RoadSignAPI::DetectionBasedImageSlicer, 23	FilterManagementLibrary::Logger, 31
RoadSignAPI::MobilenetV2RoadSignClassificator,	lowerRight
34	RoadSignAPI::DetectedSignDescriptor, 21
RoadSignAPI::RoadSignDuplicationDeleter, 62	Hoadolylini IDetectedolyliDescriptor, 21

main.cpp, 110 mainold.cpp, 112	FilterManagementLibrary::TFIntegration::Tensorflow-NNInstance, 82
main.cpp	
applyFromCVMat, 110	onNNEvaluationFinished
get_us, 110	FilterManagementLibrary::PipeSystem::TFNN←
main, 110	BasedPipeFilter, 97
openCVMatTouint8_t3DArray, 110	RoadSignAPI::MobilenetV2RoadSignClassificator,
openCVMatTouint8_tArray, 111	34
Pixel, 110	RoadSignAPI::SSDLiteRoadSignDetector, 70
readLabelsFile, 111	openCVMatTouint8_t3DArray
type2str, 111	main.cpp, 110
mainold.cpp	mainold.cpp, 112
get_us, 112	openCVMatTouint8_tArray
GetTopN, 112	main.cpp, 111
main, 112	mainold.cpp, 113
openCVMatTouint8_t3DArray, 112	originalBGRImage
openCVMatTouint8_tArray, 113	RoadSignAPI::RSAPIWorkingDataSet, 67
Pixel, 112	originallmageHeight
testNet, 113	RoadSignAPI::RSAPIWorkingDataSet, 67
type2str, 113	
manageExternalAllocatedRessources	originalImageWidth
FilterManagementLibrary::PipeSystem::Processing	RoadSignAPI::RSAPIWorkingDataSet, 67
Pipeline, 50	
markForDeletion	FilterManagementLibrary::TFIntegration::Tensorflow
RoadSignAPI::RoadSignDuplicationDeleter, 62	NNModelDescription, 88
markedNextDesiredFilter	outputTensorNames
FilterManagementLibrary::PipeSystem::PipeFilter,	FilterManagementLibrary::TFIntegration::Tensorflow
43	ResultContainer, 92
markedProcessingFinished	outputTensors
FilterManagementLibrary::PipeSystem::PipeFilter,	FilterManagementLibrary::TFIntegration::Tensorflow-
43	NNInstance, 82
minExpandPixels	$Filter Management Library :: TFIntegration :: Tensor flow \leftarrow$
RoadSignAPI::DetectionBasedImageSlicer, 24	ResultContainer, 92
minOverlapPercentage	
RoadSignAPI::RoadSignDuplicationDeleter, 63	p
	codesnippets.cpp, 102
mobilenetModelDescription	PipeFilter
RoadSignAPI::RoadSignAPI, 59	FilterManagementLibrary::PipeSystem::PipeFilter,
MobilenetV2RoadSignClassificator	39
RoadSignAPI::MobilenetV2RoadSignClassificator,	pipeRegisteredFilters
32, 33	FilterManagementLibrary::PipeSystem::PipeFilter,
modelFile	44
FilterManagementLibrary::TFIntegration::Tensorflow	RoadSignAPI::RoadSignAPI, 59
NNModelDescription, 88	pipeRegisteredFiltersHeader
nextDesiredFilter	FilterManagementLibrary::PipeSystem::Processing ←
FilterManagementLibrary::PipeSystem::PipeFilter,	Pipeline, 50
43	pipeWorkingDataSet
nnModelDescription	FilterManagementLibrary::PipeSystem::PipeFilter,
FilterManagementLibrary::TFIntegration::Tensorflow	
NNInstance, 82	Pixel
	main.cpp, 110
nnModelInputHeight	mainold.cpp, 112
RoadSignAPI::MobilenetV2RoadSignClassificator,	TensorflowOpenCVUtils.cpp, 114
36  Dood Sign A Division Lita Dood Sign Detector, 71	
RoadSignAPI::SSDLiteRoadSignDetector, 71	PortableReadFileToProto
nnModelInputWidth  ReadSignABI::MabilenetV2BaadSignClassificator	FilterManagementLibrary::TensorflowAndroidJNI  Little 72
RoadSignAPI::MobilenetV2RoadSignClassificator,	Utils, 73
36  PondSignABI::SSDI itaBondSignDatastar, 73	printfln  FilterManagementLibrary:Legger 20
RoadSignAPI::SSDLiteRoadSignDetector, 72	FilterManagementLibrary::Logger, 30
numThreads	process

FilterManagementLibrary::PipeSystem::PipeFilter,	getDetectedSignsAmount, 19
41	getGestimatedPolePositionX, 19
RoadSignAPI::ClassifiedSignsGrouper, 16	getSignsInCombination, 19
RoadSignAPI::DetectionBasedImageSlicer, 23	insertSorted, 19
RoadSignAPI::MobilenetV2RoadSignClassificator,	signs, 20
