Astrocyte AI Trainer

User Manual

sensors | cameras | frame grabbers | processors | **software** | vision solutions

Sensors | Cameras | Frame Grabbers | Processors | Software | Vision Solutions
Teledyne DALSA | Machine Vision
Astrocyte
TELEDYNE DALSA
Everywhereyoulook™
Part of the Teledyne Imaging Group

Copyright © Teledyne Digital Imaging Inc. 2018-2023
Teledyne Digital Imaging Inc. All rights reserved.

December 15, 2023

P/N: USER0

Notice

© 2019-2023 Teledyne Digital Imaging Inc.

All information provided in this manual is believed to be accurate and reliable. No responsibility is assumed by Teledyne DALSA for its use. Teledyne DALSA reserves the right to make changes to this information without notice. Reproduction of this manual in whole or in part, by any means, is prohibited without prior permission having been obtained from Teledyne DALSA.

All other trademarks or intellectual property mentioned herein belong to their respective owners.

About Teledyne DALSA

Teledyne DALSA, a business unit of Teledyne Digital Imaging Inc., is an international high-performance semiconductor and electronics company that designs, develops, manufactures, and markets digital imaging products and solutions, in addition to providing wafer foundry services.

Teledyne DALSA offers the widest range of machine vision components in the world. From industry-leading image sensors through powerful and sophisticated cameras, frame grabbers, vision processors and software to easy-to-use vision appliances and custom vision modules.

This document does not contain information whose export/transfer/disclosure is restricted by the Canadian Export Control regulation.

Contents

Astrocyte AI Trainer Overview

Description

The Astrocyte AI Trainer is a deep learning training tool for creating neural network models. It supports various architecture classes such as anomaly detection, classification, object detection and segmentation models. The Astrocyte Trainer employs supervised learning which trains models using images that have been assigned the correct prediction (ground truth).

Convolution neural networks excel at image classification, which consists of categorizing images, given a set of category labels (for example, airplane, automobile), and having the network determine the strongest category present in the image. The goal of supervised training is to have the model learn to correctly predict (identify) the categories in unlabelled images.

The Classification module supports continual learning which is the ability to perform training at inference time (run time). After a classification model has been trained with Astrocyte extra training samples can be provided to the inference engine to add new categories or improve the accuracy of existing categories. Continual learning is useful in practice when you cannot collect all image samples at training time. Continual learning is fast and only requires a very small number of samples.

The Astrocyte Object Detection module also supports Semi-Supervised Object Detection (SSOD) to automatically generate bounding boxes and labels using a set of both labelled and unlabelled images. A previously trained object detection model can also be used to generate bounding boxes and labels on individual images.

Models developed using the Astrocyte AI Trainer can be deployed to analyze and process target images using the Sapera Processing SDK or Sherlock, both Teledyne DALSA image processing software.

Key New Features for Version 1.50

Astrocyte version 1.50 includes the following new features:

- Key Features
  - Support of NVIDIA RTX 4000 family for both training and inference (image analysis using trained Astrocyte model to return results).
  - Support of multiple GPU in automatic training.
  - New service “GPUService” to automatically configure the GPU driver in Max Performance Mode when running inference
  - Significant reduction of training time via mixed-precision training
  - Support of 24-bit RGB images at inference.
- Other Changes
  - General
    - More robustness on load/save of large datasets in a limited disk space
    - More logging information when exporting datasets and models
    - Maintenance button to force Cortex shutdown for database backup
  - Object Detection
    - New hyperparameter for managing the overlap between predictions of different categories.
    - Increased maximum number of detections (instead of 200)
    - Lower default learning rate to improve training performance
    - New filter to identify fully overlapping annotations
    - Better management of image margins when adding annotations
    - New filter button to update image list when labels are changed
    - Persistent annotation colors when saved to dataset
    - Togglling of pseudo vs annotation label when using smart labeling
  - Segmentation
    - Dual display for improved result visualization in the inference tab
    - New algorithm (MANet) introduced in addition to the existing algorithm (DeepLabV3). MANet provides higher accuracy, faster training time and faster inference time.
    - New example dataset “Scratches” (locate thin scratches on brushed metal) is available.

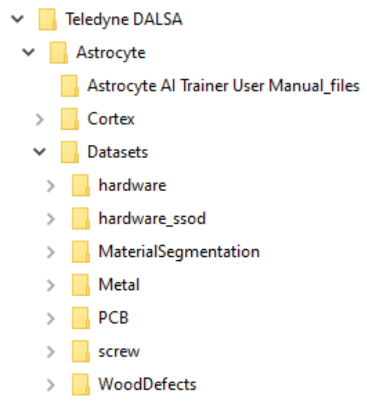
Quick Start Guides to Model Training

For quick start guides to help begin immediately with Astrocyte to experiment with training models refer to the following sections:

- Quick Start to Creating an Anomaly Detection Model
- Quick Start to Creating a Regular Classification Model
- Quick Start to Creating a Continual Learning Classification Model
- Quick Start to Creating an Object Detection Model
- Quick Start to Creating a Segmentation Model

See the Guidelines for Model Creation for more information on how to successfully train models. The Glossary of Terms may also prove useful for users to familiarize themselves with the terms used in deep learning.

The Astrocyte installation includes sample image datasets for each module (if selected during installation), located in the following directory (relative to the selected installation directory):



```
Teledyne DALSA
  Astrocyte
    Astrocyte AI Trainer User Manual_files
    Cortex
    Datasets
      hardware
      hardware_ssod
      MaterialSegmentation
      Metal
      PCB
      screw
      WoodDefects
```

Prerequisites

Supported Operating Systems

- Microsoft Windows 10 or Windows 11 64-bits

Hardware Requirements

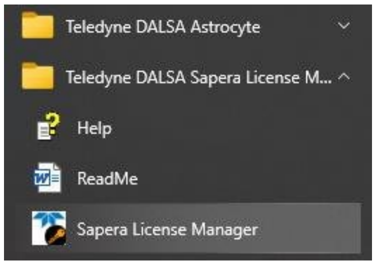
- Intel® Processor supporting EM64T technology
- Minimum 16GB of RAM (32 GB recommended)
- An NVIDIA GPU
    - With minimum 8GB of RAM
    - With graphics driver version 516.31 or later
    - Recommendations:
        - Minimum: RTX 3070/4070 or any other card with 8GB RAM
        - Very good: RTX 3080/4080 or any other card with 12GB RAM
        - Best: RTX 3090/4090 or any other card with 24GB RAM
- Notes: Refer to the Inference Acceleration section below for instructions on maximizing speed from your graphics card
- A valid “Sapera AI SDK” license
- (Optional) Sapera LT 8.71 or higher (to acquire images from Teledyne DALSA hardware)

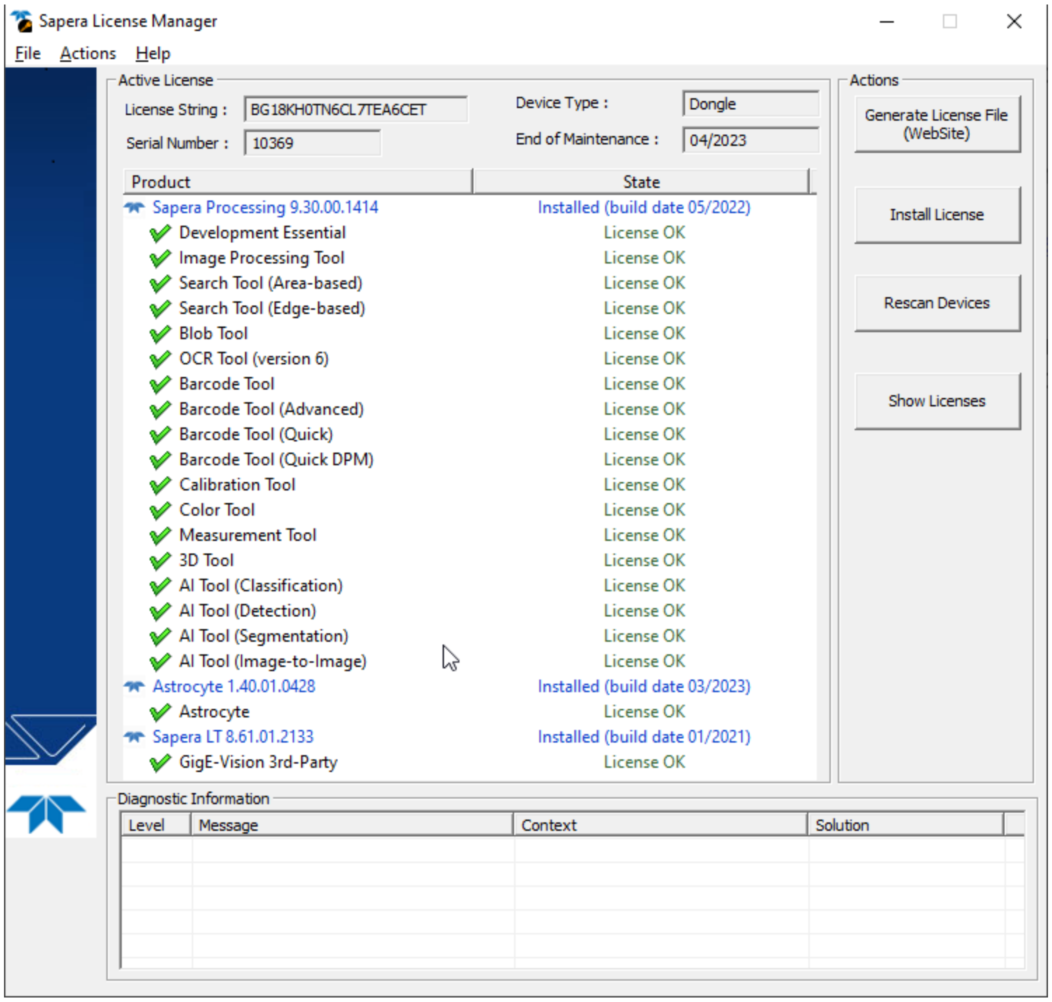
Licensing

The Astrocyte AI Trainer is protected software therefore a license is required. Development licenses use USB dongle-type protection. The Sapera AI license for Astrocyte includes Sapera Processing SDK with all options.

Licenses are activated through the Teledyne DALSA website. To obtain and activate a development or runtime license connect to the http://www.teledynedalsa.com/en/support/software-registration/sapera-software-registration/ webpage.

An evaluation period of 60 days is granted to use the Astrocyte AI Trainer without a license. The license status can be verified using the Sapera License Manager application, which is available through the Windows Start menu.





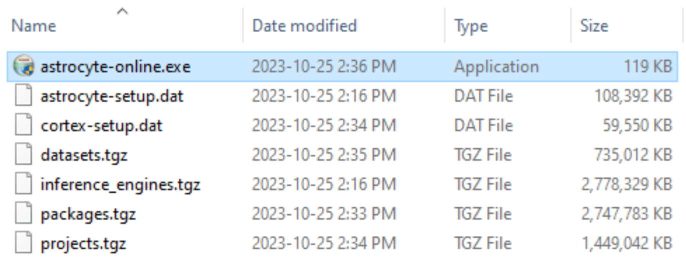
License String : BG18KH0TN6CL7TEA6CET
Serial Number : 10369
Device Type : Dongle
End of Maintenance : 04/2023

Product	State
Sapera Processing 9.30.00.1414	Installed (build date 05/2022)
✓ Development Essential	License OK
✓ Image Processing Tool	License OK
✓ Search Tool (Area-based)	License OK
✓ Search Tool (Edge-based)	License OK
✓ Blob Tool	License OK
✓ OCR Tool (version 6)	License OK
✓ Barcode Tool	License OK
✓ Barcode Tool (Advanced)	License OK
✓ Barcode Tool (Quick)	License OK
✓ Barcode Tool (Quick DPM)	License OK
✓ Calibration Tool	License OK
✓ Color Tool	License OK
✓ Measurement Tool	License OK
✓ 3D Tool	License OK
✓ AI Tool (Classification)	License OK
✓ AI Tool (Detection)	License OK
✓ AI Tool (Segmentation)	License OK
✓ AI Tool (Image-to-Image)	License OK
Astrocyte 1.40.01.0428	Installed (build date 03/2023)
✓ Astrocyte	License OK
Sapera LT 8.61.01.2133	Installed (build date 01/2021)
✓ GigE-Vision 3rd-Party	License OK

Level	Message	Context	Solution

Installation

The Astrocyte installer (astrocyte-online.exe) downloads the application from a Teledyne Imaging server (an internet connection is required); alternatively, if installation files are available in the directory where the installer is located these local files are used and downloading files is not required.

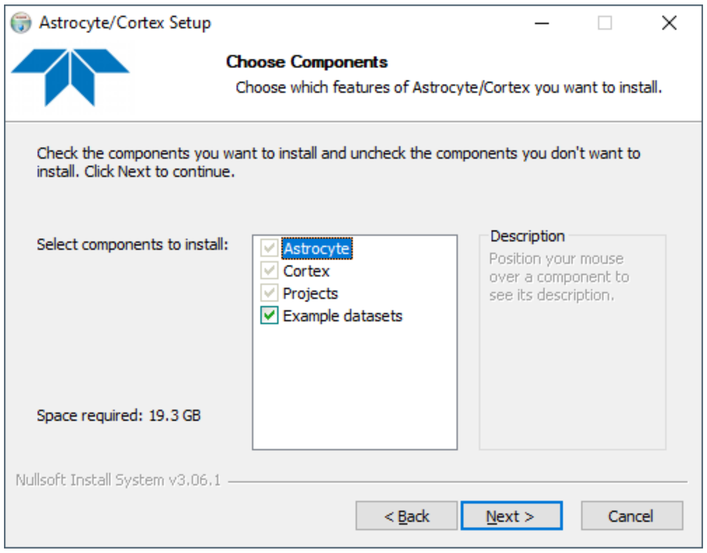


Name	Date modified	Type	Size
astrocyte-online.exe	2023-10-25 2:36 PM	Application	119 KB
astrocyte-setup.dat	2023-10-25 2:16 PM	DAT File	108,392 KB
cortex-setup.dat	2023-10-25 2:34 PM	DAT File	59,550 KB
datasets.tgz	2023-10-25 2:35 PM	TGZ File	735,012 KB
inference_engines.tgz	2023-10-25 2:16 PM	TGZ File	2,778,329 KB
packages.tgz	2023-10-25 2:33 PM	TGZ File	2,747,783 KB
projects.tgz	2023-10-25 2:34 PM	TGZ File	1,449,042 KB

To install Astrocyte follow the installation dialog prompts (computer administrative rights are required).

The Astrocyte installer installs the both the Astrocyte client and Cortex server.

The option to install the example datasets can be disabled if these files are not required.



Astrocyte/Cortex Setup

Choose Components

Choose which features of Astrocyte/Cortex you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Astrocyte
- [✓] Cortex
- [✓] Projects
- [✓] Example datasets

Description

Position your mouse over a component to see its description.

Space required: 19.3 GB

Nullsoft Install System v3.06.1

Installation layout

The Astrocyte directory structure is as follows:

<ASTRODIR>\astrocyte.exe	executable and associated DLLs and dependencies
<ASTRODIR>\Datasets	Sample datasets used in the user manual and tutorials
<ASTRODIR>\Cortex	Deep learning server, CUDA distribution and related dependencies
<ASTRODIR>\Doc	User manual, white papers, application notes

Astrocyte Software Architecture

The Astrocyte Trainer uses a client/server model. The server (Cortex server) hosts the image datasets and executes the actual training. The Astrocyte client must always be connected to a Cortex server.. This architecture allows Astrocyte clients without sufficient GPU computing power to train models on a dedicated Cortex server equipped with adequate hardware resources.

For the current distribution, the Astrocyte client and Cortex server reside on the same machine; future releases will allow deployment of the Cortex server on a separate machine to which the Astrocyte client can connect remotely.

Astrocyte Modules

Astrocyte currently supports training 4 model types using the following modules:

Module	Description
Anomaly Detection	**Short Description** A binary classifier (good/bad) trained on “good” images only. **Typical Usage** Use in defect inspection where simply finding defects is sufficient (no need to classify defects). Useful on imbalanced datasets where many “good” images and a few “bad” images are available. Does not require manual graphical annotations, hence very practical on large datasets. **Detailed Description** Creates a model trained on “normal” or “good” images which can detect instances that deviate from this norm, identifying anomalies or outliers compared to the training set. It is a binary classifier (yes/no, either/or) such that images presented to the model are labelled as “normal” or “anomaly”. When performing inference operations, it can generate heatmaps to assess what image details activated the model neurons; this information can help determine the validity of predictions (that is, if the image features that prompted the determination are what is expected)
Classification	**Short Description** A generic classifier to identify the category of an image. **Typical Usage** Use in applications where multiple category identification is required. For example, it can be used to identify several classes of defects in industrial inspection. It can train in the field using continual learning. **Detailed Description** Creates a model trained on a dataset of labelled images, with each label describing a category (class). Typically, dataset images are sorted using separate folders per category (no manual annotation required). When executing the inference operation, the model returns the category label with the highest calculated probability from among the set of classes in the training dataset. If more than one category appears in the image, the category with the highest match score is returned. In addition to the regular classifier, a Continual Learning model type is available. Using Sapera Processing, existing classes in the model can be updated or new classes added based on new image data in the field without requiring retraining in Astrocyte.

Object Detection	**Short Description** An all-in-one localizer and classifier. Object detection finds the location of an object in an image and classifies it. **Typical Usage** Use in applications where the position of objects is important. For example, it can be used to provide the location and class of defects in industrial inspection. **Detailed Description** Training object detection requires adding bounding boxes around objects, identified with category labels, in the training image dataset. The bounding boxes and associated category labels are referred to as dataset annotations. Object detection can locate multiple objects and classify them in an image. Object detection also includes a Semi-Supervised Object Detection (SSOD) Self-Training option that automatically generates bounding boxes and corresponding labels for datasets that contain a majority of unlabelled images.
Segmentation	**Short Description** A pixel-wise classifier. Segmentation associates each image pixel with a class. Connected pixels of the same category create identifiable regions in the image. **Typical Usage** Use in applications where the size and/or shape of objects are required. For example, it can be used to provide location, category and shape of defects in industrial inspection. **Detailed Description** Creates a semantic segmentation model that predicts which pixels within an image are of a certain class. The dataset identifies the regions in the image that are assigned to a particular class; unlabelled regions are considered background. Typically, at inference time Blob Analysis operators analyze the segmented image to obtain logical objects from which to extract features such as area, elongation, perimeter, etc.

Inference Acceleration

Astrocyte 1.50 (and later) and Sapera Processing 9.40 (and later) integrate optimized versions of inference engines for both NVIDIA graphics cards and x64 CPU's.

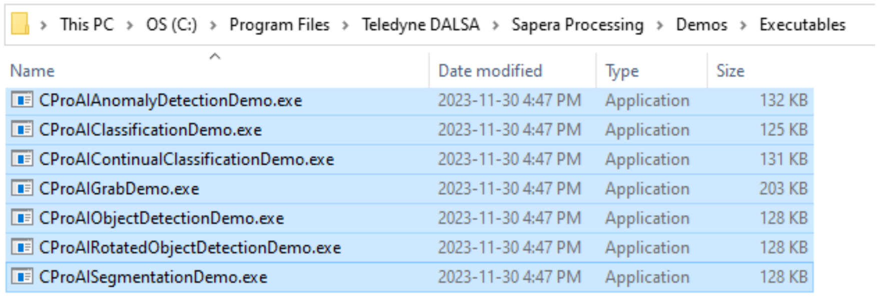
GPU Settings

For optimal performance (and to reproduce benchmark results) the system GPU needs to be configured on "Performance Mode". The default setting for NVIDIA graphics card driver is "Power-Saving Mode" which is ideal for reducing heat generation but does not provide the best performance on inference. Astrocyte installs with a Windows Service ("GPUService") that manages these settings automatically upon loading the AI module. If performance is still perceived to be suboptimal please refer to the Appendix A for manually configuring your GPU in Performance Mode

Benchmarks

The following benchmarks were performed on a range of graphics cards (low to high-end) as well as on a selection of popular CPUs. The datasets and models referred to in this benchmark are available in Astrocyte (datasets) and Sapera Processing (pre-trained models).

Benchmarks can be reproduced by running the AI pre-built demos of Sapera Processing.



This PC > OS (C:) > Program Files > Teledyne DALSA > Sapera Processing > Demos > Executables

Name	Date modified	Type	Size
CProAIAnomalyDetectionDemo.exe	2023-11-30 4:47 PM	Application	132 KB
CProAIClassificationDemo.exe	2023-11-30 4:47 PM	Application	125 KB
CProAIContinualClassificationDemo.exe	2023-11-30 4:47 PM	Application	131 KB
CProAIGrabDemo.exe	2023-11-30 4:47 PM	Application	203 KB
CProAIObjectDetectionDemo.exe	2023-11-30 4:47 PM	Application	128 KB
CProAIRotatedObjectDetectionDemo.exe	2023-11-30 4:47 PM	Application	128 KB
CProAISegmentationDemo.exe	2023-11-30 4:47 PM	Application	128 KB

Benchmark Table

MODEL				INFERENCE TIME (ms)				
Module	Dataset	Image Size	Input Size	RTX 3070	RTX 3090	RTX 4090	Intel	AMD
Anomaly Detection	Metal	2592 x 2048 x 1	1024 x 1024 x 1	21.0	13.0	9.0	275.0	644.9
Classification	Screw	768 x 512 x 1	768 x 512 x 1	3.1	2.2	1.2	31.9	41.3
Object Detection	Hardware	1228 x 920 x 3	512 x 512 x 3	3.8	3.2	3.0	31.7	46.3
Segmentation	Scratches	2048 x 2048 x 1	1024 x 1024 x 1	22.3	16.6	8.9	222.3	391.3

Intel = Intel Core-i9 12900K @ 3.2GHz
AMD = EPYC 7272 12-Core @ 2.9GHz

Module: *AI model type*

Dataset: series of images the model was train on.

Image Size: size of original image

Input Size: size of image after resized (just before entering the neural network)

Inference Time: total execution time including resizing and inference in milliseconds

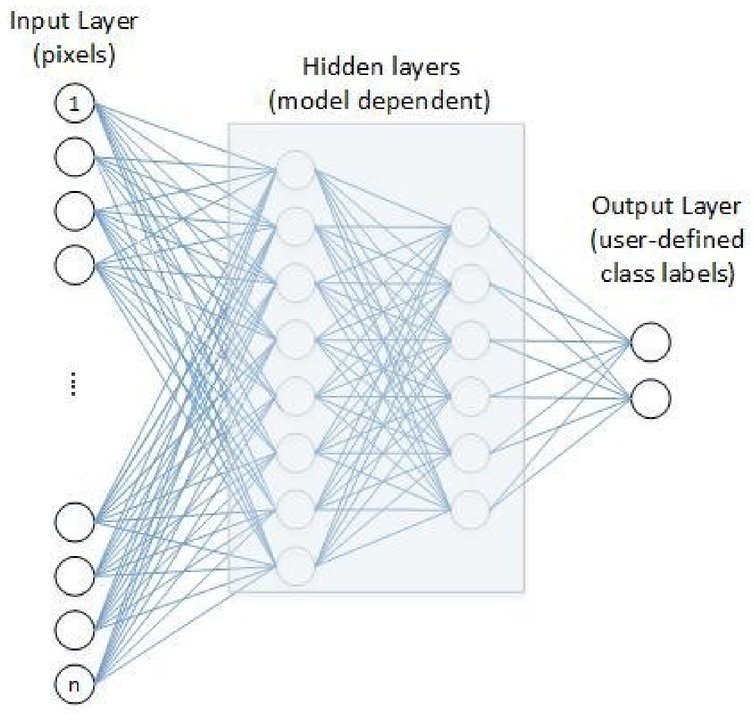
Neural Network Model

The neural network model arranges artificial neurons into layers. Neurons in any layer get input from the neurons in the layers below them; each neuron is connected to every neuron in the next layer. And, in turn, their output feeds into the neurons in the layer above. Typically, the lowest layer represents the input to a neural network; for Astrocyte this is an image.

For example, if every pixel in an input image is considered a neuron, the input layer of a 28x28 pixel image has 784 neurons. The size of the output layer is determined by the number of labels or categories that represent possible predictions. The model input layer can be non-square and of any size (though limitations exist for the object detection module).

For example, in a classification model for numbers, the output layer has 10 labels or categories (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) that the model assigns to an input based on what it predicts with the highest confidence represents the digit in the image.

The initial layers learn the low-level concepts such as edges and colors and the later, hidden level layers learn higher level concepts such as edges and contours. After computing the values of each layer, the output values are read out from the topmost layer.



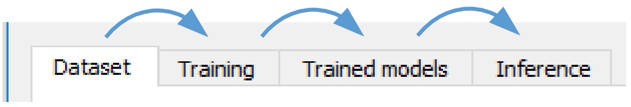
The behavior of the network is determined by the setting of weights and biases for each neuron. The weights and biases are calculated based on the expected output. The final output is then determined by the total of those weightings. Model training is the process of searching for optimal settings of these weights and biases, known as model parameters. Hyper parameters are higher level settings that control how the weights and biases are calculated during training.

During model training, each image in the dataset is presented to the model and the individual weights of its neurons are adjusted to improve its predictive performance. A complete pass of all images in the dataset is referred to as an epoch.

The number and dimensions of the layers in the neural network determine the size of the resulting model. Astrocyte provides several different popular neural network architectures. Depending on the image dataset and application requirements, different architectures may provide the optimal solution.

Astrocyte Interface Overview

The Astrocyte application is organized in tabs. The workflow usually proceeds from left to right through the Dataset, Training tabs to testing trained models on unlabelled images in the Inference tab.

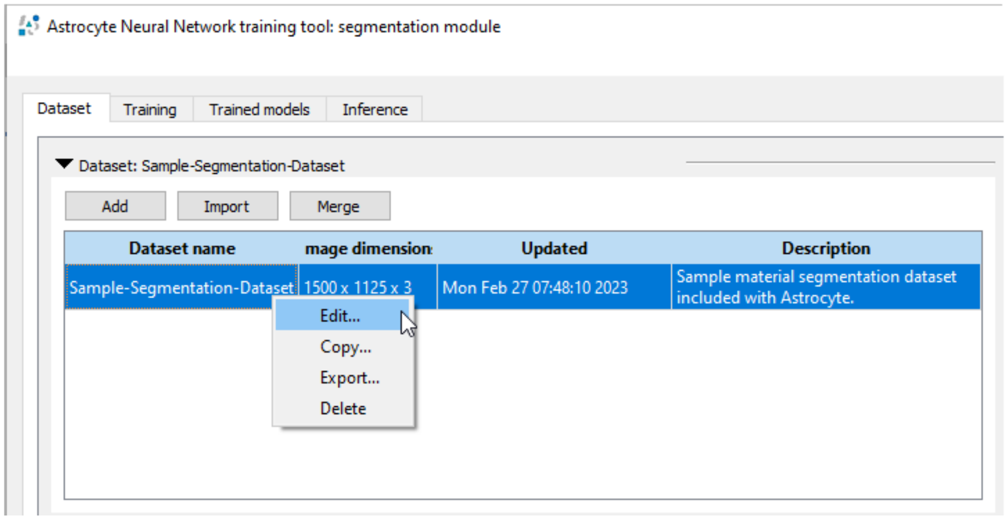


When a model is successfully trained it can be exported to use with Sapera Processing to perform inference on images within a user application.

Depending on the mode selected (anomaly detection, classification, object detection or segmentation), tabs may contain different parameters and configuration options.

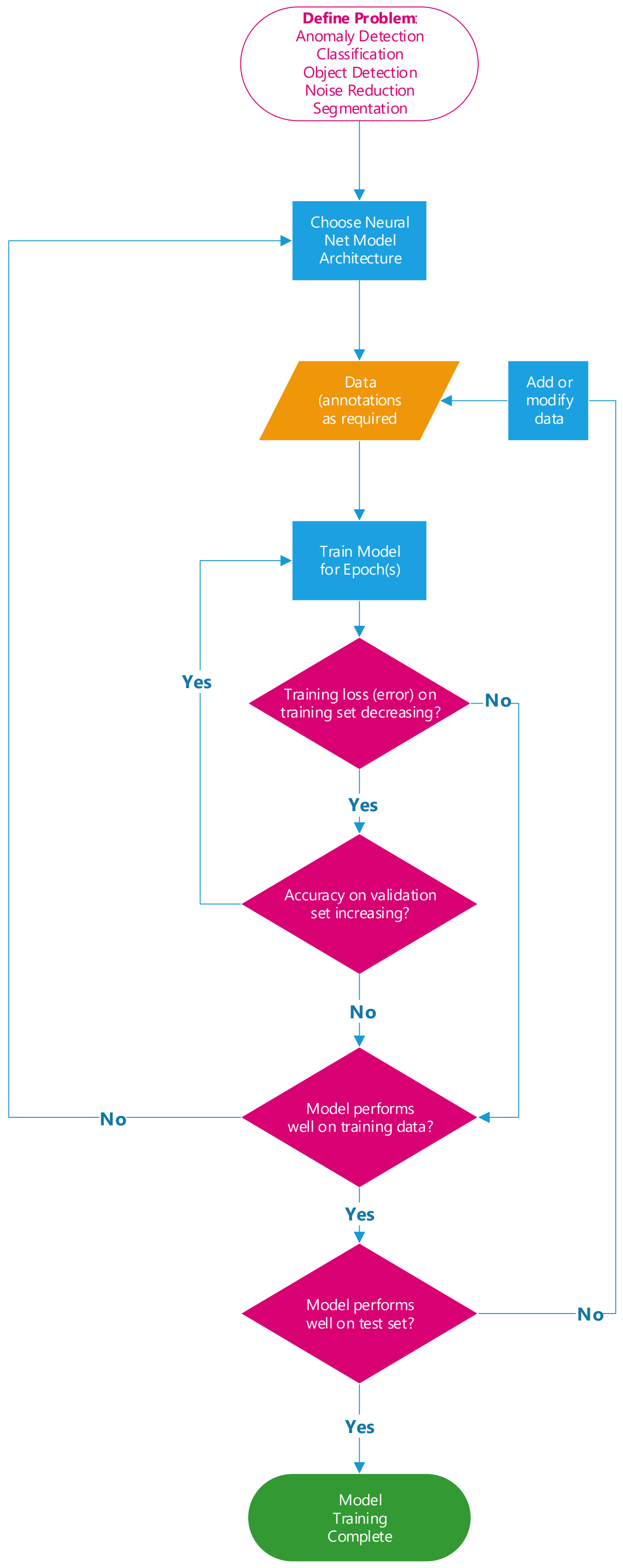
- **Dataset**: allows adding, editing, copying and deleting image datasets. It also provides access to the labels and bounding boxes associated with images in the dataset. Trained models can also be tested against datasets with various result metrics calculated.
- **Training**: configures training hyper parameters, such as the model architecture, and executes the model training on the specified image dataset.
- **Trained models**: lists available trained models, including details of relevant training parameter settings.
- **Inference**: allows testing of trained models against unlabelled images to confirm model performance.

In general, most functionality is available using right-click pop-up menus within the tab contents.


Astrocyte Neural Network training tool: segmentation module
Dataset
Training
Trained models
Inference
Dataset: Sample-Segmentation-Dataset
Add
Import
Merge
Edit...
Copy...
Export...
Delete

Dataset name	mage dimension	Updated	Description
Sample-Segmentation-Dataset	1500 x 1125 x 3	Mon Feb 27 07:48:10 2023	Sample material segmentation dataset included with Astrocyte.

Workflow for Training a Model


Define Problem:
Anomaly Detection
Classification
Object Detection
Noise Reduction
Segmentation
Choose Neural Net Model Architecture
Data (annotations as required
Add or modify data
Train Model for Epoch(s)
Training loss (error) on training set decreasing?
Yes
No
Yes
Accuracy on validation set increasing?
No
Model performs well on training data?
No
Yes
Model performs well on test set?
No
Yes
Model Training Complete

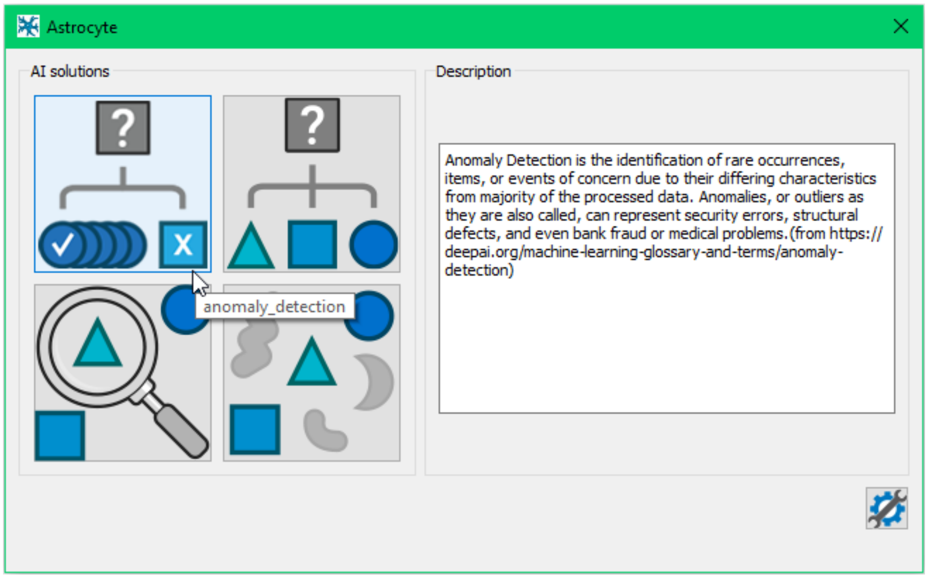
Anomaly Detection Model Training

Anomaly detection models are binary classifiers that identify images as either “Normal” or an “Anomaly”. This category of model can be used on very unbalanced datasets where using a classification model yields bad results. The model is trained only on the good or “normal” images; images labelled as “anomaly” are only used for model testing.

Quick Start to Creating an Anomaly Detection Model

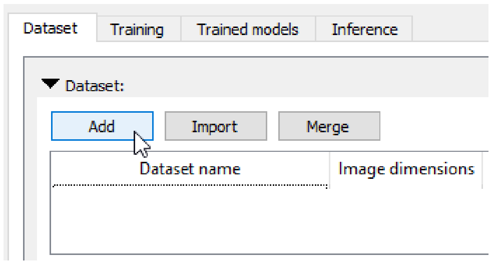
The following procedure demonstrates how to create an anomaly detection model using default values. This example uses the metal dataset included with the Astrocyte installation, available in the *<installation directory>\Teledyne DALSA\Astrocyte\Datasets* directory.

Launch the Astrocyte application and select the Anomaly Detection module from the startup screen:

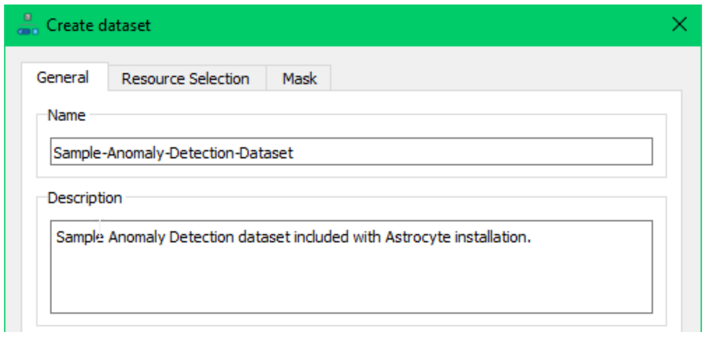


Anomaly Detection is the identification of rare occurrences, items, or events of concern due to their differing characteristics from majority of the processed data. Anomalies, or outliers as they are also called, can represent security errors, structural defects, and even bank fraud or medical problems.(from https://deepai.org/machine-learning-glossary-and-terms/anomaly-detection)

1. In the Dataset tab, click **Add**.



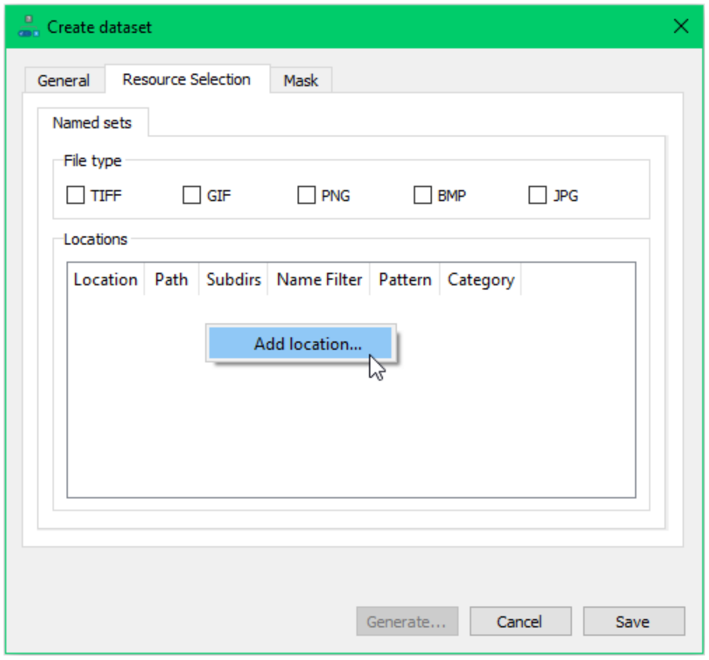
In the General tab, enter the dataset name and description.



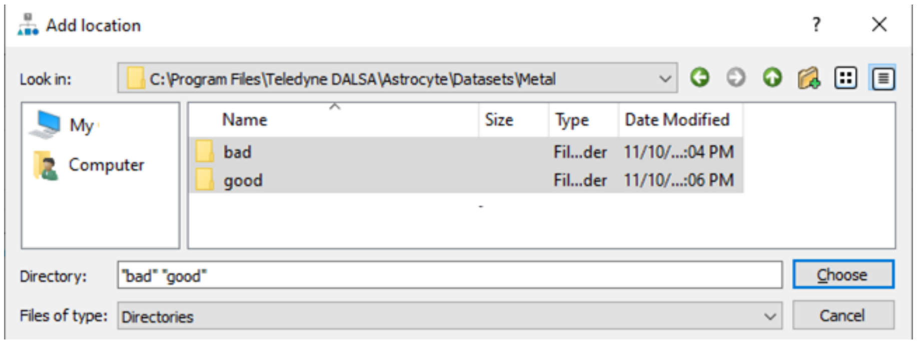
Name: Sample-Anomaly-Detection-Dataset

Description: Sample Anomaly Detection dataset included with Astrocyte installation.

In the Resource Selection tab, right-click in the Locations field and select **Add location...**.



2. In the Add location dialog, navigate to the folder containing the dataset of training images; select both normal (good) and anomaly (bad) directories and click **OK**. (The sample Anomaly Detection dataset included with the Astrocyte installation is located in the <*Install*>*\Astrocyte\Datasets\Metal* directory.)



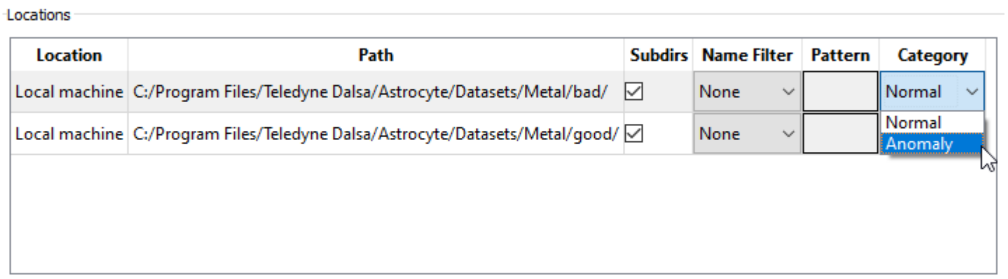
Look in: C:\Program Files\Teledyne DALSA\Astrocyte\Datasets\Metal

Name	Size	Type	Date Modified
bad		Fil...der	11/10/...:04 PM
good		Fil...der	11/10/...:06 PM

Directory: "bad" "good"

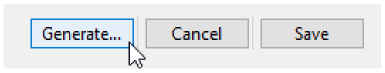
Files of type: Directories

For each directory, assign the category label using the drop-down list: Normal or Anomaly.


Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/Metal/bad/	☑	None		Normal
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/Metal/good/	☑	None		Normal / Anomaly

3. Click **Generate…** to prepare the dataset for the Cortex server.



The Dataset generation configuration dialog allows you to select the image size of the dataset; the original image size can be used, or the image dimensions reduced as required.

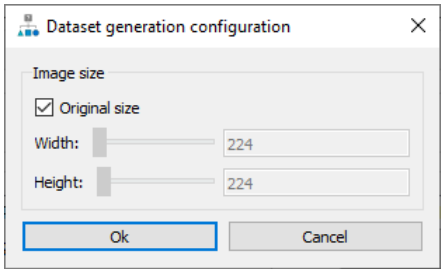

Dataset generation configuration

Image size

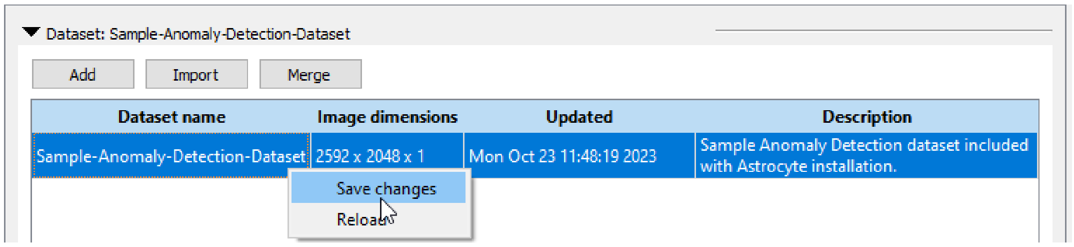
☑ Original size

Width: 224

Height: 224

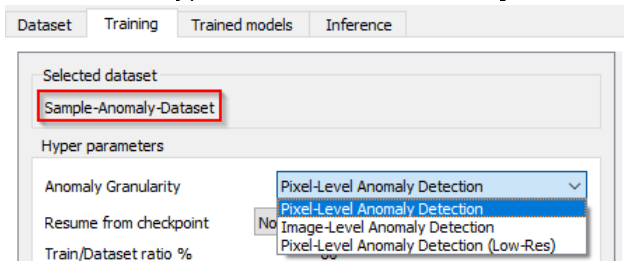
4. Click **Save** to add the dataset to the Cortex server. If all images have the same dimensions, they are automatically resized to the specified image size on the Cortex server (the original dataset is not modified); otherwise, correct images using the Image Correction dialog.

Verify the dataset images and labels; make any required changes. If the dataset is modified, right-click on the dataset entry and click **Save changes** to update and save the dataset on the Cortex server.


Dataset: Sample-Anomaly-Detection-Dataset

Dataset name	Image dimensions	Updated	Description
Sample-Anomaly-Detection-Dataset	2592 x 2048 x 1	Mon Oct 23 11:48:19 2023	Sample Anomaly Detection dataset included with Astrocyte installation.

5. In the Training tab, verify the selected dataset (the selection is made in the Dataset tab), choose the Abstract type **Pixel-Level Anomaly Detection.**


Dataset
Training
Trained models
Inference

Selected dataset

Sample-Anomaly-Dataset

Hyper parameters

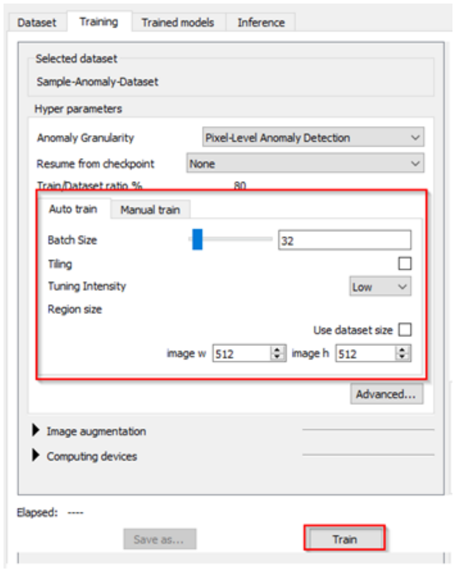
Anomaly Granularity: Pixel-Level Anomaly Detection

Pixel-Level Anomaly Detection
Image-Level Anomaly Detection
Pixel-Level Anomaly Detection (Low-Res)

Resume from checkpoint: No

Train/Dataset ratio %

6. Enable the Auto train tab by selecting it (giving it focus) and click **Train** (the **Train and save…** option automatically saves the model when training finishes).

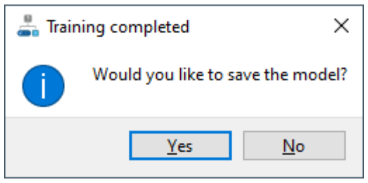


Selected dataset
Sample-Anomaly-Dataset
Hyper parameters
Anomaly Granularity: Pixel-Level Anomaly Detection
Resume from checkpoint: None
Train/Dataset ratio %: 80
Batch Size: 32
Tiling: ☐
Tuning Intensity: Low
Region size
Use dataset size ☐
image w 512 image h 512
Elapsed: ----

The model loss and metric progress graphs for each trial are displayed.

Training statistics such as current training stage and trial step are displayed in the status bar (the estimated completion time is not calculated when using auto training since early stopping is enabled).

