LumiScan Training Framework

User Guide

VISION
high dimensional vision systems

Imprint

HD Vision Systems GmbH
Palo-Alto-Platz 11
69124 Heidelberg

Tel.:	+49 (6221) 67219-00
Fax:	+49 (6221) 67219-01
E-Mail:	info@hdvisionsystems.com
WWW:	https://hdvisionsystems.com

CEO:	PD Dr. Christoph Garbe
Headquarters:	Heidelberg
Register court:	Amtsgericht Mannheim
Register number:	HRB 726917

Version_2025_10_10

Table of Contents

1 Introduction

1.1 About This Manual

This document contains important information about the installation and operation of the LumiScan Training Framework. Please read the manual carefully before using this software.

1.2 Formatting Convention

This guide uses special formatting to highlight certain words and phrases:

- Keywords/important information and buttons are highlighted in bold (e.g. Only two values are possible: **0.5** or **1.0**).
- Links and references are highlighted in yellow (e.g. info@hdvisionsystems.com).
- File and path names are highlighted in a special font (e.g. `exportDirectory`).

1.3 Disclaimer

NOTE: By accessing or using these commercial software products, you expressly agree to the following terms and conditions.

Any attempt to use a debugger to examine, analyze, or tamper with the software provided by HD Vision Systems is strictly prohibited and may have immediate and irreversible consequences.

If the software detects the presence of a debugger, security protocols will be activated to protect the intellectual property, functionality and stability of the software. This may result in, among other things, immediate suspension of the associated Software license, loss of data, and, in extreme cases, forced termination of all instances of the Software operating under the same license, as well as legal consequences.

By using the Software, you acknowledge that you have read, understood and accepted the terms of this Disclaimer (see also General Disclaimer).

2 LumiScan Training Framework

2.1 Description

The LumiScan Training Framework allows easy training of neural networks for computer vision tasks. This includes object recognition tasks. It also allows the user to train an existing network to adapt it to new data, or to train a network from scratch and convert it to ONNX.

2.2 Requirements

- CUDA Toolkit 11.8 or higher
- Code Meter 7.60 or higher
- NVIDIA GeForce RTX 2080 SUPER
- Anaconda
- Windows 10 or higher/Linux

If you have any problems or questions about the product, please contact HD Vision Systems Customer Support at +49 6221 6721905
or by e-mail: customer.success@hdvisionsystems.com

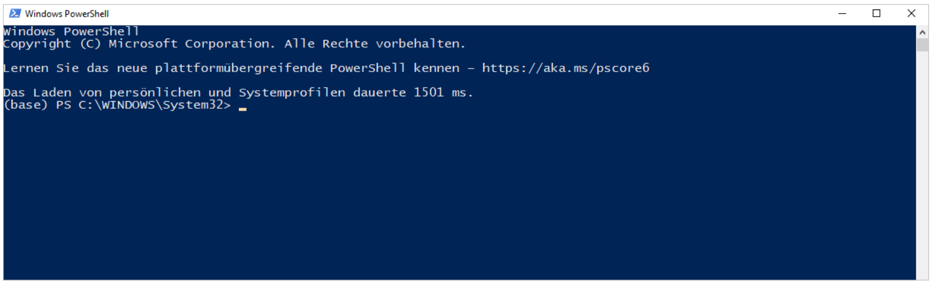
2.3 Before Installation

2.3.1 Customizing the PowerShell Execution Policy

NOTE: When in doubt, contact your IT department. They can manually run the installation script line by line to see what has changed in your system.

To start:

1. Type **PowerShell** in the Windows Search bar.
2. When Windows PowerShell appears in the list, open it.
3. PowerShell opens as a blue text/command-line interface.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen – https://aka.ms/pscore6