35	RoadSignAPI::DetectedSignDescriptor, 20
RoadSignAPI::RoadSignDuplicationDeleter, 62	classifierApprovedClassID, 21
RoadSignAPI::SSDLiteRoadSignDetector, 71	classifierConfidence, 21
processCurrentDataSet	detectionPredictedClassID, 21
FilterManagementLibrary::PipeSystem::Processing ←	detectorConfidence, 21
Pipeline, 49	lowerRight, 21
processingFinishied	upperLeft, 21
FilterManagementLibrary::PipeSystem::Processing ←	RoadSignAPI::DetectionBasedImageSlicer, 22
Pipeline, 51	castedWorkingDataSet, 24
ProcessingPipeline	expandBox, 23
FilterManagementLibrary::PipeSystem::PipeFilter,	expandPercentage, 24
43	initByPipeSetup, 23
FilterManagementLibrary::PipeSystem::Processing ←	
Pipeline, 47	process, 23
processingPipeline	RoadSignAPI::MobilenetV2RoadSignClassificator, 31
RoadSignAPI::RoadSignAPI, 59	applyImageVectorFromOpenCVMat, 33
	assetManager, 35
Read	
anonymous_namespace{TensorflowAndroidJNI←	castedWorkingDataSet, 36
Utils.cpp}::IfstreamInputStream, 25	currentCutOutImage, 36
tensorflow::android::LimitingFileInputStream, 27	currentRecognition, 36
ReadFileToProto	initByPipeSetup, 34
FilterManagementLibrary::TensorflowAndroidJNI ←	isInUnwantedClasses, 34
Utils, 73	MobilenetV2RoadSignClassificator, 32, 33
ReadFileToString	nnModelInputHeight, 36
FilterManagementLibrary::TensorflowAndroidJNI ←	nnModelInputWidth, 36
Utils, 73	onNNEvaluationFinished, 34
ReadFileToVector	process, 35
FilterManagementLibrary::TensorflowAndroidJNI←	threshold, 36
Utils, 73	validRecognition, 37
readLabelsFile	RoadSignAPI::RSAPIPipeRegisteredFilters, 64
FilterManagementLibrary::TFIntegration::Tensorflow-	_ CLASSIFIED_SIGNS_GROUPER_FILTER, 64
NNInstanceClassifier, 85	DETECTION_BASED_IMAGE_SLICER_FILTER,
main.cpp, 111	65
registerFilter	SIGN_DETECTION_FILTER, 65
FilterManagementLibrary::PipeSystem::Processing ←	SIGN_DUPLICATION_DELETER_FILTER, 65
Pipeline, 49	SIGN_RECOGNITION_FILTER, 65
registeredFilters	RoadSignAPI::RSAPIWorkingDataSet, 65
FilterManagementLibrary::PipeSystem::Processing ←	classifierApprovedSigns, 66
Pipeline, 51	cutOutImages, 66
reset	detectedSignCombinations, 66
FilterManagementLibrary::PipeSystem::PipeFilter,	detectedSigns, 66
41	detectorScaledBGRImage, 66
RoadSignAPI::ClassifiedSignsGrouper, 15	originalBGRImage, 67
castedWorkingDataSet, 16	originalImageHeight, 67
horizontalRangePercentage, 17	originalImageWidth, 67
	RoadSignAPI::RoadSignAPI, 51
imageWidth, 17	assetManager, 58
initByPipeSetup, 16	<del>-</del>
process, 16	classifiedSignsGrouper, 58
RoadSignAPI::DetectedSignCombination, 17	detectionBasedImageSlicer, 58
~DetectedSignCombination, 18	feedImage, 54
addDetectedSign, 18	getClassifierApprovedDetectedSigns, 55
DetectedSignCombination, 18	getDetectedSignCombinations, 55
estimatedPolePositionX, 20	getDetectedSigns, 55

init, 56	RoadSignAPI_Standalone_Tensorflow/header_files/←
instance, 59	$Filter Management Library/Pipe System/TFN {\leftarrow}$
mobilenetModelDescription, 59	NBasedPipeFilter.h, 104
pipeRegisteredFilters, 59	RoadSignAPI_Standalone_Tensorflow/header_ <
processingPipeline, 59	files/FilterManagementLibrary/Tensorflow ←
RoadSignAPI, 53, 54	AndroidJNIUtils.h, 105
roadSignClassificator, 59	RoadSignAPI_Standalone_Tensorflow/header_
roadSignDetector, 59	files/FilterManagementLibrary/Tensorflow ←
roadSignDuplicationDeleter, 59	Integration/TensorflowNNInstance.h, 105
ssdLiteModelDescription, 60	RoadSignAPI Standalone Tensorflow/header
staticFeedImage, 56	files/FilterManagementLibrary/Tensorflow↔
staticGetClassifierApprovedDetectedSigns, 56	Integration/TensorflowNNInstanceClassifier.
