NEURO CHECK
Industrial Vision Systems

Application Software for Industrial Vision Systems

Neural Network Tools

Plug-In

Version 6.2.11
Neckarstraße 76-1, 71686 Remseck, Germany

Phone: +49 (0) 7146 - 89 56-0
Fax: +49 (0) 7146 - 89 56-29
E-Mail: info@neurocheck.com
Web: www.neurocheck.com

Table of Contents

Introduction

General information about NeuroCheck plug-in DLLs

A plug-in DLL is a .NET assembly that serves to enhance NeuroCheck with user-defined image processing functionality. The NeuroCheck Plug-In Interface offers the opportunity to integrate user-defined check functions for image processing and data handling. A Plug-In can contain an arbitrary number of self-developed check functions.

These check functions have full access to the NeuroCheck runtime data objects such as Images, ROI Lists or Measurement Lists. The Plug-In check function can be added to a check as well as the built-in standard check functions of NeuroCheck.

Please note that for integration of a plug-in check function into your check routine, a Premium license is required. The completed check routine then can be run with any NeuroCheck license (except the Demo Version).

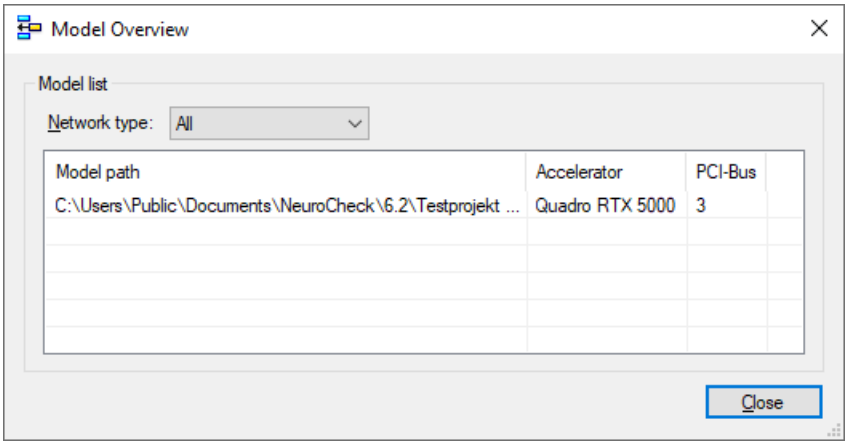
Deep Learning Inference with the PI_NeuralNetworkTools.NET

PI_NeuralNetworkTools.Net is a plug-in for the NeuroCheck Software. It can be used to deploy neural networks using the Open Neural Network Exchange (ONNX) framework. This plug-in is not a training pipeline. It is an interpreter for Open Neural Network Exchange models. The plug-in covers the tasks classification, object detection, anomaly detection, semantic segmentation and the usage of auto encoders.

Model Overview

A single model can be deployed on multiple devices, e.g., one network on each installed graphics card. To obtain a list of all models which are currently loaded by the plug-in, the user can open the model overview dialog located at `Tools > PI_NeuralNetworkTools: Model Overview`.

Screenshot of Model Overview Dialog



Models parametrized by Data Input will not be listed here.

Installation

Installation of the Plug-In

Copy the following files from the zip archive to the Plug-In directory within the desired NeuroCheck project (e.g. '`C:\Users\Public\Documents\NeuroCheck\6.2\Default\Software Extensions\PlugIns`').

- All files inside the `Binaries` directory
- All *.chm files inside the `Documentation` directory

Installation of Prerequisites (CPU and GPU)

If Microsoft Visual C++ Redistributable 2015-2019 is not installed yet, start `VC_redist.x64.exe`, which you can find in the `Prerequisites` directory. If you are unsure whether MSVC is installed, start the installer anyway as it will abort if a version is found on your PC.

Installation of Prerequisites (GPU)

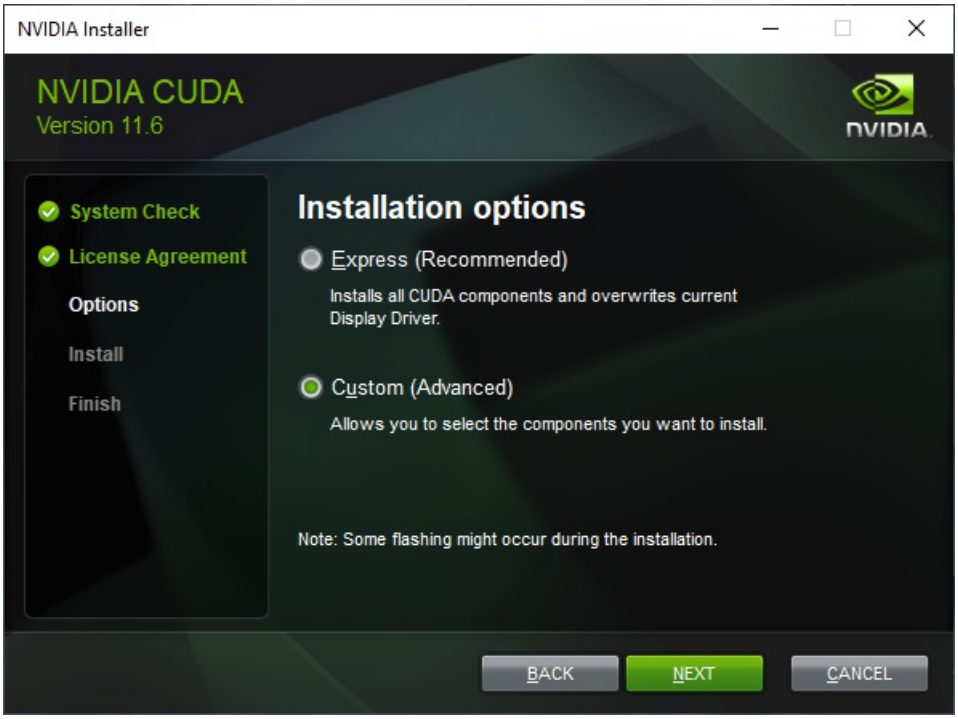
A NVIDIA GPU can be used to speed up the neural network computations. The Plug-In requires a GPU with compute capability of at least 5.0. You can find more information here.