Das Laden von persönlichen und Systemprofilen dauerte 1501 ms.
(base) PS C:\WINDOWS\System32> _
```

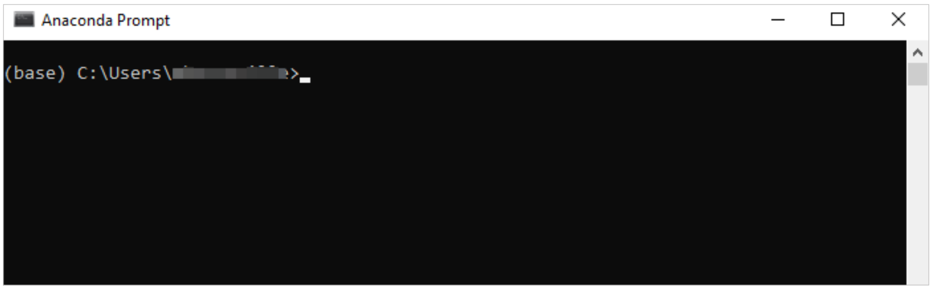
4. Disable execution policy protection with the command:
   Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope CurrentUser
5. You can now close the command prompt.

2.3.2 Activation of Annaconda for PowerShell

This requires the following steps:

1. Type **Anaconda Prompt** in the Windows search bar.
2. When the Anaconda Command Prompt (Anaconda3) appears in the list, select it with the right-click.
3. Select **Run as administrator**.

4. The command prompt opens as a black text interface (terminal).