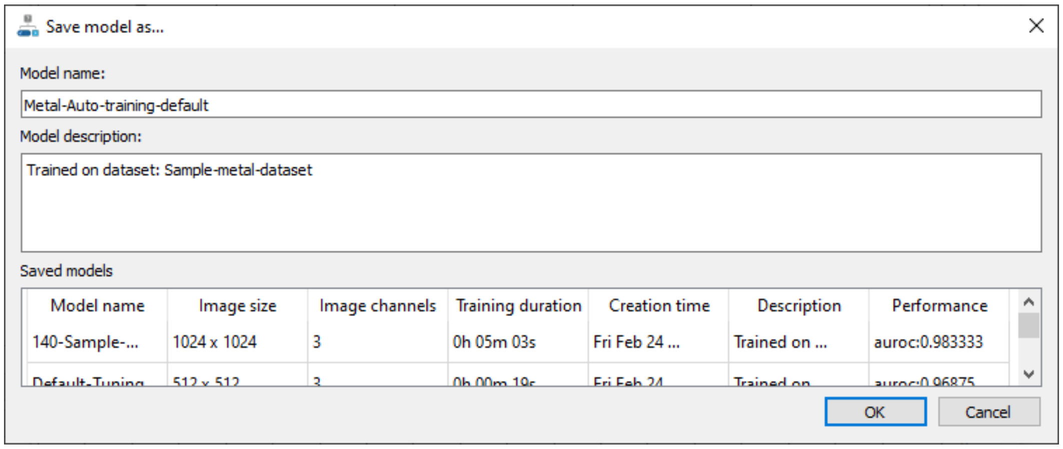
When training is complete, a prompt is displayed to save the model; click **Yes.**



Training completed

Would you like to save the model?

Enter the model name and description and click **OK**.



Save model as...

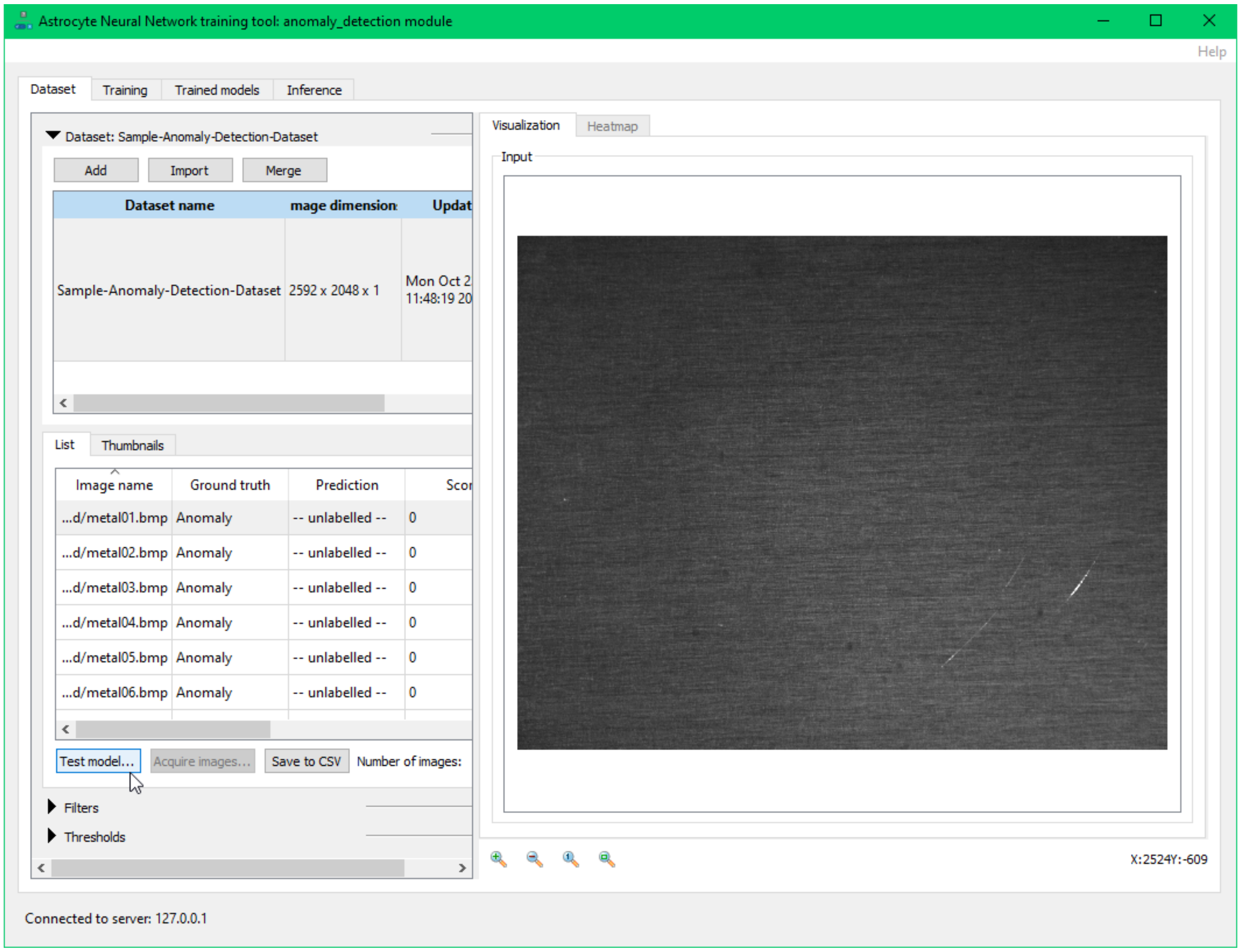
Model name:
Metal-Auto-training-default

Model description:
Trained on dataset: Sample-metal-dataset

Saved models

Model name	Image size	Image channels	Training duration	Creation time	Description	Performance
140-Sample-...	1024 x 1024	3	0h 05m 03s	Fri Feb 24 ...	Trained on ...	auroc:0.983333
Default-Tuning	512 x 512	3	0h 00m 19s	Fri Feb 24	Trained on	auroc:0.96875

7. In the Dataset tab, select the dataset and click **Test model...**.



Astrocyte Neural Network training tool: anomaly_detection module

Help

Dataset | Training | Trained models | Inference

Dataset: Sample-Anomaly-Detection-Dataset

Add | Import | Merge

Dataset name	mage dimension	Updat
Sample-Anomaly-Detection-Dataset	2592 x 2048 x 1	Mon Oct 2 11:48:19 20

List | Thumbnails

Image name	Ground truth	Prediction	Scor
...d/metal01.bmp	Anomaly	-- unlabelled --	0
...d/metal02.bmp	Anomaly	-- unlabelled --	0
...d/metal03.bmp	Anomaly	-- unlabelled --	0
...d/metal04.bmp	Anomaly	-- unlabelled --	0
...d/metal05.bmp	Anomaly	-- unlabelled --	0
...d/metal06.bmp	Anomaly	-- unlabelled --	0

Test model... | Acquire images... | Save to CSV | Number of images:

Filters

Thresholds

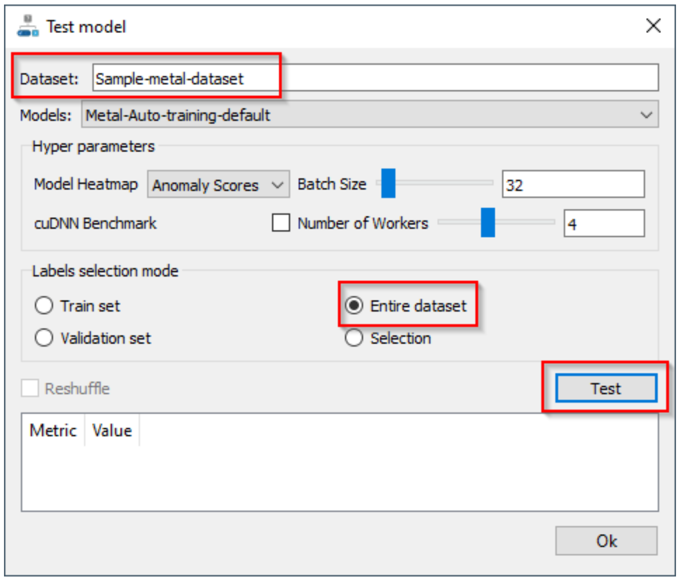
Visualization | Heatmap

Input

X:2524Y:-609

Connected to server: 127.0.0.1

8. In the Test model dialog, select the model in the drop-down list, select **Entire dataset** (to ensure both normal and anomaly images are included) and click **Test**.



Test model

Dataset: Sample-metal-dataset

Models: Metal-Auto-training-default

Hyper parameters

Model Heatmap: Anomaly Scores

Batch Size: 32

cuDNN Benchmark: [ ]

Number of Workers: 4

Labels selection mode

( ) Train set

( ) Validation set

(●) Entire dataset

( ) Selection

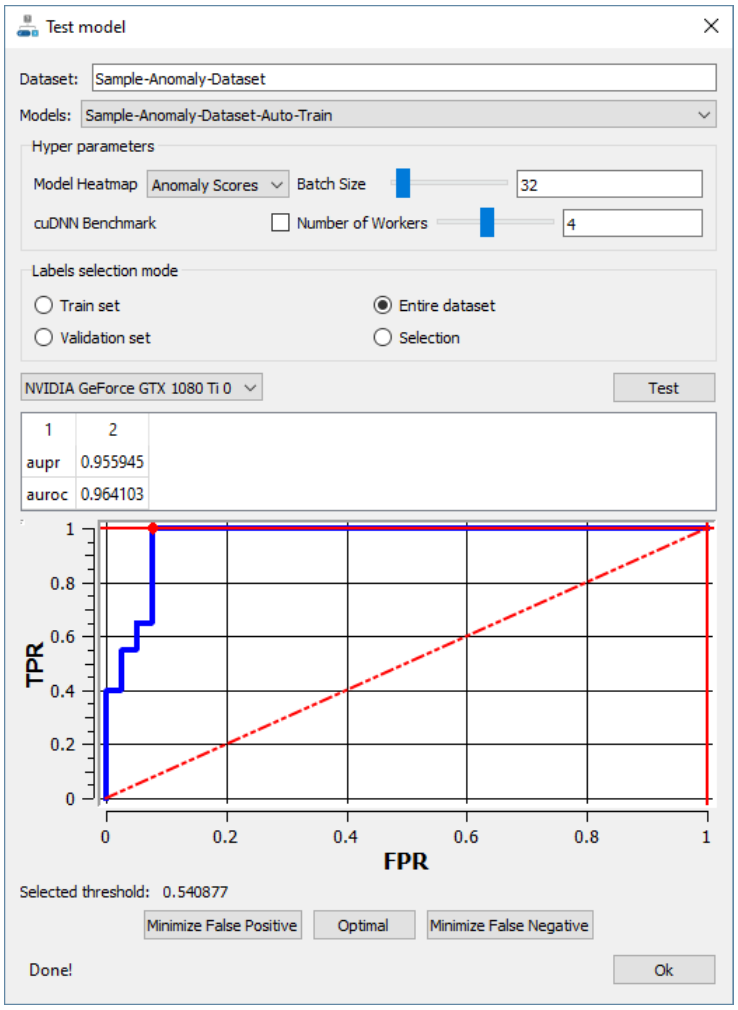
[ ] Reshuffle

Test

Metric	Value

Ok

When testing is completed AUPR (Area Under Precision-Recall curve) and AUROC (Area Under Receiver Operating Characteristic curve) are displayed in the Test model dialog.


Test model

Dataset: Sample-Anomaly-Dataset

Models: Sample-Anomaly-Dataset-Auto-Train

Hyper parameters

Model Heatmap: Anomaly Scores
Batch Size: 32
cuDNN Benchmark: ☐
Number of Workers: 4

Labels selection mode

○ Train set
[✓ ● Entire dataset]
○ Validation set
○ Selection

NVIDIA GeForce GTX 1080 Ti 0

1	2
aupr	0.955945
auroc	0.964103

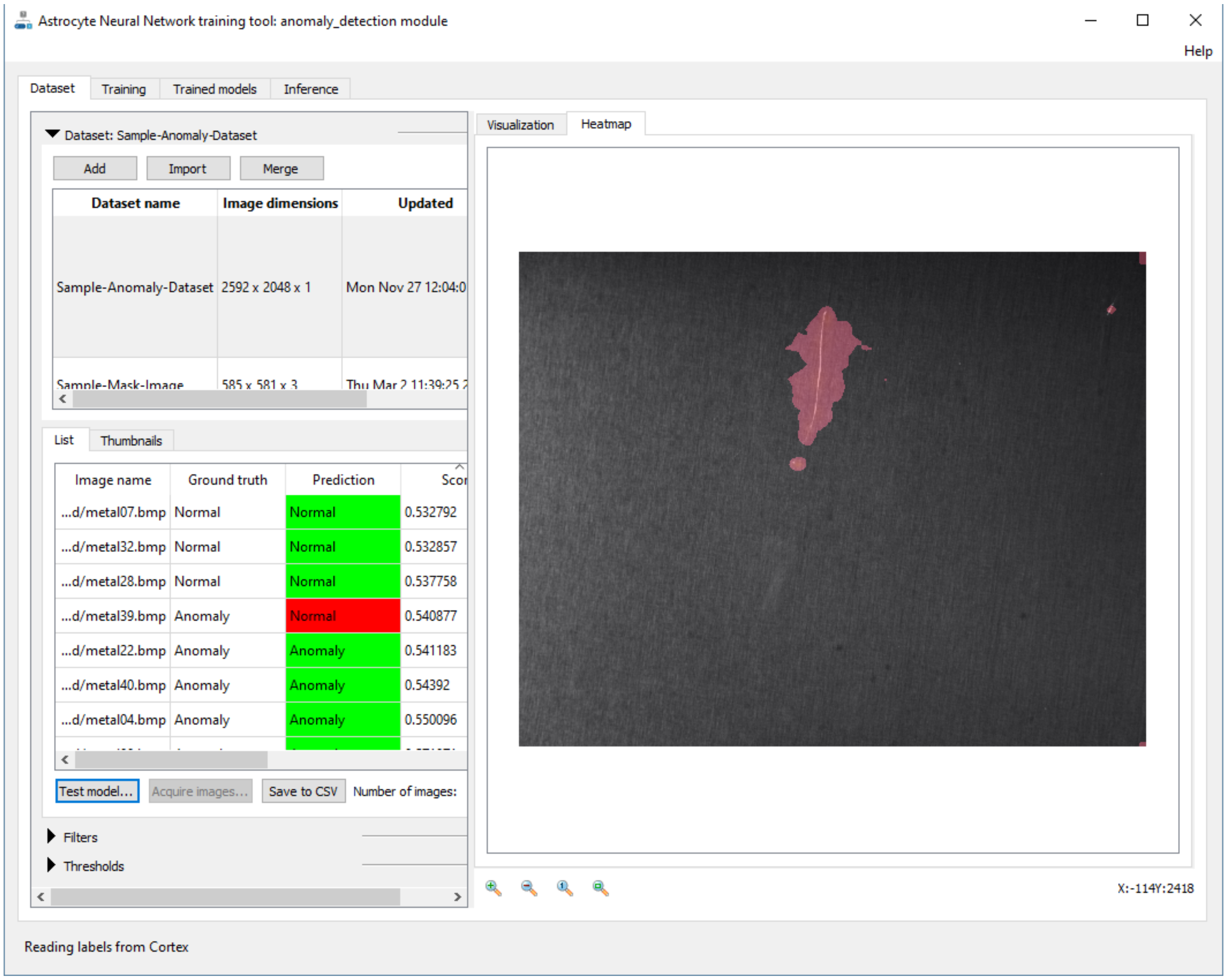
Selected threshold: 0.540877

Done!

Clicking on the TPR/FPR (True Positive Rate/ False Positive Rate) graph selects the required threshold for anomaly detection. This threshold is automatically passed as the threshold in the Inference tab.

Correct predictions are highlighted in green and erroneous predictions in red. If heatmap generation was enabled, the Heatmap tab contains the corresponding image heatmap.

For a detailed explanation of the result metrics, see the section Anomaly Detection Model Test Results.



Astrocyte Neural Network training tool: anomaly_detection module

Dataset name	Image dimensions	Updated
Sample-Anomaly-Dataset	2592 x 2048 x 1	Mon Nov 27 12:04:0
Sample-Mask-Image	585 x 581 x 3	Thu Mar 2 11:39:25 2

Image name	Ground truth	Prediction	Scor
...d/metal07.bmp	Normal	🟩 Normal	0.532792
...d/metal32.bmp	Normal	🟩 Normal	0.532857
...d/metal28.bmp	Normal	🟩 Normal	0.537758
...d/metal39.bmp	Anomaly	🟥 Normal	0.540877
...d/metal22.bmp	Anomaly	🟩 Anomaly	0.541183
...d/metal40.bmp	Anomaly	🟩 Anomaly	0.54392
...d/metal04.bmp	Anomaly	🟩 Anomaly	0.550096

Reading labels from Cortex

Classification Model Training

Classification is supervised learning where a training set of correctly identified objects is available. Astrocyte classification trains a model based on libraries of labelled images. The image label identifies the object in the image.

The resulting model can recognize multiple labels and returns the label it identifies with the highest confidence in the image. That is, if the image contains multiple types of objects for which the model has been trained, the category label of the object with highest correlation score is returned.

The probability (correlation score) always adds up to 1, regardless of the number of categories. For example, if a model contains 3 categories (A, B and C), if an object is considered 90% to be A, then B and C will account for the remaining 10% probability.

⚠	**Note**: A classification model evaluates the entire image to train; it is sensitive to the relative size of the target object in the training dataset images and will not be able to classify smaller or larger target objects which are not part of the training image dataset. However, while object detection models are relatively insensitive to object size, tiny objects in large images will be difficult to detect.

Continual Learning

The Classification module allows for the creation of models using a Continual Learning mode. Models created using this abstract type can be updated in the field as new image data is acquired. New category labels can be added to the model or new images can be added to existing categories. The model trains on these new images and updates its neural network accordingly.

The benefit of continuous learning is that the original dataset on which the model was trained does not have to be updated and the model retrained from scratch as new data is acquired but that the existing model can be refined by applying new training data. The inclusion of continuous learning allows imaging applications to adapt to a wide degree of variation and improve the system with newly acquired data as necessary.

When adding categories to a continual learning model, the neural network backbone for feature extraction is maintained and only the inference layer is updated. Each model is unique and its neural network backbone can provide excellent performance with many additional categories and new image data. Often, if the new image data does not vary considerably, only a few images are required to learn to achieve acceptable results.

While the number of new category labels that can be added to a model is not limited (though a maximum number can be specified), if new image data differs significantly from the original training image dataset the neural network backbone may be inadequate; performance should be monitored as categories are added, and if necessary, the model retrained with the new image data to improve performance.

For example, a continual learning model can provide a feedback loop for an AI inference system where a trained technician can accept or reject the output of the AI system. The expert feedback can then be used as new data to the AI system and used for future refinement. This can conceivably serve at least two purposes. The first is to ensure adequate performance and recency in the capability of the AI system. The second may provide the user of the AI system with the ability to “tune” a particular AI system to their liking and thus maximize performance and operator confidence for the image processing system. For example, as new items or defects are identified the existing model can be trained on this new data and updated in the field.

Updates in the field are performed using the AI classes provided by the Sapera Processing API; refer to the Sapera Processing documentation for more information on integrating continual learning into applications.

Refer to the Quick Start to Creating a Continual Learning Classification Model for an example of how to train and update a continual learning type classification model.

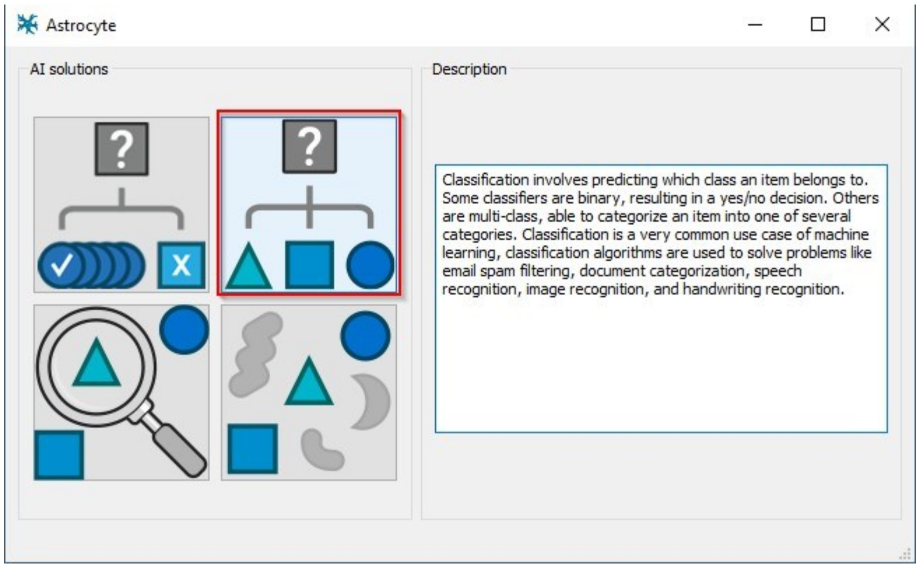
Note: The Astrocyte Continual Learning model type is resilient to “catastrophic forgetting”, where the prior knowledge is replaced or improperly distorted by new knowledge, due to maintaining the backbone layers of the neural network and only modifying the inference layer; however, catastrophic forgetting can occur if numerous images that differ considerably from those in the original training dataset are added to an existing class.

To avoid negatively impacting the performance of an existing class, when images differ significantly from those in the original training dataset, it is recommended to use a new category label.

Quick Start to Creating a Regular Classification Model

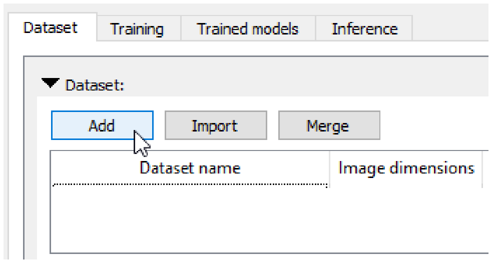
The following procedure demonstrates how to create a regular classification model using default values. This example uses the Metal dataset included with the Astrocyte installation, available in the *<installation directory>\Teledyne DALSA\Astrocyte\Datasets* directory.

Launch the Astrocyte application and select the Classification module from the startup screen:

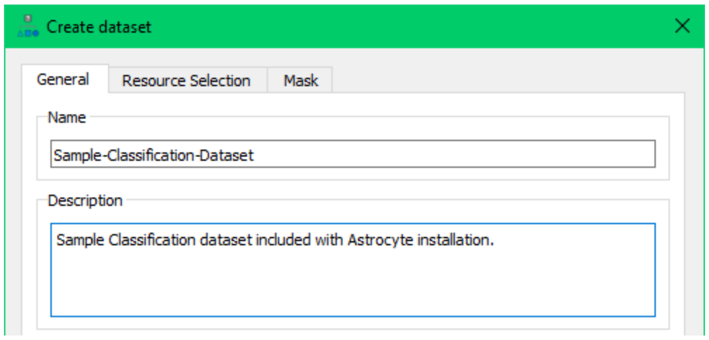


Classification involves predicting which class an item belongs to. Some classifiers are binary, resulting in a yes/no decision. Others are multi-class, able to categorize an item into one of several categories. Classification is a very common use case of machine learning, classification algorithms are used to solve problems like email spam filtering, document categorization, speech recognition, image recognition, and handwriting recognition.

1. In the Dataset tab, click **Add**.



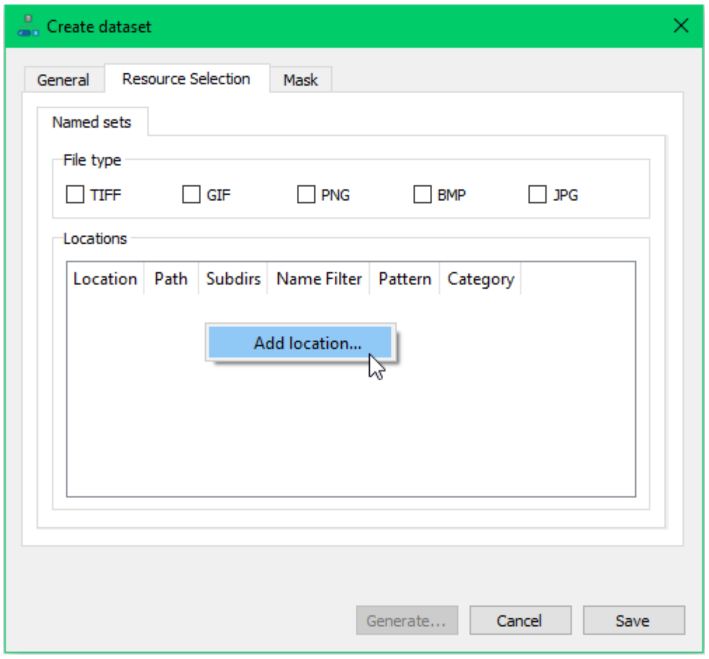
In the General tab, enter the dataset name and description.



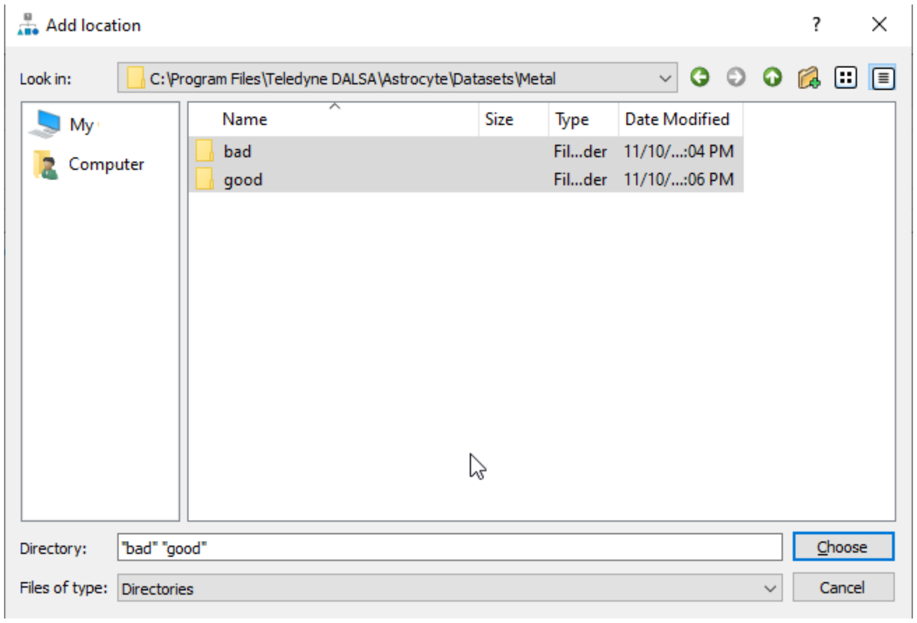
Name: Sample-Classification-Dataset

Description: Sample Classification dataset included with Astrocyte installation.

In the Resource Selection tab, right-click in the Locations field and select **Add location...**.



2. In the Add location dialog, navigate to the folder containing the dataset of training images; select all the directories and click **OK**.



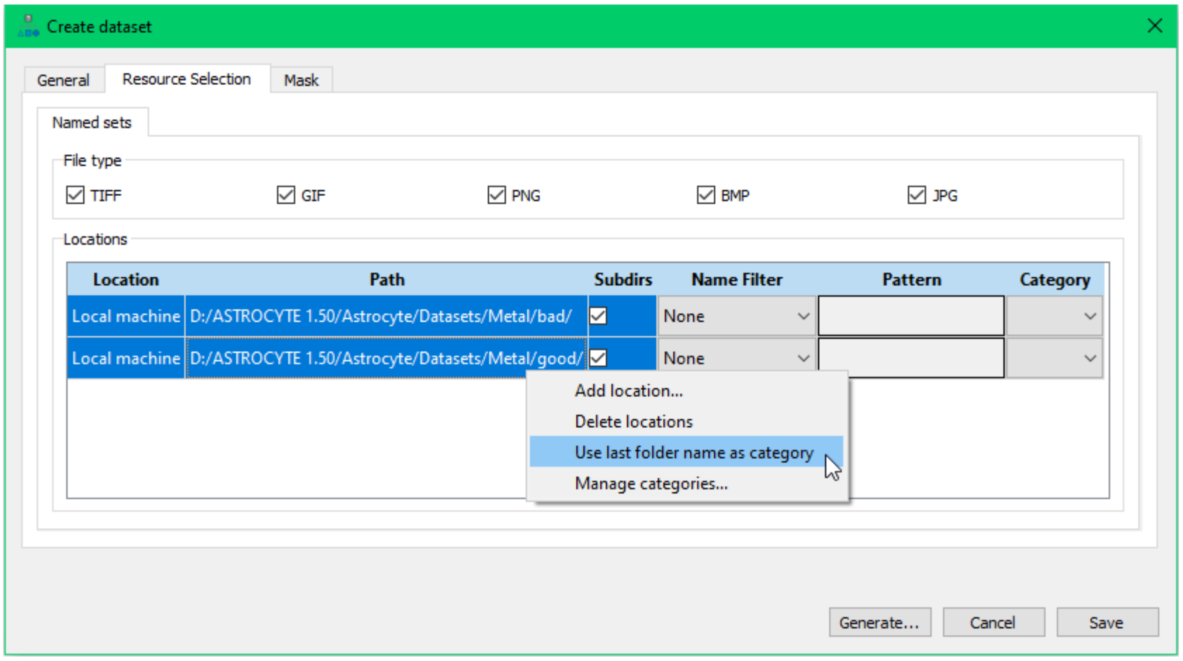
Look in: C:\Program Files\Teledyne DALSA\Astrocyte\Datasets\Metal

Name	Size	Type	Date Modified
bad		Fil...der	11/10/...:04 PM
good		Fil...der	11/10/...:06 PM

Directory: "bad" "good"

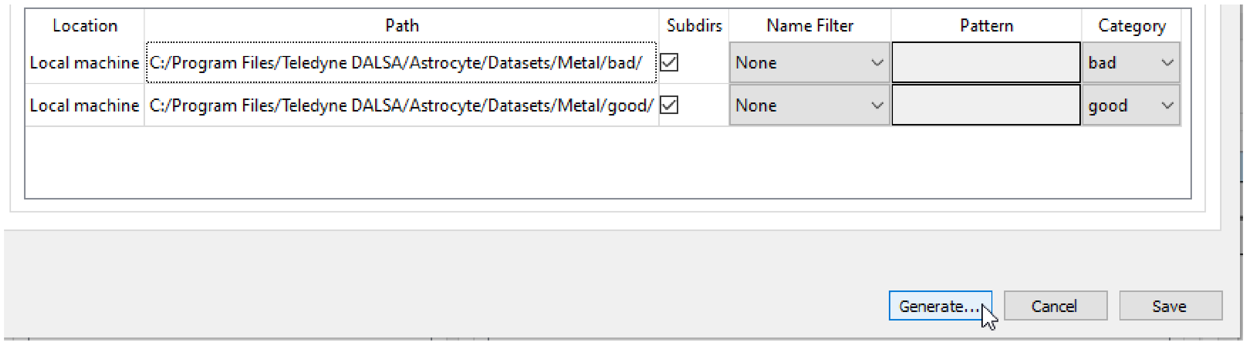
Files of type: Directories

In the Create dataset dialog Resource Selection tab, select all directories in the Location panel, right-click and select **Use last folder name as category**.



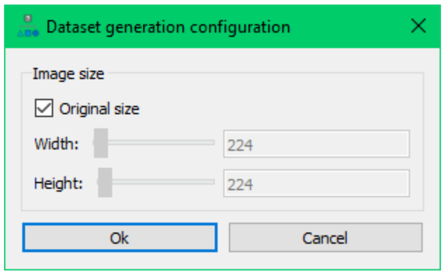
Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	D:/ASTROCYTE 1.50/Astrocyte/Datasets/Metal/bad/	☑	None		
Local machine	D:/ASTROCYTE 1.50/Astrocyte/Datasets/Metal/good/	☑	None		

3. Verify the category labels and click **Generate** to prepare the dataset for the Cortex server.



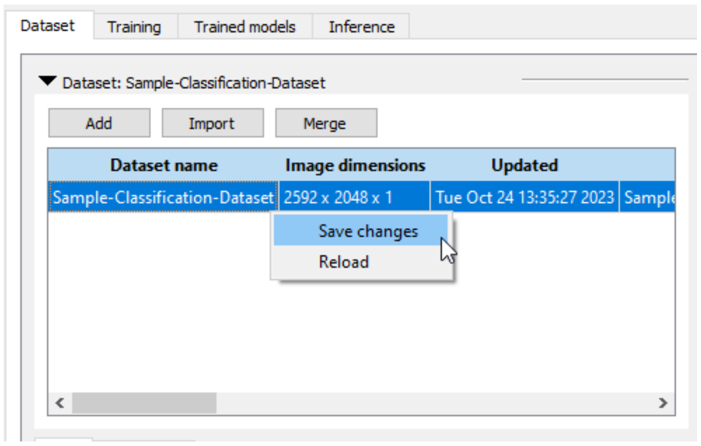
Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/Metal/bad/	☑	None		bad
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/Metal/good/	☑	None		good

The Dataset generation configuration dialog allows you to select the image size of the dataset; the original image size can be used or the image dimensions reduced as required.



4. Click **Save** to add the dataset to the Cortex server. If all images have the same dimensions, they are automatically resized to the specified image size on the Cortex server (the original dataset is not modified); otherwise, correct images using the Image Correction dialog.

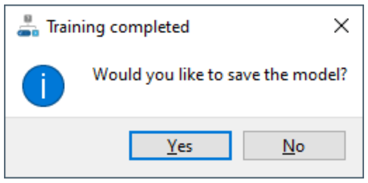
Verify the dataset images and labels and make any required changes. If the dataset is modified, right-click on the dataset entry and click **Save changes** to update and save the dataset on the Cortex server.



Dataset name	Image dimensions	Updated	
Sample-Classification-Dataset	2592 x 2048 x 1	Tue Oct 24 13:35:27 2023	Sample

5. In the Training tab, verify the selected dataset (the selection is made in the Dataset tab), choose the Abstract type Regular Classifier, enable the Auto train tab and click **Train** (the **Train and save…**option saves the model automatically when training finishes). Training starts and the training loss, accuracy graphs and progress bar are updated.

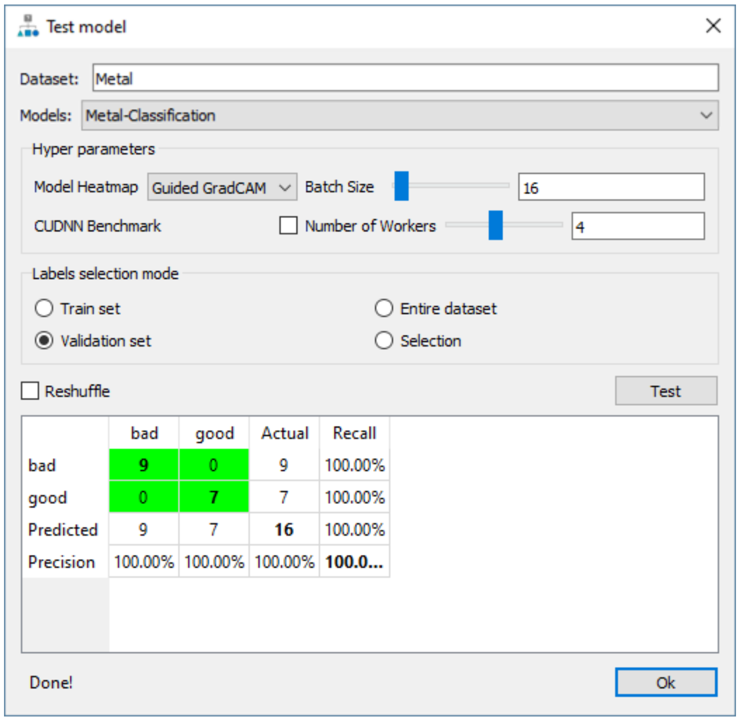
   When training is complete, a prompt is displayed to save the model.


Training completed

Would you like to save the model?

   Click **Yes**, enter the model name and description and click **OK**.

6. In the Dataset tab, select the dataset and click **Test model...**.In the Test Model dialog, select the model in the drop-down list and click **Test**.


Test model

Dataset: Metal

Models: Metal-Classification

Hyper parameters

Model Heatmap: Guided GradCAM

Batch Size: 16

CUDNN Benchmark

Number of Workers: 4

Labels selection mode

Train set

(●) Validation set

Entire dataset

Selection

Reshuffle

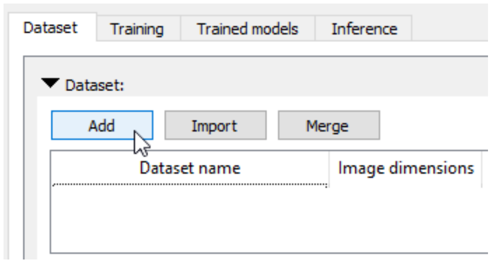
	bad	good	Actual	Recall
bad	**9**	0	9	100.00%
good	0	**7**	7	100.00%
Predicted	9	7	**16**	100.00%
Precision	100.00%	100.00%	100.00%	**100.0...**

Done!

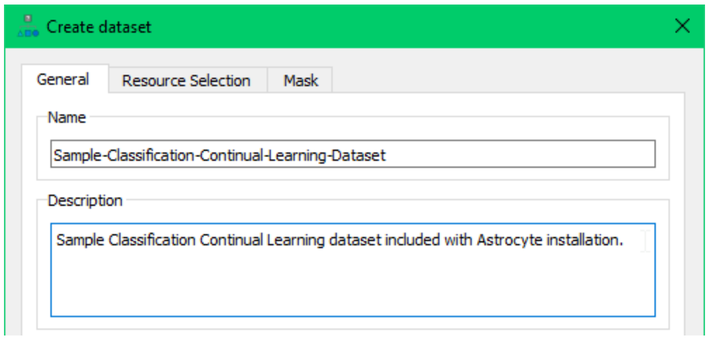
Quick Start to Creating a Continual Learning Classification Model

The following procedure demonstrates how to create a continual learning classification model using default values. This example uses the PCB components dataset included with the Astrocyte installation, available in the <installation directory>\Teledyne DALSA\Astrocyte\Datasets directory.

1. Launch the Astrocyte application and select the Classification module from the startup screen:
2. In the Dataset tab, click **Add**.



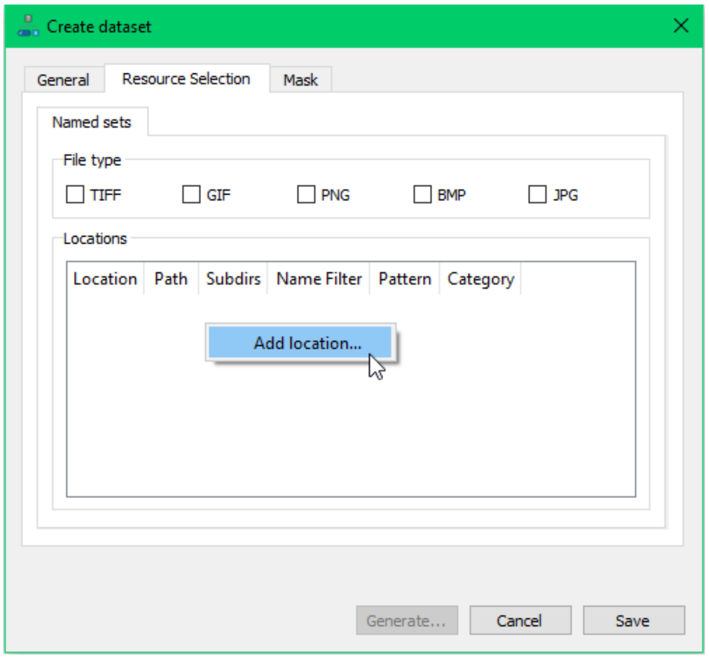
In the General tab, enter the dataset name and description.



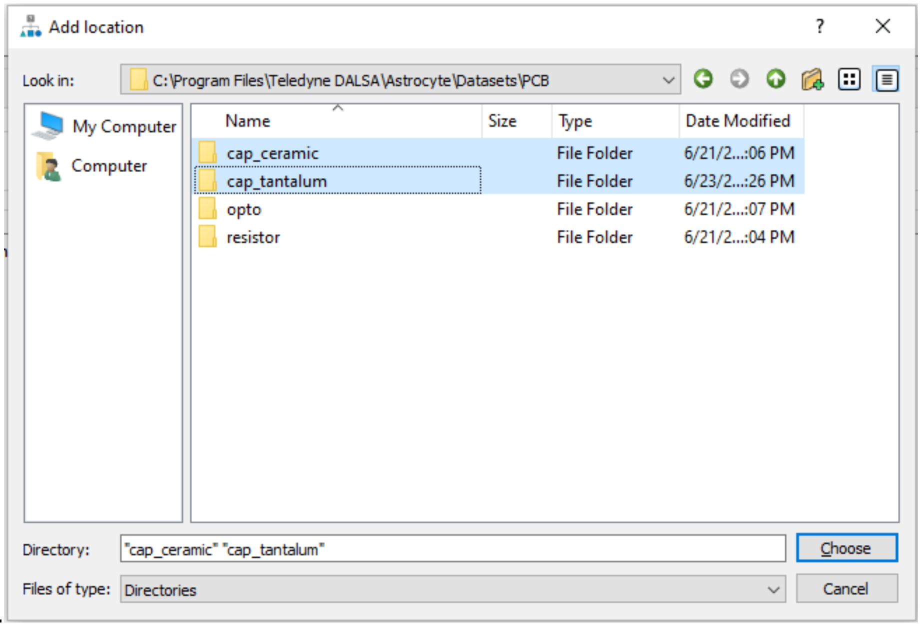
Name: Sample-Classification-Continual-Learning-Dataset

Description: Sample Classification Continual Learning dataset included with Astrocyte installation.

In the Resource Selection tab, right-click in the Locations field and select **Add location...**.



3. In the Add location dialog, navigate to the folder containing the dataset of training images; for this tutorial select the directories containing images of capacitors (cap_ceramic and cap_tantalum) and click **OK**



Look in: C:\Program Files\Teledyne DALSA\Astrocyte\Datasets\PCB

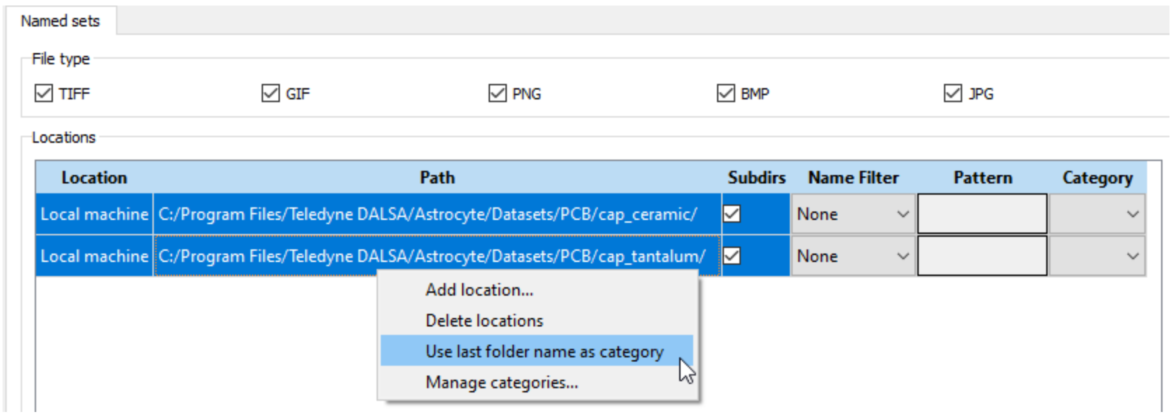
Name	Size	Type	Date Modified
cap_ceramic		File Folder	6/21/2...:06 PM
cap_tantalum		File Folder	6/23/2...:26 PM
opto		File Folder	6/21/2...:07 PM
resistor		File Folder	6/21/2...:04 PM

Directory: "cap_ceramic" "cap_tantalum"

Files of type: Directories

.

In the Locations section, select all directories in the Location panel, right-click and select **Use last folder name as category**.


Named sets

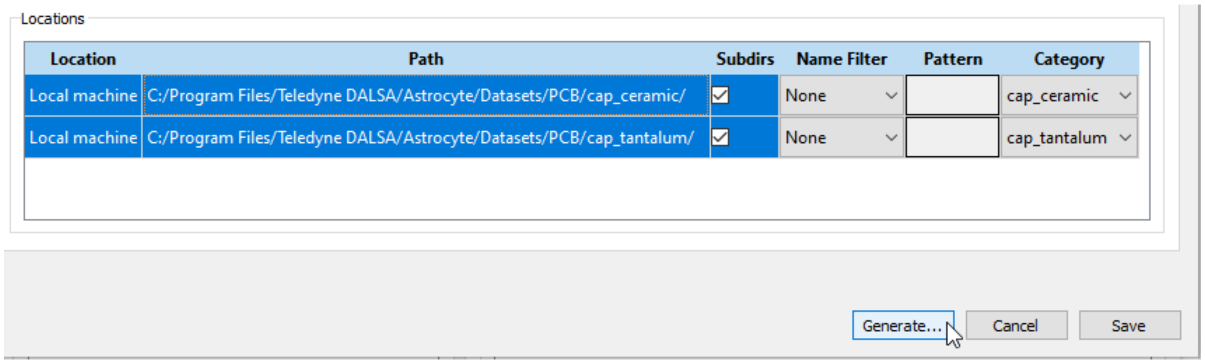
File type

☑ TIFF ☑ GIF ☑ PNG ☑ BMP ☑ JPG

Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/PCB/cap_ceramic/	✓	None		
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/PCB/cap_tantalum/	✓	None		

4. Verify the category labels and click **Generate** to prepare the dataset for the Cortex server.


Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/PCB/cap_ceramic/	✓	None		cap_ceramic
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/PCB/cap_tantalum/	✓	None		cap_tantalum

The Dataset generation configuration dialog allows you to select the image size of the dataset; the original image size can be used or the image dimensions reduced as required.