To use a GPU as a hardware accelerator you must install CUDA and cuDNN. The dependencies are located in the `Prerequisites` directory. To perform the installation please follow the description below.

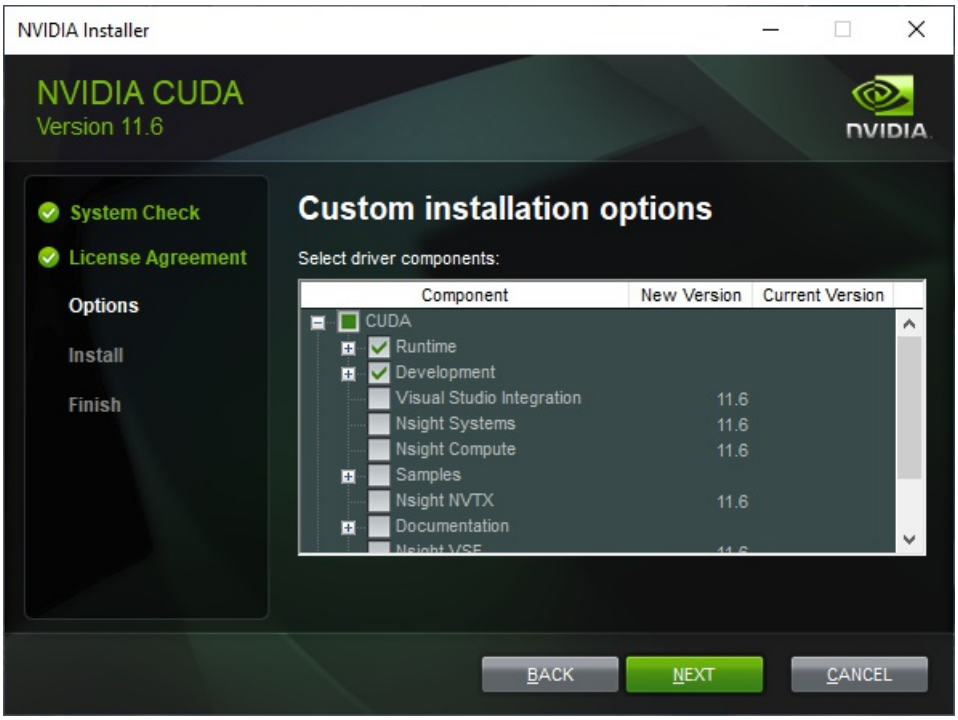
Start 'cuda_*.exe' and unpack the installation dependencies to a directory of your choice.

If done so, the installation dialog will pop up and perform a system check. If this was successful, you can install CUDA.

First select 'custom' installation:

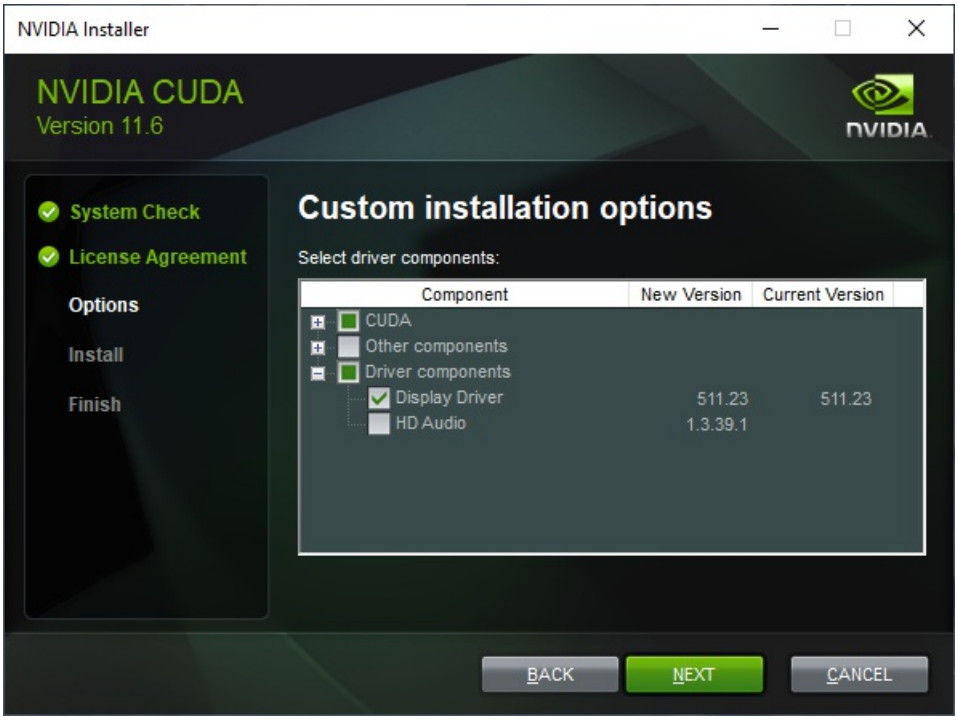


Then unselect CUDA and select 'Runtime' and 'Development'.