staticGetDetectedSignCombinations, 57	h, 105
staticGetDetectedSigns, 57	RoadSignAPI_Standalone_Tensorflow/header_
staticInit, 57, 58	files/FilterManagementLibrary/Tensorflow←
workingDataSet, 60	Integration/TensorflowNNModelDescription.h,
	106
RoadSignAPI::RoadSignDuplicationDeleter, 60	RoadSignAPI_Standalone_Tensorflow/header_
calculateBoxArea, 61	files/FilterManagementLibrary/Tensorflow↔
calculateOverlapPercentage, 61	Integration/TensorflowResultContainer.h, 106
castedWorkingDataSet, 63	RoadSignAPI_Standalone_Tensorflow/header_files/
deleteAllMarkedSignDescriptors, 61	FilterManagementLibrary/TensorflowOpen
initByPipeSetup, 62	CVUtils.h, 106
isMarkedForDeletion, 62	
markForDeletion, 62	RoadSignAPI_Standalone_Tensorflow/header_files/ FilterManagementLibrary/Utilities.h, 107
minOverlapPercentage, 63	
process, 62	RoadSignAPI_Standalone_Tensorflow/header_files/
signsDoOverlap, 63	FilterManagementLibrary/limiting_file_input
signsToDelete, 64	_stream.h, 102
RoadSignAPI::SSDLiteRoadSignDetector, 67	RoadSignAPI_Standalone_Tensorflow/header_files/
applyImageVectorFromOpenCVMat, 69	RoadSignAPI/DetectedSignCombination.h,
assetManager, 71	107
castedWorkingDataSet, 71	RoadSignAPI_Standalone_Tensorflow/header_files/
initByPipeSetup, 70	RoadSignAPI/DetectedSignDescriptor.h, 107
nnModelInputHeight, 71	RoadSignAPI_Standalone_Tensorflow/header_
nnModelInputWidth, 72	files/RoadSignAPI/Filters/ClassifiedSigns←
onNNEvaluationFinished, 70	Grouper.h, 107
process, 71	RoadSignAPI_Standalone_Tensorflow/header_files/
SSDLiteRoadSignDetector, 68, 69	$Road Sign API/Filters/Detection Based Image \leftarrow$
threshold 72	Slicer.h, 108
RoadSignAPI_Standalone_Tensorflow/_10_introduction.	_RoadSignAPI_Standalone_Tensorflow/header_files/←
dox, 101	RoadSignAPI/Filters/MobilenetV2RoadSign←
RoadSignAPI_Standalone_Tensorflow/codesnippets. ←	Classificator.h, 108
cpp, 101	RoadSignAPI_Standalone_Tensorflow/header_files/
RoadSignAPI_Standalone_Tensorflow/header_files/←	RoadSignAPI/Filters/RoadSignDuplication←
FilterManagementLibrary/Logger.h, 103	Deleter.h, 108
RoadSignAPI Standalone Tensorflow/header files/↔	RoadSignAPI_Standalone_Tensorflow/header_files/
FilterManagementLibrary/PipeSystem/Pipe	RoadSignAPI/Filters/SSDLiteRoadSign↔
Filter.h, 103	Detector.h, 108
RoadSignAPI_Standalone_Tensorflow/header_files/↔	RoadSignAPI_Standalone_Tensorflow/header_files/
FilterManagementLibrary/PipeSystem/Pipe	
RegisteredFilters.h, 103	RSAPIPipeRegisteredFilters.h, 109
RoadSignAPI_Standalone_Tensorflow/header_files/←	RoadSignAPI_Standalone_Tensorflow/header_files/
FilterManagementLibrary/PipeSystem/Pipe↔	RoadSignAPI/RSAPIWorkingDataSet/RSA
WorkingDataSet.h, 104	PIWorkingDataSet.h, 109
RoadSignAPI_Standalone_Tensorflow/header_files/←	RoadSignAPI_Standalone_Tensorflow/header_files/