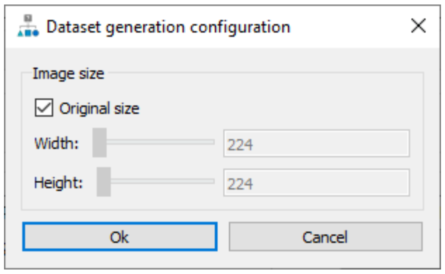

Dataset generation configuration

Image size

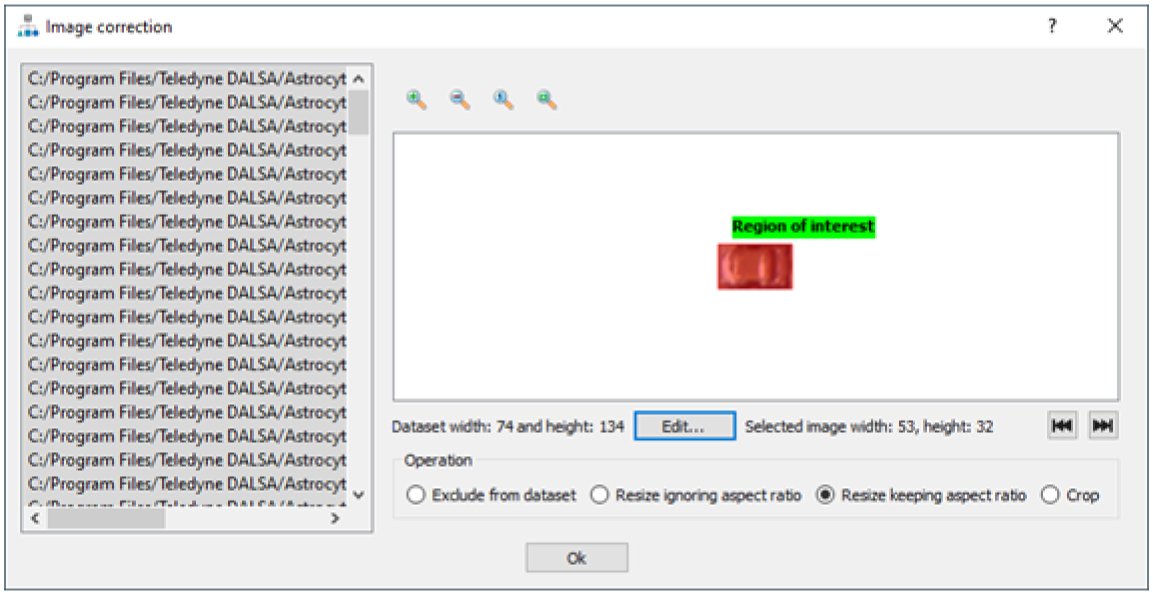
☑ Original size

Width: 224

Height: 224

5. Click Save to copy the dataset to the Cortex server. When the dataset is saved, if images all have the same dimensions, they are automatically resized to the specified maximum image size (otherwise, the Image Correction dialog is presented).

For the sample component dataset, select all images and use the **Resize keeping aspect ratio** option.



C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt
C:/Program Files/Teledyne DALSA/Astrocyt

Dataset width: 74 and height: 134

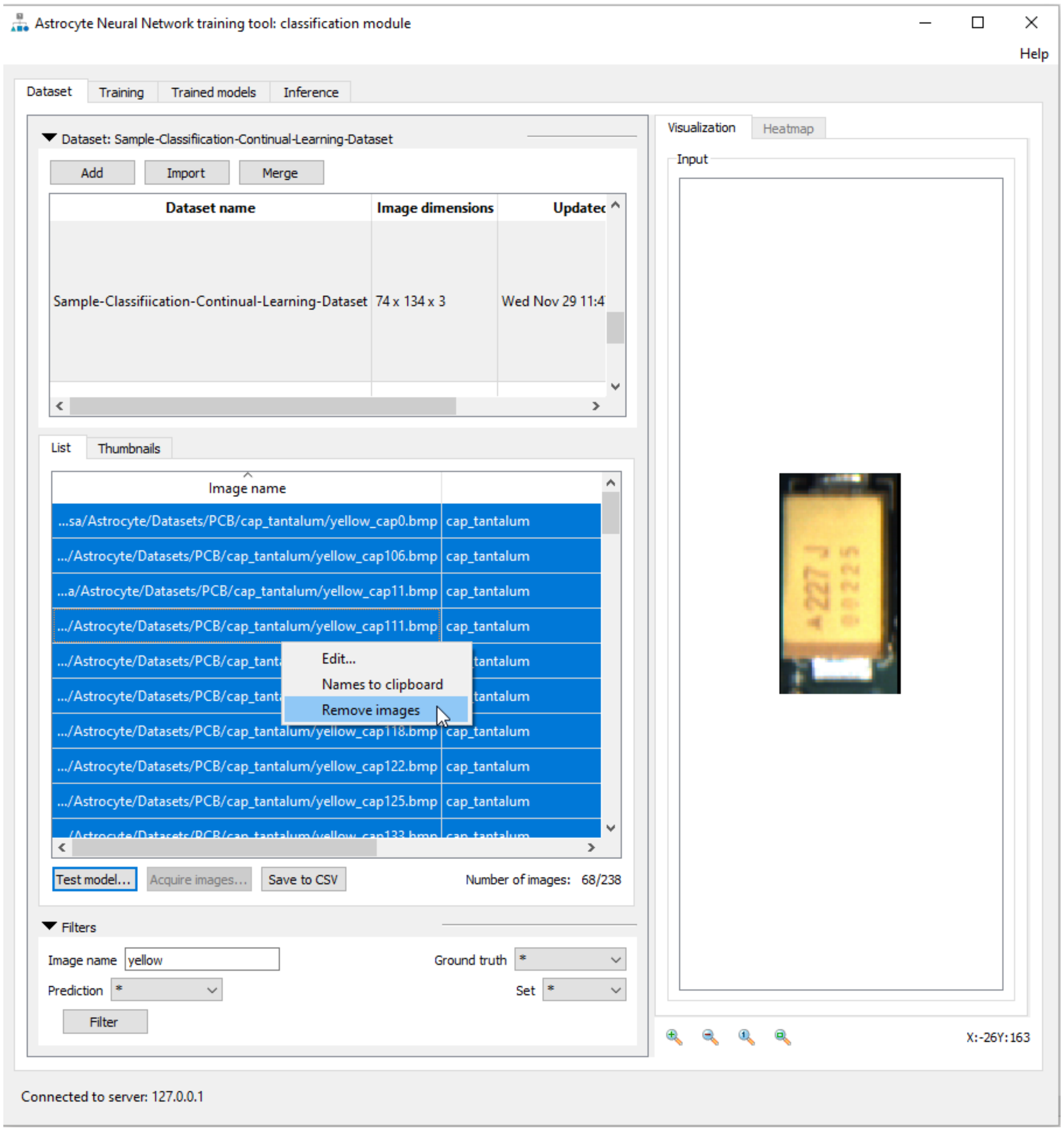
Selected image width: 53, height: 32

Operation

( ) Exclude from dataset ( ) Resize ignoring aspect ratio [✓ Resize keeping aspect ratio] ( ) Crop

6. For this tutorial delete the images of yellow tantalum capacitors from the dataset; after model training these images will be used to demonstrate continual learning by adding them to the existing *cap_tantalum* class.

   To do so, filter the images using the term "yellow" and remove these images from the dataset by right-clicking on the selected images and using the **Remove images** command.



Dataset: Sample-Classifiication-Continual-Learning-Dataset

Dataset name	Image dimensions	Updatec
Sample-Classifiication-Continual-Learning-Dataset	74 x 134 x 3	Wed Nov 29 11:4

Image name	
...sa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap0.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap106.bmp	cap_tantalum
...a/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap11.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap111.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tant	tantalum
.../Astrocyte/Datasets/PCB/cap_tant	tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap118.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap122.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap125.bmp	cap_tantalum
.../Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap133.bmp	cap_tantalum

Number of images: 68/238

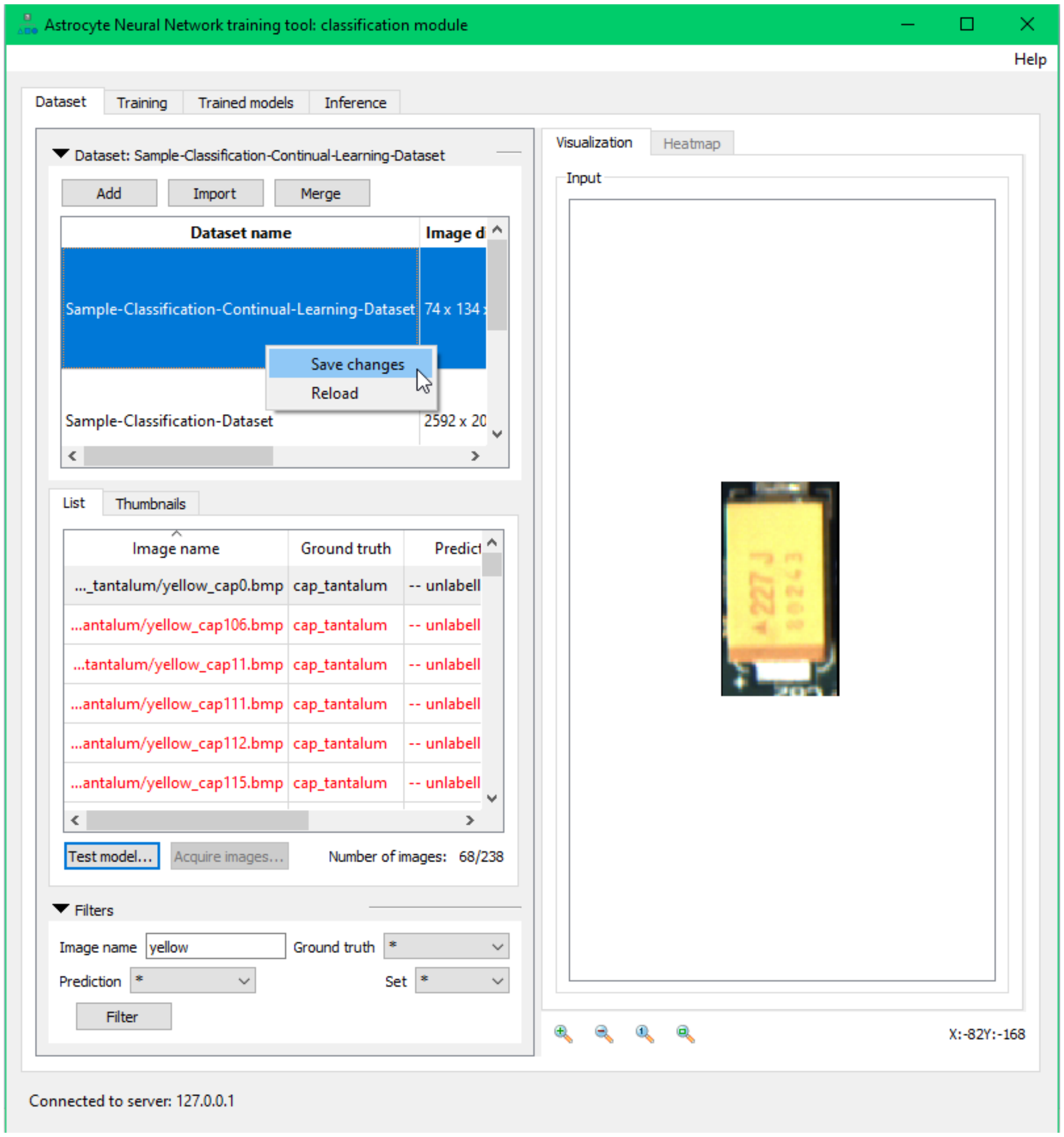
Filters

Image name: yellow
Ground truth: *
Prediction: *
Set: *

X:-26Y:163

Connected to server: 127.0.0.1

The images to delete are now shown in red; right-click on the dataset entry and click **Save changes** to update and save the dataset on the Cortex server.



Dataset name	Image d
Sample-Classification-Continual-Learning-Dataset	74 x 134
Sample-Classification-Dataset	2592 x 20

Image name	Ground truth	Predict
..._tantalum/yellow_cap0.bmp	cap_tantalum	-- unlabell
...antalum/yellow_cap106.bmp	cap_tantalum	-- unlabell
...tantalum/yellow_cap11.bmp	cap_tantalum	-- unlabell
...antalum/yellow_cap111.bmp	cap_tantalum	-- unlabell
...antalum/yellow_cap112.bmp	cap_tantalum	-- unlabell
...antalum/yellow_cap115.bmp	cap_tantalum	-- unlabell

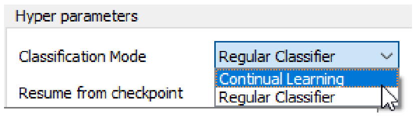
Number of images: 68/238

Image name: yellow
Ground truth: *
Prediction: *
Set: *

X:-82Y:-168

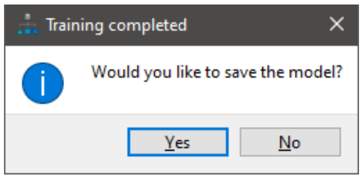
Connected to server: 127.0.0.1

In the Training tab, verify the selected dataset (the selection is made in the Dataset tab) and in the Hyper Parameters section select the Classification Mode **Continual Learning**.



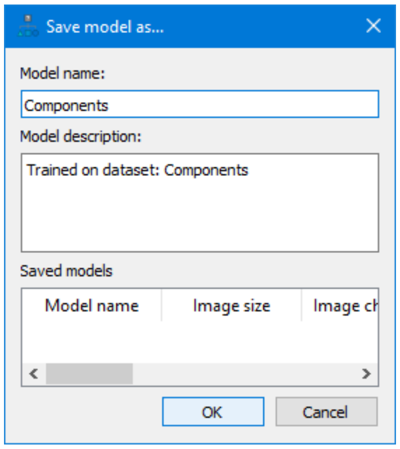
7. Click the Auto train tab (the selected training tab determines the option) and click **Train**; training starts and the training loss, accuracy graphs and progress bar are updated. Depending on the image size and available GPU memory the batch size may have to be reduced.

8. When training is complete, a prompt is displayed to save the model.


Training completed

Would you like to save the model?

Click **Yes**, enter the model name and description and click **OK**.


Save model as...

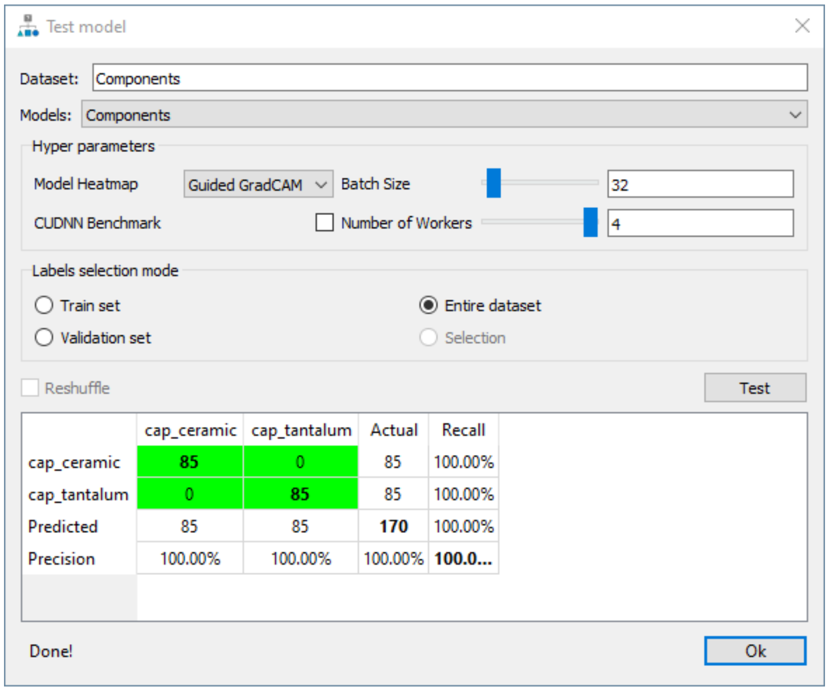
Model name: Components

Model description: Trained on dataset: Components

Saved models

Model name	Image size	Image ch

9. In the Dataset tab, select the dataset and click **Test model...**.In the Test Model dialog, select the model in the drop-down list and click **Test**.


Test model

Dataset: Components

Models: Components

Hyper parameters

Model Heatmap: Guided GradCAM

Batch Size: 32

CUDNN Benchmark: ☐

Number of Workers: 4

Labels selection mode

○ Train set [✓ Entire dataset]

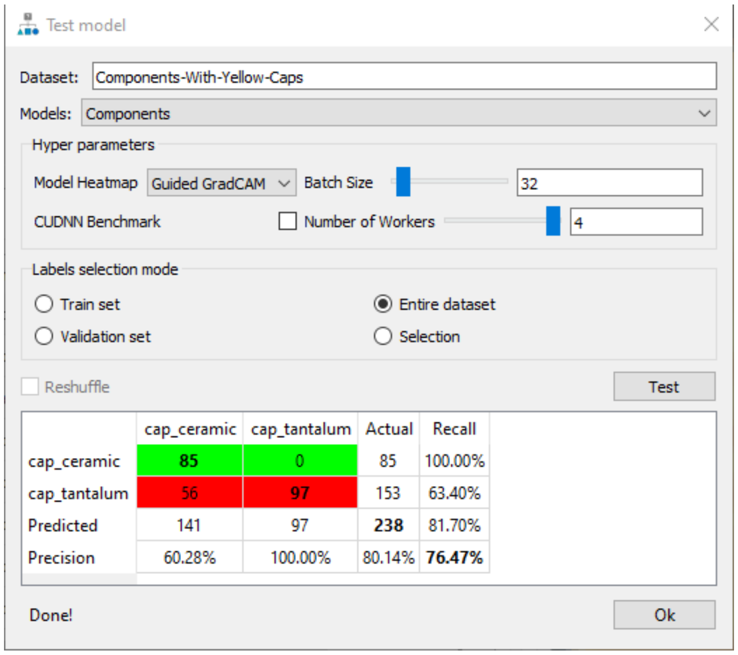
○ Validation set ○ Selection

☐ Reshuffle

	cap_ceramic	cap_tantalum	Actual	Recall
cap_ceramic	**85**	0	85	100.00%
cap_tantalum	0	**85**	85	100.00%
Predicted	85	85	**170**	100.00%
Precision	100.00%	100.00%	100.00%	**100.0...**

Done!

With the current dataset, the classifier performs very well with 100% accuracy. However, if the dataset is updated to include the yellow tantalum capacitors the performance is significantly impacted.



Test model

Dataset: Components-With-Yellow-Caps

Models: Components

Hyper parameters

Model Heatmap: Guided GradCAM; Batch Size: 32

CUDNN Benchmark; Number of Workers: 4

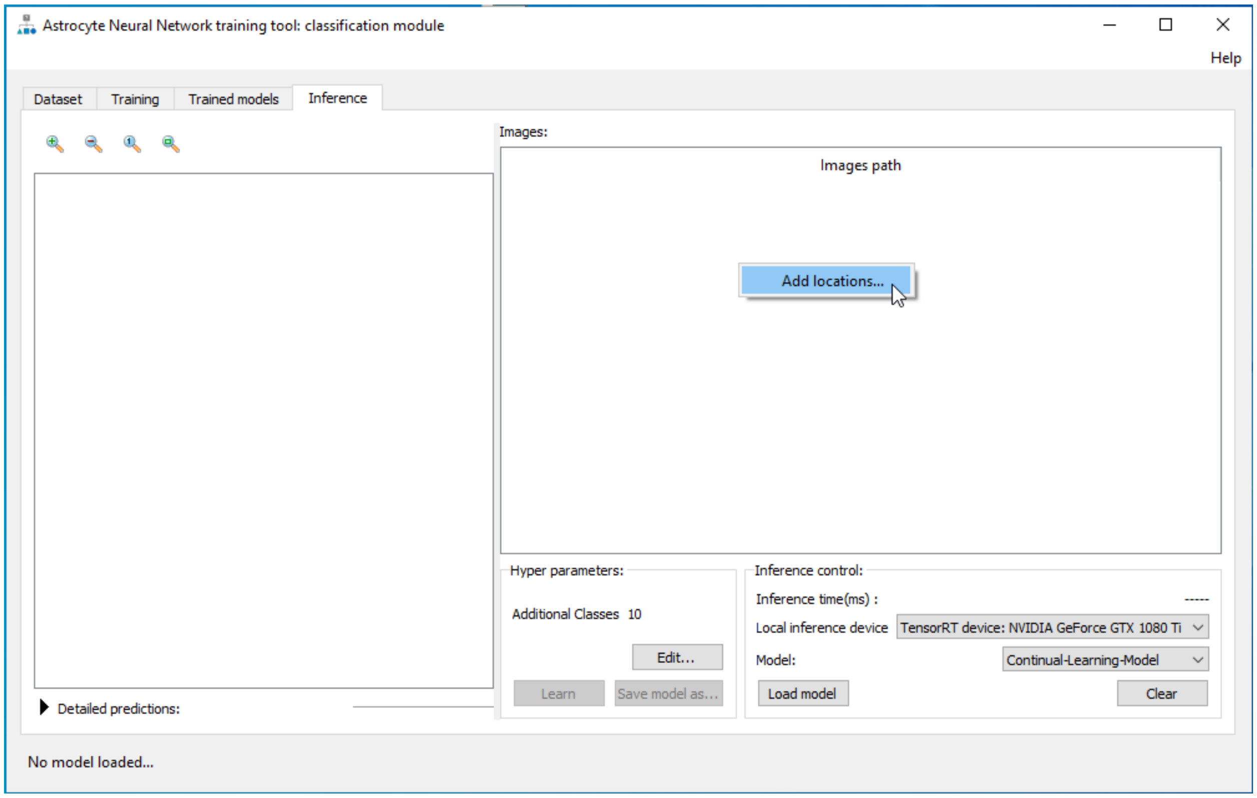
Labels selection mode: Train set, Validation set, [✓ Entire dataset], Selection

Reshuffle

	cap_ceramic	cap_tantalum	Actual	Recall
cap_ceramic	🟩 85	🟩 0	85	100.00%
cap_tantalum	🟥 56	🟥 97	153	63.40%
Predicted	141	97	**238**	81.70%
Precision	60.28%	100.00%	80.14%	**76.47%**

Done!

10. To improve model performance, update the model "cap_tantalum" category through continual learning on the yellow capacitor images. To do so, in the Inference tab, select the model from the drop-down list and click **Load model**. In the Images panel right-click and select **Add locations…**:



Astrocyte Neural Network training tool: classification module

Additional Classes 10

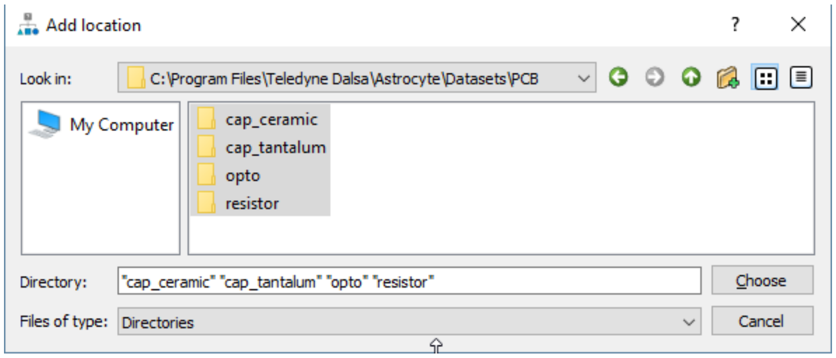
Inference time(ms) : -----

Local inference device: TensorRT device: NVIDIA GeForce GTX 1080 Ti

Model: Continual-Learning-Model

No model loaded...

Select all the folders in the PCB directory and click **OK**.


Add location

Look in: C:\Program Files\Teledyne Dalsa\Astrocyte\Datasets\PCB

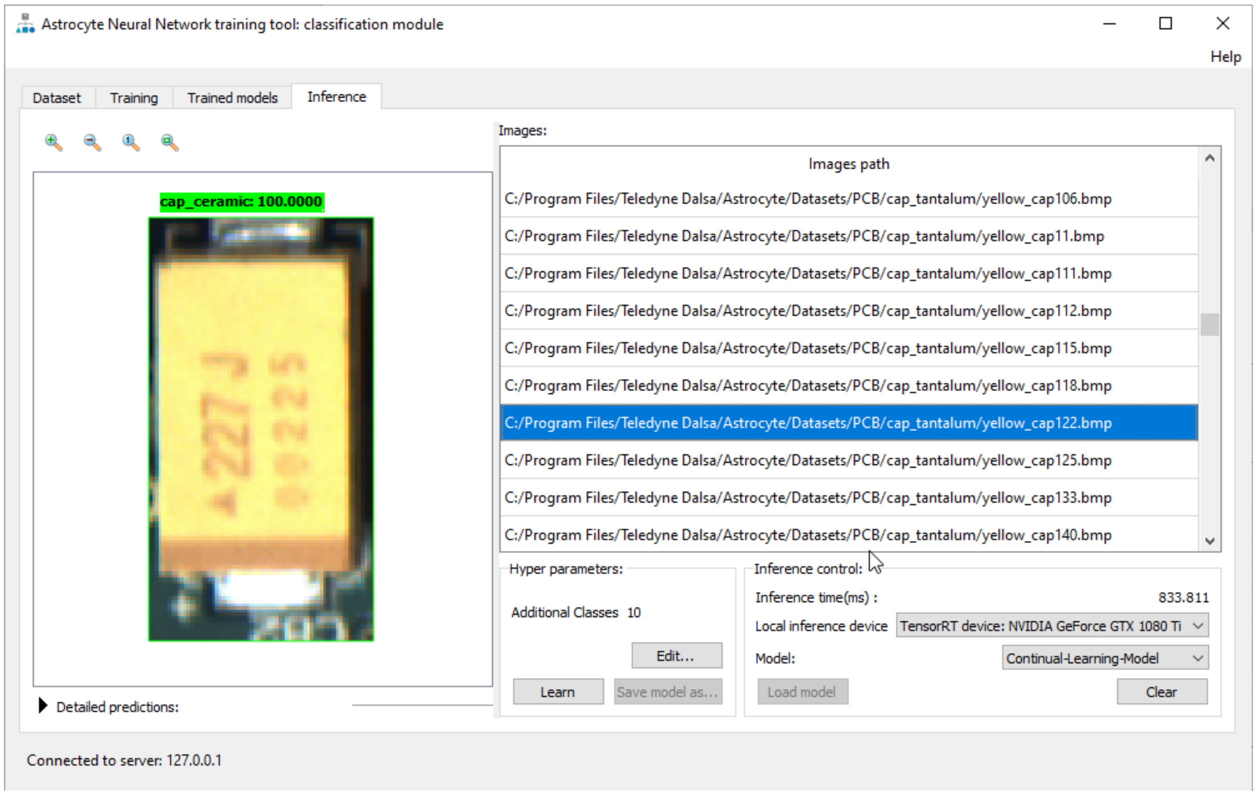
My Computer

cap_ceramic
cap_tantalum
opto
resistor

Directory: "cap_ceramic" "cap_tantalum" "opto" "resistor"

Files of type: Directories

11. Perform inference on a yellow capacitor by selecting an image that is incorrectly labelled or identified as "Undefined". For example, in this case it is misidentified as a "cap_ceramic" class.


Astrocyte Neural Network training tool: classification module

cap_ceramic: 100.0000

Images:

Images path
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap106.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap11.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap111.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap112.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap115.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap118.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap122.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap125.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap133.bmp
C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/cap_tantalum/yellow_cap140.bmp

Hyper parameters:

Additional Classes 10

Inference control:

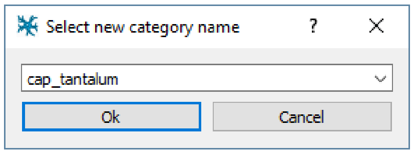
Inference time(ms) : 833.811

Local inference device: TensorRT device: NVIDIA GeForce GTX 1080 Ti

Model: Continual-Learning-Model

Connected to server: 127.0.0.1

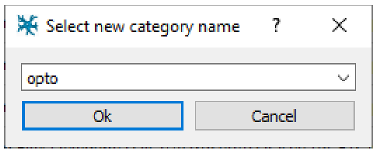
Click **Learn** and select the "cap_tantalum" category label to update the model with this image data for this category.


Select new category name

cap_tantalum

12. Continue to perform inference on the yellow capacitor images until another image is incorrectly identified; click **Learn** to add these images to the "cap_tantalum" label. The model should now correctly identify both black and yellow tantalum capacitors.

13. To demonstrate adding new categories to an existing model using continual learning, select an opto-coupler image and click **Learn**. Enter the new category label ("opto") and click **OK**.

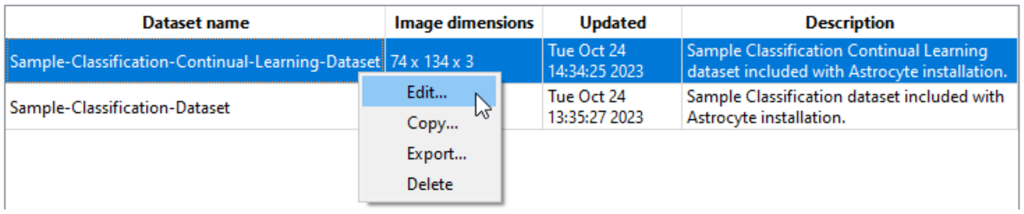


Do the same for a resistor, adding the category label “resistor” for a resistor image.

A few images are often sufficient to identify the new category if the images do not vary substantially. When updating the model it is recommended to verify that performance for previously trained categories is not adversely affected; this may occur depending on the characteristics of the added categories (if so, it may be advisable to retrain the model with the additional categories, rather than using the Learn feature).

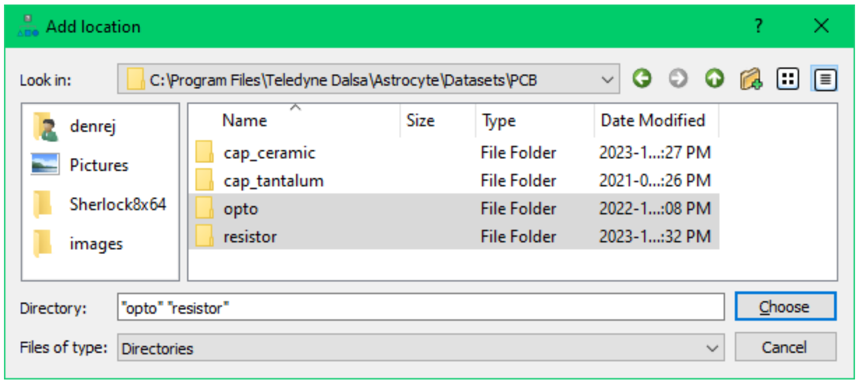
14. When model performance is satisfactory, click **Save model…**. Use a different file name if required to preserve the original model.

15. To test the updated model with all the components, the opto-coupler and resistor images can be added to the existing dataset. In the Dataset tab, right-click on the dataset and select **Edit...**



Dataset name	Image dimensions	Updated	Description
Sample-Classification-Continual-Learning-Dataset	74 x 134 x 3	Tue Oct 24 14:34:25 2023	Sample Classification Continual Learning dataset included with Astrocyte installation.
Sample-Classification-Dataset		Tue Oct 24 13:35:27 2023	Sample Classification dataset included with Astrocyte installation.

Right-click in the Locations panel, select **Add Location...** and add the directories:

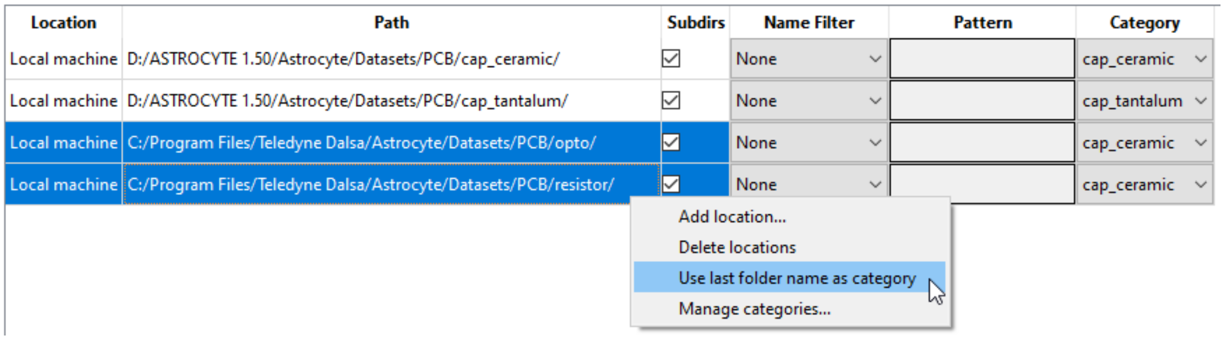

Add location
?
Look in: C:\Program Files\Teledyne Dalsa\Astrocyte\Datasets\PCB
denrej
Pictures
Sherlock8x64
images

Name	Size	Type	Date Modified
cap_ceramic		File Folder	2023-1...:27 PM
cap_tantalum		File Folder	2021-0...:26 PM
opto		File Folder	2022-1...:08 PM
resistor		File Folder	2023-1...:32 PM

Directory: "opto" "resistor"

Files of type: Directories

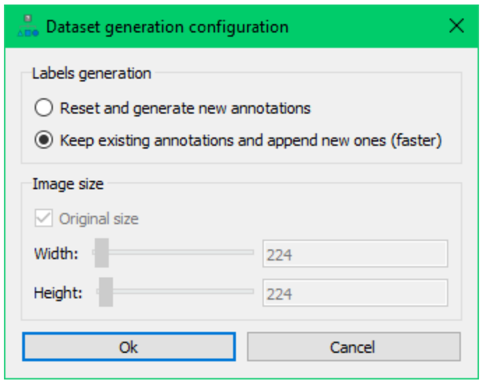
Select the directories, right-click and select **Use last folder name as category**.



Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	D:/ASTROCYTE 1.50/Astrocyte/Datasets/PCB/cap_ceramic/	☑	None		cap_ceramic
Local machine	D:/ASTROCYTE 1.50/Astrocyte/Datasets/PCB/cap_tantalum/	☑	None		cap_tantalum
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/opto/	☑	None		cap_ceramic
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/PCB/resistor/	☑	None		cap_ceramic

Note that the names must match the categories added to the model, in this case, "opto" and "resistor".

Click **Generate**, select **Keep existing annotation and append new ones**, and click **OK**.



Dataset generation configuration

Labels generation

( ) Reset and generate new annotations
(●) Keep existing annotations and append new ones (faster)

Image size

☑ Original size
Width: 224
Height: 224

Click **Save**, select the option **Resize keeping aspect ratio**, and click **Apply all**.

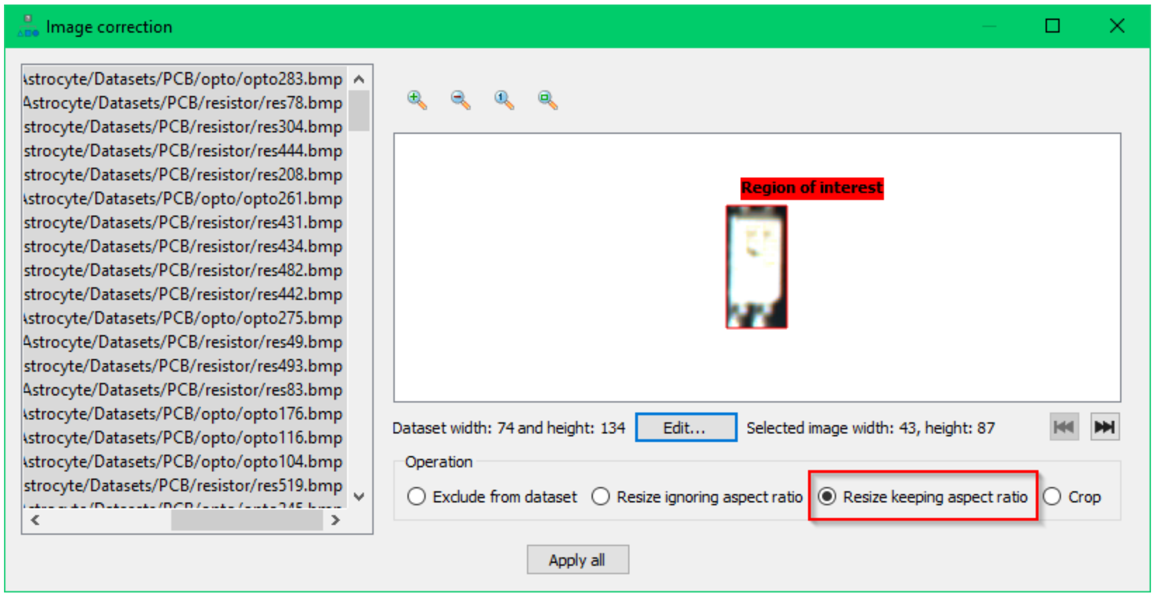


Image correction

```
strocyte/Datasets/PCB/opto/opto283.bmp
Astrocyte/Datasets/PCB/resistor/res78.bmp
strocyte/Datasets/PCB/resistor/res304.bmp
strocyte/Datasets/PCB/resistor/res444.bmp
strocyte/Datasets/PCB/resistor/res208.bmp
strocyte/Datasets/PCB/opto/opto261.bmp
strocyte/Datasets/PCB/resistor/res431.bmp
strocyte/Datasets/PCB/resistor/res434.bmp
strocyte/Datasets/PCB/resistor/res482.bmp
strocyte/Datasets/PCB/resistor/res442.bmp
strocyte/Datasets/PCB/opto/opto275.bmp
Astrocyte/Datasets/PCB/resistor/res49.bmp
strocyte/Datasets/PCB/resistor/res493.bmp
Astrocyte/Datasets/PCB/resistor/res83.bmp
strocyte/Datasets/PCB/opto/opto176.bmp
strocyte/Datasets/PCB/opto/opto116.bmp
strocyte/Datasets/PCB/opto/opto104.bmp
strocyte/Datasets/PCB/resistor/res519.bmp
```

Dataset width: 74 and height: 134 Edit... Selected image width: 43, height: 87

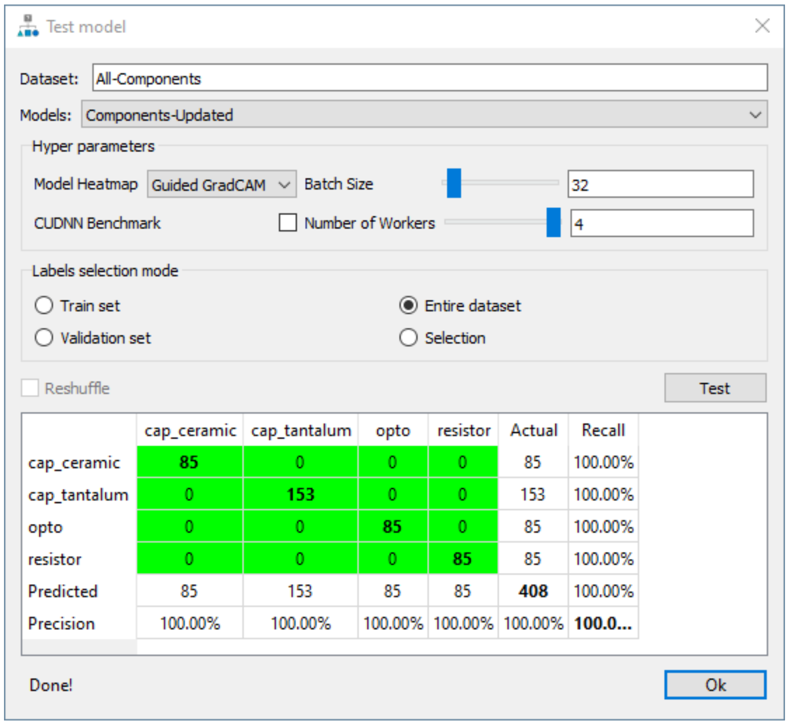
Operation

( ) Exclude from dataset ( ) Resize ignoring aspect ratio (●) Resize keeping aspect ratio ( ) Crop

When completed click **Close**.

16. Select the updated dataset with all the PCB components (408 images total, 74x134x3), click **Test model…**, select the updated model, select **Entire dataset** and click **Test**.

The test results should show that all components are correctly identified.



Test model

Dataset: All-Components

Models: Components-Updated

Hyper parameters

Model Heatmap: Guided GradCAM

Batch Size: 32

CUDNN Benchmark: ☐

Number of Workers: 4

Labels selection mode

( ) Train set [✓ Entire dataset]

( ) Validation set ( ) Selection

☐ Reshuffle

	cap_ceramic	cap_tantalum	opto	resistor	Actual	Recall
cap_ceramic	**85**	0	0	0	85	100.00%
cap_tantalum	0	**153**	0	0	153	100.00%
opto	0	0	**85**	0	85	100.00%
resistor	0	0	0	**85**	85	100.00%
Predicted	85	153	85	85	**408**	100.00%
Precision	100.00%	100.00%	100.00%	100.00%	100.00%	**100.0...**

Done!

Object Detection Model Training

Object detection model training requires that labelled bounding boxes identify the target objects in the training dataset images. Bounding box coordinates can be provided using a single bounding box database file for each image in the dataset or a single file for all the images within the dataset. The database format is specified when creating a dataset. Astrocyte supports PASCAL, MSCOCO and KITTI file formats, as well as custom-defined text file based bounding box description files.

Alternatively, bounding boxes can be created manually for each image using the tools available in the Dataset. Each image in the training set can contain multiple bounding boxes and category labels. Bounding boxes may overlap.

Astrocyte also includes a Semi-Supervised Object Detection (SSOD) Self-Training option that automatically generates bounding boxes and their corresponding category labels for unlabelled images. An assisted labelling function is also available that allows users to select an existing object detection model to annotate individual images.

When trained, an object detection model can detect and locate of multiple categories of objects within the same image. The location of objects is returned as a bounding box. Object detection can identify objects with variable sizes as opposed to classification models which is less tolerant of variations in object size in the target image.

Note: Datasets with images that are smaller than the selected input size for the model type (224x244 (minimum) for YOLOX) are automatically resized ignoring the aspect ratio. In general, it is preferable to have images in the dataset to all have comparable dimensions close to the specified input size.

Quick Start to Creating an Object Detection Model

The following procedure demonstrates how to create an object detection model using default values.

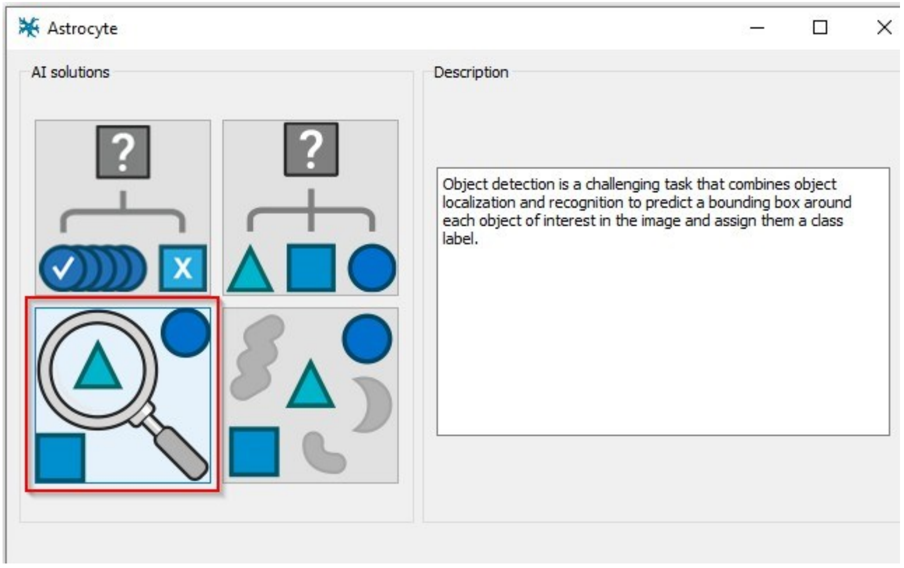
This example uses the hardware dataset included with the Astrocyte installation, available in the *<installation directory>\Teledyne DALSA\Astrocyte\Datasets* directory.

This example details three options available with the sample hardware dataset:

- *Regular:* Object detection using bounding boxes without angles
- *Rotated:* Object detection using rotated bounding boxes
- *Self-Training:* Object detection using Self-Training

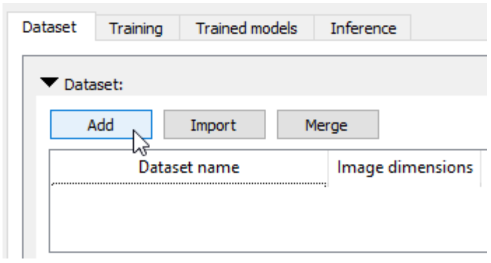
The option is selected when creating the dataset.