Unselect 'Other components' and 'Driver components'.

If you need a driver for your card or your driver is older than the provided driver, you can optionally select the 'Display Driver'.



The installation will be performed and you need to restart your computer when the installation was successfully finished.

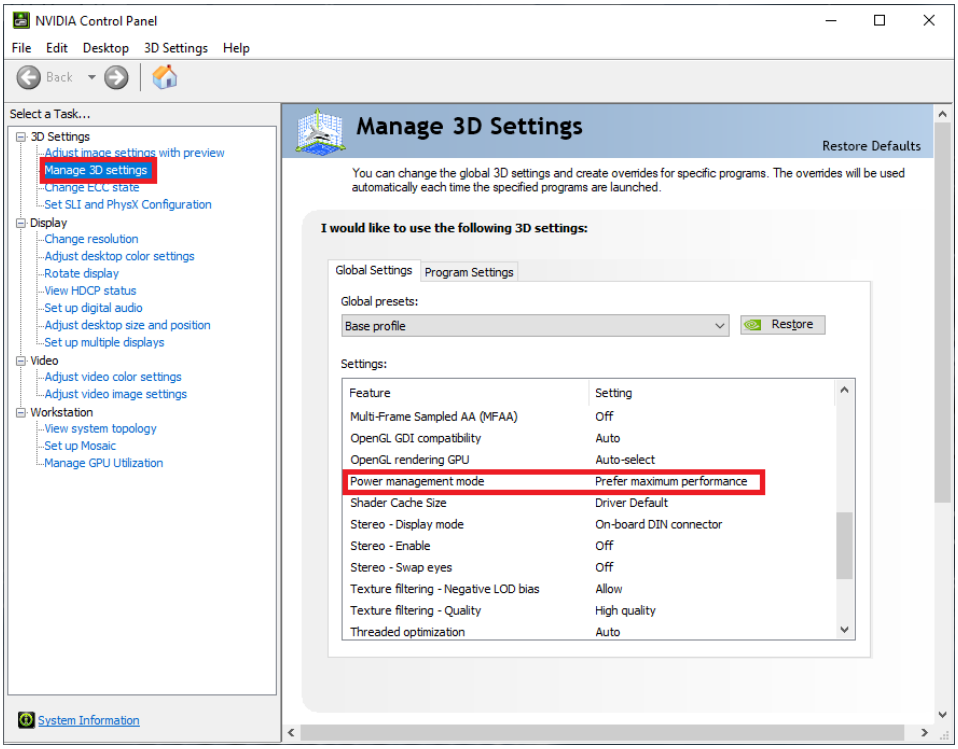
If you are using Windows 7, you additionally need to install the two patches in the `Cuda Patches` directory and replace the Onnx '*.dll' files in your NeuroCheck Plug-In directory.

After you installed CUDA, you need to install cuDNN.

First unzip the file 'cudnn-*.zip' in the `Prerequisites` directory.

Then copy the directory `bin` and the file `LICENSE` to the CUDA installation directory, e.g.: '`C:\ProgramData\NVIDIA GPU Computing Toolkit\v11.6`'.

As a last step, it is necessary to change the performance mode of the GPU for fast responses. Open the NVIDIA Control Panel by right-clicking the desktop.



Under 'Manage 3D settings', change the power management mode to 'Prefer maximum performance' as marked in the screenshot above.

Finally, you need to restart your PC one last time, so the change becomes active.

Loading the Plug-In

In order to use a Plug-In the Plug-In assembly must be loaded in NeuroCheck. The management of Plug-Ins takes place within the Software Settings dialog. The Software Settings dialog can be found in the System menu of NeuroCheck.

Please note that it is impossible to load or unload a Plug-In as long as a check routine is opened that contains the Plug-In check functions. If the currently opened check routine contains Plug-In check functions then close the check routine first.

Within the Software Settings dialog please select the node Plug-Ins and the sub-node Plug-In in the tree to the left. The loaded Plug-In assemblies are shown in the List of Plug-Ins. Press the Add button to open a file selection dialog in order to select a further Plug-In assembly.

Inserting a Plug-In check function to a check routine

A Plug-In check function is inserted using the Check Function Select dialog. All check functions of loaded Plug-Ins are listed in the Plug-In category of the Check Function Select dialog. Within the Plug-In category the check functions are ordered in sub-categories where each sub-category represents the check functions of one Plug-In.

Besides the category the user will hardly notice any difference between the usage of Plug-In check functions and built-in check functions.

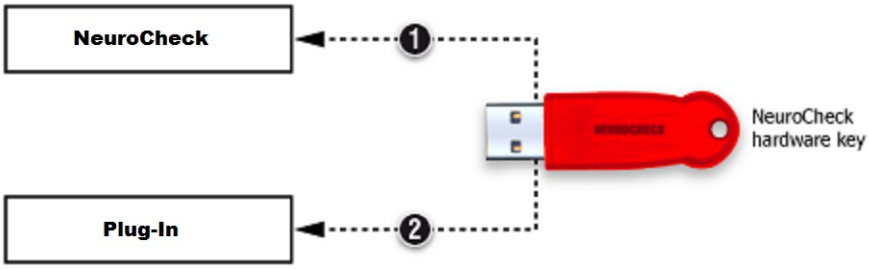
Dynamic Inputs on GPU

If model inference is performed on GPU and the number of ROIs that the network is evaluating changes every iteration, users may experience *multiple* slow inferences in the beginning (e.g. over 1 second). In these cases, close NeuroCheck, open `Configuration\NC62CFG.PI.CFGX` and change the value of `HeuristicInitialisation` to 1. Usually, it is recommended to leave

the value at 0, as this allows for additional optimizations which will pay off in the long run.