```
(base) C:\Users\[redacted]>_
```

5. Enable Anaconda for Powershell with the command:
   conda init powershell
6. You can now close the command prompt.

2.4 LumiScan Training Framework Installation

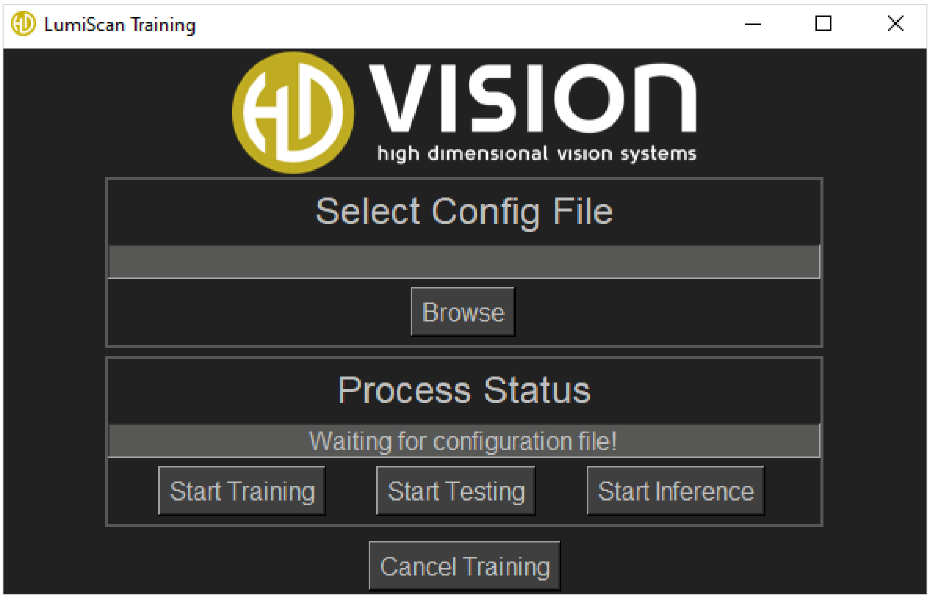
To install LumiScan Training Framework:

1. Navigate to the location where the LumiScan Training Framework installer is stored.
2. Double-click: `Installer_LumiScan_Training_Framework_vx_x_x_x.exe` **(x stands for the version number)**
3. Select the option **Installation for all users**.
4. Confirm execution as administrator.
5. Select the desired language (currently only English is supported).
6. Select the desired installation path. Navigieren Sie zum Speicherort des Installers für das LumiScan Training Framework.
   - It is recommended to use the default path.
7. Decide whether to create a desktop icon.
8. Confirm the installation configuration with **Install**.
9. Complete the installation by clicking **Finish**.

Please contact HD Vision Systems if you have any problems.

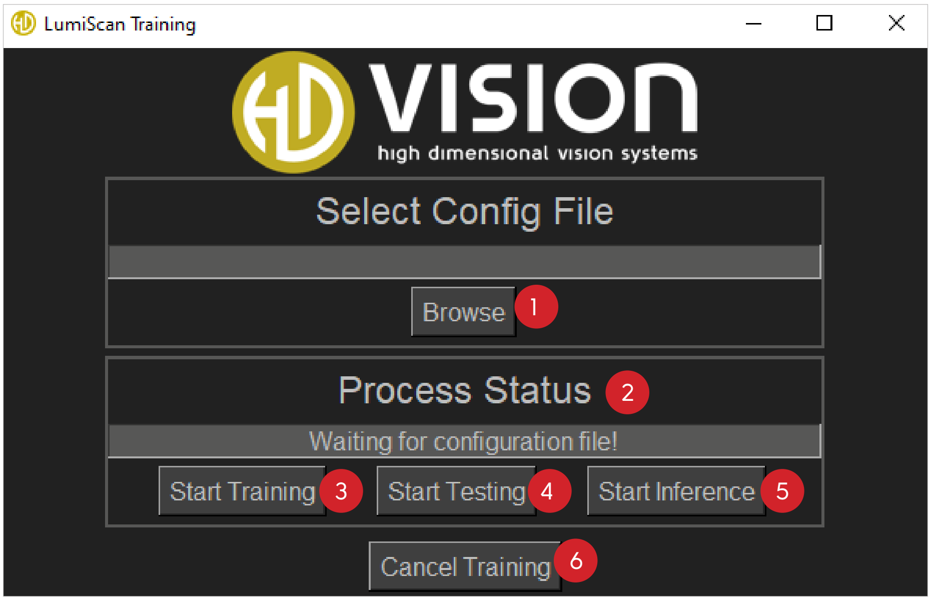
2.5 User Interface

After starting the program, the standard user interface opens.



Waiting for configuration file!

The various elements of the user interface are annotated in Figure below and explained in Table 1.



Waiting for configuration file!

Table 1: LumiScan Training Framework: Described user interface

Nr.	Description
1	**Select Config File/Browse** is used to select the training data.
2	**Process Status** displays the current status of the process. Possible messages: • **Waiting for configuration file!** The program is waiting for training data selection. • **Testing Network!** appears after clicking on **Start Testing**. Indicates that an existing network is being tested with a new test data set. • **Waiting for deployment!** After training, the network must be converted and exported to an onnx file. This message be displayed during the process. • **Finished cleanup** marks the end of the training. • **Time Remaining** displays the remaining process time. • **Latest test scores** is metric for evaluating networks. Should not be used as an absolute value, but as a means of comparing different networks tested with the same data set.
3	**Start Training** starts training the data.
4	**Start Testing** starts the test of the network configuration specified in the Configuration Test Data Set. The results are stored in the Export folder.
5	**Start Inference** Clicking this button opens a pop-up window that lets you select a folder containing images in which the network results will be displayed. The images in this folder will then be extracted from the network, and the results will be saved under predictions in the export directory.
6	**Cancel Training** interrupts an ongoing training, test, or inference process.

For correct and successful training, it is necessary to configure all relevant parameters in the JSON settings beforehand. An example of such a setting is given in the next section.

2.6 JSON Settings

This section describes how to configure the training process for object detection with orientation (rotation and direction). The following options allow the model to be tailored to specific requirements based on the nature and behavior of the objects to be detected.

The tool supports two main use cases:

- **Rotated Object Detection** - Detect the location of objects along with the smallest bounding box that encapsulates each object, allowing for rotation
- **Oriented Object Detection** - Detect the position and orientation of objects, including identifying which side of the bounding box aligns with the top of the object. (Oriented detection identifies not just rotation, but also direction — e.g., top vs bottom.)

An example of application configuration JSON settings is shown below.