Launch the Astrocyte application and select the Object Detection module from the startup screen:

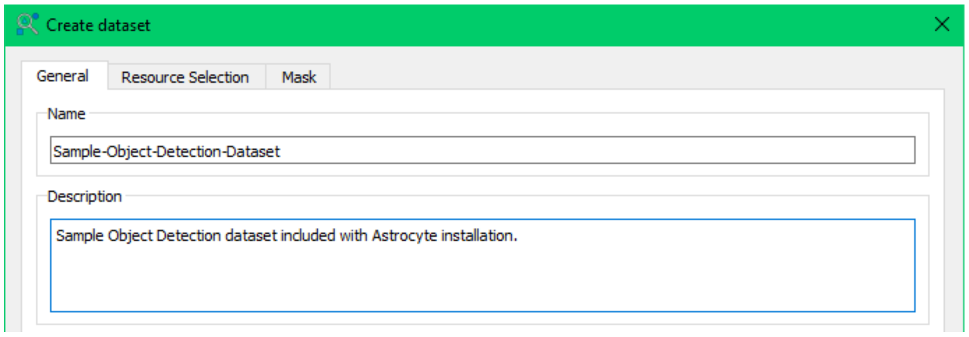


Object detection is a challenging task that combines object localization and recognition to predict a bounding box around each object of interest in the image and assign them a class label.

1. In the Dataset tab, click **Add**.



2. In the Create dataset dialog General tab, enter the dataset name and description.

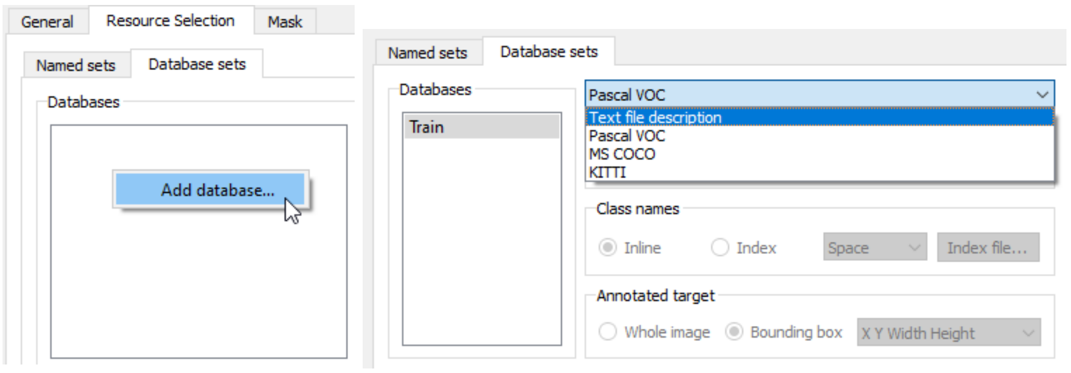


Name: Sample-Object-Detection-Dataset

Description: Sample Object Detection dataset included with Astrocyte installation.

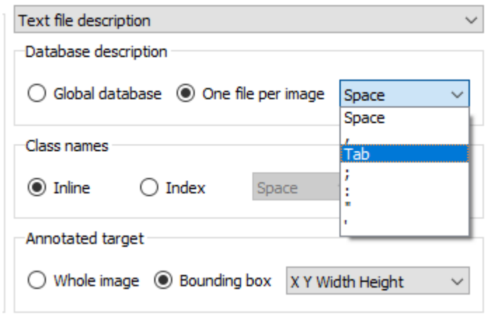
3. In the Resource Selection tab, select the Database sets tab, right-click and use the **Add database…** command and enter the database name (for example, “Train”).

   - *For the Regular option:* In the Text file description drop-down list select **Text file description**.

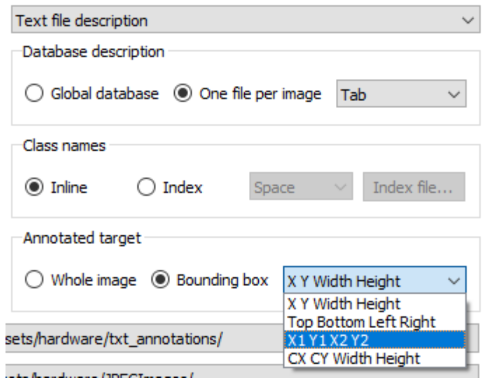


   - *For the Rotated and Self-Training options:* In the Text file description drop-down list select **Pascal VOC** and skip step 4.

4. Enable **One file per image** and select **Tab** as the descriptor separator from the drop-down list.



For the bounding box format select **X1 Y1 X2 Y2**.



Text file description

Database description: ○ Global database ◉ One file per image — Tab

Class names: ◉ Inline ○ Index — Space — Index file...

Annotated target: ○ Whole image ◉ Bounding box — X Y Width Height

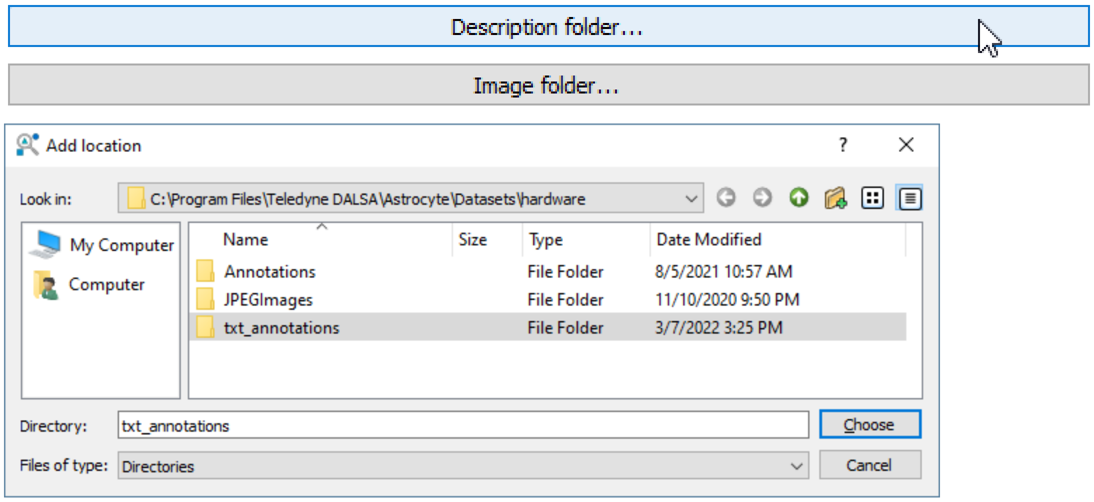
X Y Width Height
Top Bottom Left Right
X1 Y1 X2 Y2
CX CY Width Height

sets/hardware/txt_annotations/

5. Click the **Description folder...** button and navigate to location where the annotation files are located and click **Choose**.

- *For the Regular Option:* select the following folder.

  <installation directory>\Teledyne DALSA\Astrocyte\Datasets\hardware\txt_annotations



Add location

Look in: C:\Program Files\Teledyne DALSA\Astrocyte\Datasets\hardware

My Computer
Computer

Name	Size	Type	Date Modified
Annotations		File Folder	8/5/2021 10:57 AM
JPEGImages		File Folder	11/10/2020 9:50 PM
txt_annotations		File Folder	3/7/2022 3:25 PM

Directory: txt_annotations — Choose

Files of type: Directories — Cancel

For the Rotated option: select the following folder.

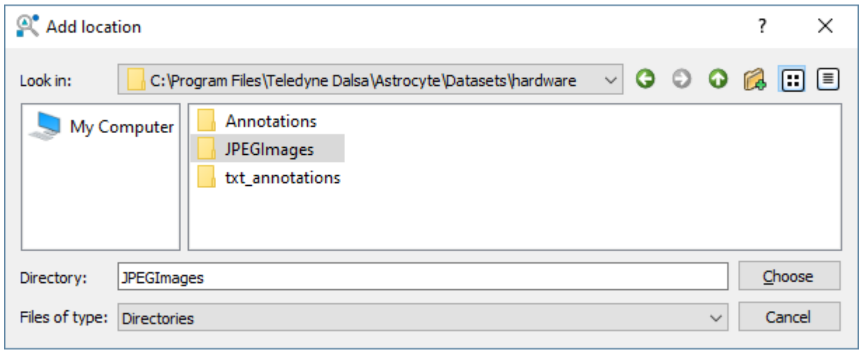
<installation directory>\Teledyne DALSA\Astrocyte\Datasets\hardware_obb\Annotations

For the Self-Training option: select the following folder. Note that the annotation files contain annotations for 40 of the 100 images in the dataset.

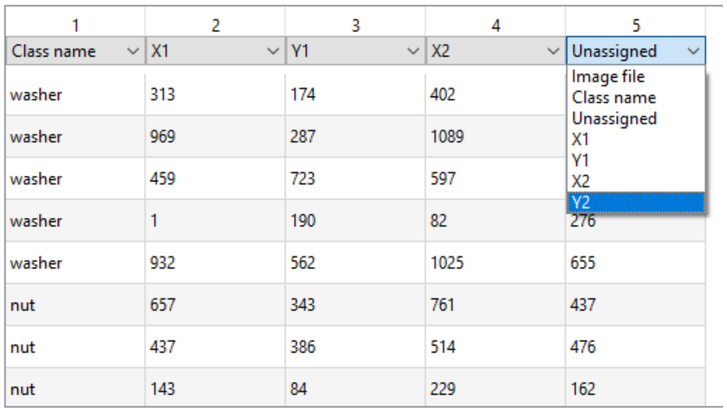
<installation directory>\Teledyne DALSA\Astrocyte\Datasets\hardware_ssod\Annotations

6. Click the **Image folder...** button and navigate to location where the dataset image files are located and click **Choose**. For all examples this is:

<Install Directory>\Teledyne DALSA\Astrocyte\Datasets\hardware\JPEGImages

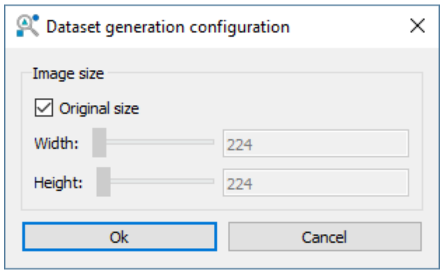

Add location
Look in: C:\Program Files\Teledyne Dalsa\Astrocyte\Datasets\hardware
My Computer
Annotations
JPEGImages
txt_annotations
Directory: JPEGImages
Choose
Files of type: Directories
Cancel

7. This step is required for the regular option only. For the Rotated and Self-Training options Pascal VOC annotations do not require this step. Use the drop-down list to assign the data type. For the example hardware dataset, the five columns entries are Class name, X1, Y1, X2, Y2.



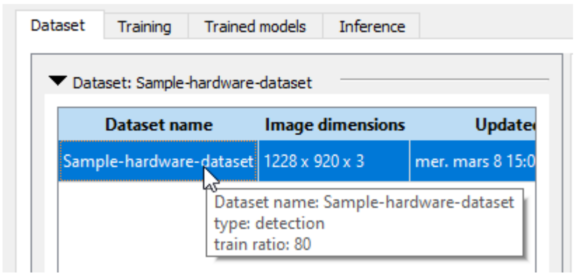
1	2	3	4	5
Class name	X1	Y1	X2	Unassigned
washer	313	174	402	
washer	969	287	1089	
washer	459	723	597	
washer	1	190	82	276
washer	932	562	1025	655
nut	657	343	761	437
nut	437	386	514	476
nut	143	84	229	162

8. Click **Generate…** to prepare the dataset for the Cortex server. In the Dataset generation configuration dialog click **OK** to use the default values.



9. Click **Save** to add the dataset to the Cortex server. If all images have the same dimensions, they are automatically resized to the specified image size on the Cortex server (the original dataset is not modified); otherwise, correct images using the [Image Correction](#) dialog.

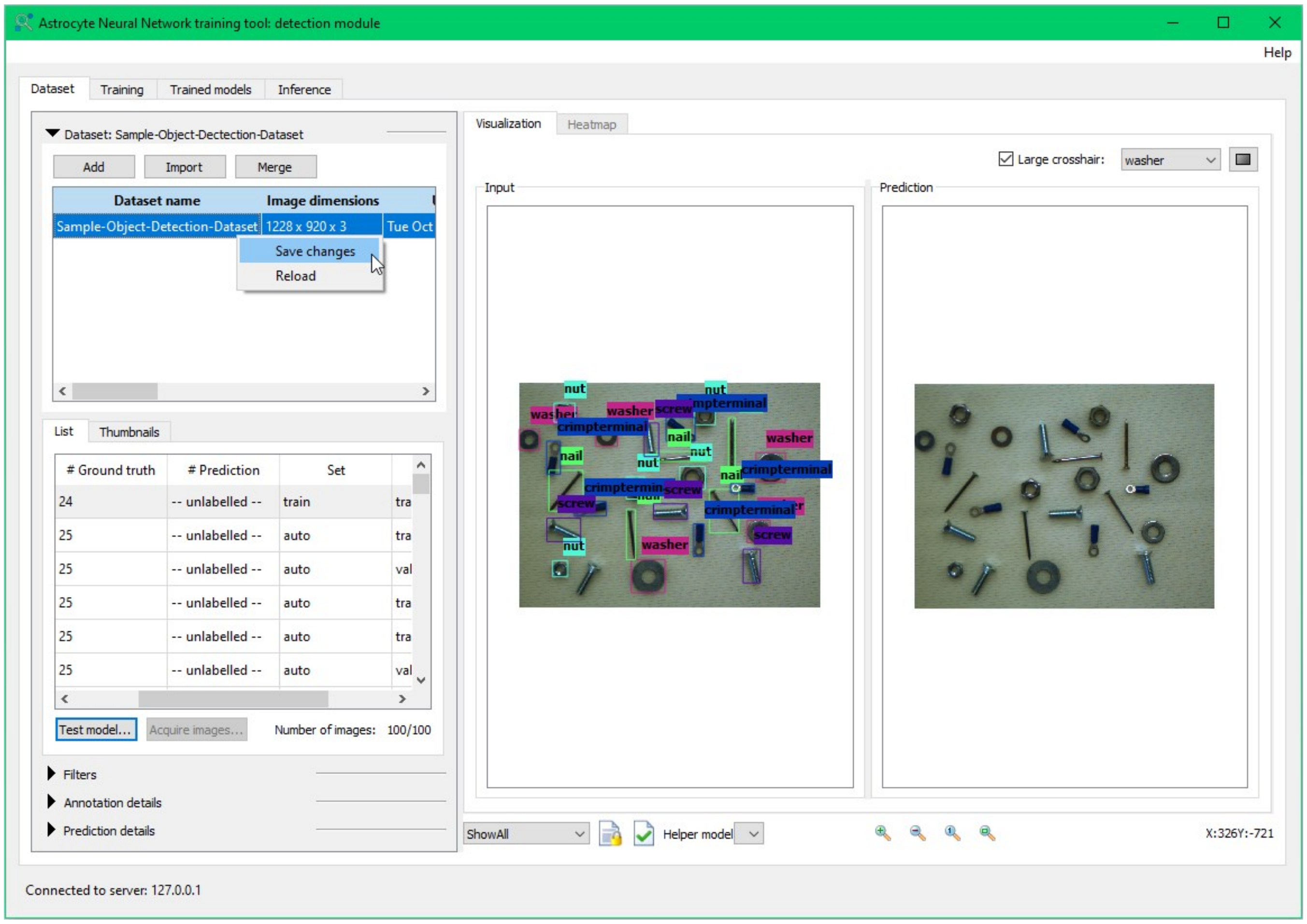
In the Dataset tab, select the dataset by double-clicking on it.



Dataset name	Image dimensions	Update
Sample-hardware-dataset	1228 x 920 x 3	mer. mars 8 15:0

Dataset name: Sample-hardware-dataset
type: detection
train ratio: 80

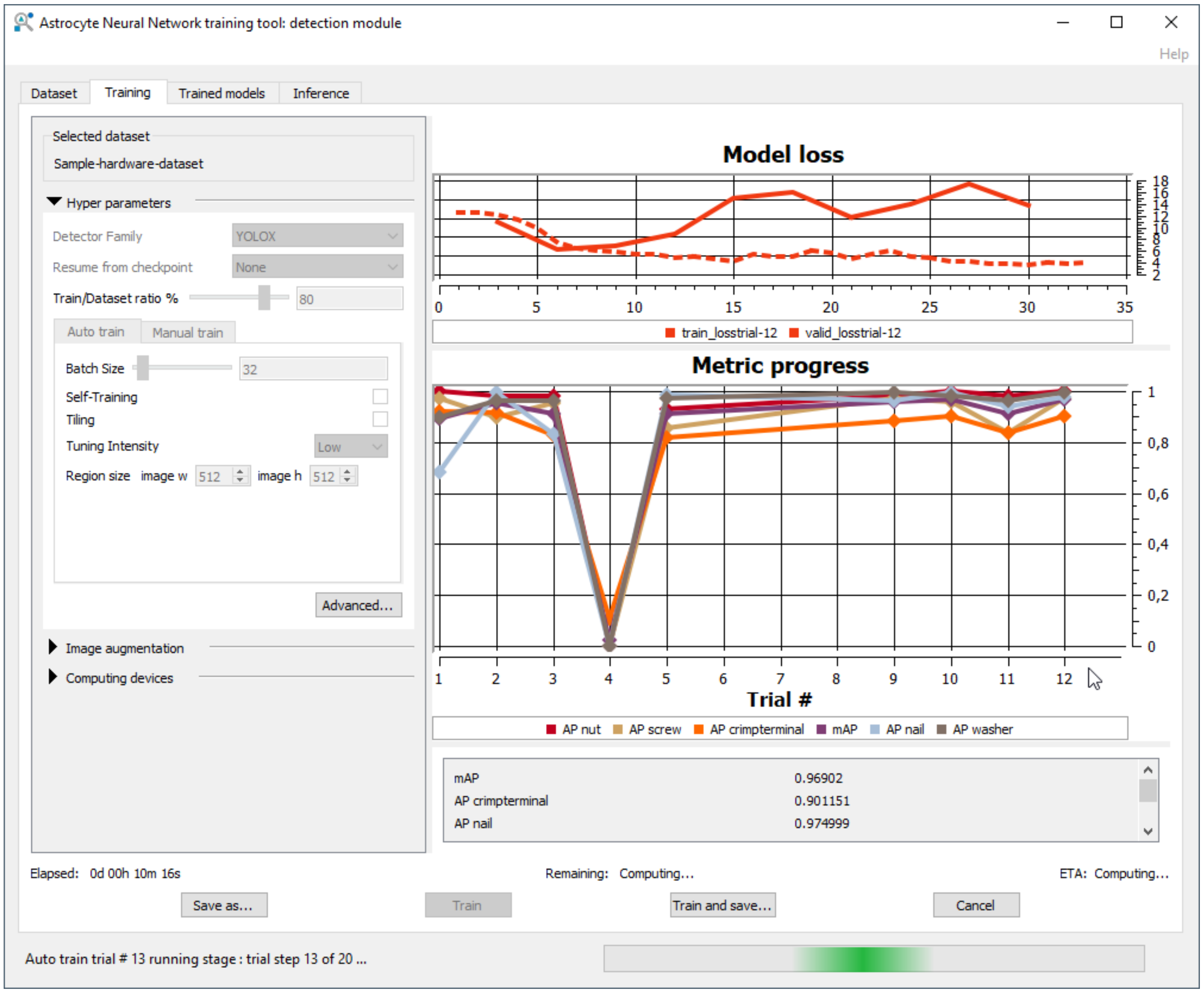
Verify the dataset images, labels and bounding boxes and make any required changes. If the dataset is modified, right-click on the dataset entry and click **Save changes** to update and save the dataset on the Cortex server.



Dataset name	Image dimensions	
Sample-Object-Detection-Dataset	1228 x 920 x 3	Tue Oct

# Ground truth	# Prediction	Set	
24	-- unlabelled --	train	tra
25	-- unlabelled --	auto	tra
25	-- unlabelled --	auto	val
25	-- unlabelled --	auto	tra
25	-- unlabelled --	auto	tra
25	-- unlabelled --	auto	val

10. In the Training tab, verify the selected dataset (the selection is made in the Dataset tab), enable the Auto train tab and click Train; the Model loss graph and Metric progress graphs are updated at the completion of batches during trials (the estimated completion time is not calculated since Early stopping is enabled).


Astrocyte Neural Network training tool: detection module

Selected dataset
Sample-hardware-dataset

Hyper parameters

Detector Family: YOLOX
Resume from checkpoint: None
Train/Dataset ratio %: 80

Batch Size: 32
Self-Training
Tiling
Tuning Intensity: Low
Region size image w 512 image h 512

mAP	0.96902
AP crimpterminal	0.901151
AP nail	0.974999

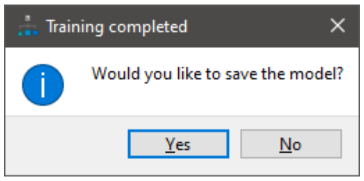
Elapsed: 0d 00h 10m 16s Remaining: Computing... ETA: Computing...

Auto train trial # 13 running stage : trial step 13 of 20 ...

For the Self-Training option click the Self-Training checkbox to enable it and set the number of epochs to 100 to ensure generation of pseudo-labels for unlabeled images)

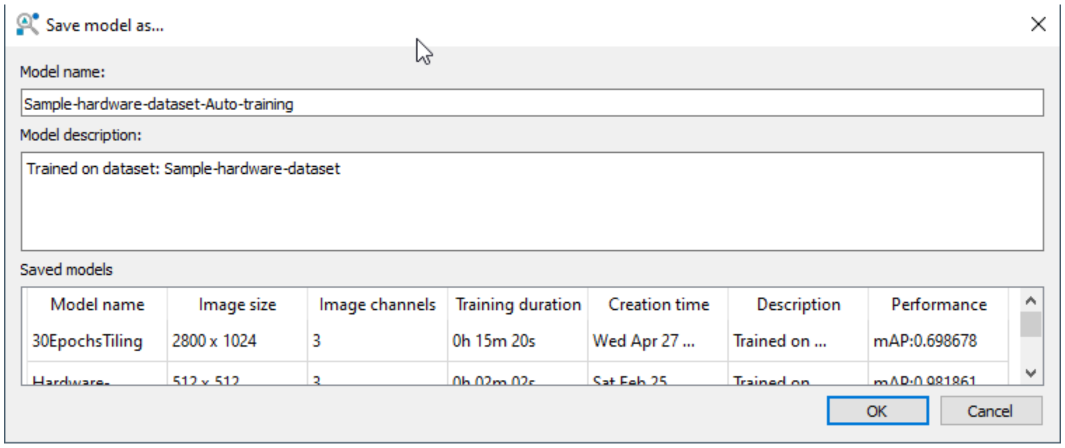
Depending on the image size and available GPU memory the batch size may have to be reduced. For the sample hardware dataset using the YOLOX-Nano architecture a batch size of 32 should be within the memory resource capabilities of an 8GB GPU.

When training is complete, a prompt is displayed to save the model; click **Yes.**


Training completed

Would you like to save the model?

Enter the model name and description and click **OK**.



Save model as...

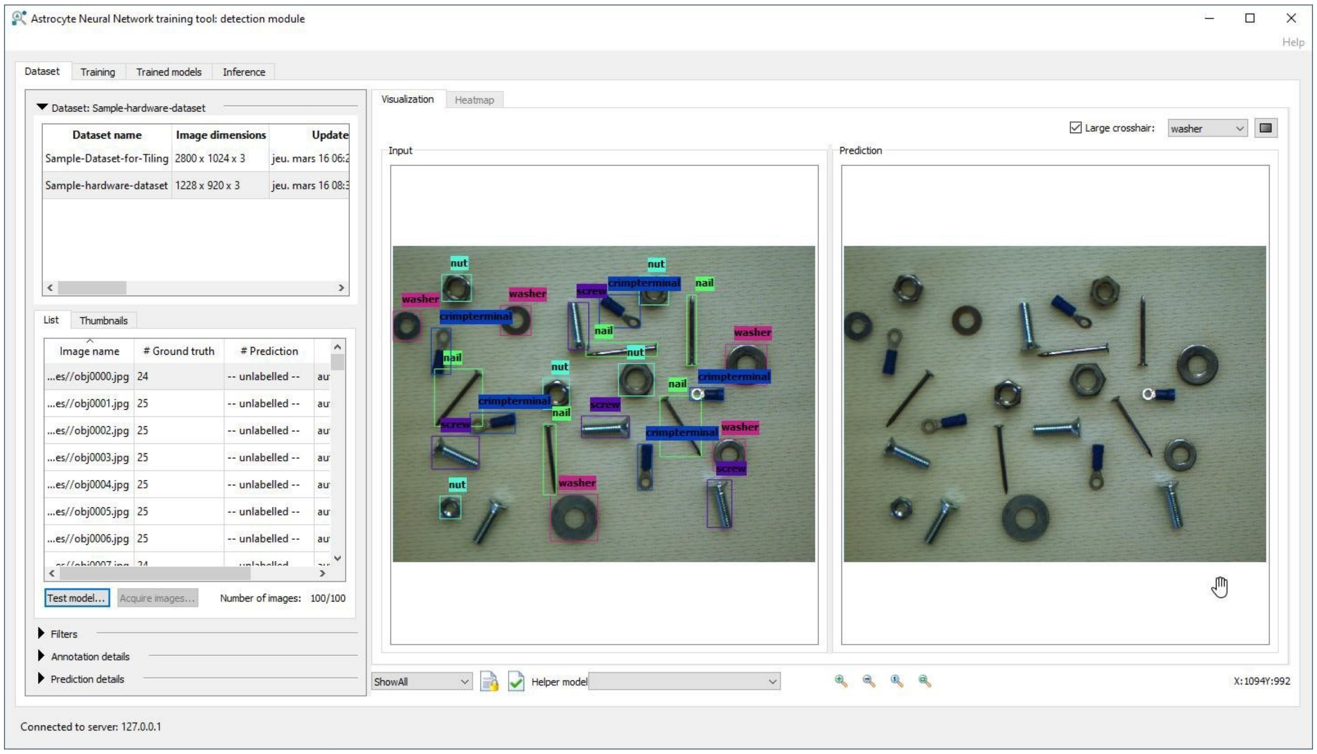
Model name: Sample-hardware-dataset-Auto-training

Model description: Trained on dataset: Sample-hardware-dataset

Saved models

Model name	Image size	Image channels	Training duration	Creation time	Description	Performance
30EpochsTiling	2800 x 1024	3	0h 15m 20s	Wed Apr 27 ...	Trained on ...	mAP:0.698678
Hardware-	512 x 512	3	0h 02m 02s	Sat Feb 25	Trained on	mAP:0.981861

11. In the Dataset tab, select the dataset and click **Test model...**.



Astrocyte Neural Network training tool: detection module

Dataset: Sample-hardware-dataset

Dataset name	Image dimensions	Update
Sample-Dataset-for-Tiling	2800 x 1024 x 3	jeu. mars 16 06:2
Sample-hardware-dataset	1228 x 920 x 3	jeu. mars 16 08:3

Image name	# Ground truth	# Prediction	
...es//obj0000.jpg	24	-- unlabelled --	au
...es//obj0001.jpg	25	-- unlabelled --	au
...es//obj0002.jpg	25	-- unlabelled --	au
...es//obj0003.jpg	25	-- unlabelled --	au
...es//obj0004.jpg	25	-- unlabelled --	au
...es//obj0005.jpg	25	-- unlabelled --	au
...es//obj0006.jpg	25	-- unlabelled --	au

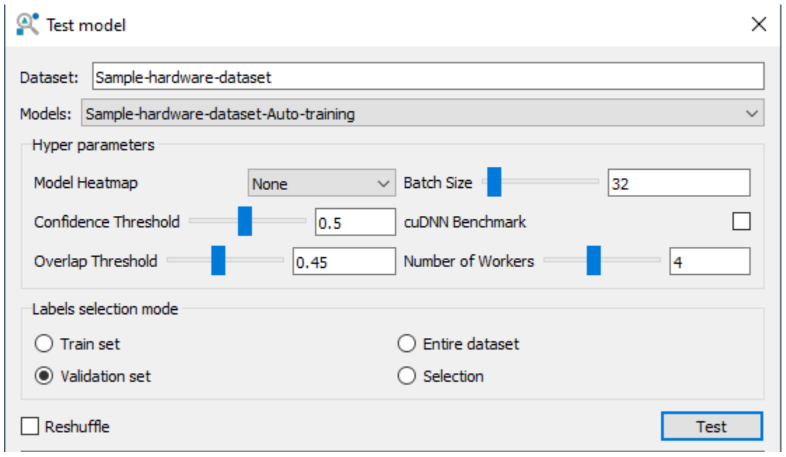
Number of images: 100/100

X: 1094Y:992

Connected to server: 127.0.0.1

For the Self-Training option, results for the pseudo-labels generated are shown in pink and can be sorted using the Pseudo labels checkbox.

12. In the Test model dialog, select the model in the drop-down list and click **Test**. By default, testing is performed on the validation subset of the training images.



Test model

Dataset: Sample-hardware-dataset

Models: Sample-hardware-dataset-Auto-training

Hyper parameters

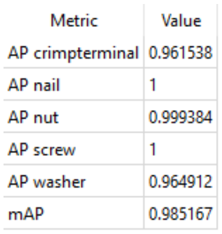
Model Heatmap	None	Batch Size	32
Confidence Threshold	0.5	cuDNN Benchmark	☐
Overlap Threshold	0.45	Number of Workers	4

Labels selection mode

Train set [✓ Validation set] Entire dataset Selection

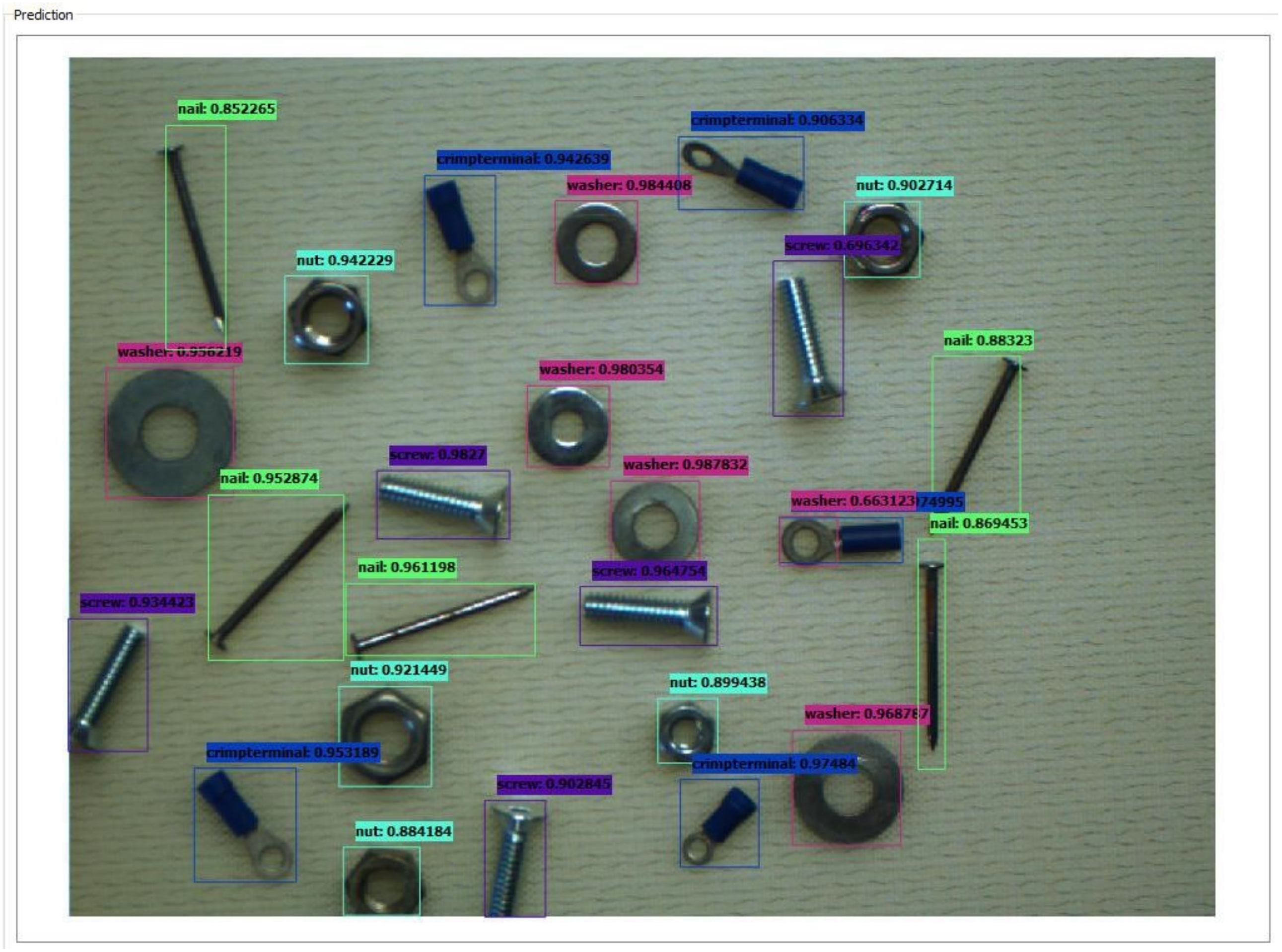
☐ Reshuffle

When testing is completed, the average precision (AP) results for each category and the model total mean average precision (mAP) are displayed in the Test model dialog.



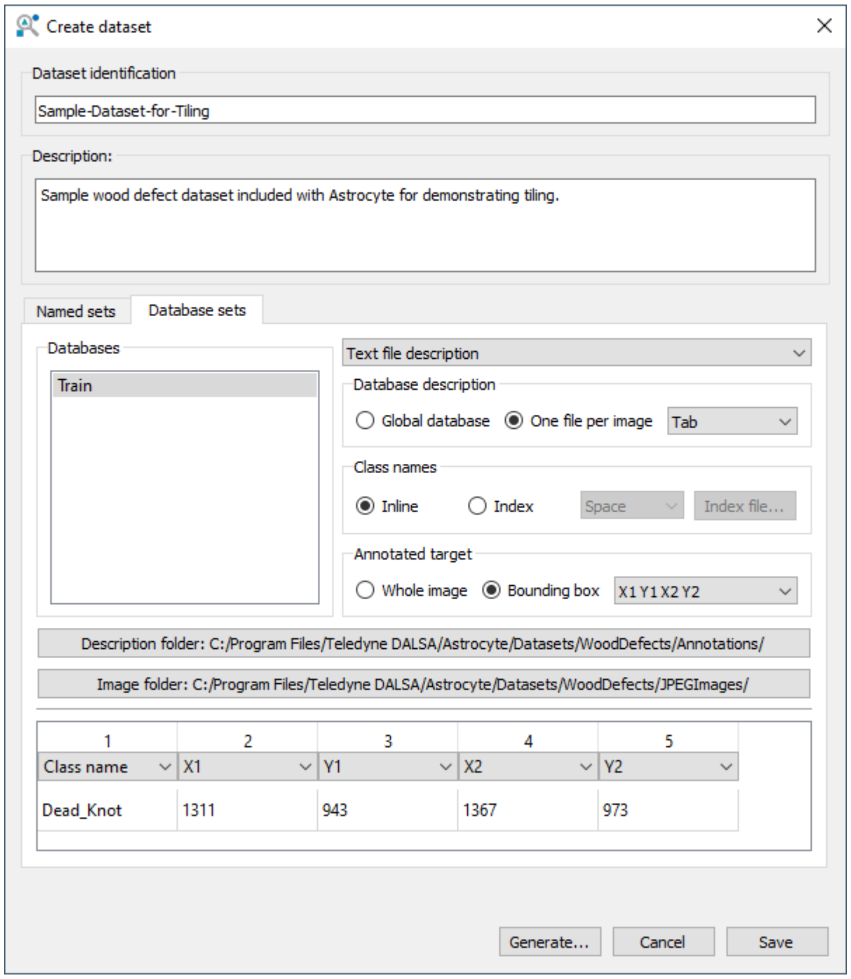
Metric	Value
AP crimpterminal	0.961538
AP nail	1
AP nut	0.999384
AP screw	1
AP washer	0.964912
mAP	0.985167

Prediction bounding boxes include the category and confidence score.


Prediction
nail: 0.852265
crimpterminal: 0.906334
crimpterminal: 0.942639
washer: 0.984408
nut: 0.902714
nut: 0.942229
screw: 0.696342
washer: 0.956219
washer: 0.980354
nail: 0.88323
screw: 0.9827
washer: 0.987832
nail: 0.952874
washer: 0.663123
nail: 0.869453
nail: 0.961198
screw: 0.964754
screw: 0.934423
nut: 0.921449
nut: 0.899438
washer: 0.968787
crimpterminal: 0.953189
crimpterminal: 0.97484
screw: 0.902845
nut: 0.884184

Tiling Example

To demonstrate the use of tiling, Astrocyte includes a dataset (WoodDefects) composed of high-resolution images with tiny defects to locate. Follow the steps outlined in the Quick Start to Creating an Object Detection Model using this dataset.



Dataset identification: Sample-Dataset-for-Tiling

Description: Sample wood defect dataset included with Astrocyte for demonstrating tiling.

Databases: Train

Text file description

Database description: Global database [✓ One file per image] Tab

Class names: [✓ Inline] Index Space

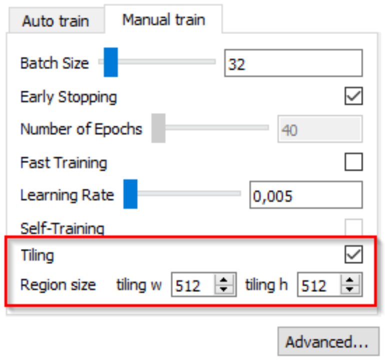
Annotated target: Whole image [✓ Bounding box] X1 Y1 X2 Y2

Description folder: C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/WoodDefects/Annotations/

Image folder: C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/WoodDefects/JPEGImages/

1	2	3	4	5
Class name	X1	Y1	X2	Y2
Dead_Knot	1311	943	1367	973

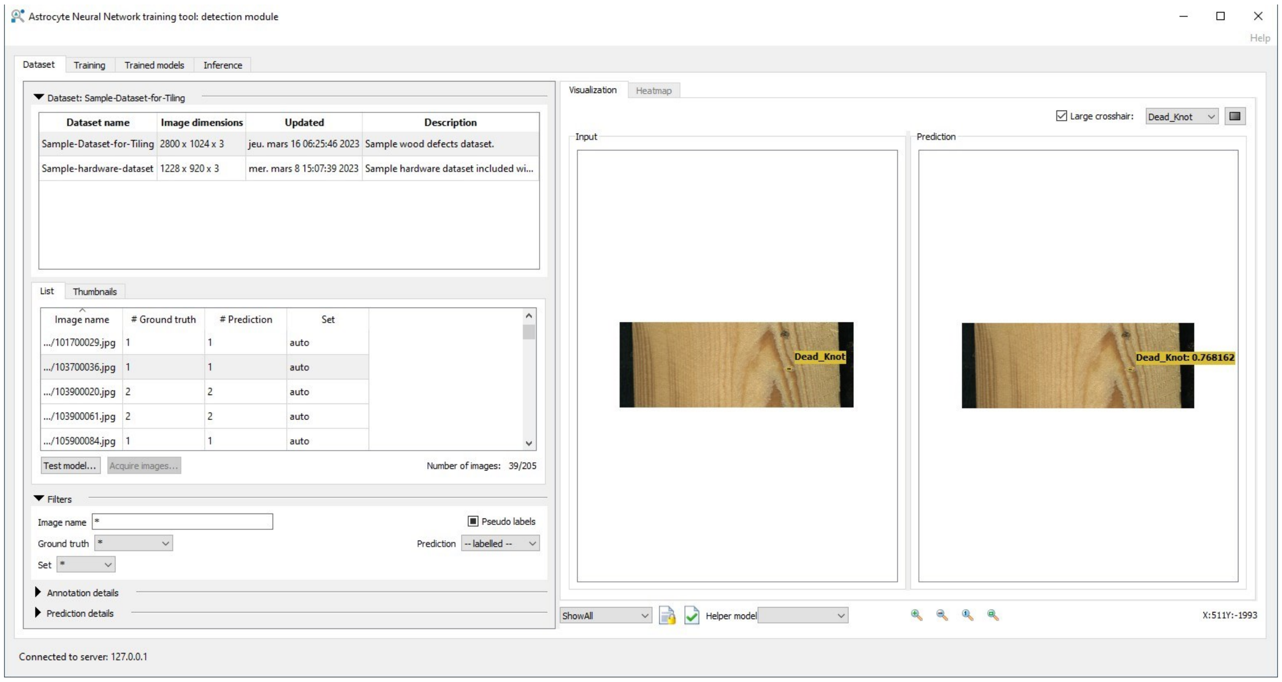
In the Training tab, enable the Tiling option.



Batch Size: 32
Early Stopping: ✓
Number of Epochs: 40
Fast Training: ☐
Learning Rate: 0,005
Self-Training: ☐
Tiling: ✓
Region size: tiling w 512 tiling h 512

Depending on the image size and available GPU memory the batch size may have to be reduced. For this example, a batch size of 16 using 512x512 tiles should be within the memory resource capabilities of an 8GB GPU.

When training is complete the model can be tested as with all other models.



Astrocyte Neural Network training tool: detection module

Dataset: Sample-Dataset-for-Tiling

Dataset name	Image dimensions	Updated	Description
Sample-Dataset-for-Tiling	2800 x 1024 x 3	jeu. mars 16 06:25:46 2023	Sample wood defects dataset.
Sample-hardware-dataset	1228 x 920 x 3	mer. mars 8 15:07:39 2023	Sample hardware dataset included wi...

Image name	# Ground truth	# Prediction	Set
.../101700029.jpg	1	1	auto
.../103700036.jpg	1	1	auto
.../103900020.jpg	2	2	auto
.../103900061.jpg	2	2	auto
.../105900084.jpg	1	1	auto

Number of images: 39/205

Filters

Image name: *

Pseudo labels

Ground truth: *

Prediction: -- labelled --

Set: *

Large crosshair: Dead_Knot

X:511Y:-1993

Connected to server: 127.0.0.1

Segmentation Model Training

For object segmentation model training, each image in the dataset has a corresponding mask image where pixel values (color) identify the object type and associated category label. That is, each category is associated with a specific mask pixel value. The entire image is segmented; that is, divided into regions for each object type.

Quick Start to Creating a Segmentation Model

The following procedure demonstrates how to create a semantic segmentation model using default values. This example uses the MaterialSegmentation dataset included with the Astrocyte installation, available in the *<installation directory>\Teledyne DALSA\Astrocyte\Datasets* directory.

Launch the Astrocyte application and select the Segmentation module from the startup screen:

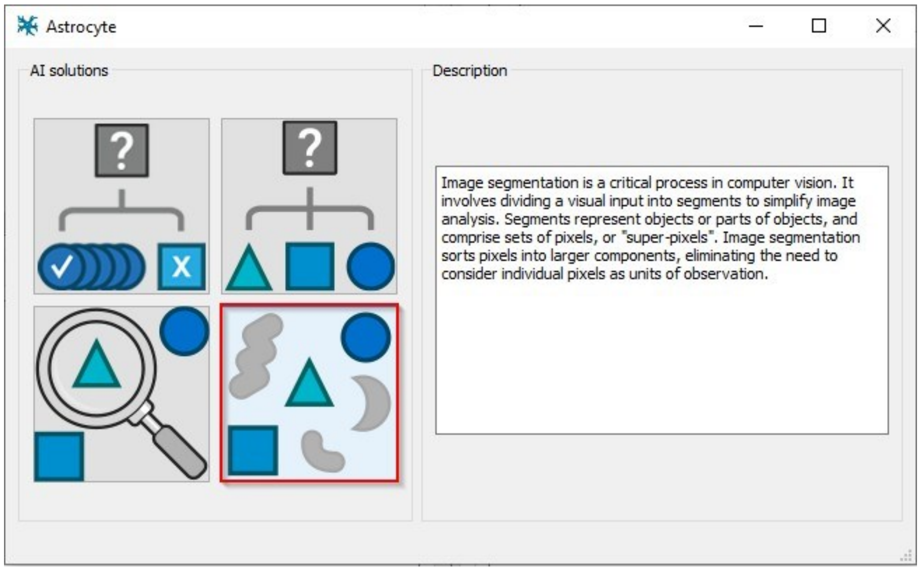
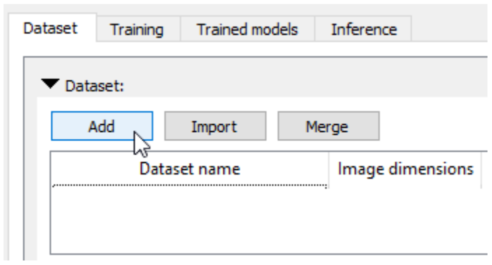
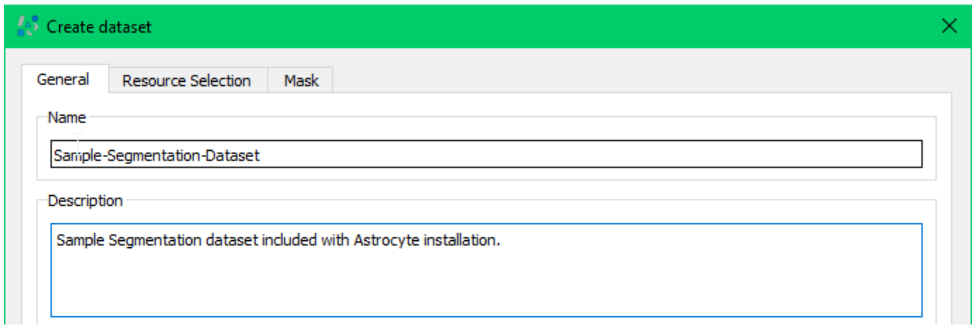


Image segmentation is a critical process in computer vision. It involves dividing a visual input into segments to simplify image analysis. Segments represent objects or parts of objects, and comprise sets of pixels, or "super-pixels". Image segmentation sorts pixels into larger components, eliminating the need to consider individual pixels as units of observation.