Licensing

This section describes the licensing mechanism for this NeuroCheck Plug-In.



1. Protection of NeuroCheck

NeuroCheck requires a valid license which is provided as hardware security key (dongle). USB and LPT dongles are available. Please note that a Premium license is required in order to integrate a plug-in check function into your check routine. If the check routine is completed once (including the plug-in functions) it can be run with any NeuroCheck license (except demo).

You obtain the standard NeuroCheck license when you purchase the software from your local NeuroCheck partner.

2. Protection of Plug-In

In addition to the standard NeuroCheck license, also a license for the NeuroCheck Plug-In is required. The protection of the plug-in is stored as a special flag in the same dongle as for the NeuroCheck license. If the plug-in cannot detect the special flag, the execution of the plug-in check functions in the automatic mode will always return NOK and the check functions in manual mode will periodically return NOK with a license error.

In order to get the license for the plug-in, please contact your local NeuroCheck partner. The license can be added to a standard NeuroCheck license by remote-programming of the dongle. The remote-programming works in the same way as a NeuroCheck update.

Input Definition

Data Format

The input layer of the network should be of type "Uint8" or "Uint16". Note that you cannot pass 16 bit images to a "Uint8" input layer. The reverse case will only work correctly if the network uses adaptive normalization.

If the input layer is of any other type than "Uint8" or "Uint16", the input image needs to casted to that specific type. This will have an negative impact on the inference speed.

Dimension

The input layer of the neural network is expected to be a placeholder with four dimensions (see section Format).

Format

The images will be passed to the Open Neural Network Exchange runtime in the "NHWC" or "NCHW format. N = number of batches, H = height of image, W = width of image and C = image channels.

Notice: If you use NHWC format the convertion between NeuroCheck images and the tensors will be slower than the NCHW format, by a factor of (H x W) ^ 2.

Normalizing

If the neural network expects a normalized image, the input layer should take care of the normalization.

Scaling

If the input images NeuroCheck passes to the neural network need to be scaled, this can be done beforehand, using the NeuroCheck tools or the network scales the images.

Classification

The output of the classification network should be a tensor coming from the softmax function. The output tensor is defined as $X \in \mathbb{R}^{n \times c}$ where $n \in \mathbb{N}^*$ is the number of minibatches and $c \in \mathbb{N}^*$ the number of classes.

The datatype must be Float32.

Object Detection

The object detection task has four output layers. One for the number of detections, the box coordinates, the scores and the classes.

Number of detections

The layer containing the number of detections must match the shape $X \in \mathbb{N}^{n}_{>0}$ with $n \in \mathbb{N}^{*}$ as the number of minibatches. The preferred datatype is Float32, but can be Int32 as well. Otherwise the datatype will be casted to the target data type.

Box coordinates

The layer providing the box coordinates must match the shape $X \in \mathbb{R}^{n \times d \times 4}$ with $n \in \mathbb{N}^{*}$ as the number of minibatches, $d \in \mathbb{N}^{*}$ as the number of detections and four coordinates in order ymin, xmin, ymax, xmax of shape $\mathbb{R}$ providing non normalized coordinates. The preferred datatype is Float32. Otherwise the datatype will be casted to the target data type.

Scores

The layer providing the scores must match the shape $X \in \mathbb{R}^{n \times d}$ with $n \in \mathbb{N}^{*}$ as the number of batches and $d \in \mathbb{N}^{*}$ as the number of detections. The datatype is Float32.

Classes

The layer providing the classes must match the shape $X \in \mathbb{R}^{n \times d}$ with $n \in \mathbb{N}^{*}$ as the number of minibatches and $d \in \mathbb{N}^{*}$ as the number of detections. The preferred datatype is Float32. Otherwise the datatype will be casted to the target data type.

Semantic Segmentation

The output layer must produce a segmentation map of shape $X \in \mathbb{N}^{n \times h \times w}$ with $n \in \mathbb{N}^*$ as the number of minibatches, $h \in \mathbb{N}^*$ as the height of the segmentation map, $w \in \mathbb{N}^*$ as the width of the segmentation map.

The data type of the segmentation map must be Uint8, otherwise the segmentation map must be casted. Note: since the datatype of the segmentation map will be casted to Uint8, the maximum number of classes will be limited to 256.

VariationalAuto Encoders

The output layer of an auto encoder must produce an tensor of shape $X \in \mathbb{N}^{n \times h \times w \times c}$ or $\mathbb{N}^{n \times c \times h \times w}$ with $n \in \mathbb{N}^*$ as the number of minibatches, $h \in \mathbb{N}^*$ as the height of the encoded image, $w \in \mathbb{N}^*$ as the width of the encoded image and $c \in \mathbb{N}^*$ as the channels of the encoded image. The channels can either be one or three.

Note that the output image of an variational auto encoder needs to be non normalized and all its values must lie in the range $X = \{x \in \mathbb{N} | 0 \leq x \leq 255\}$.

The data type of the encoded tensor must be Uint8, otherwise the encoded image must be casted.