FilterManagementLibrary/PipeSystem/←	RoadSignAPI/RoadSignAPI.h, 109
ProcessingPipeline.h, 104	RoadSignAPI_Standalone_Tensorflow/main.cpp, 109
1 100000mgr ipomio.ii, 104	Troducing in a local data of the following in the first of the first o

RoadSignAPI_Standalone_Tensorflow/mainold.cpp,  111	roadSignDuplicationDeleter RoadSignAPI::RoadSignAPI, 59
RoadSignAPI_Standalone_Tensorflow/source_files/	runInference
FilterManagementLibrary/Logger.cpp, 113	FilterManagementLibrary::TFIntegration::Tensorflow-
RoadSignAPI_Standalone_Tensorflow/source_files/↔	NNInstance, 80
<del>-</del>	Taranstance, ou
FilterManagementLibrary/PipeSystem/Pipe  Filter and 110	SIGN_DETECTION_FILTER
Filter.cpp, 113	RoadSignAPI::RSAPIPipeRegisteredFilters, 65
$RoadSignAPI\_Standalone\_Tensorflow/source\_files/{\leftarrow}$	SIGN_DUPLICATION_DELETER_FILTER
FilterManagementLibrary/PipeSystem/←	RoadSignAPI::RSAPIPipeRegisteredFilters, 65
ProcessingPipeline.cpp, 113	
RoadSignAPI_Standalone_Tensorflow/source_files/←	SIGN_RECOGNITION_FILTER
FilterManagementLibrary/PipeSystem/TFN←	RoadSignAPI::RSAPIPipeRegisteredFilters, 65
NBasedPipeFilter.cpp, 113	SSDLiteRoadSignDetector
RoadSignAPI_Standalone_Tensorflow/source_	RoadSignAPI::SSDLiteRoadSignDetector, 68, 69
files/FilterManagementLibrary/Tensorflow↔	scaledImage
	codesnippets.cpp, 102
AndroidJNIUtils.cpp, 113	sessionOptions
RoadSignAPI_Standalone_Tensorflow/source_	FilterManagementLibrary::TFIntegration::Tensorflow
files/FilterManagementLibrary/Tensorflow←	NNInstance, 82
Integration/TensorflowNNInstance.cpp, 114	setCredentials
RoadSignAPI_Standalone_Tensorflow/source_←	
files/FilterManagementLibrary/Tensorflow←	FilterManagementLibrary::PipeSystem::PipeFilter,
Integration/TensorflowNNInstanceClassifier.	41 <del>-</del>
cpp, 114	setLogTag
RoadSignAPI_Standalone_Tensorflow/source_	FilterManagementLibrary::Logger, 30
files/FilterManagementLibrary/Tensorflow←	setup
Integration/TensorflowResultContainer.cpp,	FilterManagementLibrary::PipeSystem::Processing ←
	Pipeline, 50
114 D 10: 17: 0: 11: T (1: /	setupModelFromAssets
RoadSignAPI_Standalone_Tensorflow/source_files/←	FilterManagementLibrary::PipeSystem::TFNN←
FilterManagementLibrary/TensorflowOpen ←	BasedPipeFilter, 97
CVUtils.cpp, 114	FilterManagementLibrary::TFIntegration::Tensorflow-
RoadSignAPI_Standalone_Tensorflow/source_files/←	NNInstance, 80
FilterManagementLibrary/Utilities.cpp, 115	
RoadSignAPI_Standalone_Tensorflow/source_files/	setupModelFromFile
RoadSignAPI/DetectedSignCombination.cpp,	FilterManagementLibrary::PipeSystem::TFNN←
115	BasedPipeFilter, 98
RoadSignAPI_Standalone_Tensorflow/source_←	FilterManagementLibrary::TFIntegration::Tensorflow-
<del>-</del>	NNInstance, 81
files/RoadSignAPI/Filters/ClassifiedSigns↔	signs