1. In the Dataset tab, click **Add**.



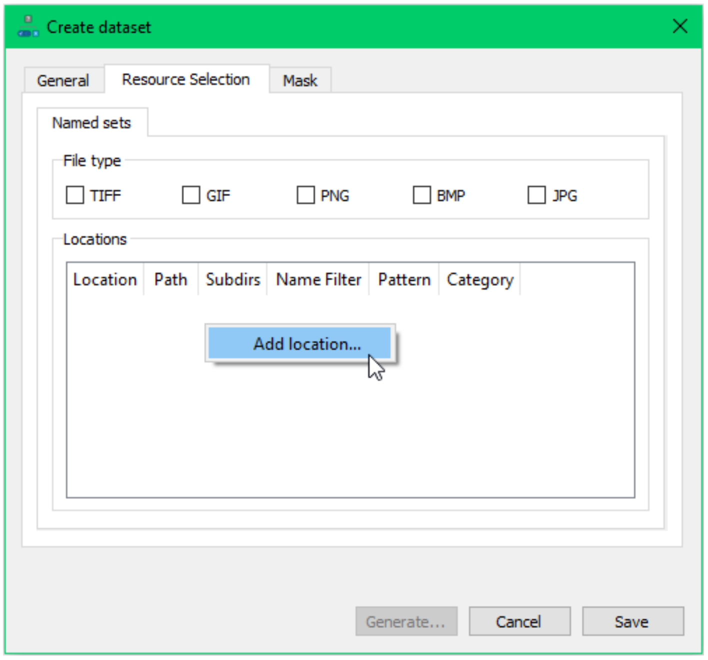
2. In the Create dataset dialog General tab, enter the dataset name and description.



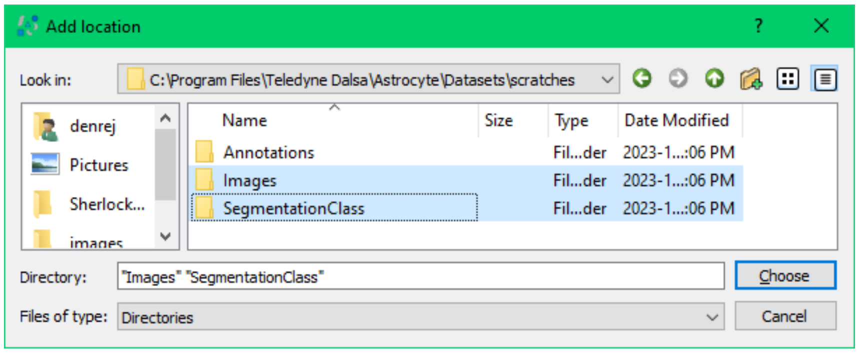
Name: Sample-Segmentation-Dataset

Description: Sample Segmentation dataset included with Astrocyte installation.

3. In the Resource Selection tab, right-click in the Locations field and select **Add location...**.



Navigate to the directories containing the image files and corresponding mask images and click **OK**.



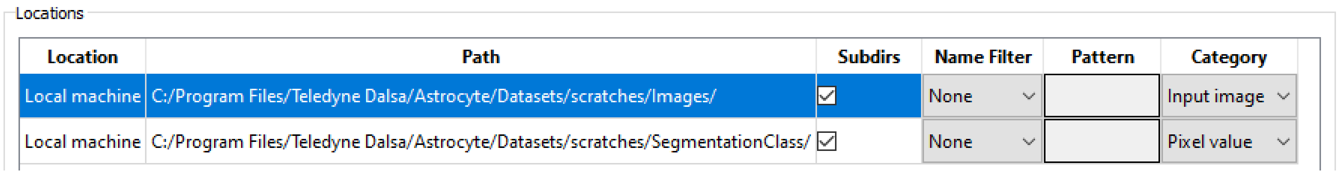
Look in: C:\Program Files\Teledyne Dalsa\Astrocyte\Datasets\scratches

Name	Size	Type	Date Modified
Annotations		Fil...der	2023-1...:06 PM
Images		Fil...der	2023-1...:06 PM
SegmentationClass		Fil...der	2023-1...:06 PM

Directory: "Images" "SegmentationClass"

Files of type: Directories

4. In the Locations section, assign the category “Input image” to the input images and “Pixel value” to the segmentation masks.


Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/scratches/Images/	☑	None		Input image
Local machine	C:/Program Files/Teledyne Dalsa/Astrocyte/Datasets/scratches/SegmentationClass/	☑	None		Pixel value

5. Click **Generate…** to prepare the dataset for the Cortex server. In the Dataset generation configuration dialog click **OK** to use the default values.

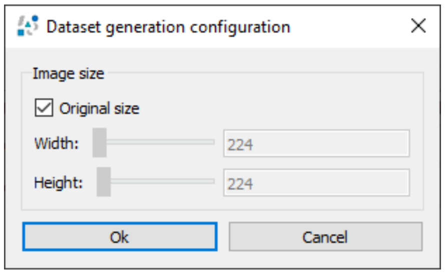

Dataset generation configuration

Image size

☑ Original size

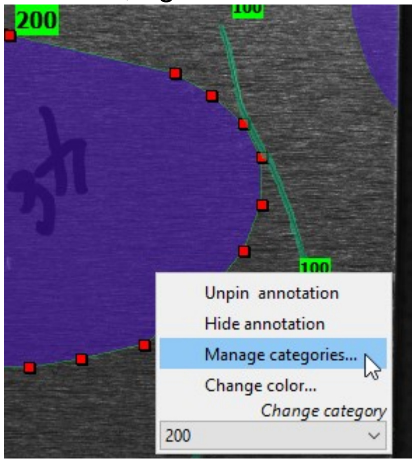
Width: 224

Height: 224

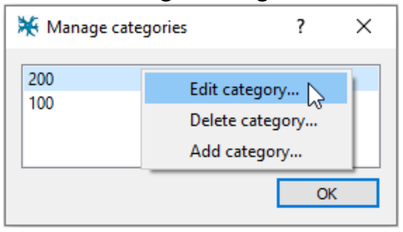
Ok · Cancel

6. Click **Save** to add the dataset to the Cortex server. If all images have the same dimensions, they are automatically resized to the specified image size on the Cortex server (the original dataset is not modified); otherwise, correct images using the Image Correction dialog.
7. Verify the dataset images and segmentation mask images and make any required changes. To easily identify pixel categories, it is typical to modify the category labels (default = mask pixel value).

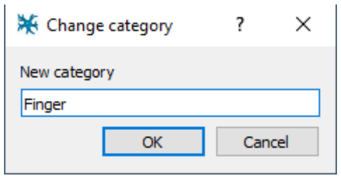
   To do so, right-click on a mask bounding box and select **Manage categories...**

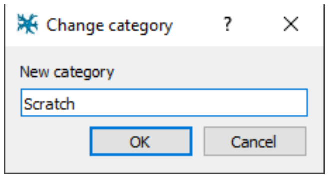


   In the Manage categories dialog, select a category and click **Edit category...**

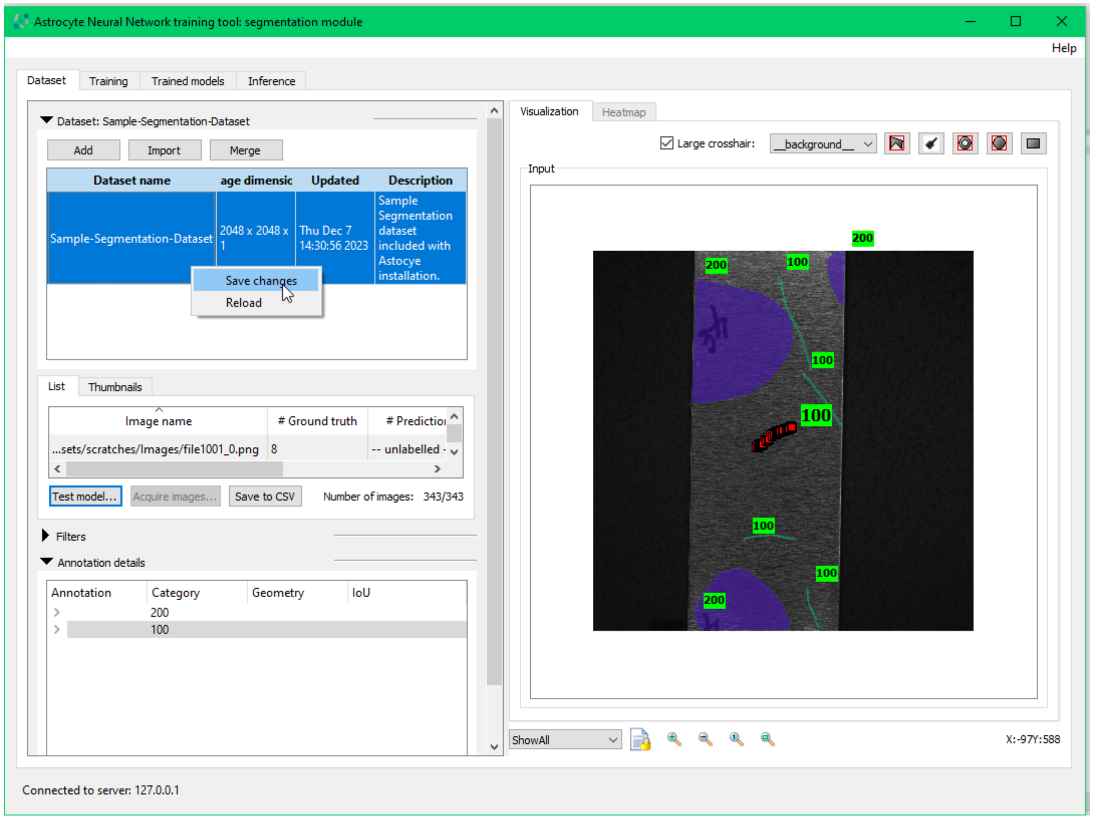


   In the Change category dialog enter the new descriptive category name (in this case, the mask identifies region defects as “Finger” and “Scratch”).





After the dataset is modified, right-click on the dataset entry and use the **Save changes** command to update and save the dataset on the Cortex server.



Dataset name	age dimensic	Updated	Description
Sample-Segmentation-Dataset	2048 x 2048 x 1	Thu Dec 7 14:30:56 2023	Sample Segmentation dataset included with Astocye installation.

Image name	# Ground truth	# Predictio
...sets/scratches/Images/file1001_0.png	8	-- unlabelled -

Number of images: 343/343

Annotation	Category	Geometry	IoU
	200		
	100		

8. In the Training tab, verify the selected dataset (the selection is made in the Dataset tab), enable the **Auto train** tab. For the example dataset, to replicate the results of the Sapera Processing sample model (and ensure that typical hardware resources are adequate given the dataset image size), set the image width and height to 1024 x 1024.

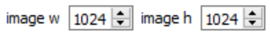
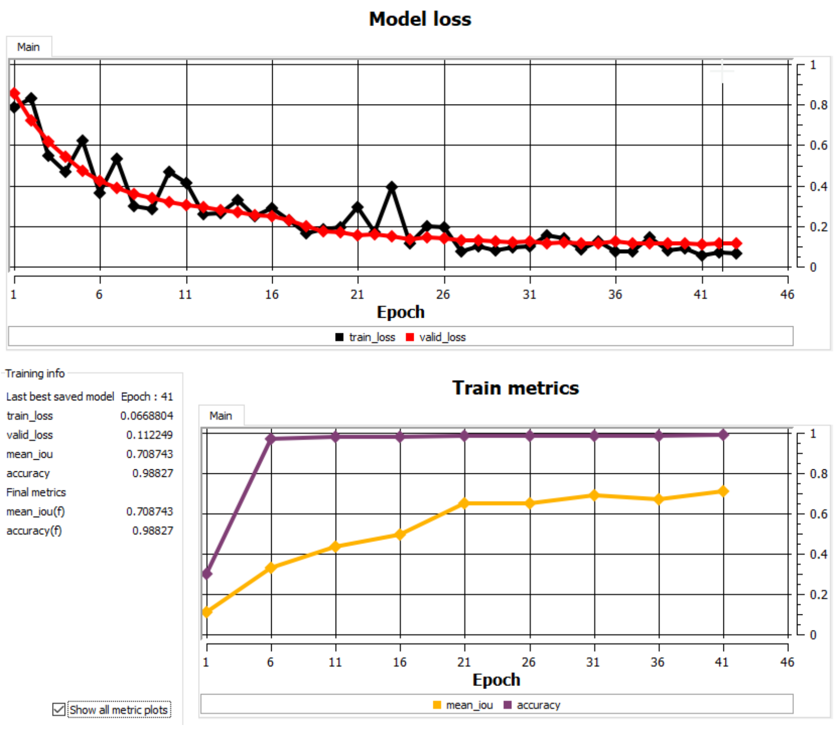


image w 1024 image h 1024

Click **Train**; the training loss graph and the metrics are displayed.



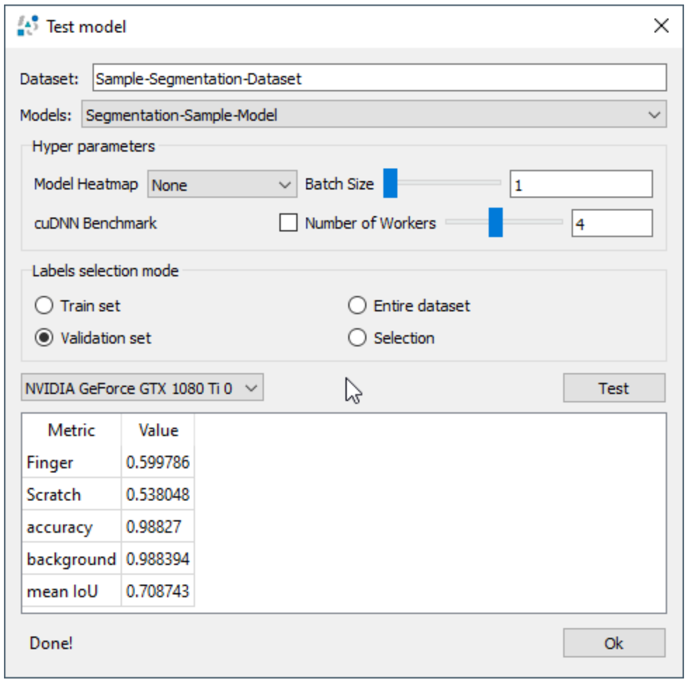
Training info

Last best saved model	Epoch : 41
train_loss	0.0668804
valid_loss	0.112249
mean_iou	0.708743
accuracy	0.98827
Final metrics	
mean_iou(f)	0.708743
accuracy(f)	0.98827

When training is complete, a prompt is displayed to save the model; click **Yes**, enter the model name and description and click **OK**.

9. In the Dataset tab, select the dataset and click **Test model...**.

In the Test model dialog, select the model in the drop-down list and click **Test**. By default, testing is performed on the validation subset of the training images. When testing is completed, the IoU (Intersection-over-Union) results for each category and the mean IoU are displayed in the Test model dialog.



Dataset: Sample-Segmentation-Dataset

Models: Segmentation-Sample-Model

Hyper parameters

Model Heatmap: None
Batch Size: 1
cuDNN Benchmark: ☐
Number of Workers: 4

Labels selection mode

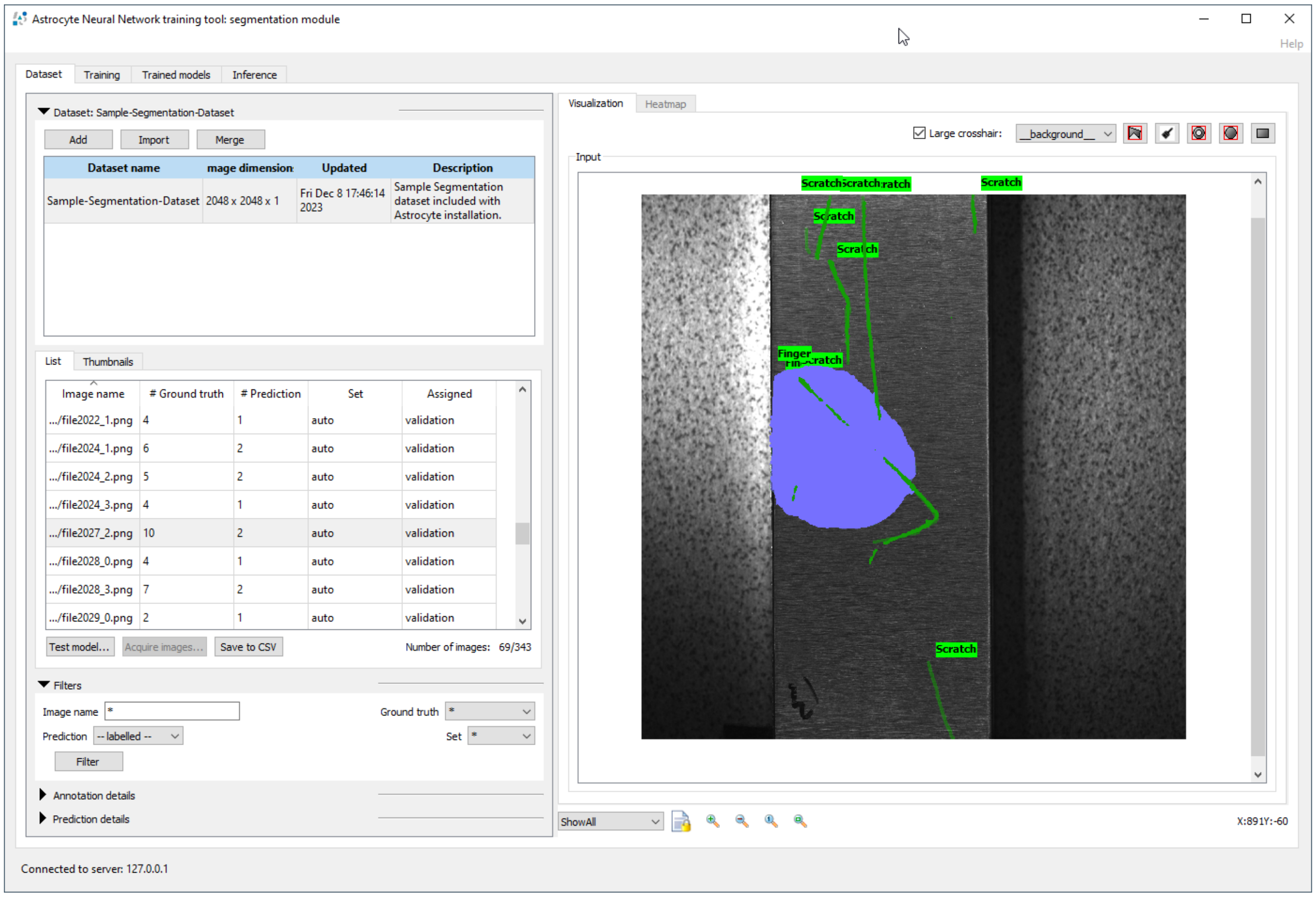
○ Train set
● Validation set
○ Entire dataset
○ Selection

NVIDIA GeForce GTX 1080 Ti 0

Metric	Value
Finger	0.599786
Scratch	0.538048
accuracy	0.98827
background	0.988394
mean IoU	0.708743

Done!

Prediction segmentation results are shown as opaque overlays, with each category identified by color, with the original ground truth shown in a transparent color.



Dataset: Sample-Segmentation-Dataset

Dataset name	mage dimension	Updated	Description
Sample-Segmentation-Dataset	2048 x 2048 x 1	Fri Dec 8 17:46:14 2023	Sample Segmentation dataset included with Astrocyte installation.

Image name	# Ground truth	# Prediction	Set	Assigned
.../file2022_1.png	4	1	auto	validation
.../file2024_1.png	6	2	auto	validation
.../file2024_2.png	5	2	auto	validation
.../file2024_3.png	4	1	auto	validation
.../file2027_2.png	10	2	auto	validation
.../file2028_0.png	4	1	auto	validation
.../file2028_3.png	7	2	auto	validation
.../file2029_0.png	2	1	auto	validation

Number of images: 69/343

Filters

Image name: *
Ground truth: *
Prediction: -- labelled --
Set: *

Large crosshair: __background__

Connected to server: 127.0.0.1

Working with Image Datasets

A well-curated image dataset is a critical component to successful model training. It must reflect the target inference images on which the model is to be deployed. The Astrocyte application is used to load user image datasets and copy them to the Cortex server; the image dataset that resides on the Cortex server is used to train models.

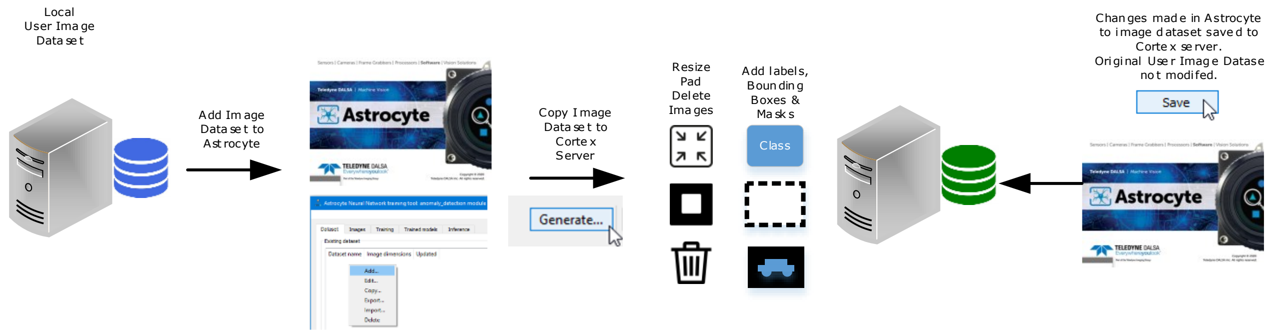

Local User Image Dataset
Add Image Dataset to Astrocyte
Astrocyte
Copy Image Dataset to Cortex Server
Generate...
Resize Pad Delete Images
Add labels, Bounding Boxes & Masks
Class
Changes made in Astrocyte to image dataset saved to Cortex server. Original User Image Datase not modifed.
Save
Astrocyte

Image datasets that reside on the Cortex server are only accessible through the Astrocyte application; datasets can be exported from the Cortex server, using Astrocyte, to be imported by other instances of Astrocyte applications and added to other instances of Cortex servers.

Exported datasets are only accessibly through the Astrocyte application; original user image datasets are never modified.

	Note: To undo any saved changes to an image dataset that resides on a Cortex server, in Astrocyte, open the dataset in the Dataset tab using the Edit pop-up menu command and regenerate the original image dataset by clicking Generate. This restores the original image dataset from the user location on the Cortex server.

Astrocyte supports most image formats (8, 10, 12, 16-bit). All images must have the same number of channels (either all monochrome or all color).

When training, the **Region size** determines the input image size to the neural network. This allows a single high-resolution dataset to be trained on lower levels of resolution to determine the optimal input size without having to create multiple datasets.

Guidelines on Preparing Dataset

The following are general guidelines to prepare a dataset in order to achieve good model performance.

Dataset Size (number of images)

For a specific dataset size, if the training loss cannot be decreased to near zero, the model is considered to have high avoidable bias (underfitting). In this scenario, adding more image samples will not solve the problem. If the model performs well on the training set (low training loss) but not on the validation set (high validation loss), then it has a high variance (overfitting) and adding more data will help overcome that problem. Before adding more image samples (usually a labor-intensive expensive task), it is a best practice to make sure the model can fit the training samples well by checking the training loss.

Image Resolution

Increasing image resolution at training time has the potential of increasing neural network performance for various machine-learning tasks. Usually, the rule is “the higher resolution the better”.

However, although it may help to have higher resolution images, model performance is highly dependent on the feature characteristics and the dataset distribution. If the features are few and clear even a low-resolution image will convey those features correctly to the model and increasing resolution will not change the model performance but may result in slower training.

In general, it is recommended to try smaller model architectures first and evaluate performance before trying larger architectures or increasing input image resolution.

Image Content

The images of the training set should contain the details that are trying to be identified (objects of interest, defects, and so forth). Also, the training set should cover the distribution of the problem to learn (for example, there should be enough variation on the objects of interest and the background so the algorithm can generalize.) A good rule to follow is to train on images taken in the same environment as the final setup.

Adding an Image Dataset to the Cortex Server

When adding an image dataset to the Cortex server, image files are copied from the user-defined location (for example, a local disk) to the Cortex server after the Generate operation is executed and the dataset is saved.

Note: Subsequent modifications to the dataset from the Astrocyte Dataset tab, when saved, only apply to the image dataset that resides on the Cortex server; the image dataset in the originally designated user-defined location is not modified.

When generating the dataset, a configuration dialog allows the original size of the images to be used, or images can be resized; the width and height setting determines the maximum size of images used as the input size for training; larger images are resized when the dataset is generated and added to the Cortex server (the original files are not modified).

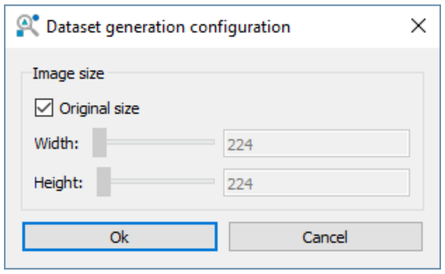

Dataset generation configuration

Image size

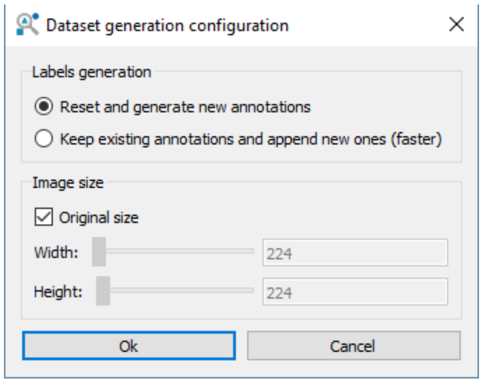
[✓] Original size

Width: 224

Height: 224

When using the original size option, if the dataset contains different image sizes, when saving the image dataset to the Cortex server the Image correction dialog is presented to adjust images that differ from the most common size in the dataset; refer to the Adjusting Image Sizes section.

When editing or adding images to an existing dataset, label generation options are provided to reset and generate new annotations or keep existing annotation and append new ones (faster).


Dataset generation configuration

Labels generation

[● Reset and generate new annotations]

○ Keep existing annotations and append new ones (faster)

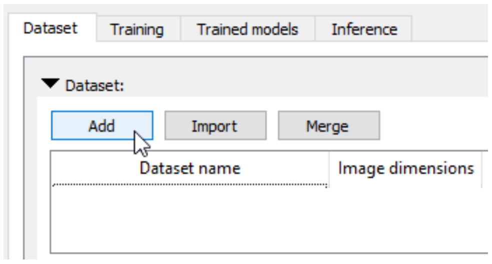
Image size

[✓] Original size

Width: 224

Height: 224

Astrocyte can search directories for images by file type and using regular expressions within filenames. To add a dataset, in the Dataset tab, click **Add**:

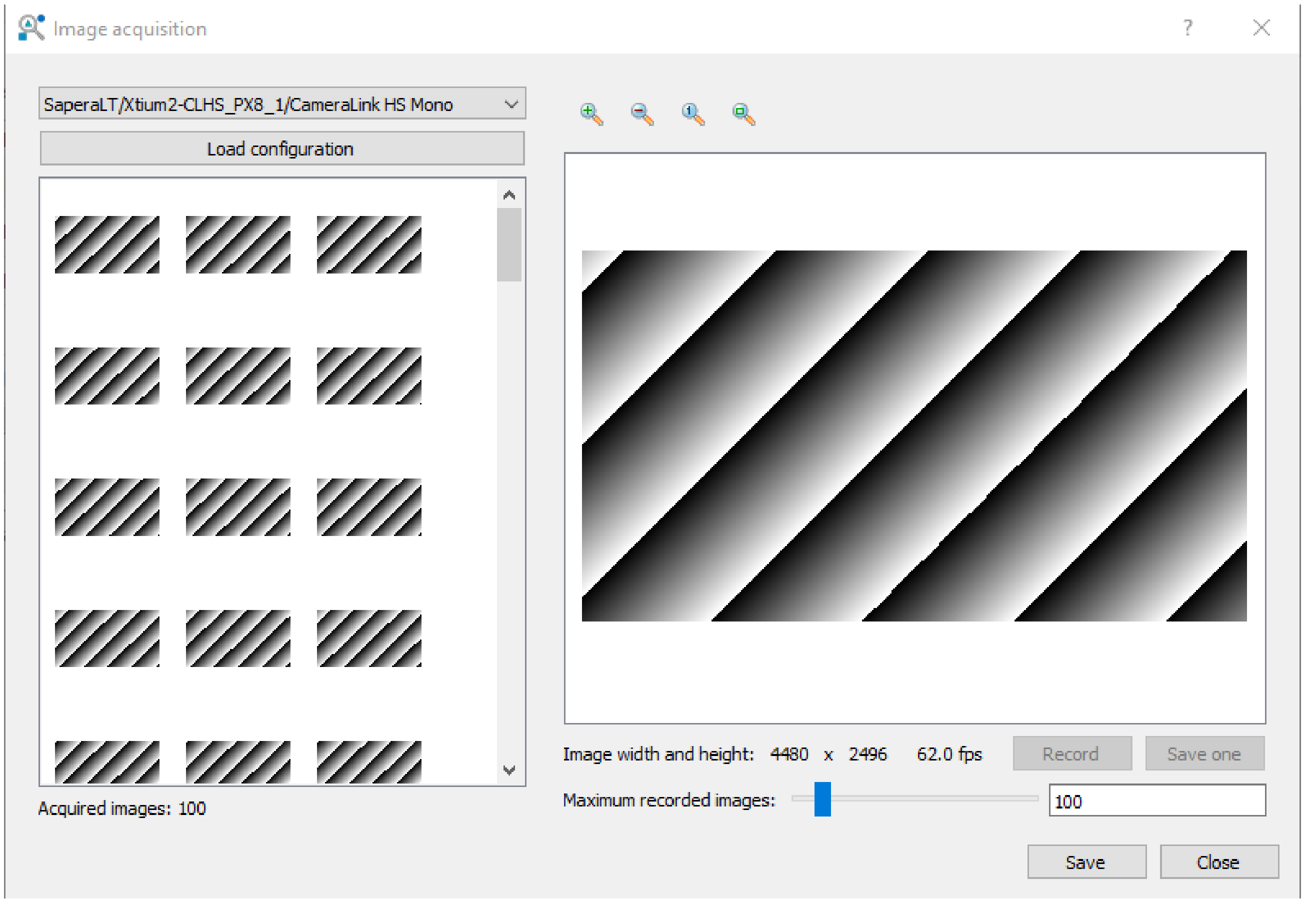


The Create dataset dialog provides fields to enter the dataset name and description. The optional description field can add further information regarding the dataset, as required.

Acquiring Images from Cameras and Frame Grabbers

Astrocyte supports acquisition of images from cameras and frame grabbers supported by Teledyne DALSA's Sapera LT platform, including GigE and USB3 cameras.

The image acquisition dialog displays the live acquisition, as well as the current frame rate. The image display can be adjusted using the zoom icons or using standard Windows controls and the mouse (for example, CTRL + mouse wheel).

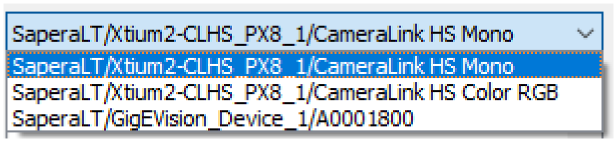


Acquired images: 100

Image width and height: 4480 x 2496 62.0 fps

Maximum recorded images: 100

If more than one device is available, use the drop-down list to select a device:



SaperaLT/Xtium2-CLHS_PX8_1/CameraLink HS Mono
SaperaLT/Xtium2-CLHS_PX8_1/CameraLink HS Mono
SaperaLT/Xtium2-CLHS_PX8_1/CameraLink HS Color RGB
SaperaLT/GigEVision_Device_1/A0001800

If necessary, adjust the camera and/or frame grabber parameters, such as exposure time and frame rate, using Sapera LT's CamExpert tool, so that the imaging setup is satisfactory and reflects actual operating conditions; refer the Sapera LT documentation for more information.

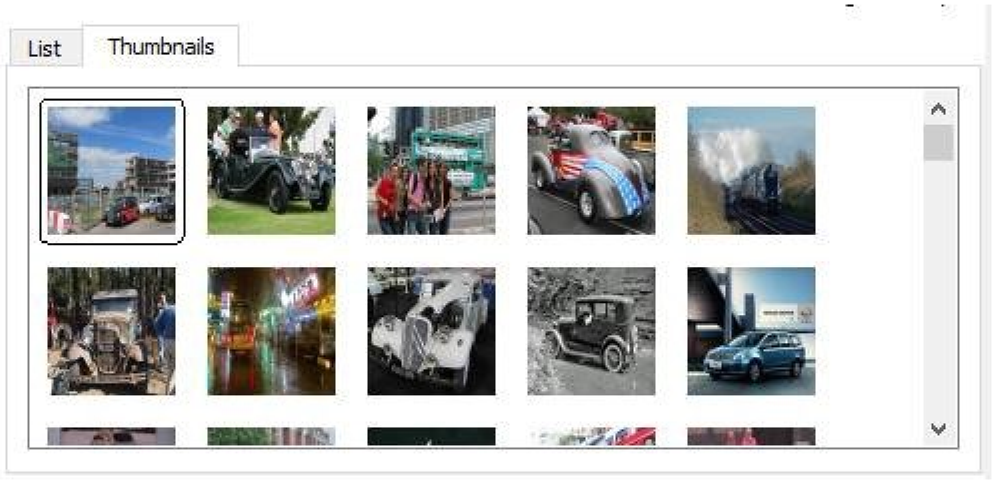
Single images or a sequence of images (maximum 1000) can be acquired using the **Save one** or **Record** buttons; any image resolution can be acquired.

Thumbnails of the acquired images are displayed. If the images are satisfactory, click **Save** and use the standard Windows dialog to select a directory location; these images can now be loaded into a new or existing Astrocyte dataset.

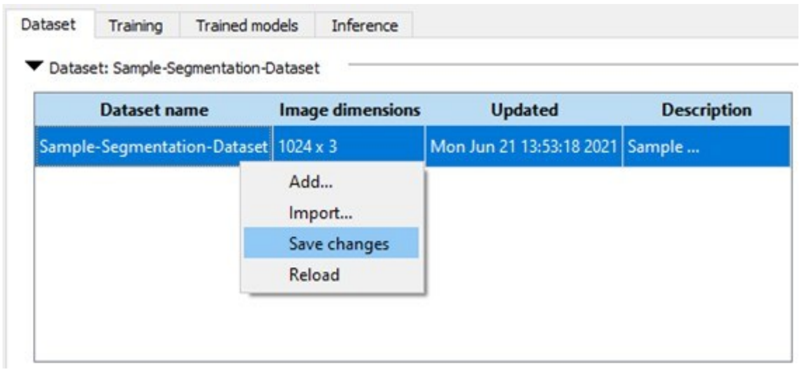
Working with Images in the Dataset Tab

The Dataset tab provides many functions to manipulate image datasets on the Cortex server. These various functions allow category labels to be verified, and if necessary modified, or images to be removed from the dataset altogether. Astrocyte also supports drag-and-drop of images from Windows File Explorer.

The Thumbnails tab shows the image thumbnails for quick reference compared to the List view.

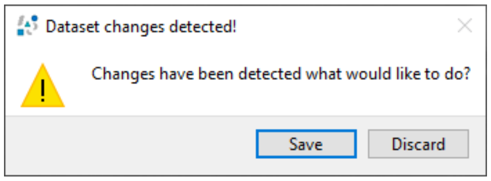


Any changes or modifications to the dataset can be applied to the dataset by right-clicking on the dataset entry and using the **Save changes** command; to discard or undo any changes that have been made, but not yet applied, use the **Reload** command.



Dataset name	Image dimensions	Updated	Description
Sample-Segmentation-Dataset	1024 x 3	Mon Jun 21 13:53:18 2021	Sample ...

When navigating away from the Dataset tab or closing the application, if any unsaved changes to the dataset are detected a prompt appears to choose to save or discard changes.


Dataset changes detected!

Changes have been detected what would like to do?

Types of Datasets

Two types of datasets are available:

- **Named**: images are not annotated; category labels are applied individually to each image or can be batch assigned using the folder label. Named sets are available on all modules. When using named sets for object detection and segmentation, graphical annotations (such as bounding boxes, polygons, masks) and labels can be manually added in the right panel of the Dataset tab.
- **Database**: image annotations are contained in separate text files or XML files. Database datasets are used with the object detection and segmentation modules. For example, bounding boxes written as numbers in a text file. Databases can follow a standard (for example, PASCAL VOC) or custom format.

A single dataset can have multiple sets of different databases of different type (named or database). For example, when using object detection with Self-Learning, a named set can be added to the dataset that contains unlabelled images and a database set with an annotation file identifying bounding boxes and labels; when generating the dataset images from both sets are added to the dataset and copied to the Cortex server.

Named sets

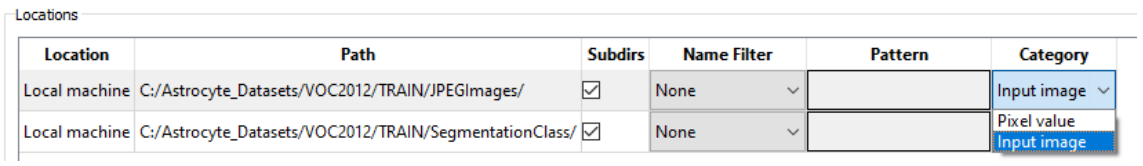
Named sets are typically used for training classification and anomaly detection models where each image is associated with a single category label.

For the object detection module, named sets are only used for image datasets that do not have associated annotations; bounding boxes and labels are to be manually added in the Dataset tab after adding the dataset.

For the segmentation module, named sets are used for image datasets that have corresponding mask files, but do not have associated annotation files; Astrocyte automatically generates bounding polygons based on the mask images.

Segmentation Named Sets

Segmentation requires a set of mask images be provided for each corresponding image, with each pixel value representing an object category. Use the Category label drop-down list to designate the image folder as containing either the Input images or Pixel value (that is, the mask images).


Locations

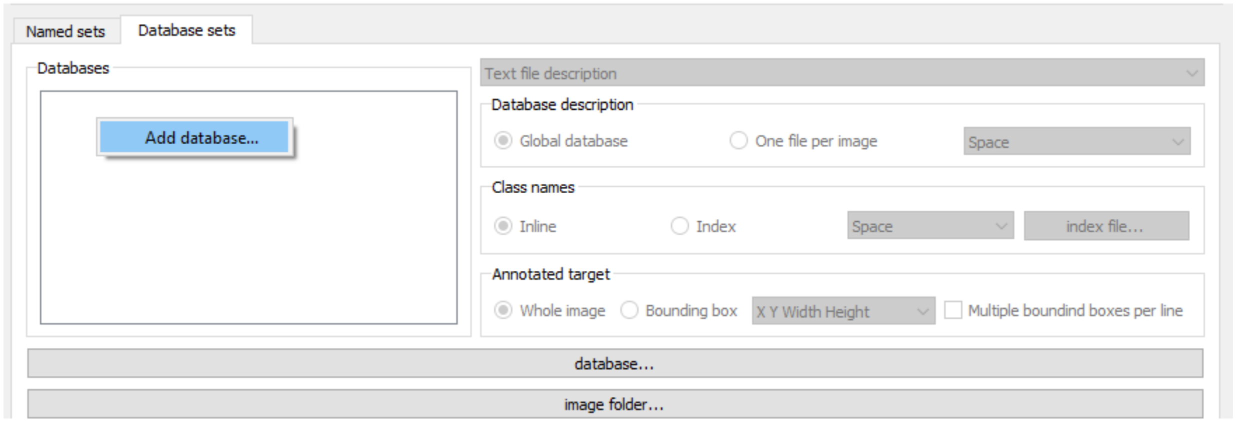
Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Astrocyte_Datasets/VOC2012/TRAIN/JPEGImages/	✓	None		Input image
Local machine	C:/Astrocyte_Datasets/VOC2012/TRAIN/SegmentationClass/	✓	None		Pixel value Input image

Typically, to easily identify the category associated with a pixel value, descriptive labels are added; see the [Adding, Editing or Deleting Category Labels]() section.

Database Sets

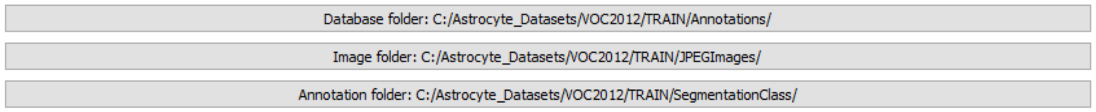
Database sets for the Object Detection and Segmentation modules include both images and annotation file(s) describing the bounding boxes or image masks and associated category labels in each image.

In the Create dataset dialog, right-click in the Databases panel and select Add database... to define a new database of images and annotations.



Use the Database... and Image folder... buttons to specify the path to the required database annotation file and associated images. The currently assigned path is then displayed in the button field.

For the Segmentation module an additional database folder is required containing the mask files corresponding to the images in the dataset.



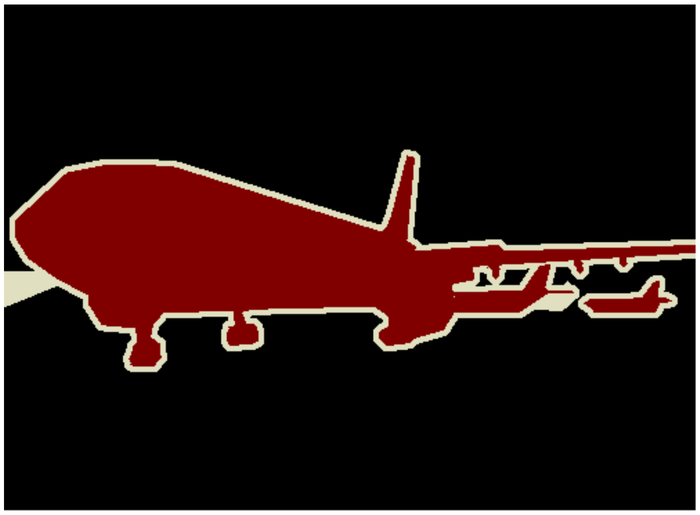
Database folder: C:/Astrocyte_Datasets/VOC2012/TRAIN/Annotations/

Image folder: C:/Astrocyte_Datasets/VOC2012/TRAIN/JPEGImages/

Annotation folder: C:/Astrocyte_Datasets/VOC2012/TRAIN/SegmentationClass/

Mask Files

For segmentation, mask files use specific pixel values to denote category labels of objects in the corresponding images, with each pixel value denoting a category label and a single value designated for background pixels. For example, the following is a mask image taken from the PASCAL VOC dataset.



Training, Validation and Test Image Datasets

In general, for model training, three sets images are used:

- **Training**: The training set is the collection of images on which the model learns.
- **Validation**: The validation set is a subset of the training set which is used during training to assess the performance of the model training. The model is not trained on the validation set; at the end of an epoch, the model is tested against the validation set to verify how well the model performs on data it has not seen.
- **Test**: The test set is a collection of images on which to test the performance of the trained model. It should not contain any images from the training set.

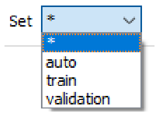
The number of images for training, validation and test is arbitrary, however the greater the number of images the more robust the model. In general, it is recommended to train on the highest number of images possible. For example, the MNIST database of handwritten digits contains 70000 images (60000 training set (including a percentage as validation) and 10000 test set).

Before training verify that the bounding boxes and label annotations are valid and correctly assigned. Wrongly labelled or irrelevant images can corrupt the training process and affect model performance.

Training and Validation Image Sets

Astrocyte uses a percentage of the training set to create a randomly selected validation set that is used during the training session. Images can be assigned explicitly to the training set or validation set; remaining images for the validation are assigned automatically by the Astrocyte trainer.

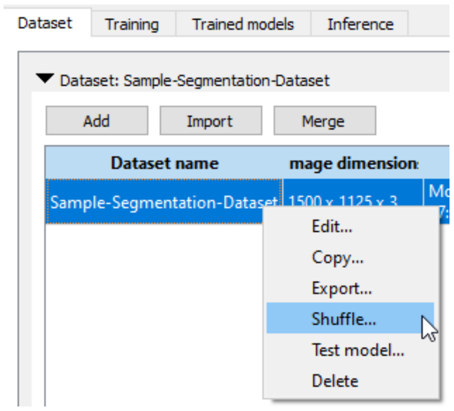
In the Dataset tab, images can be specifically assigned to the training or validation set as required using the Set filter drop-down list. By default, all images are assigned automatically to either the train or validation set according to the specified ratio.