Anomaly Detection

The anomaly detection task has two output layers. One for the anomaly scores and one for the anomaly maps.

Anomaly Score

The layer containing the anomaly scores must match the shape $X \in \mathbb{N}^{n}_{>0}$ with $n \in \mathbb{N}^{*}$ as the number of minibatches. The preferred datatype is Float32.

Anomaly map

The output layer for the anomaly maps must produce an tensor of shape $X \in \mathbb{N}^{n \times h \times w \times c}$ or $\mathbb{N}^{n \times c \times h \times w}$ with $n \in \mathbb{N}^{*}$ as the number of minibatches, $h \in \mathbb{N}^{*}$ as the height of the encoded image, $w \in \mathbb{N}^{*}$ as the width of the encoded image and $c \in \mathbb{N}^{*}$ as the channels of the encoded image. The channels can either be one or three.

Note that the anomaly map needs to be non normalized and all its values must lie in the range $X = \{x \in \mathbb{N} | 0 \leq x \leq 255\}$.

The data type of the tensor must be Uint8, otherwise the encoded image must be casted.

Model Format

Open Neural Network Exchange (Onnx)

To deploy Onnx models, the models can be trained using your favorite framework and then be converted to the Onnx format via the transformation API.

Auto Encode Objects: Introduction

Function overview

This function can be used to auto encode the objects in the input list of ROIs.

Input data

This check function requires an image and a list of ROIs as input data objects.

Output data

An image containing the encoded objects.

Result view

The result view shows an image of the encoded objects.

Properties

 Check function group Plug-In.

 The check function has a Parameter Dialog.

 The check function has own result Visualizations.

Auto Encode Objects : How to Use

You can deploy your pre-trained auto encoder on a single image or compute more than one images at once.

1. **Single image**

   To deploy your auto encoder on a single image, create an image and a list of ROIs with exactly one entry.

   The size of the ROI must match the input layers dimensions.

2. **Multi image**

   To deploy your auto encoder on more than one image at once, create an image and a list of ROIs with as many entries you want.

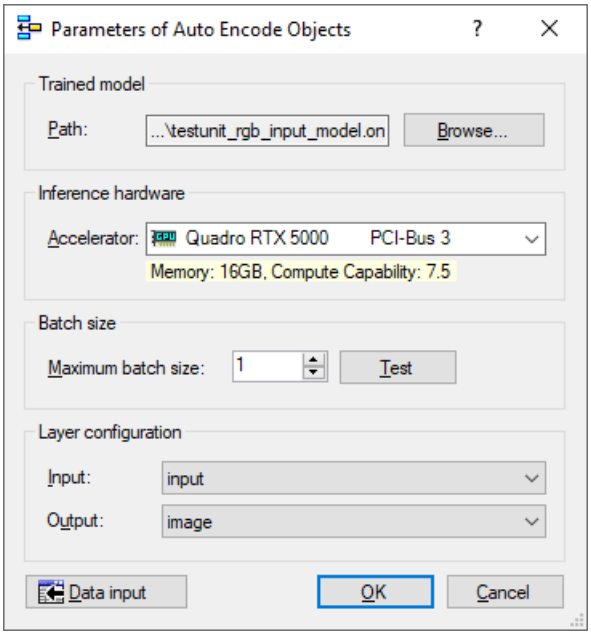
   Note that the size of each ROI must match the input layers dimensions.

If the neural network was created with NeuroCheck Toolkit Version 2.1.6 or newer, the ROI size can be arbitrary and multiple ROIs do not need to have identical dimensions.

Auto Encode Objects Objects: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

Screenshot of Parameter Dialog



The **Parameter** dialog contains the following elements:

Element	Description
Path	The full path to the pre-trained model.
Accelerator	The hardware accelerator to use for inference (Default CPU).
Batch size	The maximum number of images to be processed at once.
Test	Test if memory limit is reached when predicting using the given batch size.
Input	The input layer name.
Output	The output layer name.

Auto Encode Objects: Visualization

This section describes the result visualizations the check function "Auto Encode Objects" provides.

Element	Description
Input Image	Displays the input image of the check function.
Input ROI Collection	Displays a list of the input ROI collection data of the check function.
Output Image	Displays the auto-encoded image.

Classify Objects: Introduction

Function overview

This function can be used to classify the objects in the input list of ROIs.

Input data

This check function requires an image and a list of ROIs as input data objects.

Output data

None (classification results will be attached to the input list of ROIs).

Result view

The result view shows the input image and a list of the classified input ROI collection.

Properties

 Check function group Plug-In.

 The check function has a Parameter Dialog.

 The check function has own result Visualizations.

Classify Objects : How to Use

You can deploy your pre-trained classification network on a single image or compute more than one images at once.

1. **Single image**

   To deploy your classification network on a single image, create an image and a list of ROIs with exactly one entry.

   The size of the ROI must match the input layers dimensions.

2. **Multi images**

   To deploy your classification network on more than one image at once, create an image and a list of ROIs with as many entries you want.