Grouper.cpp, 115	RoadSignAPI::DetectedSignCombination, 20
$RoadSignAPI\_Standalone\_Tensorflow/source\_files/{\leftarrow}$	signsDoOverlap
RoadSignAPI/Filters/DetectionBasedImage←	RoadSignAPI::RoadSignDuplicationDeleter, 63
Slicer.cpp, 115	signsToDelete
RoadSignAPI_Standalone_Tensorflow/source_files/←	RoadSignAPI::RoadSignDuplicationDeleter, 64
RoadSignAPI/Filters/MobilenetV2RoadSign←	Skip
Classificator.cpp, 115	tensorflow::android::LimitingFileInputStream, 27
RoadSignAPI_Standalone_Tensorflow/source_files/←	
RoadSignAPI/Filters/RoadSignDuplication ←	ssdLiteModelDescription
Deleter.cpp, 115	RoadSignAPI::RoadSignAPI, 60
···	start_time
RoadSignAPI_Standalone_Tensorflow/source_files/←	codesnippets.cpp, 102
RoadSignAPI/Filters/SSDLiteRoadSign←	staticFeedImage
Detector.cpp, 115	RoadSignAPI::RoadSignAPI, 56
RoadSignAPI_Standalone_Tensorflow/source_files/←	staticGetClassifierApprovedDetectedSigns
RoadSignAPI/RoadSignAPI.cpp, 115	RoadSignAPI::RoadSignAPI, 56
RoadSignAPI, 13	staticGetDetectedSignCombinations
RoadSignAPI::RoadSignAPI, 53, 54	RoadSignAPI::RoadSignAPI, 57
roadSignClassificator	staticGetDetectedSigns
RoadSignAPI::RoadSignAPI, 59	RoadSignAPI::RoadSignAPI, 57
	-
roadSignDetector	staticInit
RoadSignAPI::RoadSignAPI, 59	RoadSignAPI::RoadSignAPI, 57, 58

```
TFNNBasedPipeFilter
     FilterManagementLibrary::PipeSystem::TFNN←
         BasedPipeFilter, 94
tensorflow, 13
tensorflow::android, 13
tensorflow::android::LimitingFileInputStream, 26
     ~LimitingFileInputStream, 27
    bytes_left_, 27
     errno, 27
    fd_, 28
     LimitingFileInputStream, 26
     Read, 27
     Skip, 27
TensorflowAndroidJNIUtils.cpp
     ASSET_PREFIX, 114
TensorflowNNInstance
     FilterManagementLibrary::TFIntegration::Tensorflow←
         NNInstance, 76
TensorflowNNInstanceClassifier
     FilterManagementLibrary::TFIntegration::Tensorflow←
         NNInstanceClassifier, 84
TensorflowOpenCVUtils.cpp
     Pixel, 114
TensorflowResultContainer
     FilterManagementLibrary::TFIntegration::Tensorflow←
         ResultContainer, 91
tensorflowSession
     FilterManagementLibrary::TFIntegration::Tensorflow←
         NNInstance, 82
testNet
     mainold.cpp, 113
tfNNInstance
     FilterManagementLibrary::PipeSystem::TFNN←
         BasedPipeFilter, 98
threshold
     RoadSignAPI::MobilenetV2RoadSignClassificator,
     RoadSignAPI::SSDLiteRoadSignDetector, 72
type2str
     main.cpp, 111
     mainold.cpp, 113
     RoadSignAPI::DetectedSignDescriptor, 21
validRecognition
     RoadSignAPI::MobilenetV2RoadSignClassificator,
workingDataSet
     FilterManagementLibrary::PipeSystem::Processing -
         Pipeline, 51
     RoadSignAPI::RoadSignAPI, 60
WriteProtoToFile
     FilterManagementLibrary::TensorflowAndroidJNI ←
         Utils, 73
```