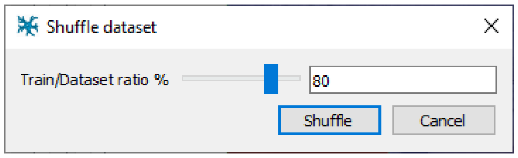


Shuffle Command

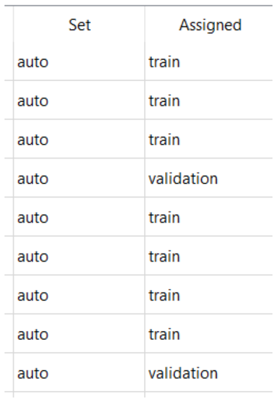
The shuffling operation consists in randomly allocating a predefined percentage of images to the training set and the rest to the validation set. This allocation is automatically performed once the dataset is created for the first time. To redo the allocation use the Shuffle command. Note that the random selection only applies to images set in the “Auto” mode.

In the Dataset tab, right-click on the selected dataset, select **Shuffle…** and use the **Train/Dataset ratio %** slider to set the percentage of images to assign to the Train portion (and inversely, the Validation portion). After the shuffle operation has been applied the allocation results can be viewed in the Assigned column of the dataset image list.





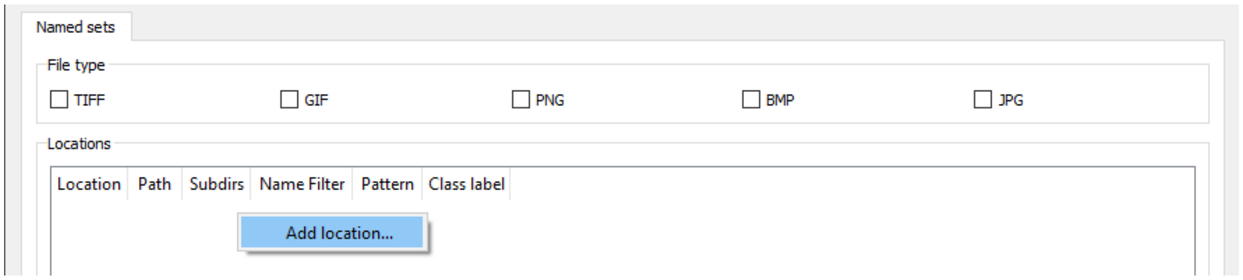
Train/Dataset ratio % 80



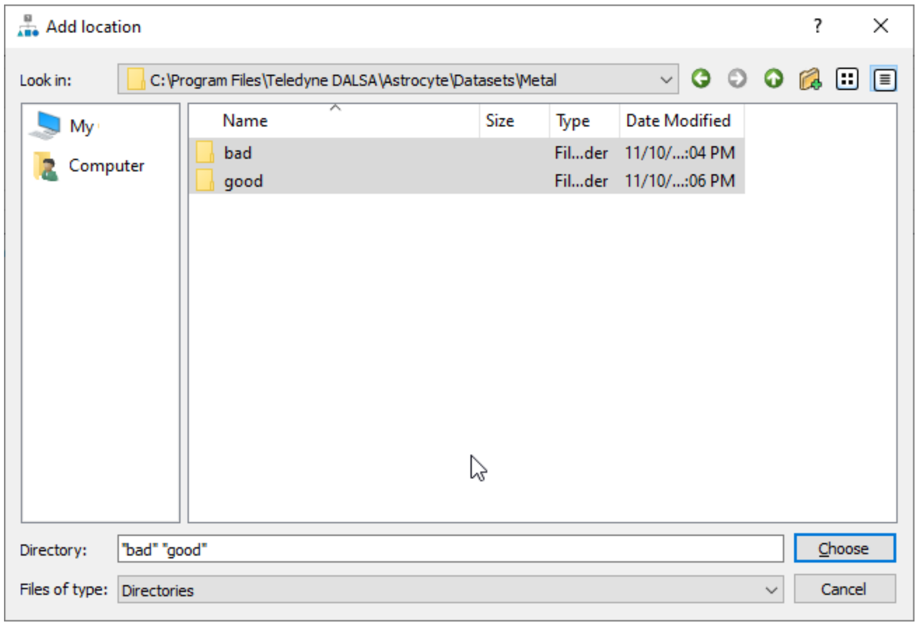
Set	Assigned
auto	train
auto	train
auto	train
auto	validation
auto	train
auto	train
auto	train
auto	train
auto	validation

Adding Image Directories to a Dataset

To add images to a dataset, right-click in the Locations panel and click **Add location...**. Use the file type checkboxes to filter image files by file type in the selected directories; by default Astrocyte adds all image file types without having to check these boxes.



Multiple directories can be selected in the Add location dialog using the standard Windows key combinations (for example, <SHIFT> or <CTRL> + select).



Look in: C:\Program Files\Teledyne DALSA\Astrocyte\Datasets\Metal

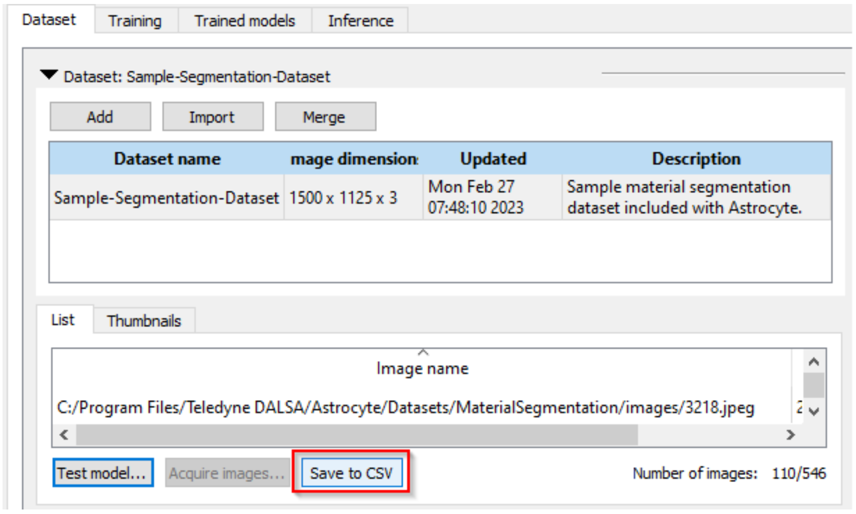
Name	Size	Type	Date Modified
bad		Fil...der	11/10/...:04 PM
good		Fil...der	11/10/...:06 PM

Directory: "bad" "good"

Files of type: Directories

Saving Dataset Information to CSV File

Dataset information can be saved to a Microsoft Excel file (*.csv*) by clicking the Save to CSV button.



Dataset name	mage dimension	Updated	Description
Sample-Segmentation-Dataset	1500 x 1125 x 3	Mon Feb 27 07:48:10 2023	Sample material segmentation dataset included with Astrocyte.

Image name
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/3218.jpeg

Number of images: 110/546

The file includes all columns: Image name, # Ground truth, # Prediction, Set and Assigned.

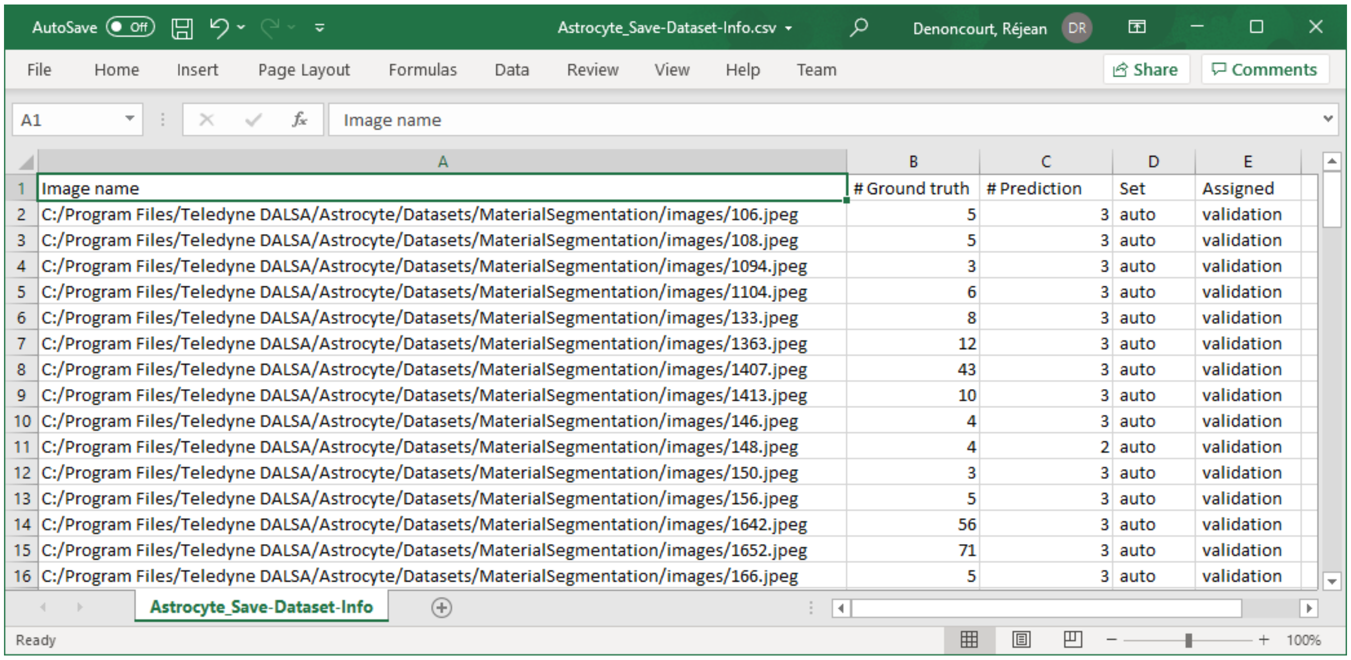
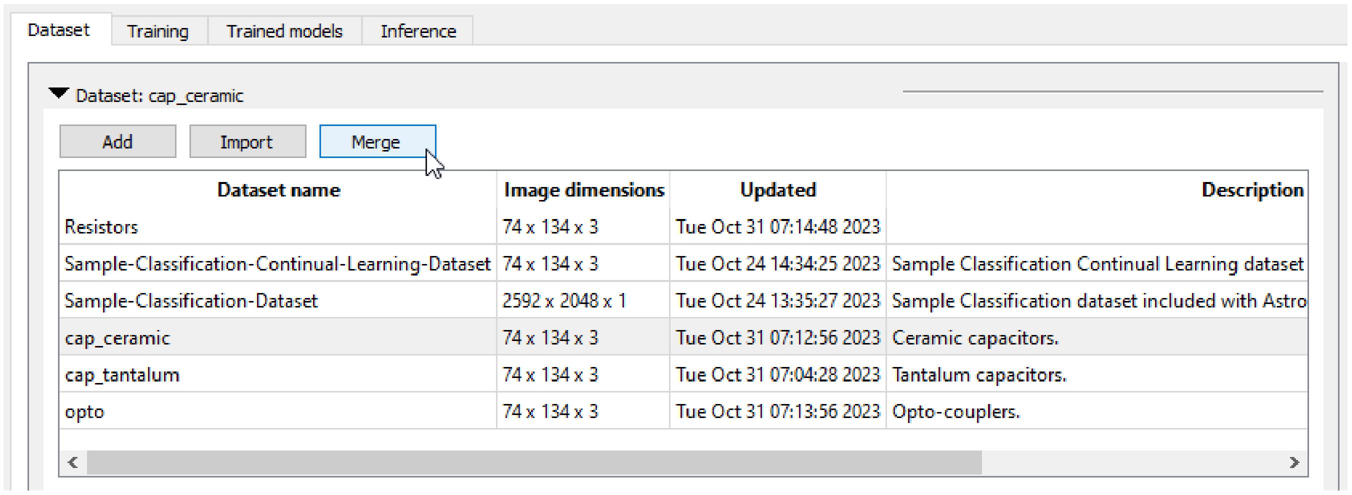


Image name	# Ground truth	# Prediction	Set	Assigned
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/106.jpeg	5	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/108.jpeg	5	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1094.jpeg	3	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1104.jpeg	6	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/133.jpeg	8	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1363.jpeg	12	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1407.jpeg	43	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1413.jpeg	10	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/146.jpeg	4	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/148.jpeg	4	2	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/150.jpeg	3	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/156.jpeg	5	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1642.jpeg	56	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/1652.jpeg	71	3	auto	validation
C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/MaterialSegmentation/images/166.jpeg	5	3	auto	validation

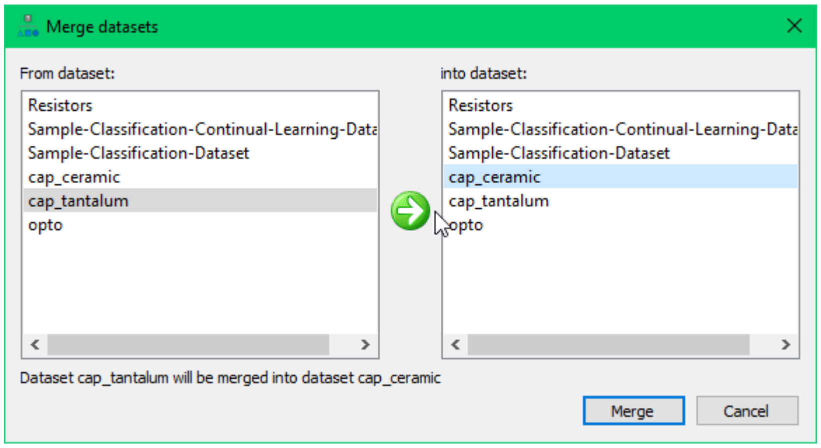
Merging Datasets

Astrocyte datasets can be merged together to create a single larger dataset. This may be useful as new types or more training images become available. Only datasets with the same image size can merged.



Dataset name	Image dimensions	Updated	Description
Resistors	74 x 134 x 3	Tue Oct 31 07:14:48 2023	
Sample-Classification-Continual-Learning-Dataset	74 x 134 x 3	Tue Oct 24 14:34:25 2023	Sample Classification Continual Learning dataset
Sample-Classification-Dataset	2592 x 2048 x 1	Tue Oct 24 13:35:27 2023	Sample Classification dataset included with Astro
cap_ceramic	74 x 134 x 3	Tue Oct 31 07:12:56 2023	Ceramic capacitors.
cap_tantalum	74 x 134 x 3	Tue Oct 31 07:04:28 2023	Tantalum capacitors.
opto	74 x 134 x 3	Tue Oct 31 07:13:56 2023	Opto-couplers.

Clicking the Merge button opens the Merge datasets dialog:

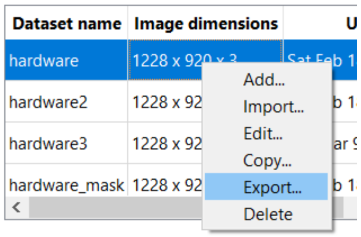

Merge datasets
From dataset:
Resistors
Sample-Classification-Continual-Learning-Data
Sample-Classification-Dataset
cap_ceramic
cap_tantalum
opto
into dataset:
Resistors
Sample-Classification-Continual-Learning-Data
Sample-Classification-Dataset
cap_ceramic
cap_tantalum
opto
Merge
Cancel

Dataset cap_tantalum will be merged into dataset cap_ceramic

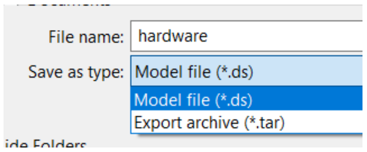
As a precaution, it is recommended to make a copy of the dataset that is being merged into since the operation cannot be undone and a return to the original dataset is needed.

Importing and Exporting a Dataset

Datasets can be exported to files. This is useful to transfer datasets from one system to another. To export a dataset right-click on the required dataset from the dataset list and select **Export…**



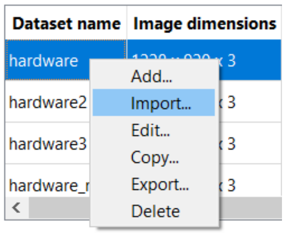
Name the dataset and select the file format (.ds or .tar).


File name: hardware

Save as type: Model file (*.ds)

- **.ds format:** binary proprietary format. It is the preferred format for transferring a dataset to another system (especially because it is the only format that can be imported into Astrocyte).
- **.tar format:** zip package including images and annotations in PASCAL VOC standard. Use this format to merge two datasets together. The .tar can be unzipped to extract images and annotations.

To import a dataset (**.ds** format only) right-click anywhere in the dataset list and select **Import…**

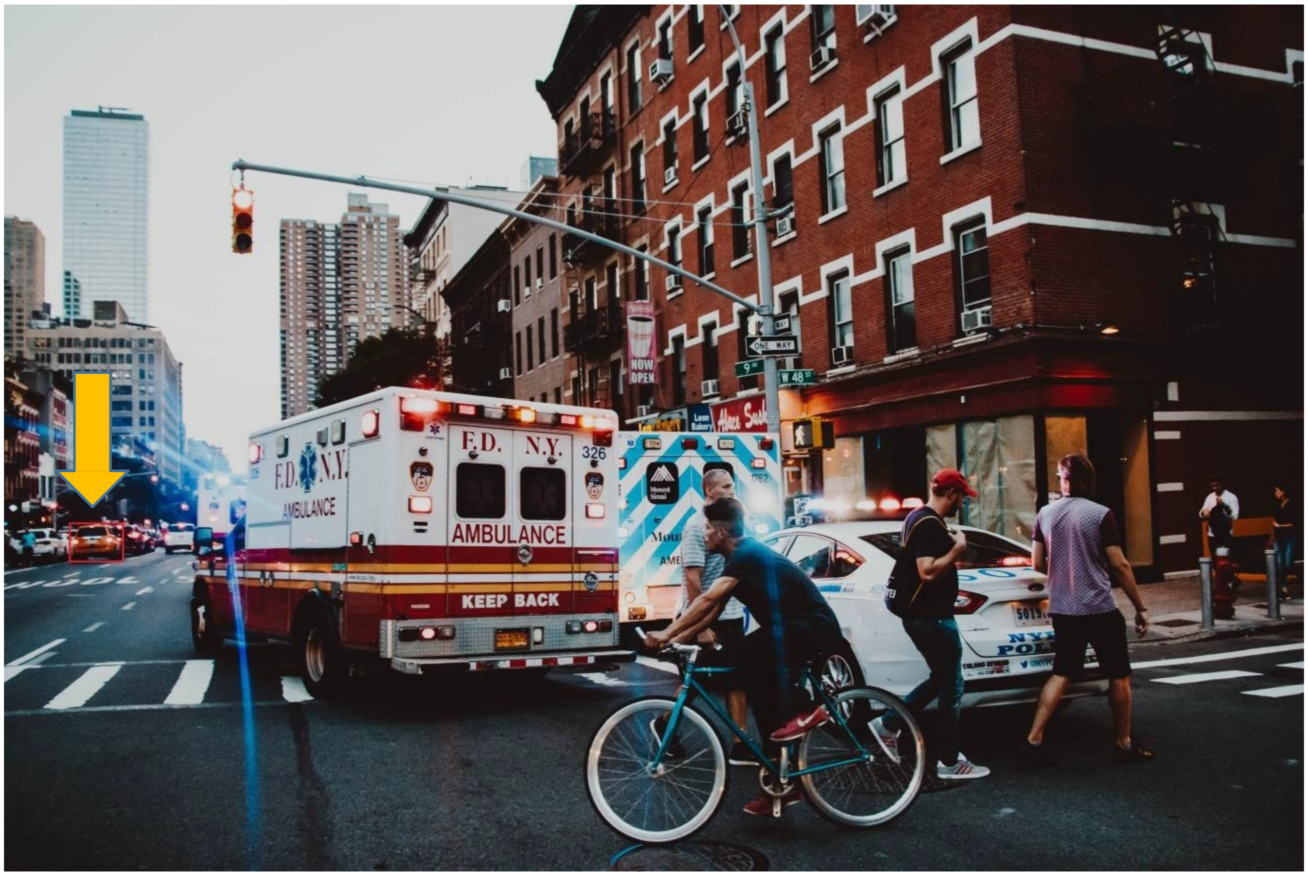


Browse for the **.ds** file and select it, then give the dataset a name and the dataset will be imported and added to the dataset list. You can immediately use it for training.

Image Size Considerations

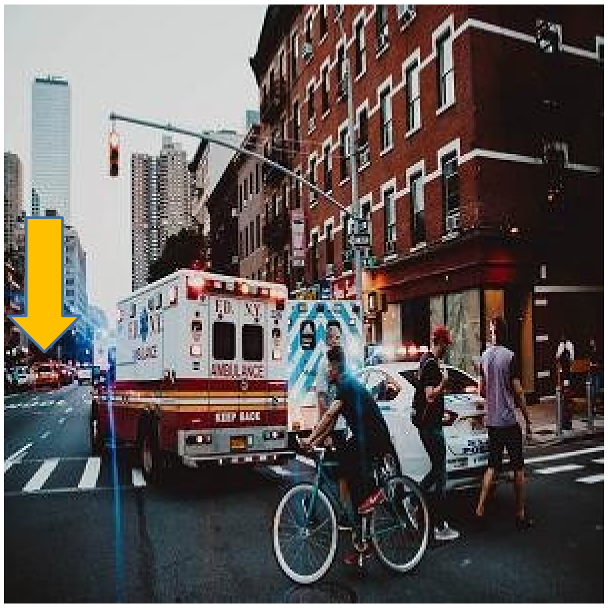
Astrocyte supports any input image size. Higher resolution images require greater training and processing time during inference. Depending on the module and the target features to identify, higher resolution images may not provide any significant benefit in terms of actual performance. However, if very small details are necessary to identify these may be lost if images are decimated to smaller sizes; in this case, higher resolution images may be required.

For example, consider the bounding box shown in this high-resolution image (5472x3648):

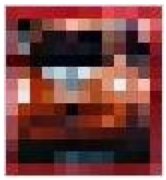


The actual level of detail in the high-resolution image is quite good for the relatively small bounding box in this example (232x169):

However, when the image is resampled and resized to 300x300 pixels, most details are lost.



The bounding box image is now only 13x14 pixels with a very low detail level:

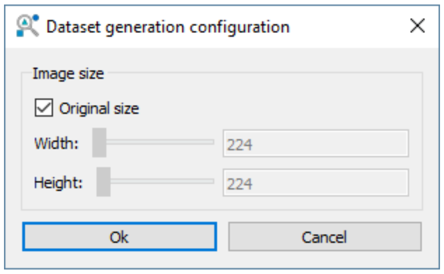


For Anomaly Detection, Object-Detection and Segmentation models, datasets with high-resolution images with relatively small target objects to detect can improve results by using the "Tiling" option during training.

The maximum input size for model training is not limited except for the object detection module; YOLOX architectures support non- square input sizes greater than 224x224, in increments of 32 pixels.

Adjusting Image Sizes

After clicking **Generate…**the maximum input image size (height or width) is determined using the settings presented in the **Dataset generation configuration** dialog.



Original size: ✓
Width: 224
Height: 224

After clicking **Save** to copy dataset to the Cortex server Astrocyte presents, when using the "Original size" option, the Image Correction dialog if images in the dataset have differing dimensions compared to the majority of images in the dataset.

If the dataset contains images with different numbers of color channels, the number of channels used by the majority of images is taken and images with a different number of channels are converted.

Image Correction

The image correction options include:

- Exclude from dataset
- Resize ignoring aspect ratio
- Resize keeping aspect ratio
- Crop

Note: The crop and resize keeping aspect ratio image correction methods can introduce padding pixels depending on the original image size and the dataset image size.

Multiple images can be selected in the image panel using the common windows key combinations (CTRL or shift); right-click on the selections to choose an operation to apply.

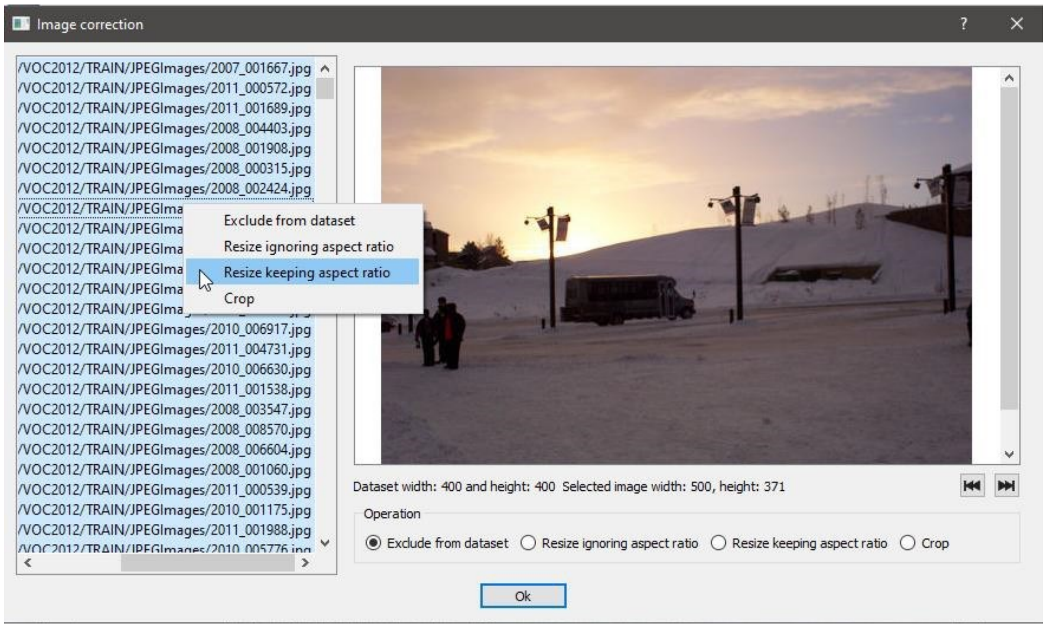
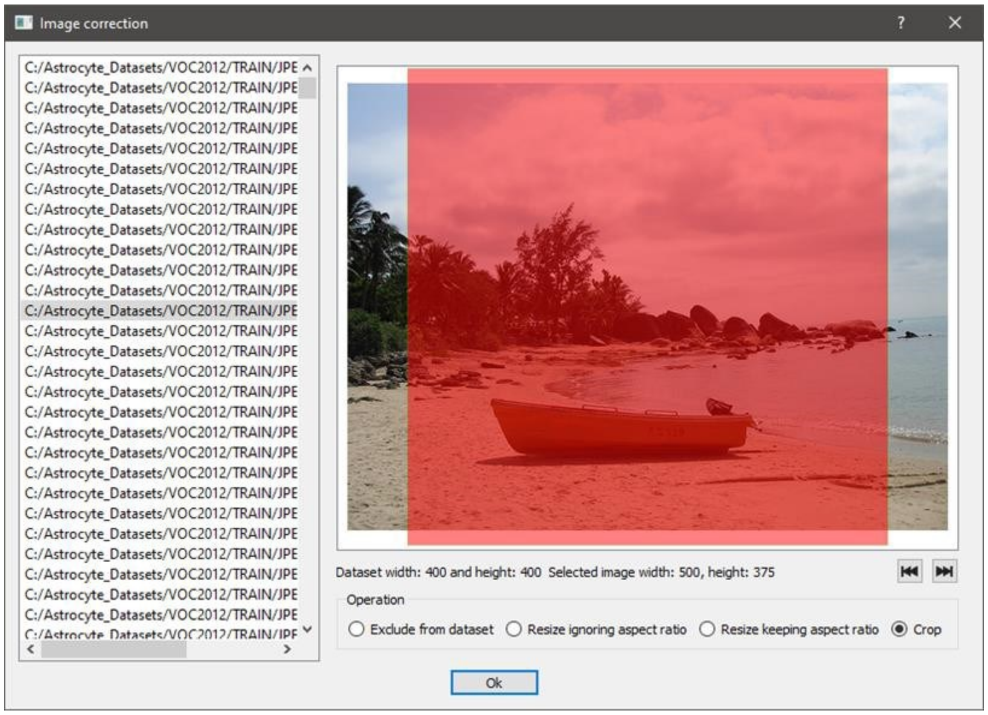


Image correction
/VOC2012/TRAIN/JPEGImages/2007_001667.jpg
/VOC2012/TRAIN/JPEGImages/2011_000572.jpg
/VOC2012/TRAIN/JPEGImages/2011_001689.jpg
/VOC2012/TRAIN/JPEGImages/2008_004403.jpg
/VOC2012/TRAIN/JPEGImages/2008_001908.jpg
/VOC2012/TRAIN/JPEGImages/2008_000315.jpg
/VOC2012/TRAIN/JPEGImages/2008_002424.jpg
/VOC2012/TRAIN/JPEGImages/2010_006917.jpg
/VOC2012/TRAIN/JPEGImages/2011_004731.jpg
/VOC2012/TRAIN/JPEGImages/2010_006630.jpg
/VOC2012/TRAIN/JPEGImages/2011_001538.jpg
/VOC2012/TRAIN/JPEGImages/2008_003547.jpg
/VOC2012/TRAIN/JPEGImages/2008_008570.jpg
/VOC2012/TRAIN/JPEGImages/2008_006604.jpg
/VOC2012/TRAIN/JPEGImages/2008_001060.jpg
/VOC2012/TRAIN/JPEGImages/2011_000539.jpg
/VOC2012/TRAIN/JPEGImages/2010_001175.jpg
/VOC2012/TRAIN/JPEGImages/2011_001988.jpg
Exclude from dataset
Resize ignoring aspect ratio
Resize keeping aspect ratio
Crop
Ok

Dataset width: 400 and height: 400 Selected image width: 500, height: 371

Operation: [✓ Exclude from dataset] Resize ignoring aspect ratio Resize keeping aspect ratio Crop

Alternatively, images can be selected individually and a specific operation applied using the radio buttons. Images that have been designated for correction are highlighted in transparent red.



Dataset width: 400 and height: 400 Selected image width: 500, height: 375

Operation: Exclude from dataset Resize ignoring aspect ratio Resize keeping aspect ratio [✓ Crop]

When selection of correction operations for images is complete, click OK to apply the changes to the dataset (original images are not modified; only those that reside on the Cortex server are affected).

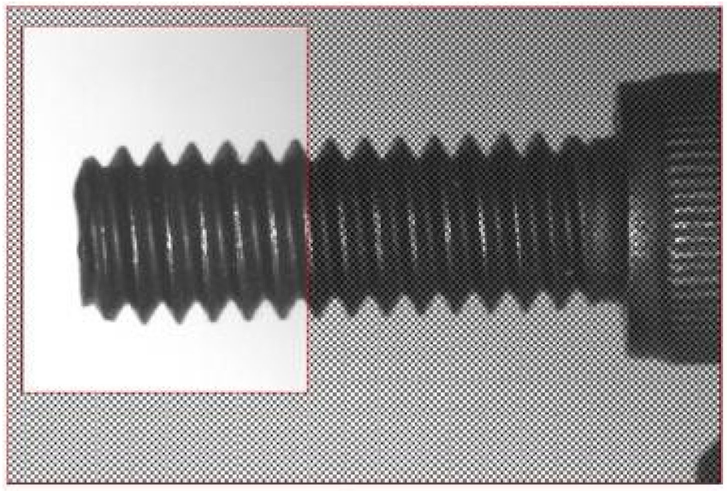
	Note: Images not specifically assigned an operation are by default excluded from the dataset.

Input Image Cropping

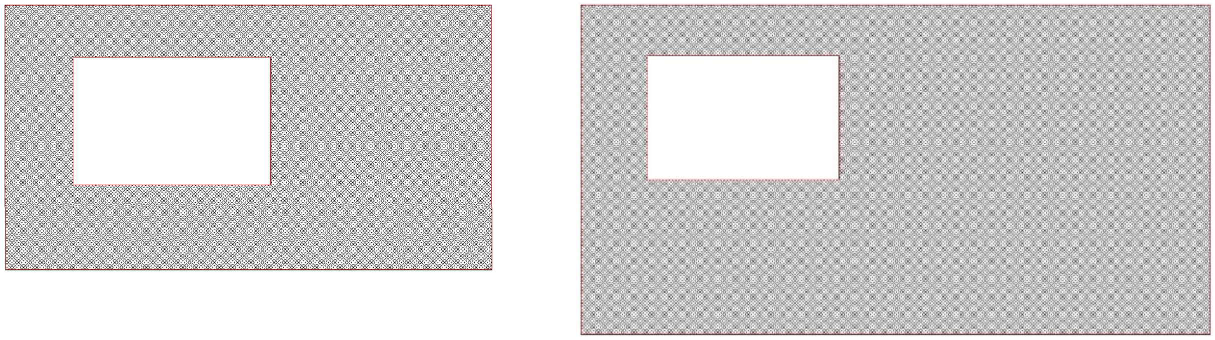
Astrocyte automatically crops the input image to reduce size if ROIs are such that it allows a smaller image width and height. That is, if outer image regions are masked (excluded/ignored). This in turn reduces the training time required to create a model.

Only regions defined as unmasked (“background” or active regions) are presented to the model for training are processed; masked regions are ignored.

For example, the images in the sample screw dataset (included with Astrocyte) are 768 x 512. The region to be processed defined by the ROI is 306 x 392.



At inference, the position of the ROI relative to the top-left corner of the image is always used, regardless of the size of the input image.

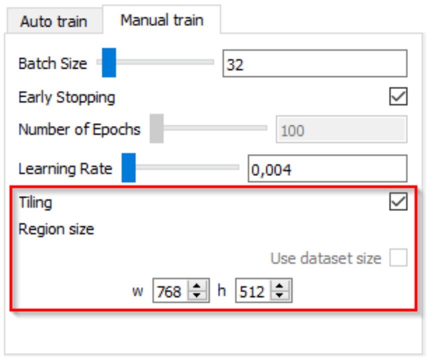


Tiling

When it is necessary to use high-resolution images, tiling maintains the original image resolution, dividing the image into “tiles” that correspond to the selected architecture’s input size. In this manner no image data is lost by sub-sampling or decimating the image, which can cause small details in larger images to be flattened or effaced.

Tiling is supported by the anomaly detection, object detection and segmentation modules.

Tile size can be specified at training by enabling the Tiling checkbox (available in the Hyper parameters section in Auto-Train or Manual tab) and specifying the region size:



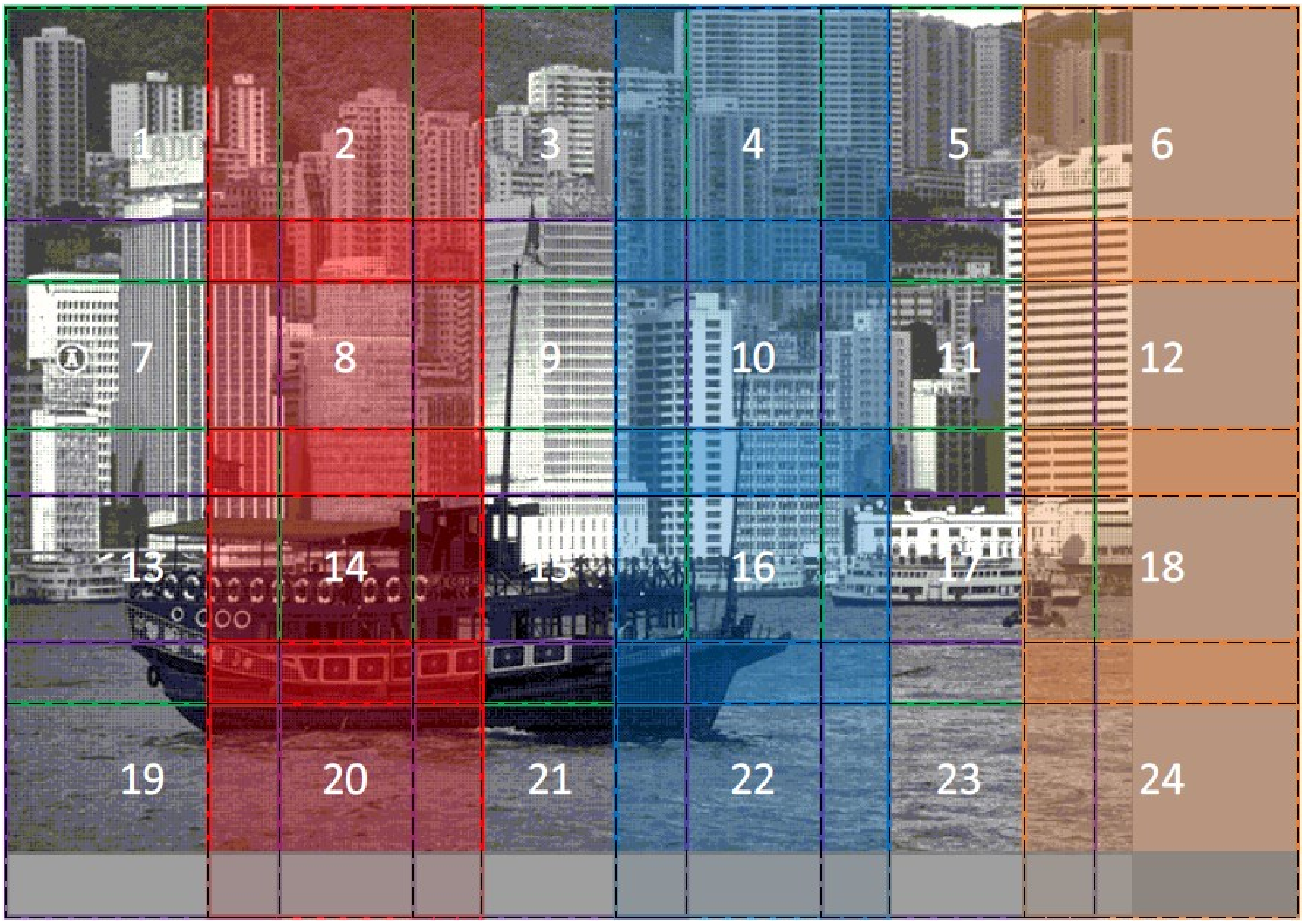
Batch Size 32
Early Stopping ✓
Number of Epochs 100
Learning Rate 0,004
Tiling ✓
Region size
Use dataset size ☐
w 768 h 512

For object detection the minimum tile size is that of the input architecture (YOLOX = 224).

The first tile starts at the image origin (top-left corner), proceeding horizontally and vertically. Tiles that surpass the right and bottom image sides are padded with the average value of image pixels in the tile.

To avoid slicing coherent objects that border tiles, tiles overlap to a variable degree depending on the maximum size of the labelled objects in the dataset (maximum object size + 10 pixels up to 25% of tile size).

For example, a 2064x1544 image can result in 24 tiles, depending on the tile size and the overlap:



If the maximum object size exceeds 25% of the height or width of the tile, when performing inference, in addition to searching the tiles, another search is performed subsampling the entire image to the architecture input size to ensure that larger objects are not missed. Note that this additional step impacts model performance slightly.

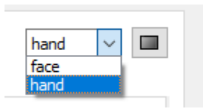
For object detection, architectures perform well without tiling if the minimum object size is sufficiently large to be identified after sub-sampling of the original image. If objects of interest are too small (or effectively disappear) after sub-sampling, then tiling is beneficial and should provide better results by successfully locating objects that would otherwise lack sufficient detail (pixels) to be recognized.

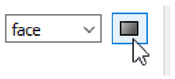
Manually Adding or Adjusting Bounding Boxes

For the object detection and segmentation modules, bounding boxes or segmentation masks can be added or adjusted in the Dataset.

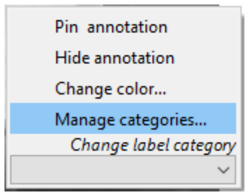
To quickly navigate through images while annotating and maintaining the mouse operation use the **Spacebar** to advance to the next image and the **Backspace** button to move the previous image.

To add a bounding box to an image, select the category label from the drop-down list (or use the number keys [0-9] to quickly select from the first 10 categories) and then click then click then Bounding Box button.

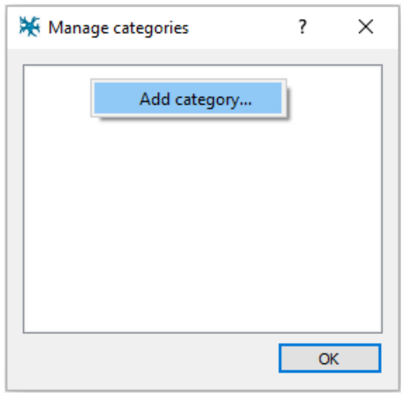


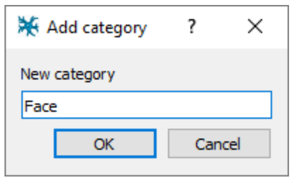


If the dataset does not currently have any category labels, add a bounding box, right-click on it and select **Manage categories**.

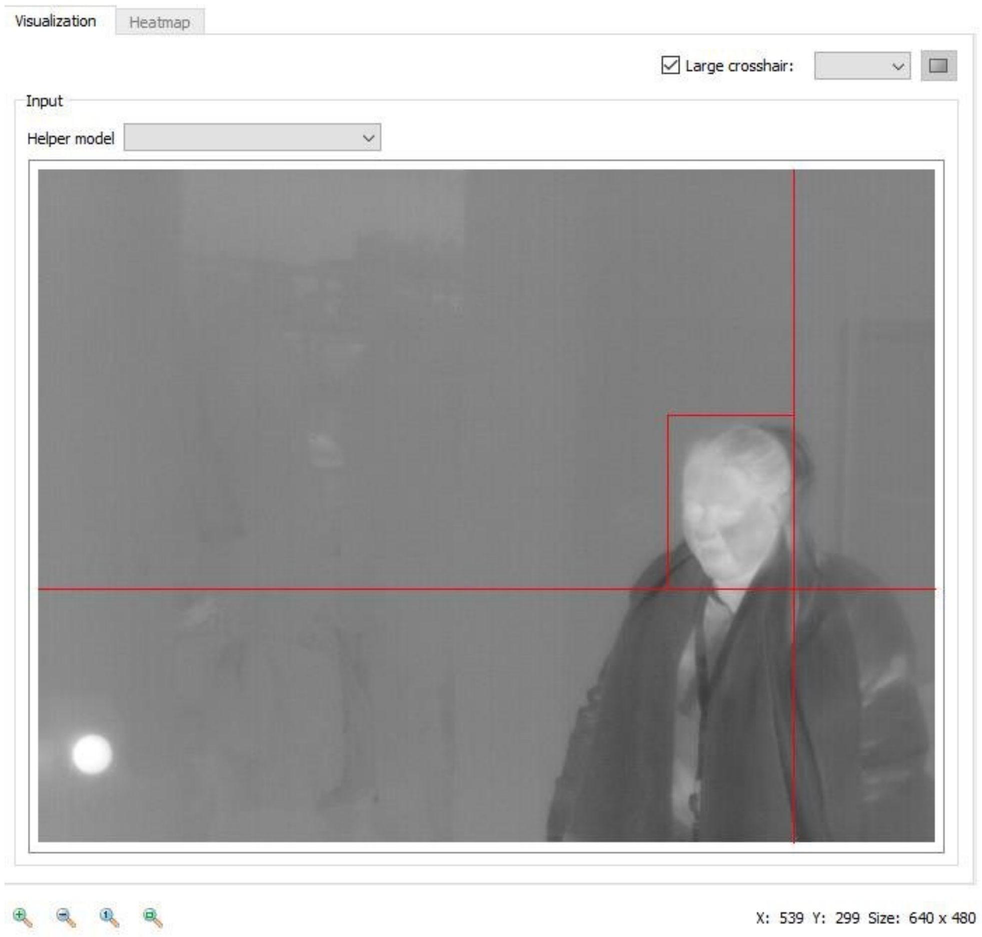


Right-click in the **Manage categories** dialog and select **Add category…** and add the category label.





For object detection, use the mouse to draw a rectangular bounding box in the image, starting with the top-left corner. When the **Large** crosshair option is enabled, the cursor is shown as a two intersecting lines, with the X and Y coordinates displayed in the bottom-right corner of the panel.



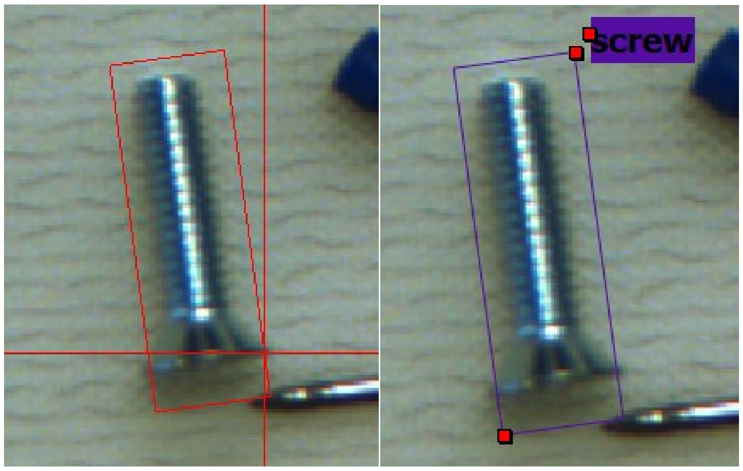
Click the cursor at the required bottom-right point of the bounding box.