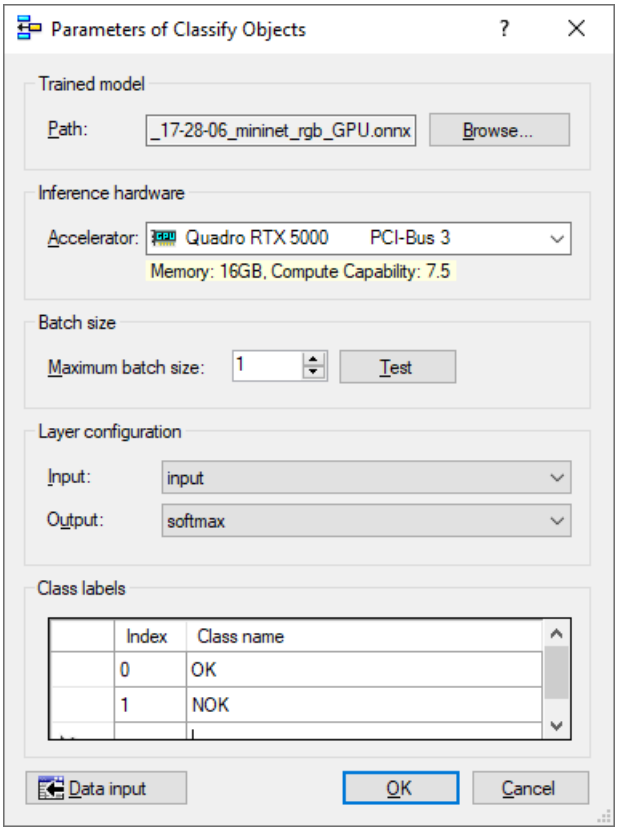
   Note that the size of each ROI must match the input layers dimensions.

If the neural network was created with NeuroCheck Toolkit Version 2.1.6 or newer, the ROI size can be arbitrary and multiple ROIs do not need to have identical dimensions.

Classify Objects: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

Screenshot of Parameter Dialog



The **Parameter** dialog contains the following elements:

Element	Description
Path	The full path to the pre-trained model.
Accelerator	The hardware accelerator to use for inference (Default CPU).
Batch size	The maximum number of images to be processed at once.
Test	Test if memory limit is reached when predicting using the given batch size.
Input	The input layer name.
Output	The output layer name.
Class labels	(optional) Mapping of class indices to class names. The class indices and names can be pasted from the clippboard. Examples for allowed patterns: • 1,Class1 • 2;Class2 • 3 Class3 • 4|Class4

Element	Description

Classify Objects: Visualization

This section describes the result visualizations the check function "Classify Objects" provides.

Element	Description
Input Image	Displays the input image of the check function.
Input ROI Collection	Displays a list of the input ROI collection data of the check function and the appended class scores.

Detect Anomalies: Introduction

Function overview

This function can be used to detect anomalies in the input list of ROIs. It offers an option to binarize the detected anomalies. If the option is enabled, the check function will perform binarization of the anomaly map based on a user-defined anomaly threshold. If the option is disabled, the check function will instead show an anomaly score for each ROI, where a score higher than 50% hints at an anomaly.

Input data

This check function requires an image and a list of ROIs as input data objects.

Output data

An image containing the anomaly map and a list of ROIs. If binarization is disabled, the list of ROIs contains the anomaly scores of the network. If binarization is enabled, the list of ROIs contains the result of binarization.

Result view

The result view shows the input image and a list of the classified input ROI collection.

Properties

 Check function group Plug-In.

 The check function has a Parameter Dialog.

 The check function has own result Visualizations.

Detect Anomalies : How to Use

You can deploy your pre-trained classification network on a single image or compute more than one images at once.

1. **Single image**

   To deploy your classification network on a single image, create an image and a list of ROIs with exactly one entry.

   The size of the ROI must match the input layers dimensions.

2. **Multi images**

   To deploy your classification network on more than one image at once, create an image and a list of ROIs with as many entries you want.

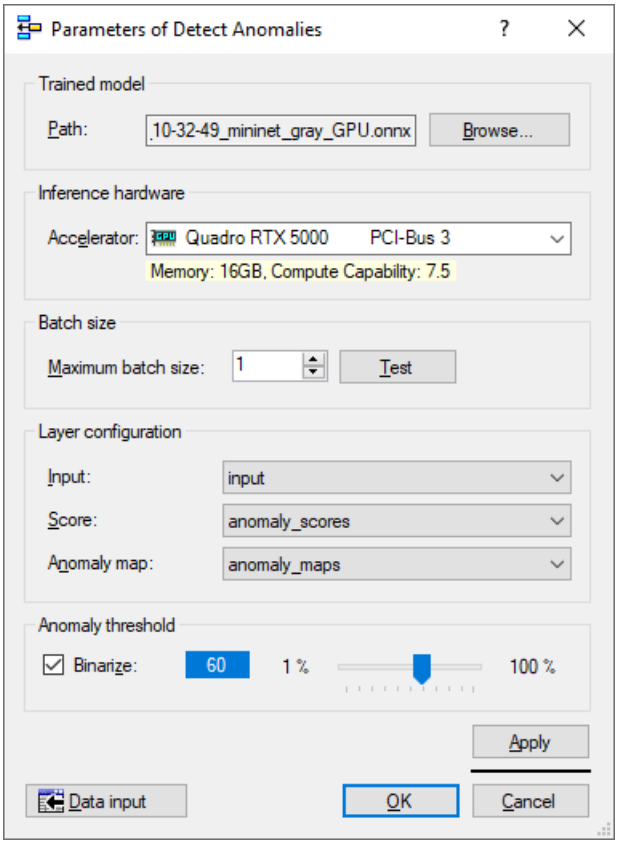
   Note that the size of each ROI must match the input layers dimensions.

If the neural network was created with NeuroCheck Toolkit Version 2.1.6 or newer, the ROI size can be arbitrary and multiple ROIs do not need to have identical dimensions.