```
{

    "trainingData": "/path/to/training_data",
    "validationData": "/path/to/validation_data",
    "exportDirectory": "/path/to/export_directory",

    "classes": [
        "class1",
        "class2"
    ],
    "networkType": "ORTMS",
    "schedule": "Full",
}
```

These are standard elements of the JSON configuration.
All parameters in Table 2 **must be** specified to train a model.
Parameters in Table 3 are optional and allow further customization of training anddeployment.

Table 2: LumiScan Training Framework: JSON-Settings (Required)

Parameter	Description
trainingData	Path to the training dataset.* Specifies the dataset on which the model will be trained.
validationData	Path to the validation dataset. Specifies the dataset used for validation during training. This can be the same as the training dataset, but ideally should be different from the test dataset to ensure proper evaluation. Results of the validation are saved in the network description.
exportDirectory	Path where the trained model and associated files will be saved. The final model, test outputs, and a description file will be exported to this directory.
classes	List of object class names to be detected. This does **not** need to include all classes present in the dataset. Only include labels that the model should detect.
networkType	Specifies the base model architecture and task type, including-network size and whether the task is **Oriented Object Detection** or **Rotated Object Detection**. Supported values: • **ORTMXS** – Oriented Object Detection, extra small (tiny) model • **ORTMS** – Oriented Object Detection, small model (**recommended**) • **ORTMM** – Oriented Object Detection, medium model • **ORTML** – Oriented Object Detection, large model • **RRTMXS** – Rotated Object Detection, extra small (tiny) model • **RRTMS** – Rotated Object Detection, small model (**recommended**) • **RRTMM** – Rotated Object Detection, medium model • **RRTMM** – Rotated Object Detection, medium model • **OOD** – Rotated Object Detection for Hailo (deprecated on otherplatforms)

* NOTE: When specifying the path, pay special attention to the direction of the slashes. All **backslashes (\)** must be replaced with **slashes (/)**.

Parameter	Description
	- **ROD** – Rotated Object Detection for Hailo (deprecated on otherplatforms) - **LumiScanOOD** – Rotated Object Detection (deprecated - **LumiScanROD** – Rotated Object Detection (deprecated)
schedule	Defines the training schedule strategy. Supported values: **Full** – Train a model from scratch using the full dataset. **FullSimple** – Accelerated training for simple datasets, typically used when: - The same object instance is repeated throughout the dataset - Object positions and orientations are consistent - All objects are fully visible (i.e., occlusions are not relevant) **QC** – Full training for complex quality control models, such as: - Detection of abstract defects (e.g., scratches, texture anomalies). - Cases where defects have no definitive or consistent shape **Finetune** – Fine-tune an existing model on new data. - Trains faster than full schedules - Ideal for rapid solutions during production - Requires **finetuneData** and **networkBase** to be set

Table 3: LumiScan Training Framework: JSON-Settings (Optional)

Parameter	**Description**
testData	Path to the test dataset. Specifies the dataset used for testing. This can be the same as the validation dataset, but ideally should be different to ensure proper evaluation. The test dataset is used when selecting **Start Testing**. Results are exported to `test_results.json`.
finetuneData	Path to the fine-tuning dataset. Specifies the dataset used to fine-tune a model. The fine-tune dataset is used together with the original dataset to adapt an existing network to new object appearances (e.g., variations of already known classes). After fine-tuning, a full training schedule should be executed, incorporating the fine-tune data into the complete training set.
networkBase	Pfad zur `.pth-Datei` des Basismodells, das feinabgestimmt werden soll. Bei der Feinabstimmung eines Netzwerks sollte dieser Pfad auf die `.pth-Datei` des vortrainierten Modells verweisen. Die `.pth-Datei` wird immer zusammen mit dem ONNX-Modell und seiner Beschreibungsdatei exportiert.
trainingProfile	Controls batch size, training iterations, and GPU usage. Currently, only single-GPU training is supported. Adjust this if the default batch size causes memory issues. Supported values: • **B8x1** – Batch size of 8 • **B4x1** – Batch size of 4
imageScale	Defines the scaling factor applied to the original image to reduce processing time while maintaining acceptable detection accuracy*. Resolution vs. Object Size ~**75** pixels (object size in image): • Sufficient for most standard use cases • Effective when detecting visually similar objects • Suitable if **occlusions are not a concern**

***** NOTE: Ensure that all relevant features remain visible after scaling, especially:

- Thin or elongated defects (e.g., scratches)
- Small visual cues used for determining rotation, orientation, or classification.

Parameter	Description
	<100 pixels: • May provide slight improvements for certain tasks • Recommended for **occlusion-aware detection**.