Deleting Bounding Boxes

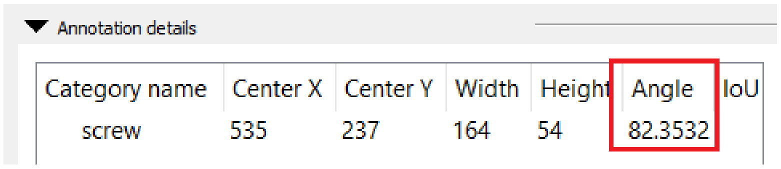
To delete a bounding box, simply select it in the image and press the keyboard **Delete** key.

Rotated Bounding Boxes

To create a rotated rectangle, click two of the four corners, then mouse your mouse to enlarge the rectangle to the desired width and finally make a third click to complete as shown below.



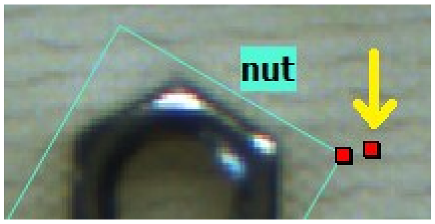
- In the **Annotation details** note the presence of the new **Angle** field.



Annotation details

Category name	Center X	Center Y	Width	Height	Angle	IoU
screw	535	237	164	54	82.3532	

- If needed, adjust the angle of the rectangle by using the extra handle as shown here.

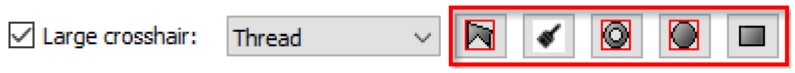


- Repeat the above operation for all annotations in the dataset images.

The Smart Label and/or Self-Training methods can be used to reduce the number of manual annotations to perform.

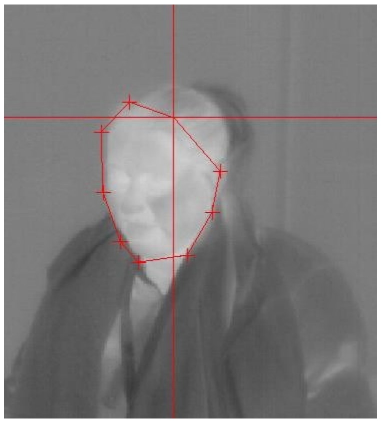
Segmentation Masks

The segmentation module supports several common annotation tools to define bounding box masks. These include polygon, paint brush, annulus, circle and rectangle tools.

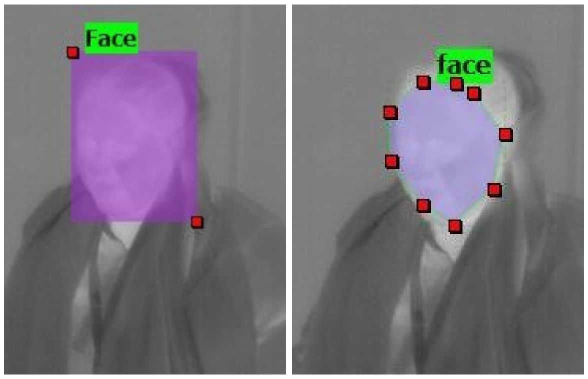


Use the drop-down list to select the mask category.

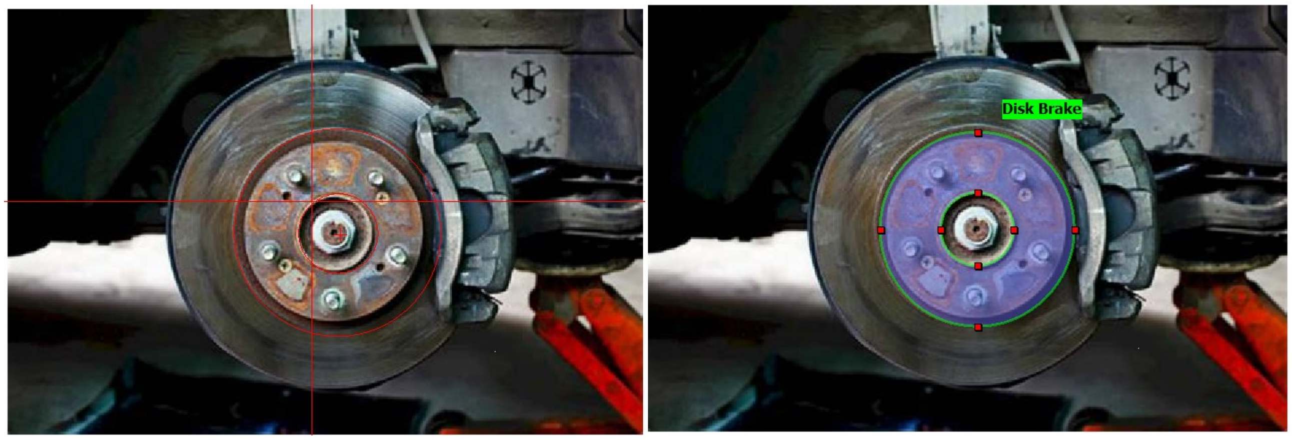
For polygon bounding boxes use mouse clicks to add additional vertices and double-click to close the bounding box.



When the bounding box is added it is shown with the category label and bounding box handles.



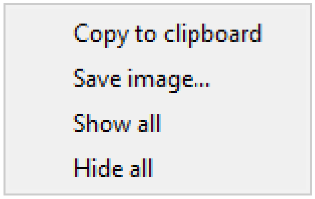
For the annulus tool, place the mouse in the center of the target area and create the outer circle, click and then place the inner circle.



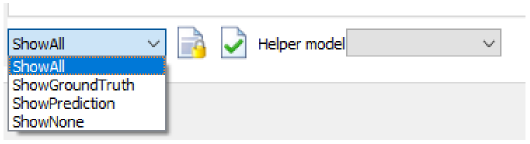
Click ESC or another function of the Astrocyte interface to exit the bounding box operation.

Additional Annotation Commands

The right-click pop-up menu provides several additional commands to help create and select annotations.



The Visualization panel includes a drop-down list for showing and hiding annotations (ground truth or predictions)



Masking

Astrocyte's implementation of masking allows users to create static regions of interest (ROIs) of pixels to include/exclude within the images contained in the dataset. Masking is available for all Astrocyte modules.

Masking is used to restrict the inspection area to a portion of the image only. It has the effect of:

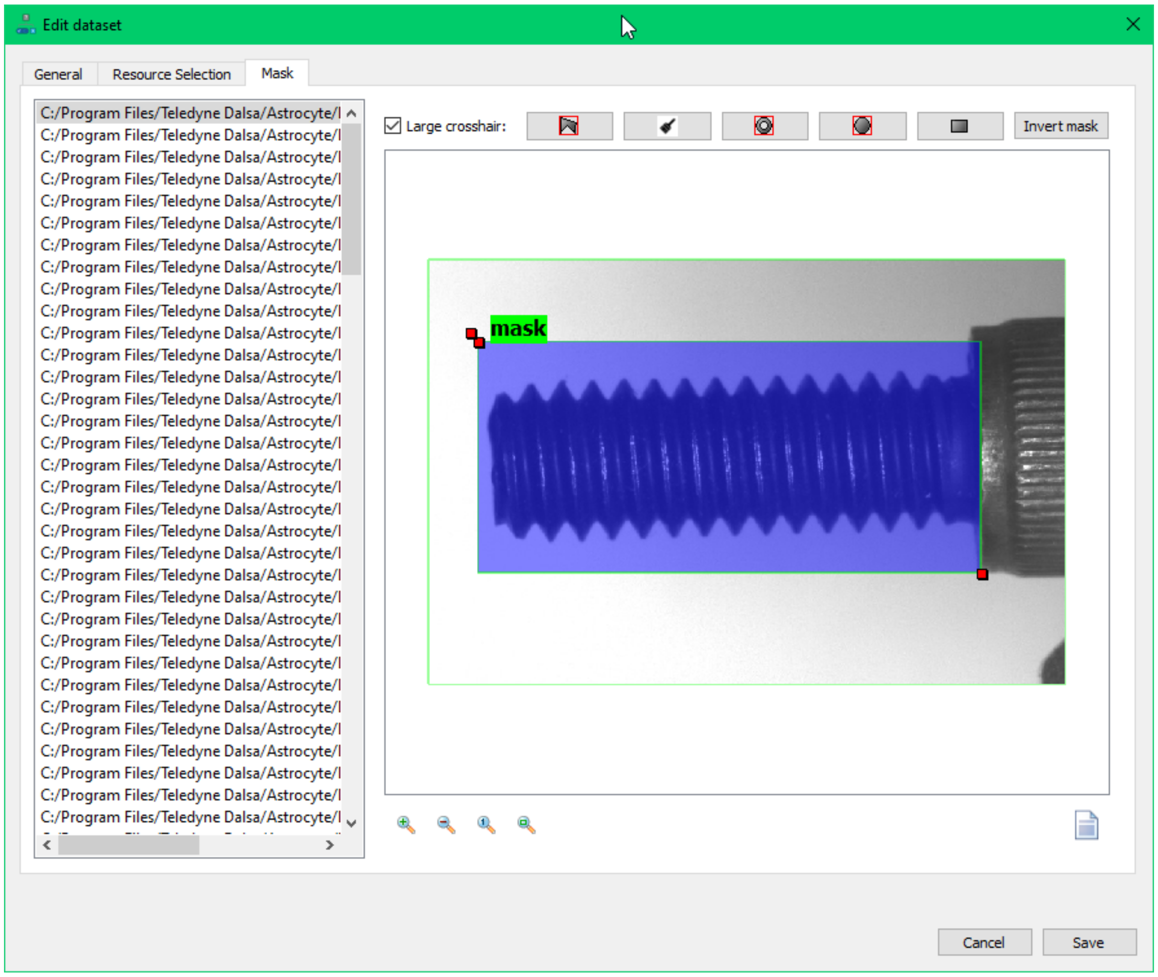
- Increasing accuracy
- Reducing training time, memory usage and inference time

By static, it means that one set of ROIs are applied to all the images. Therefore, images in the dataset should all contain the target pixels in the area(s) defined by an ROI in a single typical image (rotation may not be possible).

The mask is saved along with the model and applied at inference time. Note that the selected model architecture for the module continues to determine the model size (that is, the number of neurons).

Creating Masks

Masks are defined for an image dataset in the Mask tab of the Create dataset (or Edit dataset) dialog.

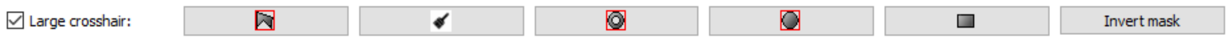


C:/Program Files/Teledyne Dalsa/Astrocyte/I (repeated 33 times in file list)

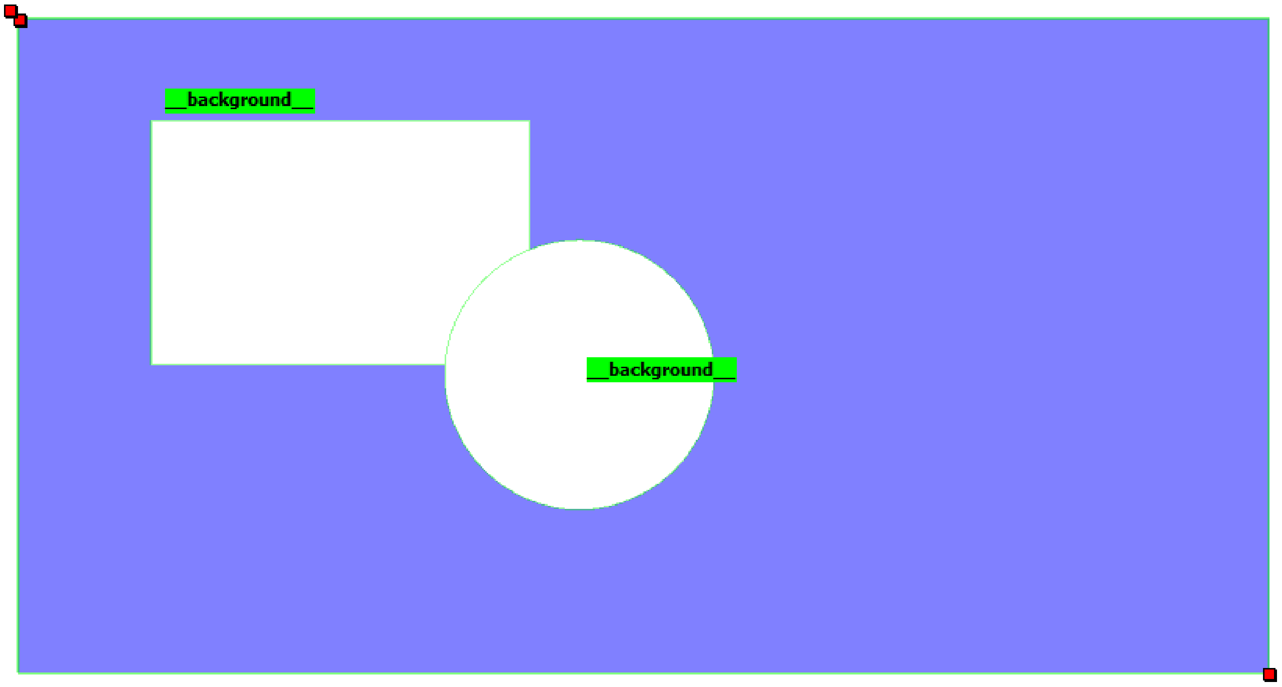
Masks are defined as a binary image with regions labelled:

- **mask**: masked pixels (ignored)
- **background**: unmasked or active pixels ("background" relative to the mask image)

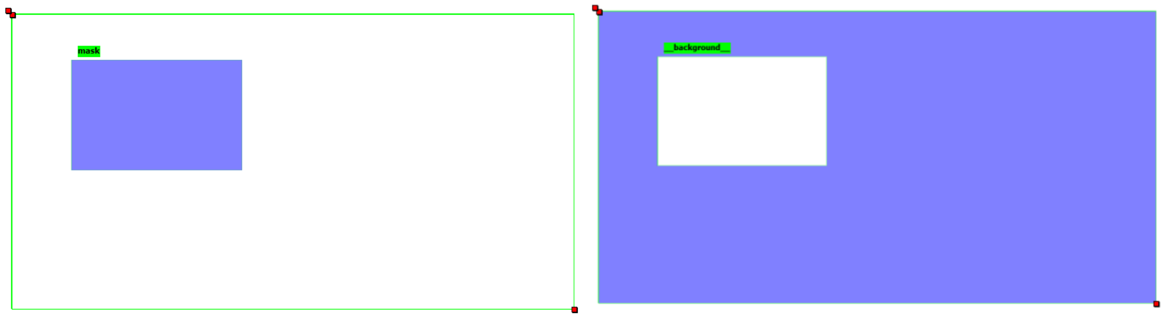
The annotation tools for creating a mask are located at the top of the image display. These include typical shapes such as polygon, paint brush, annulus, circle and rectangle.



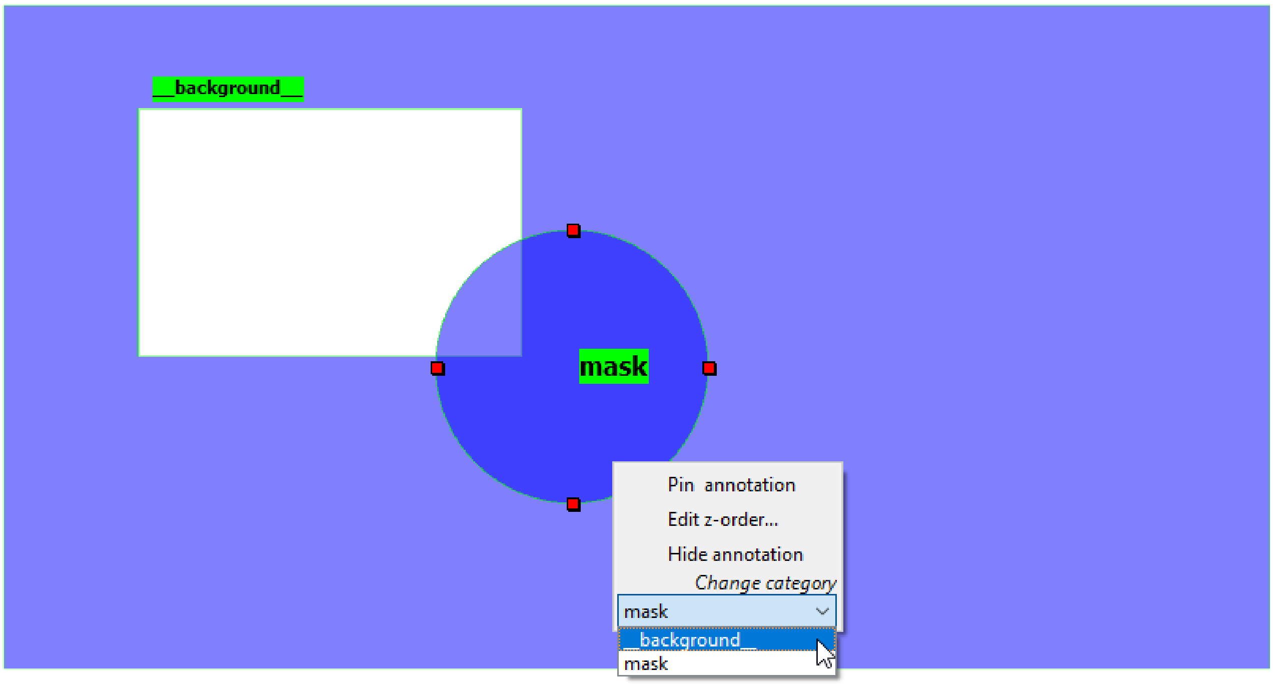
Multiple regions can be drawn. Regions can overlap to create complex shapes.



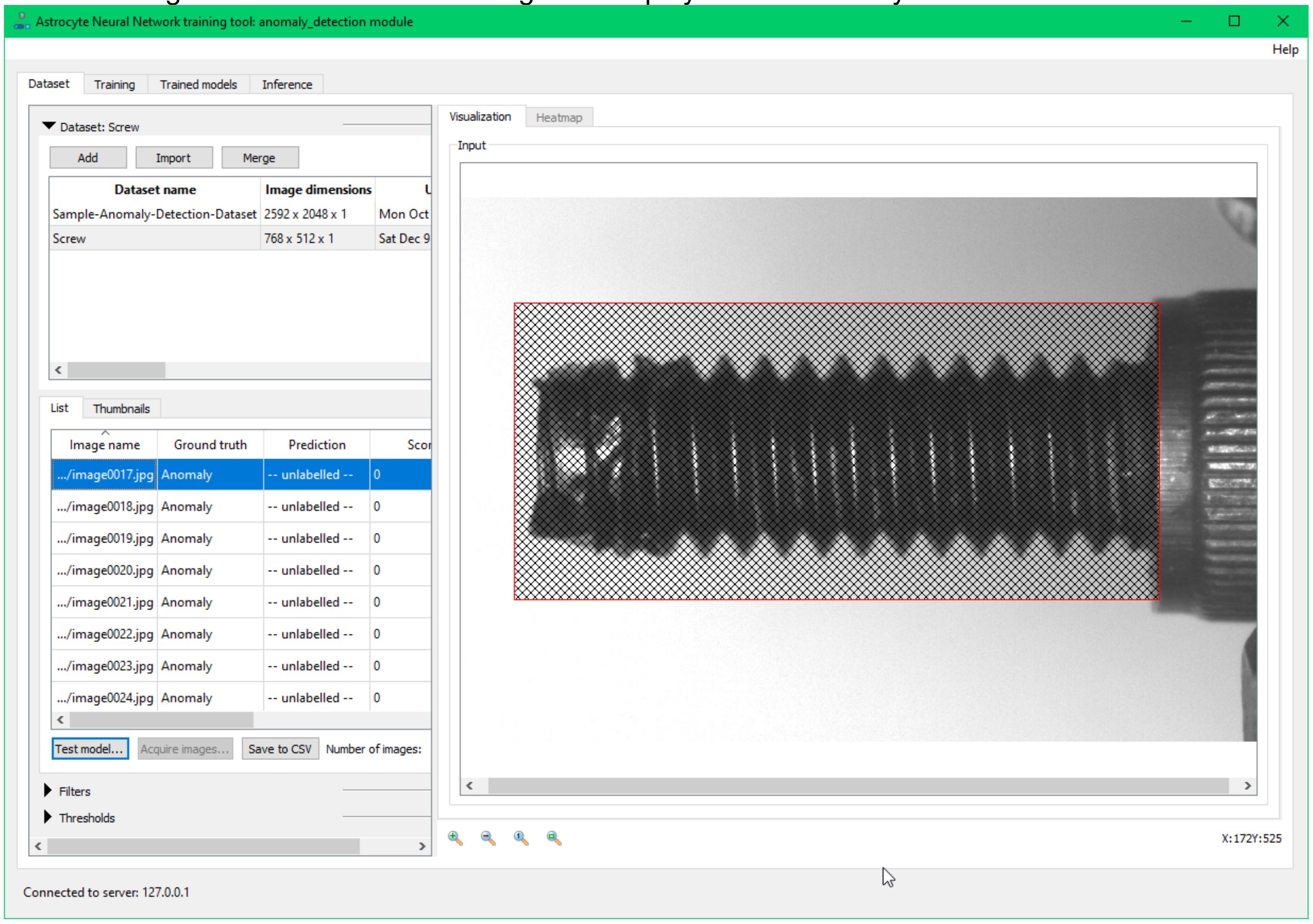
The **Invert mask** button changes (flips) the current mask labels.



Alternatively, the category for individual regions can be switched using the right-click menu **Change category** command.



When viewing the dataset the masked region is displayed as an overlay:


Astrocyte Neural Network training tool: anomaly_detection module

Help

Dataset | Training | Trained models | Inference

Dataset: Screw

Add | Import | Merge

Dataset name	Image dimensions	U
Sample-Anomaly-Detection-Dataset	2592 x 2048 x 1	Mon Oct
Screw	768 x 512 x 1	Sat Dec 9

List | Thumbnails

Image name	Ground truth	Prediction	Scor
.../image0017.jpg	Anomaly	-- unlabelled --	0
.../image0018.jpg	Anomaly	-- unlabelled --	0
.../image0019.jpg	Anomaly	-- unlabelled --	0
.../image0020.jpg	Anomaly	-- unlabelled --	0
.../image0021.jpg	Anomaly	-- unlabelled --	0
.../image0022.jpg	Anomaly	-- unlabelled --	0
.../image0023.jpg	Anomaly	-- unlabelled --	0
.../image0024.jpg	Anomaly	-- unlabelled --	0

Test model... | Acquire images... | Save to CSV | Number of images:

Filters

Thresholds

Visualization | Heatmap

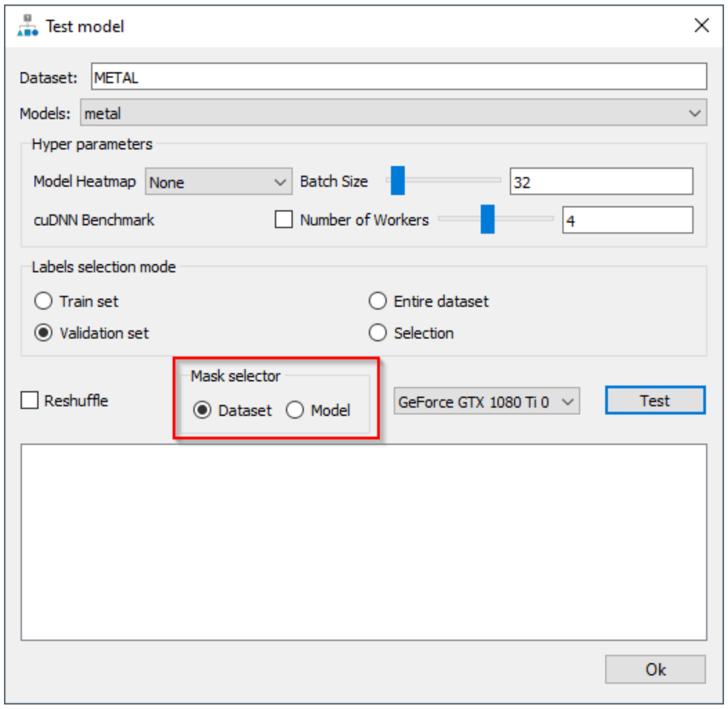
Input

X:172Y:525

Connected to server: 127.0.0.1

Using a Different Mask for Testing

When testing models, the dataset mask can be used instead of the mask saved with the model. This can be useful if the target region in the image is not in the same area as in the dataset on which the model was trained.



Test model

Dataset: METAL

Models: metal

Hyper parameters

Model Heatmap: None

Batch Size: 32

cuDNN Benchmark

[ ] Number of Workers: 4

Labels selection mode

( ) Train set
( ) Entire dataset
(●) Validation set
( ) Selection

[ ] Reshuffle

Mask selector

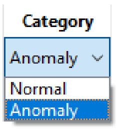
(●) Dataset ( ) Model

GeForce GTX 1080 Ti 0

Adding, Editing or Deleting Category Labels

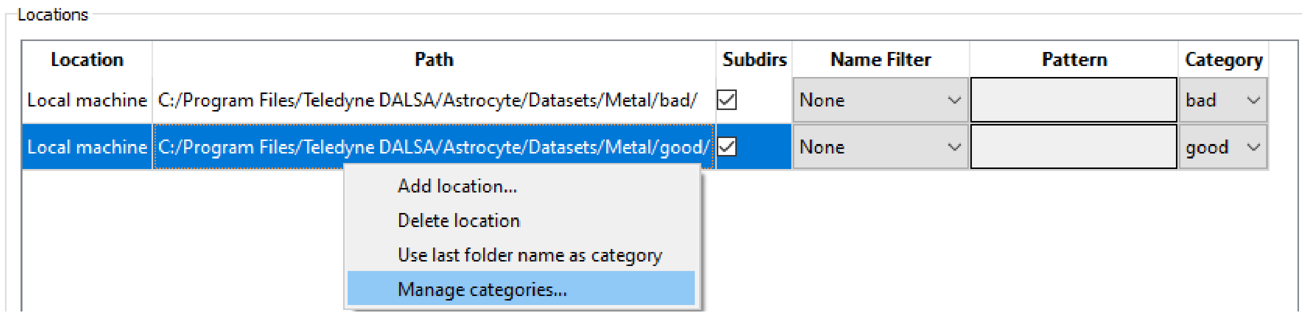
Anomaly Detection Category Labels

For the Anomaly Detection module, two default category labels are available in the Dataset tab: Normal and Anomaly.



Classification Category Labels

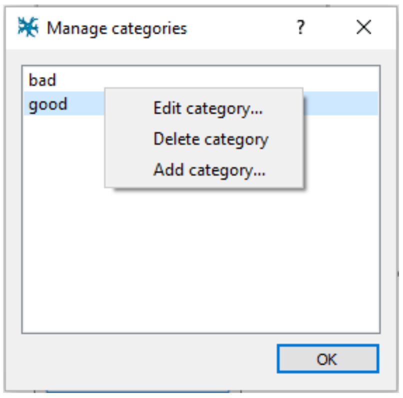
For the classification module, to add, edit or delete category labels right-click on an entry in the Dataset tab and select **Manage categories…**:



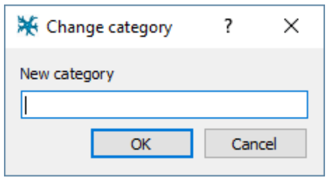
Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/Metal/bad/	✓	None		bad
Local machine	C:/Program Files/Teledyne DALSA/Astrocyte/Datasets/Metal/good/	✓	None		good

Right-click to use the pop-up menu to add, edit or delete category labels.

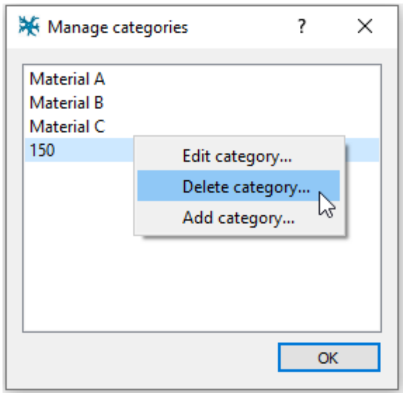


To edit a category name, enter the new category name in the Change category dialog box:

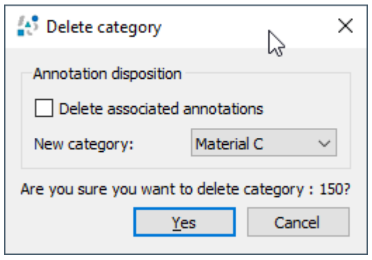


Alternatively, the category label can be directly edited in the drop-down list.

When deleting a category, existing annotations with that category can be migrated to a different category.



Material A
Material B
Material C
150



☐ Delete associated annotations

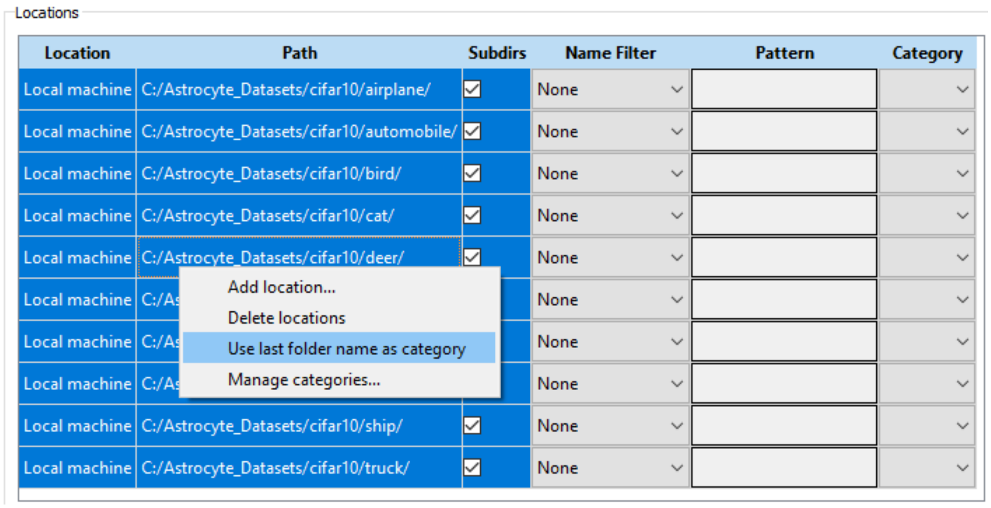
New category: Material C

Are you sure you want to delete category : 150?

Use Folder Name as Category Label

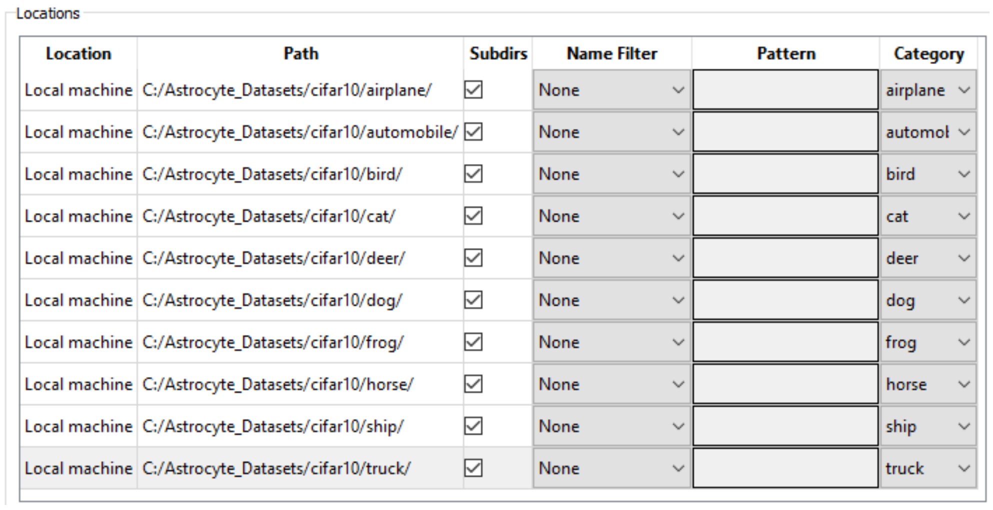
In the classification module, images of the same category can be placed in a single directory folder and the folder name used as the category label. The folder name is applied as the category label for all images within the folder.

To do so, select the named directories, right-click to open the pop-up menu and choose **Use last folder name as category**.



Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Astrocyte_Datasets/cifar10/airplane/	✓	None		
Local machine	C:/Astrocyte_Datasets/cifar10/automobile/	✓	None		
Local machine	C:/Astrocyte_Datasets/cifar10/bird/	✓	None		
Local machine	C:/Astrocyte_Datasets/cifar10/cat/	✓	None		
Local machine	C:/Astrocyte_Datasets/cifar10/deer/	✓	None		
Local machine	C:/As		None		
Local machine	C:/As		None		
Local machine	C:/As		None		
Local machine	C:/Astrocyte_Datasets/cifar10/ship/	✓	None		
Local machine	C:/Astrocyte_Datasets/cifar10/truck/	✓	None		

The Category label drop-down list displays the assigned category label for each directory in the dataset.



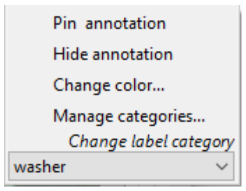
Locations

Location	Path	Subdirs	Name Filter	Pattern	Category
Local machine	C:/Astrocyte_Datasets/cifar10/airplane/	☑	None		airplane
Local machine	C:/Astrocyte_Datasets/cifar10/automobile/	☑	None		automol
Local machine	C:/Astrocyte_Datasets/cifar10/bird/	☑	None		bird
Local machine	C:/Astrocyte_Datasets/cifar10/cat/	☑	None		cat
Local machine	C:/Astrocyte_Datasets/cifar10/deer/	☑	None		deer
Local machine	C:/Astrocyte_Datasets/cifar10/dog/	☑	None		dog
Local machine	C:/Astrocyte_Datasets/cifar10/frog/	☑	None		frog
Local machine	C:/Astrocyte_Datasets/cifar10/horse/	☑	None		horse
Local machine	C:/Astrocyte_Datasets/cifar10/ship/	☑	None		ship
Local machine	C:/Astrocyte_Datasets/cifar10/truck/	☑	None		truck

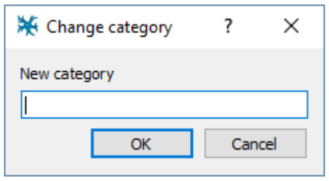
Object Detection and Segmentation Category Labels

For the object detection and segmentation modules category labels can also be modified, deleted or new category labels added in the Dataset.

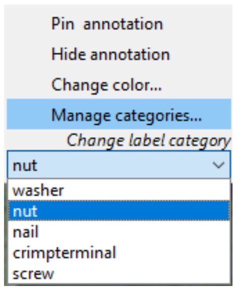
Right-click on an existing bounding box to open the pop-up menu and select **Manage categories…** to edit, delete or add a category.



To edit or add a class, enter the new category label in the corresponding dialog box:



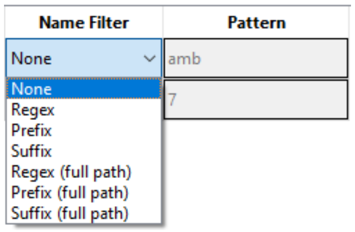
The assigned category label to a bounding box can be changed using the right-click pop-up menu for the selected annotation and using the **Change label category** drop-down list.



Filtering Images Using Regular Expressions

Astrocyte supports filtering images within directories by prefix, suffix or using a regular expression (using common regular expression syntax); only images corresponding to the filter are added to the dataset. The full path options apply the filter to the full image directory path.

To do so, use the Name Filter drop-down list to select the filter and use the Pattern field to enter the required filter. For example, with a Prefix or Suffix filter, only those images whose filename starts or ends with the specified pattern are added to the dataset.



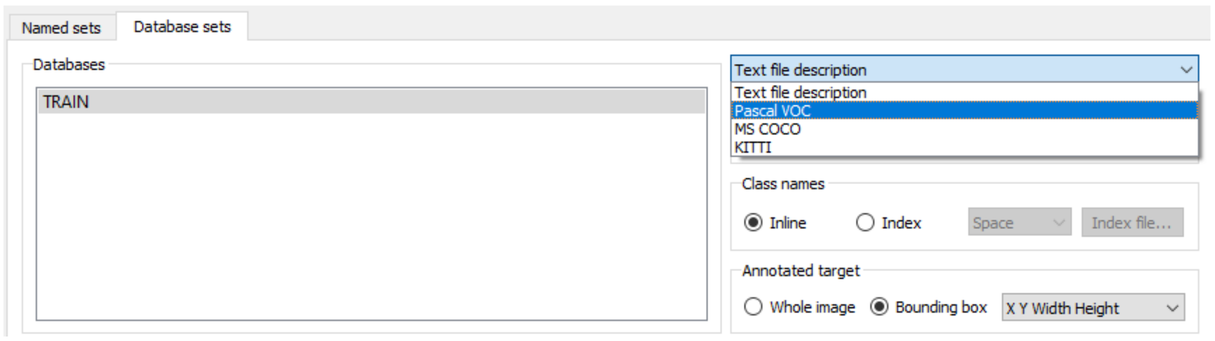
Annotation Files

Annotation files describe the bounding boxes or image masks and associated category labels located in the dataset images.

Astrocyte supports the following annotation file types:

File Type	Description
Text File	Text files containing annotations with separated values for bounding boxes,category labels and image file paths can be used (for example, .csv files). For more information, refer to the Using Text Files for Annotations section.
PASCAL VOC	Pattern Analysis, Statistical Modelling and Computation Learning (PASCAL) Visual Object Classification (VOC) annotated image dataset. PASCAL VOC uses an individual XML file to describe the bounding boxes and labels for each image. Astrocyte uses the object bounding box only; bounding boxes identifying parts within an object bounding box are not used.
MS COCO	Microsoft (MS) Common Object in Context (COCO) annotated image dataset. MS COCO uses a single JSON file to provide bounding box and category label information for all images in the dataset. Astrocyte uses the MS COCO JSON file identifying instances; caption or person keypoint JSON files are not used.
KITTI	The KITTI Vision Benchmark Suite file format for 3D object tracklet labels (cars, trucks, trams, pedestrians, cyclists) stored as xml file.

After adding the database, select it and specify the annotation file format using the drop-down list:

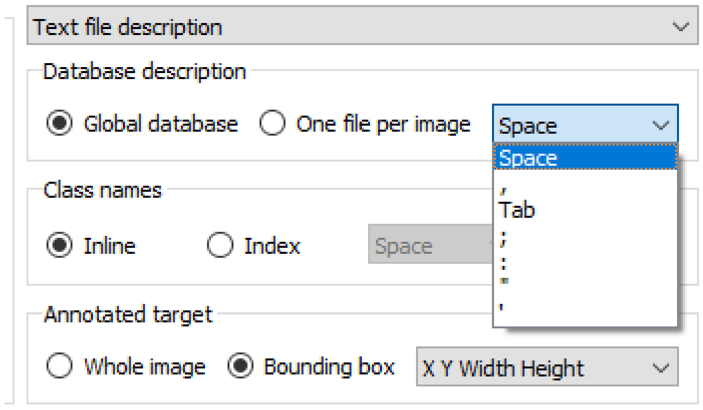


When a valid database is defined, the Generate button is enabled; click **Generate** to add the dataset and close the dialog.

Using Text Files for Annotations

In the Database description section, use the radio buttons to set whether a single file containing annotations for all images (Global database) or individual annotation files for each image are used (One file per image).

Use the drop-down list to specify the separator character that delimits entries in the text file.


Text file description

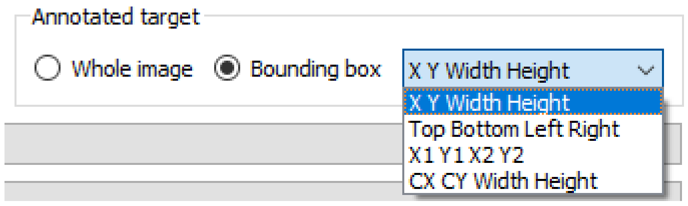
Database description: [✓ Global database] One file per image — Space

Class names: [✓ Inline] Index — Space

Annotated target: Whole image [✓ Bounding box] — X Y Width Height

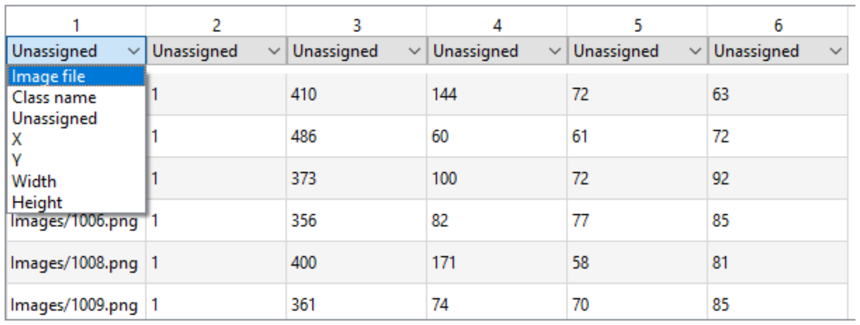
Category names can be assigned directly in the annotation file, or if numbers are used, an associated index file can assign text labels to the numerical labels.

Several bounding box formats are supported; select the format that corresponds to the information available in the annotation text file using the Bounding Box drop-down list. Alternatively, the bounding box can be considered as the whole image, in which case only category labels need to be provided and bounding box coordinates are not needed.

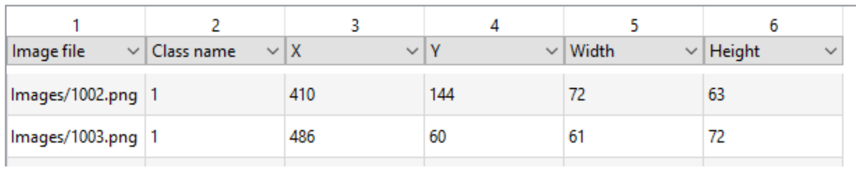

Annotated target: Whole image [✓ Bounding box] — X Y Width Height

Format	Description
X Y Width Height	Top-left corner X coordinate, top-left corner Y coordinate, width in pixels, height in pixels
Top Bottom Left Right	Top: Y coordinate of top of bounding box Bottom: Y coordinate of bottom of bounding box Left: X coordinate of left of bounding boxRight: X coordinate of right of bounding box
X1 Y1 X2 Y2	X1, Y1: top-left coordinate X2, Y2: bottom-right coordinate
CX CY Width Height	Center coordinate X, center coordinate Y, width in pixels, height in pixels

Each line in the annotation defines a single image bounding box; use the drop-down lists for each column to assign the entry type.



1	2	3	4	5	6
Unassigned	Unassigned	Unassigned	Unassigned	Unassigned	Unassigned
	1	410	144	72	63
	1	486	60	61	72
	1	373	100	72	92
Images/1006.png	1	356	82	77	85
Images/1008.png	1	400	171	58	81
Images/1009.png	1	361	74	70	85



1	2	3	4	5	6
Image file	Class name	X	Y	Width	Height
Images/1002.png	1	410	144	72	63
Images/1003.png	1	486	60	61	72

If multiple bounding boxes are present in an image a separate entry is required for each bounding box.

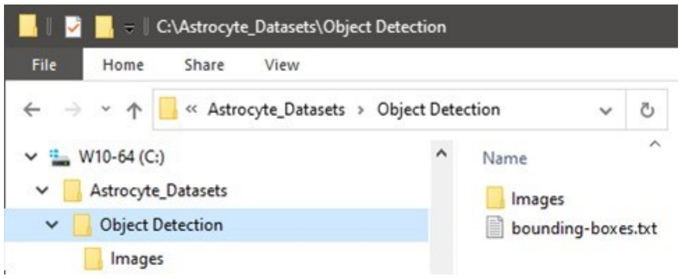
Images/101.png 1 446 83 70 85

Images/101.png 1 227 177 57 59

When multiple entries for the same image are present in the annotation file Astrocyte does not make multiple copies of the image on the Cortex server; only one instance of the image is required.

The image path specified in the annotation file is used in conjunction with that assigned using the Image Folder button to provide the full path to the image location. That is, any relative path in the annotation file is based on the assigned Image Folder path.

For example, if the directory structure of the dataset is:

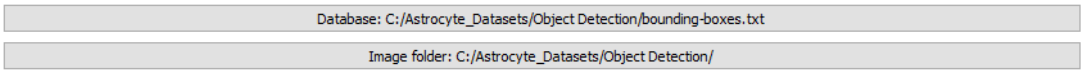


The annotation file uses the following relative path:

Images/100.png 1 461 210 56 80

To provide the full path to the images folder, the Image Folder must be assigned to:

C:/Astrocyte_Datasets/Object Detection/



Database: C:/Astrocyte_Datasets/Object Detection/bounding-boxes.txt

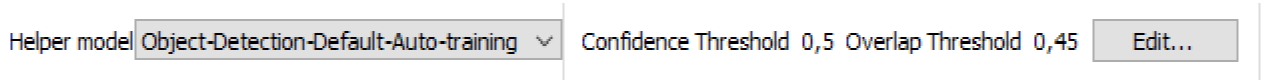
Image folder: C:/Astrocyte_Datasets/Object Detection/

That is, the Image Folder path is not necessarily the full path to the actual images folder, but is used in conjunction with the relative path defined in the annotation text file to define the full path to the images. This allows a certain flexibility in directory structure without having to edit existing annotation file image path entries.