Detect Anomalies: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

Screenshot of Parameter Dialog



The **Parameter** dialog contains the following elements:

Element	Description
Path	The full path to the pre-trained model.
Accelerator	The hardware accelerator to use for inference (Default CPU).
Batch size	The maximum number of images to be processed at once.
Test	Test if memory limit is reached when predicting using the given batch size.
Input	The input layer name.
Score	The scores layer name.
Anomaly map	The anomaly map layer name.
Binarize	If binarization is computed or anomaly scores are returned.
Anomaly Threshold	The threshold of what is considered an anomaly during binarization.

Detect Anomalies: Visualization

This section describes the result visualizations the check function "Detect Anomalies" provides.

Element	Description
Input Image	Displays the input image of the check function.
Input ROI Collection	Displays an image of the input ROI collection data of the check function.
Output Image	Displays the anomaly response map.
Output ROI Collection	Displays the input ROIs with anomaly class scores or the anomaly regions created by binarization.

Detect Objects: Introduction

Function overview

This function can be used to detect the objects within the input list of ROIs.

Input data

This check function requires an image and a list of ROIs as input data objects.

Output data

List of ROIs containing classified AOIs.

Properties

Check function group Plug-In.

The check function has a Parameter Dialog.

Detect Objects : How to Use

You can deploy your pre-trained detection network on a single image or compute more than one images at once.

1. **Single image**

   To deploy your detection network on a single image, create an image and a list of ROIs with exactly one entry.

   The size of the ROI must match the input layers dimensions.

2. **Multi images**

   To deploy your detection network on more than one image at once, create an image and a list of ROIs with as many entries you want.

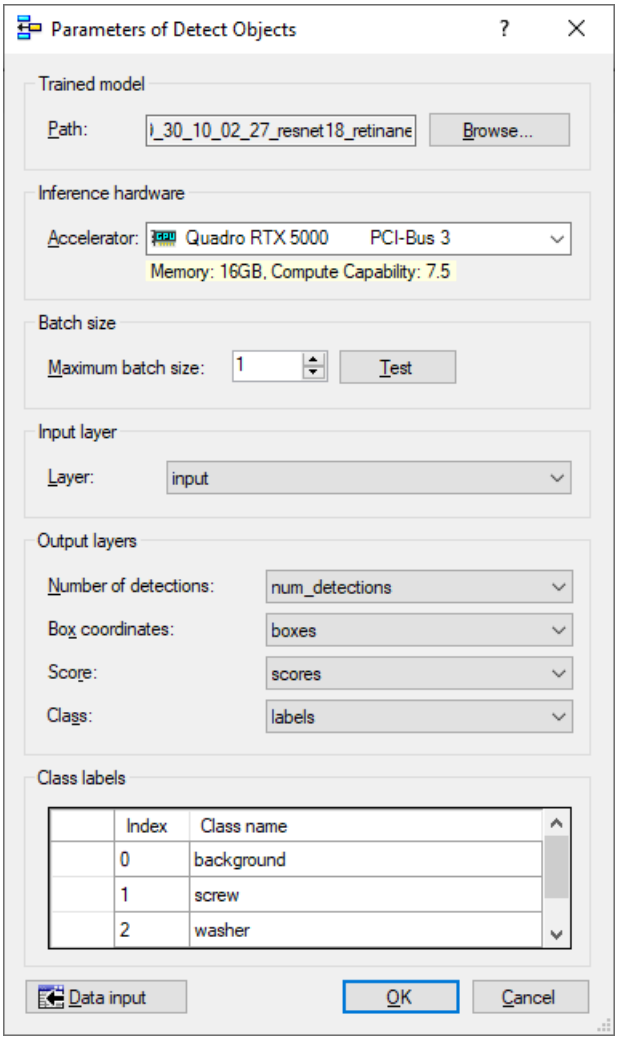
   Note that the size of each ROI must match the input layers dimensions.

If the neural network was created with NeuroCheck Toolkit Version 2.1.6 or newer, the ROI size can be arbitrary and multiple ROIs do not need to have identical dimensions.

Detect Objects: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

Screenshot of Parameter Dialog



The **Parameter** dialog contains the following elements:

Element	Description
Path	The full path to the pre-trained model.
Accelerator	The hardware accelerator to use for inference (Default CPU).
Batch size	The maximum number of images to be processed at once.
Test	Test if memory limit is reached when predicting using the given batch size.
Input Layer	The input layer name.
Number of detections	The layer name containing the number of detections.
Box coordinates	The layer name containing the box coordinates.
Score	The layer name containing the scores.
Class	The layer name containing the classes.
Class labels	Mapping of class indices to class names. For custom models, it is important to define all classes. Otherwise, the class feature may contain wrong values. The class indices and names can be pasted from the clippboard. Examples for allowed patterns: • 1,Class1 • 2;Class2 • 3 Class3 • 4|Class4

Detect Objects: Visualization

This section describes the result visualizations the check function "Detect Objects" provides.

Element	Description
Input Image	Displays the input image of the check function.
Input ROI Collection	Displays a list of the input ROI collection data of the check function.
Output ROI Collection	Displays a list of the detected objects with their class scores.

Segment Objects: Introduction

Function overview

This function can be used to segment the objects in the input list of ROIs.

Input data

This check function requires an image and a list of ROIs as input data objects.

Output data

List of ROIs containing segmented objects as boundary ROIs.

Result view

The result view shows the ROI objects.

Properties

 Check function group Plug-In.

 The check function has a Parameter Dialog.