imageSizeTraining	Specifies the image dimensions usedduring training. The default size corresponds to the typical downscaled resolution from LumiScan^{X} . Smaller training image sizes reduce GPU memory usage and acceleratetraining but must still be large enough to capture all relevant features. If **imageSizeTraining** is smaller than the scaled-down input image, randomcrops will be used during training. **If occlusions are not relevant:** • Training at a different resolution than inference is acceptable • Enables faster training and lower hardware requirements **If occlusion-aware filtering is used:** • Training and inference should use **full images**, even ifdownscaled • Occlusion relationships must remain intact
imageSizeDeployment	Defines theimage dimensions used during **deployment** (**inference**). This allows setting a specific image size for inference. If possible, set it tothe **full input image size** to ensure best results. Only reduce this ifhardware constraints require it (e.g., embedded devices).*

* NOTE: The model expects images to be scaled according to the imageScale parameter. If full images cannot be used for inference, inferencemust be performed in patches to ensure complete coverage.

3 FAQ

I am getting a FileNotFoundError or does training fail immediately?

Check all paths in your config file (e.g., `trainingData, validationData, exportDirectory`). They must be absolute and accessible from the machine running the training. Relative or broken paths often cause this issue.

Detection is accuracy poor or are objects missing during detection?

Ensure the class names in the classes list exactly match those used in your dataset annotations. Only include the classes you want the model to detect. Class name mismatches lead to poor results or missed detections.

My model is failing to detect obvious objects, or detection scores are very low?

Check your training schedule setting. Use **Full** for most cases. For simple datasets with consistent object appearances, **FullSimple** may also work. An inappropriate schedule can significantly limit model performance.

Detected objects are incorrectly sized or positioned off-center?

- Ensure the imageScale or imageSizeTraining settings do not distort object proportions.
- Review your dataset annotations for consistency in bounding boxes and object placement.
- Follow best practices outlined in the Data Acquisition and Annotation Guidelines.

My model is failing to detect small objects?

Avoid setting the imageScale too low*.

* NOTE: Too much downscaling can cause small objects to disappear from training.

After scaling:

- The smallest important features should be **at least 10 pixels** in size.
- The smallest relevant objects should be **at least 75 pixels**.

Inference is extremely slow or crashing on my hardware-constrained device (e.g., Hailo)?

Use a smaller imageSizeDeployment value suitable for your hardware. Make sure this deployment size matches the scaled input image, not the training cutouts. For large input images, consider patch-based inference to handle processing in smaller sections.

4 General Disclaimer

1. The manufacturer is not liable for damage to life, body or health or damage to property resulting from improper use. Please note that operating and/or connection errors are beyond our control. We cannot accept any liability for damage resulting from this.
2. Any damage caused by unauthorized debugging activities to inspect, analyze or manipulate the software provided by HD Vision Systems is the sole responsibility of the user. Any attempt to use a debugger is strictly prohibited and may result in irreversible consequences such as loss of data, interruption of service, and even legal action. The manufacturers, developers and distributors of the software are not liable for any damage or loss resulting from the user‘s failure to comply with this warning.
3. Claims for damages on the part of the purchaser arising from culpa in contrahendo, breach of secondary contractual obligations and tort are excluded, unless they are based on intent or gross negligence on the part of HD Vision Systems GmbH itself or one of our vicarious agents.
4. Recourse claims in the sense of §12 of the Product Liability Act (Produkthaftungsgesetz) are excluded, unless the party entitled to recourse proves that the defect was caused by HD Vision Systems GmbH and was due to at least gross negligence.
5. If any provision of this Agreement is or becomes invalid or unenforceable, the validity of the remaining provisions shall not be affected. The invalid or unenforceable provision shall be replaced by a valid and enforceable provision whose effects come as close as possible to the economic purpose pursued by the parties with the invalid or unenforceable provision. The foregoing provisions shall apply mutatis mutandis in the event that the contract proves to be incomplete. German courts shall have exclusive jurisdiction over all disputes arising in connection with this disclaimer.
6. For more information about the warranty, please contact the manufacturer of the product.

VISION

high dimensional vision systems

HD Vision Systems GmbH
Palo-Alto-Platz 11
69124 Heidelberg

T +49 6221 6721900
F +49 6221 6721901

info@hdvisionsystems.com
www.hdvisionsystems.com