 The check function has own result Visualizations.

Segment Objects : How to Use

You can deploy your pre-trained segmentation network on a single image or compute more than one images at once.

1. **Single image**

   To deploy your segmentation network on a single image, create an image and a list of ROIs with exactly one entry.

   The size of the ROI must match the input layers dimensions.

2. **Multi images**

   To deploy your segmentation network on more than one image at once, create an image and a list of ROIs with as many entries you want.

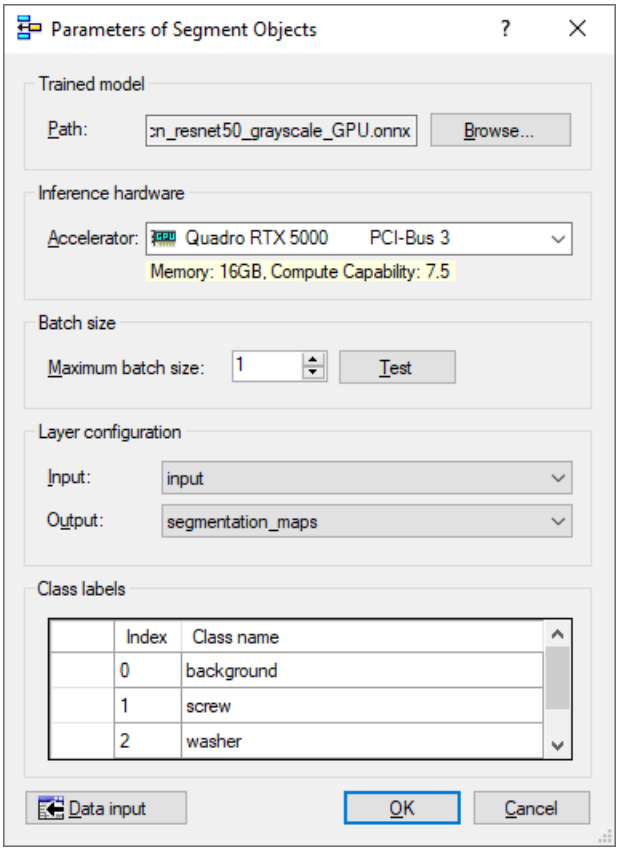
   Note that the size of each ROI must match the input layers dimensions.

If the neural network was created with NeuroCheck Toolkit Version 2.1.6 or newer, the ROI size can be arbitrary and multiple ROIs do not need to have identical dimensions.

Segment Objects: Parameter Dialog

This plug-in check function has a **Parameter** dialog.

Screenshot of Parameter Dialog



The **Parameter** dialog contains the following elements:

Element	Description
Path	The full path to the pre-trained model.
Accelerator	The hardware accelerator to use for inference (Default CPU).
Batch size	The maximum number of images to be processed at once.
Test	Test if memory limit is reached when predicting using the given batch size.
Input	The input layer name.
Output	The output layer name.

Element	Description
Class labels	Mapping of class indices to class names. For custom models, it is important to define all classes. Otherwise, the class feature may contain wrong values. The class indices and names can be pasted from the clippboard. Examples for allowed patterns: • 1,Class1 • 2;Class2 • 3 Class3 • 4|Class4

Segment Objects: Visualization

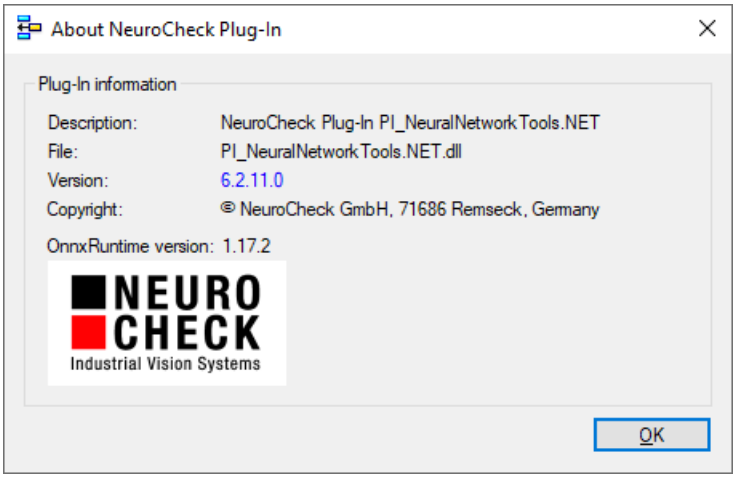
This section describes the result visualizations the check function "Segment Objects" provides.

Element	Description
Input Image	Displays the input image of the check function.
Input ROI Collection	Displays a list of the input ROI collection data of the check function.
Output ROI Collection	Displays a list of the regions containing objects which are not part of the background.

About Dialog

This dialog displays version information about the NeuroCheck Plug-In **PI_NeuralNetworkTools.NET.dll**.

Screenshot of About Dialog



Support Services

For technical support, please contact your local NeuroCheck partner or NeuroCheck GmbH:

Phone: +49 (0) 7146 - 89 56-40
E-Mail: support@neurocheck.com
Web: www.neurocheck.com

Before contacting us, please provide some important information about your system:

Information about your NeuroCheck installation and your PC setup:

- Use the NeuroCheck Diagnostics tool to check your installation and computer configuration.
- The NeuroCheck Diagnostics is installed in the "Tools" folder within your NeuroCheck installation.

Log file information:

- Logging for NeuroCheck can be activated in **System** > **Software Settings** > **Diagnosis** > **Logging**.