Smart Label Function: Assisted Labelling

For object detection, the Smart Label function uses an existing object detection model to automatically locate objects (using the specified confidence threshold) and create bounding boxes and labels in the selected image.

To do so, in the Visualization panel select the model using the **Helper model** drop-down list, click **Edit…** and set the confidence threshold.


Helper model Object-Detection-Default-Auto-training Confidence Threshold 0,5 Overlap Threshold 0,45

Double-click in the target input image or right-click on the image name and select **Smart label** (multiple images can be selected in this manner); the **#Ground truth** is updated and refreshing the image display shows the bounding boxes and labels of the located objects.

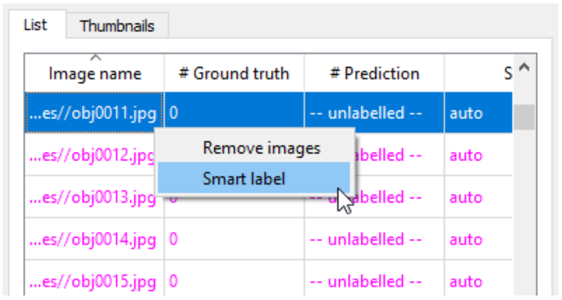


Image name	# Ground truth	# Prediction	S
...es//obj0011.jpg	0	-- unlabelled --	auto
...es//obj0012.jpg		belled --	auto
...es//obj0013.jpg		belled --	auto
...es//obj0014.jpg	0	-- unlabelled --	auto
...es//obj0015.jpg	0	-- unlabelled --	auto

If necessary, adjust any bounding boxes or labels.

The Smart Label function generates labels called “Pseudo Labels” as opposed to “Ground Truth Labels” which are created manually. Pseudo labels are shown in pink color in the image list while Ground Truth labels are shown in black.

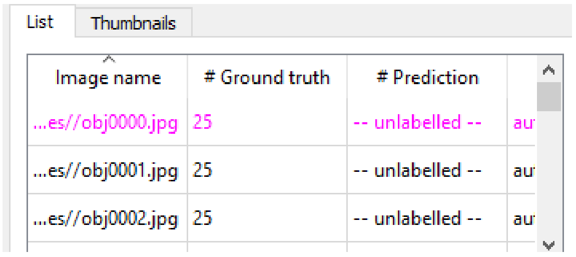
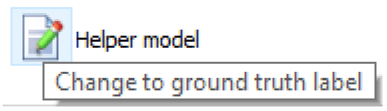


Image name	# Ground truth	# Prediction	
...es//obj0000.jpg	25	-- unlabelled --	au
...es//obj0001.jpg	25	-- unlabelled --	au
...es//obj0002.jpg	25	-- unlabelled --	au

To use Pseudo Labels in the next training session, they must be converted to Ground Truth Labels using the button shown below.

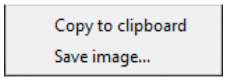


This is to ensure label positions are reviewed visually and confirmed as acceptable ground truth.

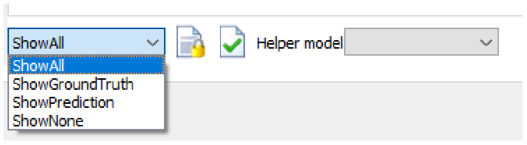
Note that this manual approval operation must be performed on each image where Assisted Labeling is applied. The button is a toggle that allows converting Ground Truth Labels back to Pseudo Labels as required.

Additional Commands

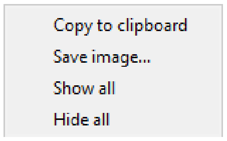
Right-clicking on an image provides access to additional commands to copy and save images.



For modules that support annotations, such as Object Detection and Segmentation, the Visualization panel includes a drop-down list for showing and hiding annotations (ground truth or predictions).

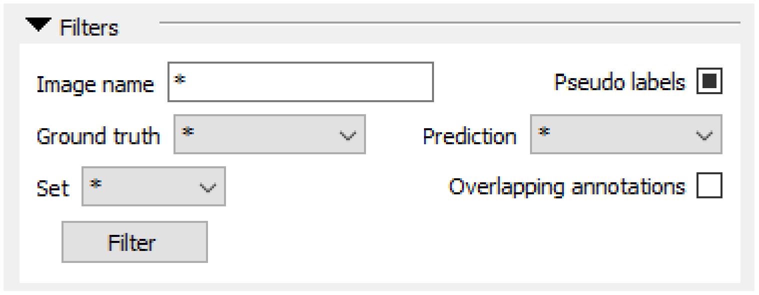


The right-click menu includes the following commands:



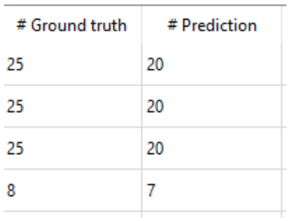
Filtering & Sorting Images

The image dataset can be navigated using the column filters to sort images.



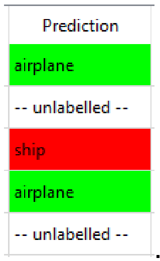
Note: the above filters are immediately applied upon changing states. However if you change annotations in the view window (for example, adding or deleting annotations) then you can click to Filter button to apply the filters again.

For the object detection and segmentation modules the **# Ground Truth** column displays the number of labelled bounding boxes (that is, the ground truth) in the image; the **#Prediction** column displays the number of predicted labels following model testing.



# Ground truth	# Prediction
25	20
25	20
25	20
8	7

For the classification module, the **Predicted** column displays the predicted category label (or “unlabelled” if no inference has been performed on the image).



Prediction
airplane
-- unlabelled --
ship
airplane
-- unlabelled --

.

The **Set** column displays the assigned image set for validation, training or whether images are automatically assigned to a set by Astrocyte.

For the applied filters, the number of corresponding images within the dataset is displayed, which can be used to provide information regarding the ratio of images within the entire dataset to avoid unbalanced datasets.

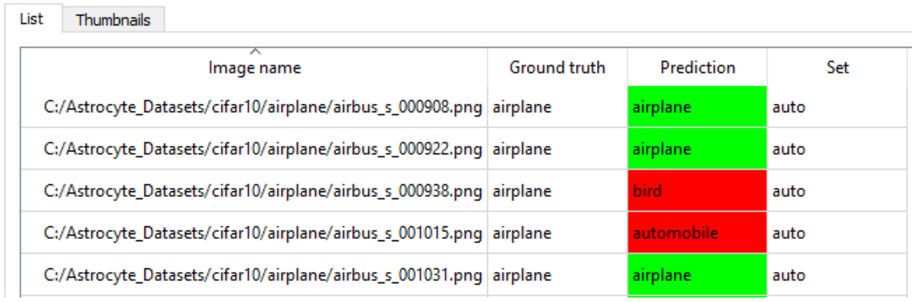


Image name	Ground truth	Prediction	Set
C:/Astrocyte_Datasets/cifar10/airplane/airbus_s_000908.png	airplane	airplane	auto
C:/Astrocyte_Datasets/cifar10/airplane/airbus_s_000922.png	airplane	airplane	auto
C:/Astrocyte_Datasets/cifar10/airplane/airbus_s_000938.png	airplane	bird	auto
C:/Astrocyte_Datasets/cifar10/airplane/airbus_s_001015.png	airplane	automobile	auto
C:/Astrocyte_Datasets/cifar10/airplane/airbus_s_001031.png	airplane	airplane	auto

Using multiple filters can extract information about testing performance on the image dataset. For example, ground truth and prediction filters can be used together to view only those results of interest; the Number of images field is updated according to the filters applied.

Images can be sorted to display only those images whose filename contains the string entered in the Image name column drop-down list. For example, in the CIFAR dataset, filtering the filename by "tug" shows that there are 62 images of tugboats in the entire dataset of 50000 images.

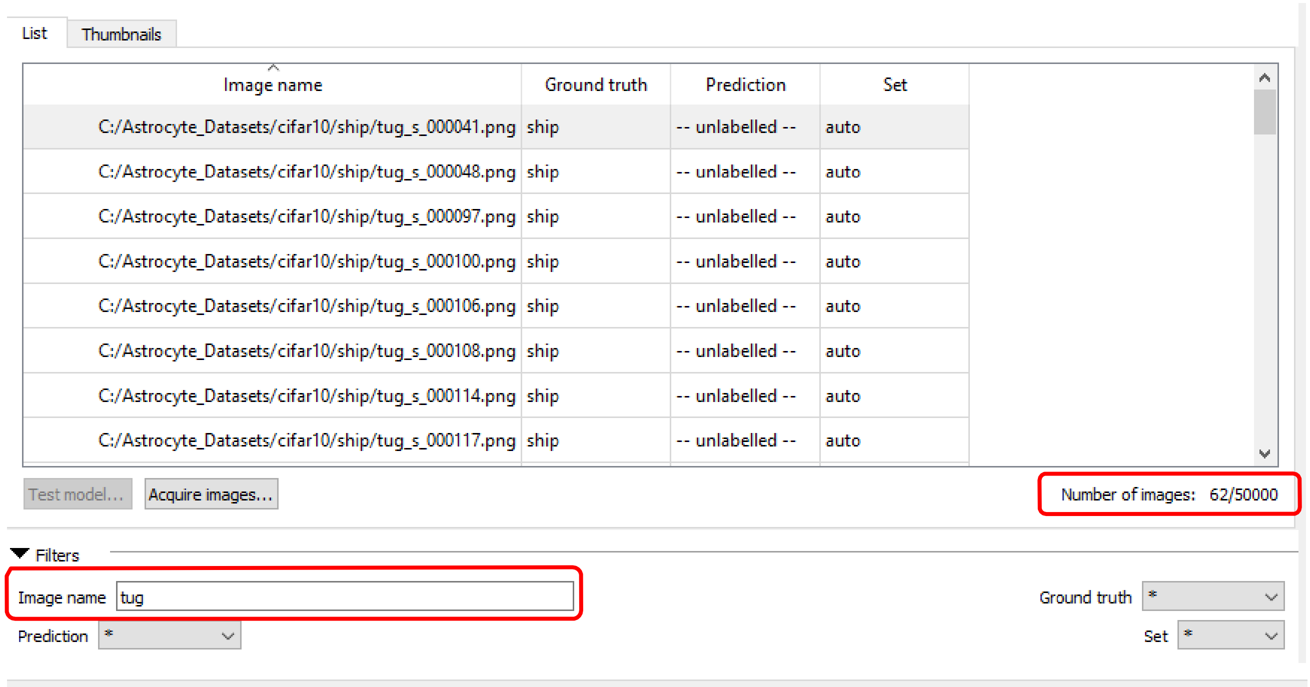


Image name	Ground truth	Prediction	Set
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000041.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000048.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000097.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000100.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000106.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000108.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000114.png	ship	-- unlabelled --	auto
C:/Astrocyte_Datasets/cifar10/ship/tug_s_000117.png	ship	-- unlabelled --	auto

Number of images: 62/50000

Filters

Image name: tug

Ground truth: *

Prediction: *

Set: *

Pseudo Labels Filter

Object detection also includes a **Pseudo labels** filter to sort images that have bounding boxes and labels created using the Self-Training option. This is a tri-state checkbox that allows for 3 sorting methods:

☐ Pseudo labels	Exclude images with pseudo labels.
▣ Pseudo labels	Display all images (ground truth and pseudo labels).
☑ Pseudo labels	Display only images with pseudo-labels.

For example, using the sample hardware dataset with 40 labelled images and 60 pseudo labelled images (shown in pink font), the Number of images field shows the effect of each Pseudo label checkbox state:

Exclude images with pseudo labels.

☐ Pseudo labels

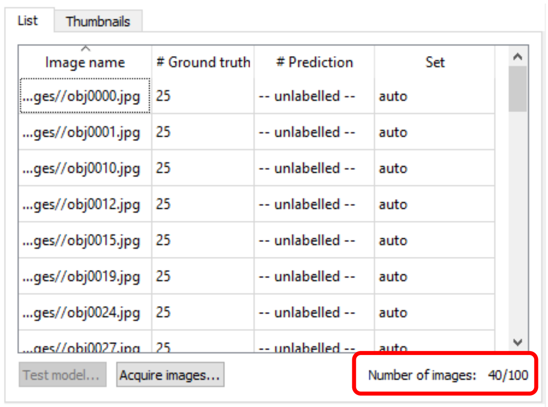


Image name	# Ground truth	# Prediction	Set
...ges//obj0000.jpg	25	-- unlabelled --	auto
...ges//obj0001.jpg	25	-- unlabelled --	auto
...ges//obj0010.jpg	25	-- unlabelled --	auto
...ges//obj0012.jpg	25	-- unlabelled --	auto
...ges//obj0015.jpg	25	-- unlabelled --	auto
...ges//obj0019.jpg	25	-- unlabelled --	auto
...ges//obj0024.jpg	25	-- unlabelled --	auto
...ges//obj0027.jpg	25	-- unlabelled --	auto

Number of images: 40/100

All images.

▣ Pseudo labels

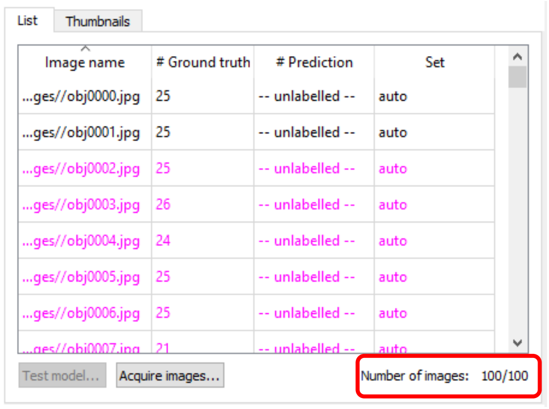
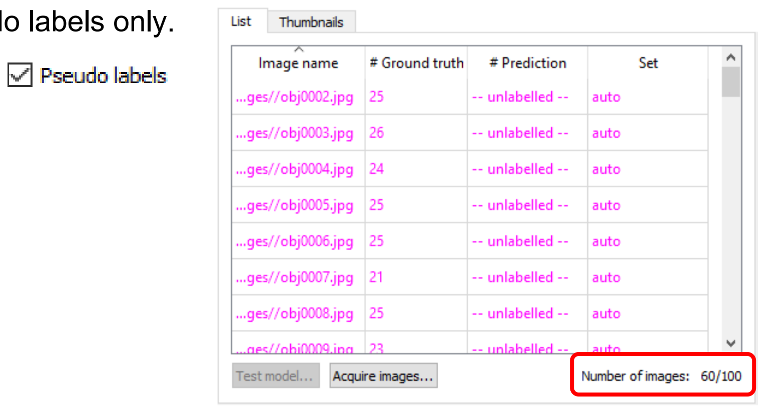


Image name	# Ground truth	# Prediction	Set
...ges//obj0000.jpg	25	-- unlabelled --	auto
...ges//obj0001.jpg	25	-- unlabelled --	auto
...ges//obj0002.jpg	25	-- unlabelled --	auto
...ges//obj0003.jpg	26	-- unlabelled --	auto
...ges//obj0004.jpg	24	-- unlabelled --	auto
...ges//obj0005.jpg	25	-- unlabelled --	auto
...ges//obj0006.jpg	25	-- unlabelled --	auto
...ges//obj0007.jpg	21	-- unlabelled --	auto

Number of images: 100/100

Pseudo labels only.


☑ Pseudo labels

List | Thumbnails

Image name	# Ground truth	# Prediction	Set
...ges//obj0002.jpg	25	-- unlabelled --	auto
...ges//obj0003.jpg	26	-- unlabelled --	auto
...ges//obj0004.jpg	24	-- unlabelled --	auto
...ges//obj0005.jpg	25	-- unlabelled --	auto
...ges//obj0006.jpg	25	-- unlabelled --	auto
...ges//obj0007.jpg	21	-- unlabelled --	auto
...ges//obj0008.jpg	25	-- unlabelled --	auto
...ges//obj0009.jpg	23	-- unlabelled --	auto

Test model... | Acquire images... | Number of images: 60/100

Note: If pseudo label bounding boxes or category labels are modified, and the image dataset saved, they are considered as ground truth and no longer identified as pseudo labels.

In addition, a pseudo label Training / Validation set cannot be modified and is always set to “auto”.

Overlapping Annotations Filter

Object detection also includes an **Overlapping annotations** filter to sort images that have bounding boxes overlapping each other. The bounding boxes need to be almost fully overlapped to be detected by the filter. The example below shows two sets of two overlapping bounding boxes sorted by the filter. This is useful to quickly locate bounding boxes that might have been mistakenly annotated twice.

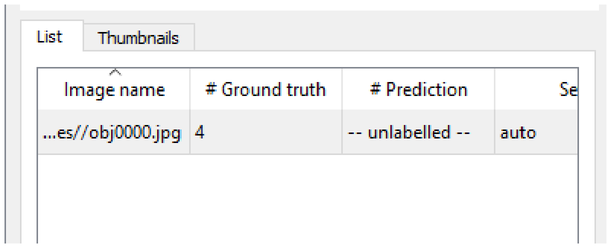
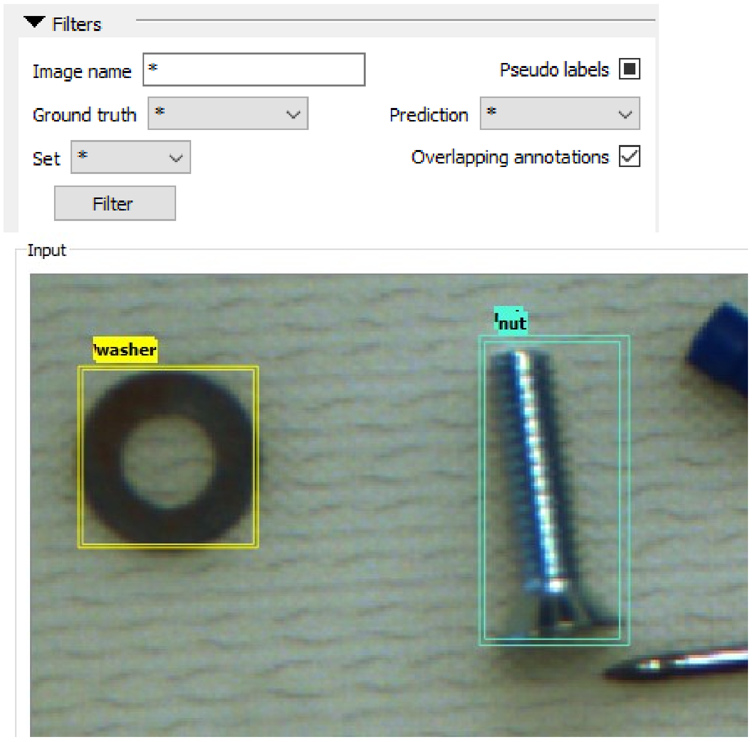

List | Thumbnails

Image name	# Ground truth	# Prediction	Se
...es//obj0000.jpg	4	-- unlabelled --	auto



Filters

Image name *

Pseudo labels ■

Ground truth *

Prediction *

Set *

Overlapping annotations ✓

Filter

Removing Images from the Dataset

To remove (delete) images from the dataset, select the image to remove, right-click and select **Remove images**. Multiple images can be selected using the common Windows mouse and key combinations (<shift> or <ctrl>).

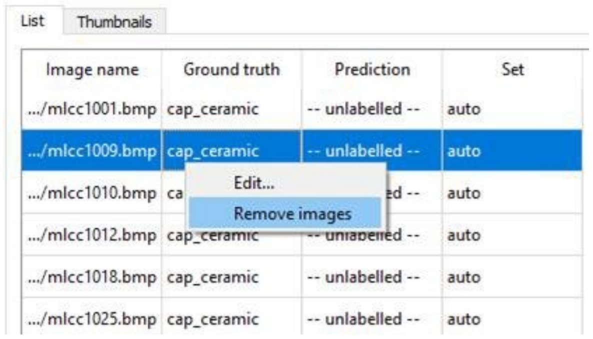


Image name	Ground truth	Prediction	Set
.../mlcc1001.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1009.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1010.bmp	ca	ed --	auto
.../mlcc1012.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1018.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1025.bmp	cap_ceramic	-- unlabelled --	auto

Images selected for removal are highlighted in red.

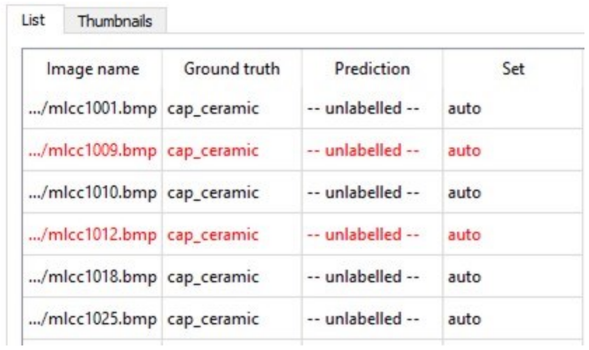
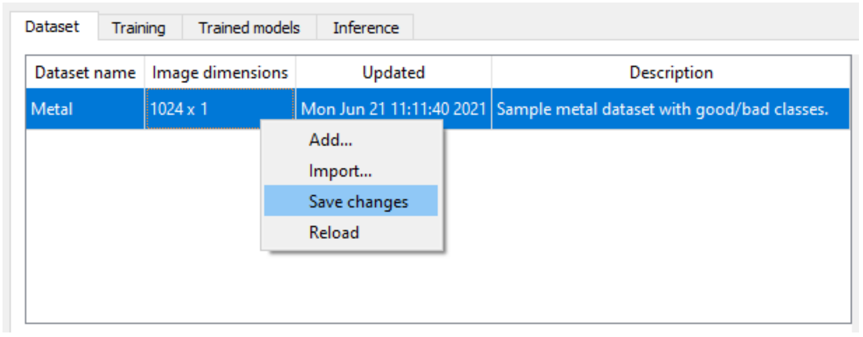


Image name	Ground truth	Prediction	Set
.../mlcc1001.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1009.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1010.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1012.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1018.bmp	cap_ceramic	-- unlabelled --	auto
.../mlcc1025.bmp	cap_ceramic	-- unlabelled --	auto

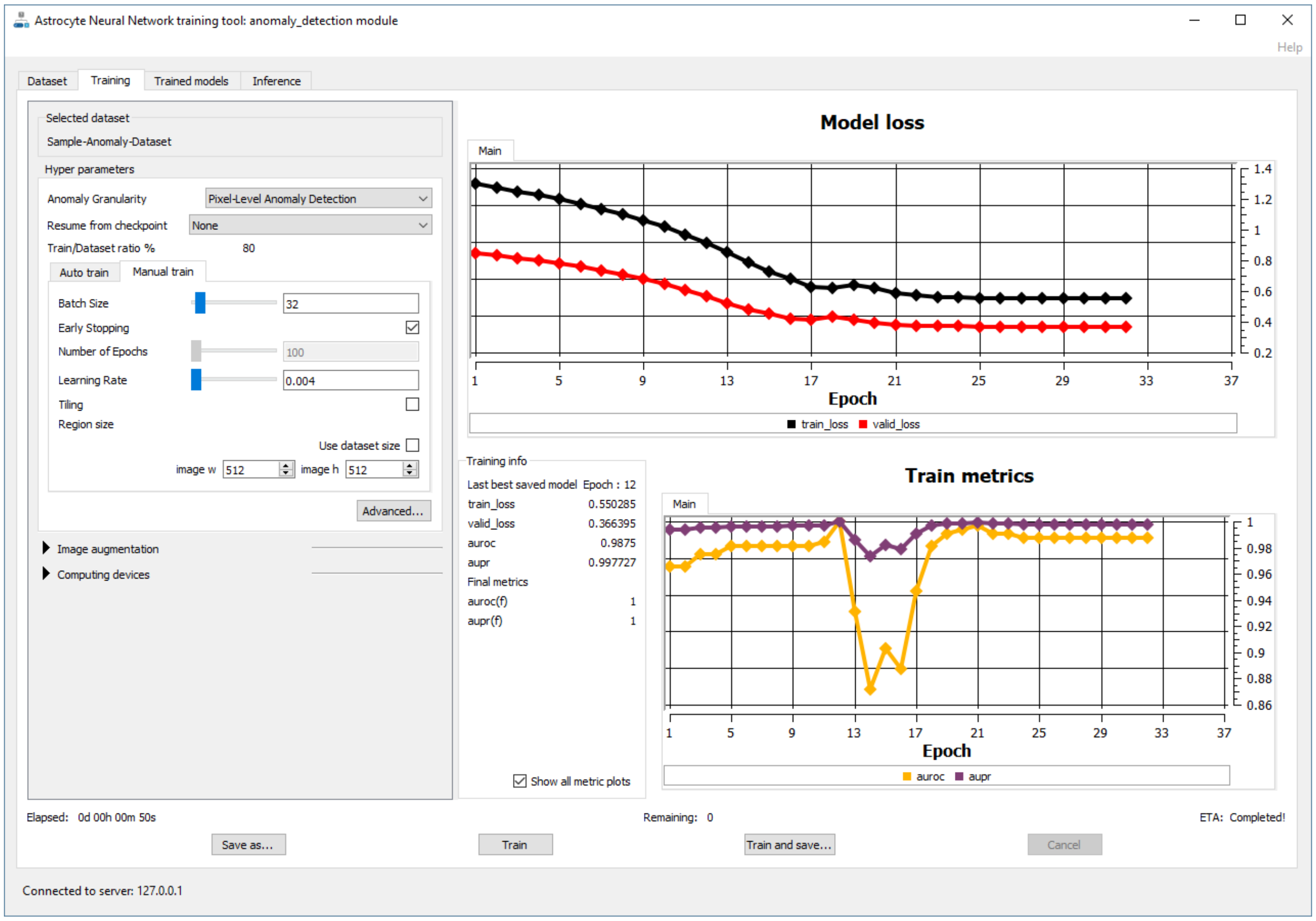
Images tagged for removal are only deleted when the **Save changes** command is executed (available in the dataset pop-up menu by right-clicking on the dataset entry).



Dataset name	Image dimensions	Updated	Description
Metal	1024 x 1	Mon Jun 21 11:11:40 2021	Sample metal dataset with good/bad classes.

Training Models

The Training tab groups the settings and functions for training models. The main sections provide for selection of dataset, the computing device on which to train and current memory usage during training, the training hyper parameters, image augmentation, as well as graphs, such as the training loss and validation loss, that display real-time feedback on the training process. See the Guidelines for Model Creation for more information on how to successfully train models.



Astrocyte Neural Network training tool: anomaly_detection module

Selected dataset
Sample-Anomaly-Dataset

Hyper parameters
Anomaly Granularity: Pixel-Level Anomaly Detection
Resume from checkpoint: None
Train/Dataset ratio %: 80

Auto train | Manual train
Batch Size: 32
Early Stopping: ☑
Number of Epochs: 100
Learning Rate: 0.004
Tiling: ☐
Region size
Use dataset size ☐
image w 512 image h 512

Training info

Last best saved model	Epoch : 12
train_loss	0.550285
valid_loss	0.366395
auroc	0.9875
aupr	0.997727
Final metrics	
auroc(f)	1
aupr(f)	1

☑ Show all metric plots

Elapsed: 0d 00h 00m 50s
Remaining: 0
ETA: Completed!

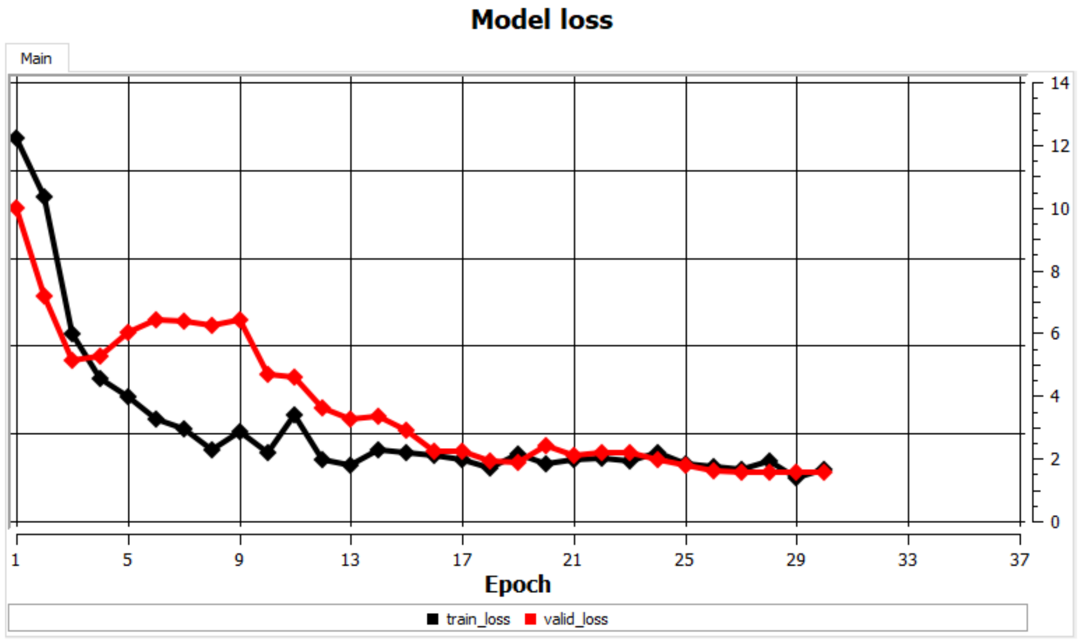
Connected to server: 127.0.0.1

Training Loss Function

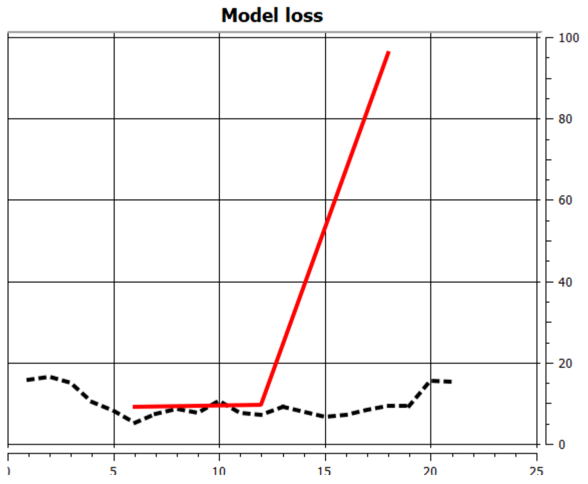
The goal of the model training is to best estimate a target function (f) that maps input data (X) onto output variables (Y). This measures how accurate the model is during training.

The training loss (cost) function represents how well the model fits the given inputs (the training set of images) to the correct output (label). During training the cost function ideally converges over epochs to the optimal solution with the lowest cost (false outputs).

Astrocyte displays a real-time graph of the training loss (cost function) and validation loss during training. The training loss is calculated after processing of each batch (subset of training images) and the graph is then updated with the result. Validation loss (in red) is calculated after each epoch.



As training proceeds the cost function should converge to the lower bound; if it diverges or varies greatly from the lower bound this indicates that the training image set is inadequate or that hyper parameters need to be adjusted. In this case, Astrocyte automatically stops training (a message box is displayed); adjust the image dataset or hyper parameters changed before restarting the training.



In general, the most likely issue is related to deficiencies in the image dataset (for example, misidentified category labels or poor image selection).

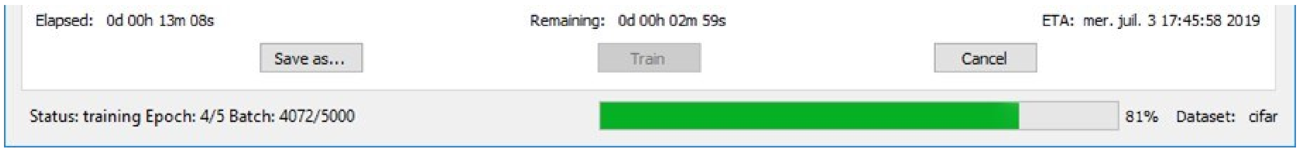
A very low training loss can indicate that the model has effectively memorized the training set (overfitting); in this case, additional images may need to be included in the dataset or a smaller model architecture may provide better results.

Overfitting

When training a model, it is possible that the model stops learning important features but instead starts overfitting to the training set. That is, it becomes very good at recognizing the precise images in the training set but does not perform well on images it has not seen. For example, to identify overfitting during training, the accuracy on the training set continues to increase while the accuracy on the validation set stays the same (or decreases). To avoid overfitting stop the training process as soon as overfitting is recognized.

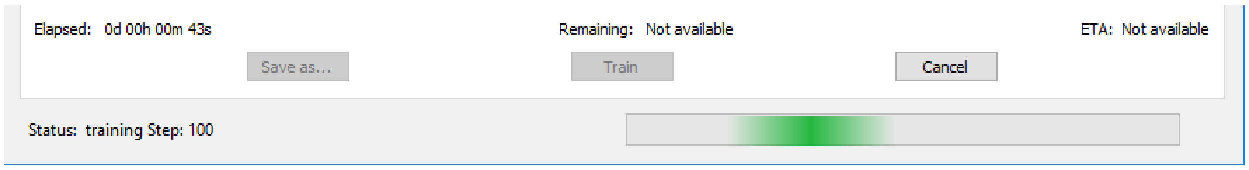
Training Status Bar

The Status bar indicates the total number of training and validation steps for the current training session. The number of training steps varies depending on the dataset size, number of epochs, batch size and other factors. The elapsed and remaining time, including the estimated time of arrival (ETA) of completion are also provided.


Elapsed: 0d 00h 13m 08s Remaining: 0d 00h 02m 59s ETA: mer. juil. 3 17:45:58 2019

Status: training Epoch: 4/5 Batch: 4072/5000 81% Dataset: cifar

However, if Early Stopping is enabled for training the total Step count, Remaining and ETA times are not available; only the Elapsed time and current training step are displayed.


Elapsed: 0d 00h 00m 43s Remaining: Not available ETA: Not available

Status: training Step: 100

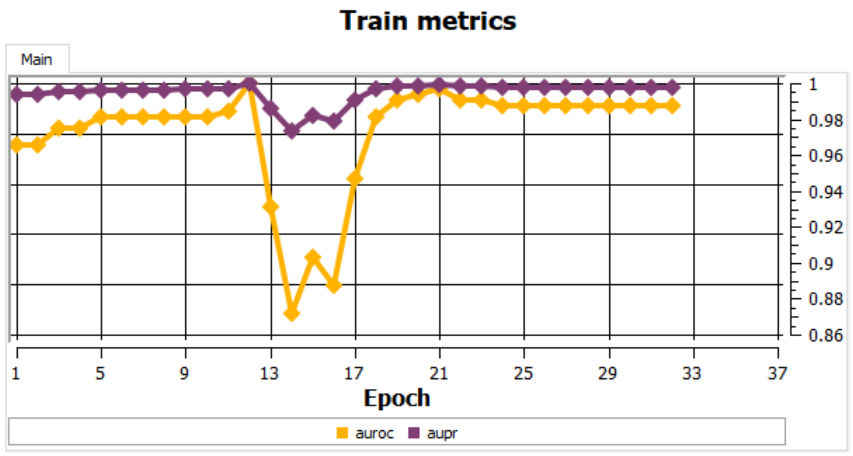
Training Analysis Graphs

In addition to the training and validation loss graph, the anomaly detection, classification and segmentation modules have additional training analysis graphs that display real-time feedback during training.

To enable the display of all analysis graphs, use the **Show all metric plots** checkbox.

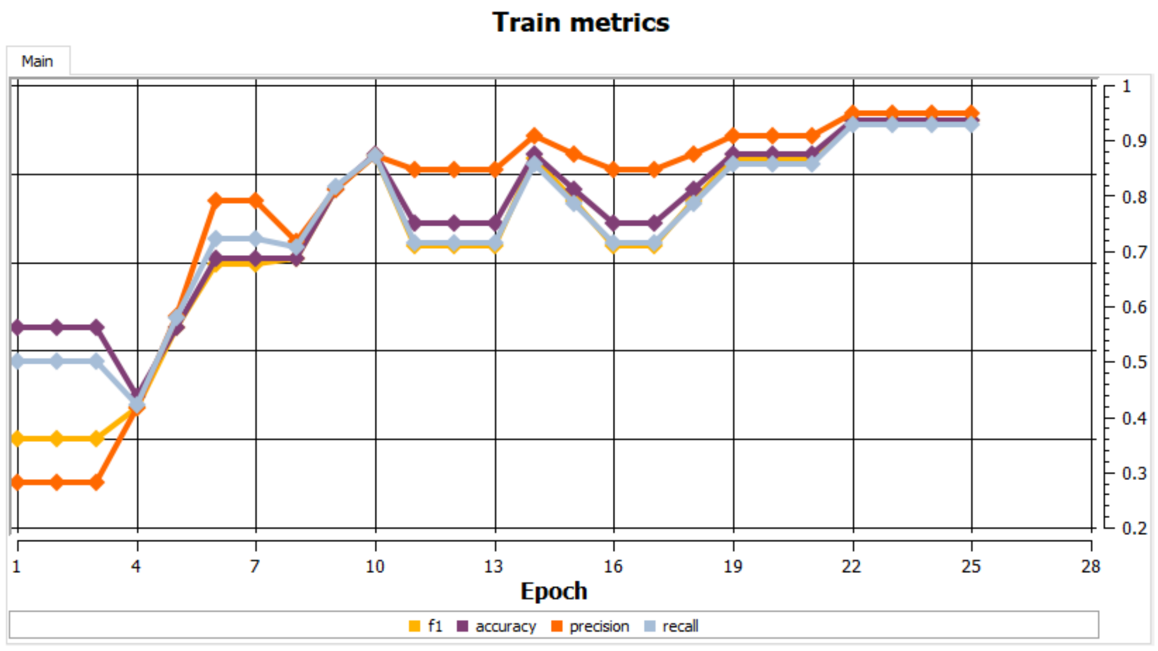
Anomaly Detection Training Analysis Graphs

When segmentation training is completed AUPR and AUROC graphs are displayed. For more information on AUPR and AUROC, see the Anomaly Detection Model Test Results section.



Classification Training Analysis Graphs

For classification training, additional graphs display the accuracy, f1, precision and recall statistics calculated after each training epoch.



Accuracy Training Metric

Accuracy indicates how well the model accurately identifies the labelled object correctly; it is the fraction of images that are correctly classified. That is, accuracy = number of correct predictions / total number of predictions for all categories:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 Score

The F1 score is used to measure a test's accuracy and is the harmonic mean between precision and recall. The F1 score tries to find the balance between precision and recall. The range for F1 Score is [0, 1] and indicates how precise the classification model is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). The greater the F1 Score, the better is the performance of the model. Mathematically, it can be expressed as:

$$F1 = 2\,x\,\frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

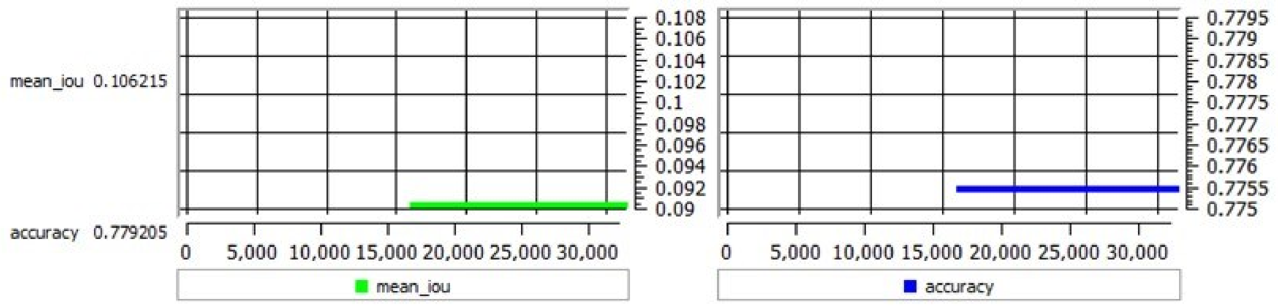
For example, high precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify.

Segmentation Training Analysis Graphs

When segmentation training is completed Mean IOU and Accuracy graphs are displayed.

Accuracy gives the percentage of correctly predicted segment pixels compared to the ground truth segment pixels.

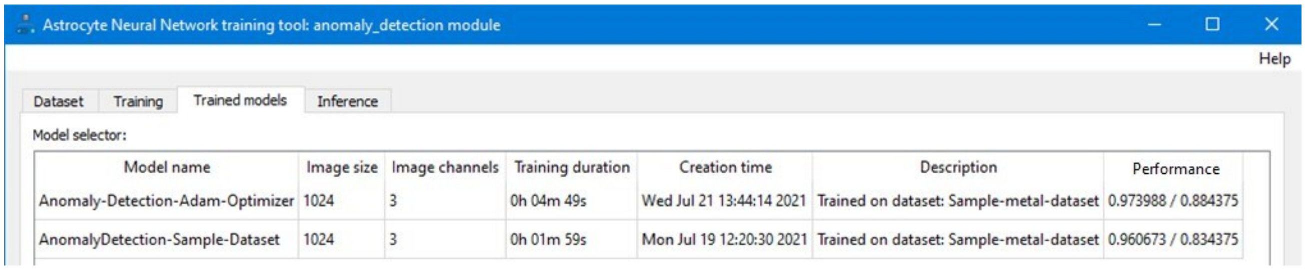
The mean IoU measures the average IoU for all labelled segments (including the background).



Trained Models Tab

The Trained models tab lists the previously trained models for the Astrocyte module.

Information included the creation time and the relevant training test metrics, for example accuracy and mean IoU for segmentation models.


Astrocyte Neural Network training tool: anomaly_detection module
Help
Dataset
Training
Trained models
Inference
Model selector:

Model name	Image size	Image channels	Training duration	Creation time	Description	Performance
Anomaly-Detection-Adam-Optimizer	1024	3	0h 04m 49s	Wed Jul 21 13:44:14 2021	Trained on dataset: Sample-metal-dataset	0.973988 / 0.884375
AnomalyDetection-Sample-Dataset	1024	3	0h 01m 59s	Mon Jul 19 12:20:30 2021	Trained on dataset: Sample-metal-dataset	0.960673 / 0.834375

Model Details

To display the image augmentation and hyper parameters used to train the model, double-click on the entry to open the Model details dialog. For image augmentation, only those enabled for training are included.

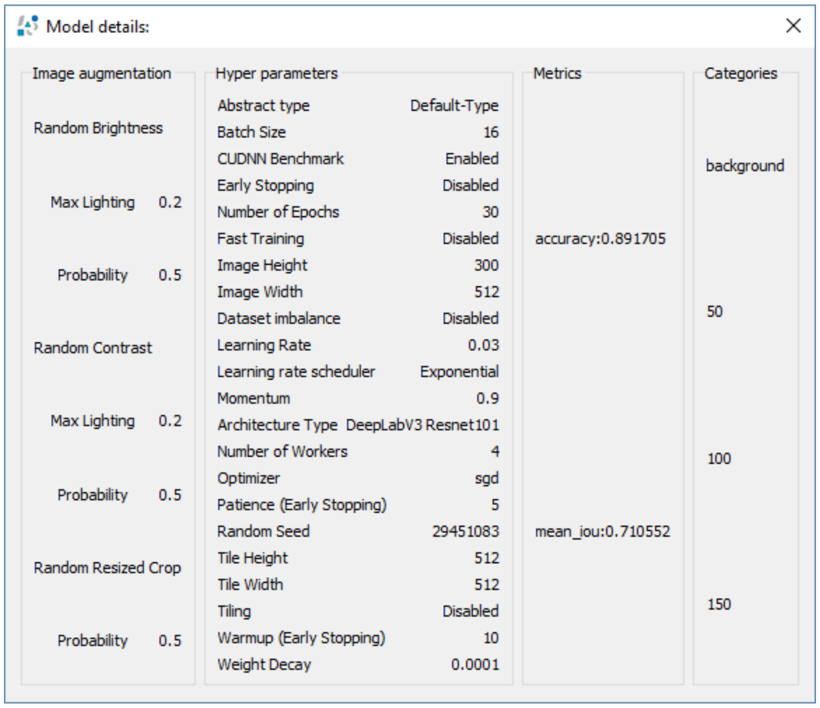


Image augmentation

Random Brightness	
Max Lighting	0.2
Probability	0.5
Random Contrast	
Max Lighting	0.2
Probability	0.5
Random Resized Crop	
Probability	0.5

Hyper parameters

Abstract type	Default-Type
Batch Size	16
CUDNN Benchmark	Enabled
Early Stopping	Disabled
Number of Epochs	30
Fast Training	Disabled
Image Height	300
Image Width	512
Dataset imbalance	Disabled
Learning Rate	0.03
Learning rate scheduler	Exponential
Momentum	0.9
Architecture Type	DeepLabV3 Resnet101
Number of Workers	4
Optimizer	sgd
Patience (Early Stopping)	5
Random Seed	29451083
Tile Height	512
Tile Width	512
Tiling	Disabled
Warmup (Early Stopping)	10
Weight Decay	0.0001

Metrics

accuracy:0.891705

mean_iou:0.710552

Categories

background

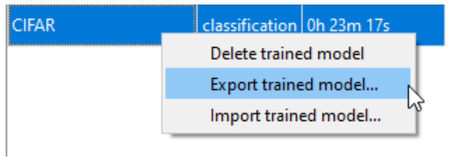
50

100

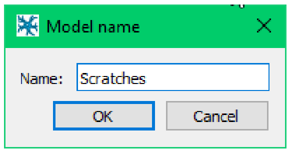
150

Importing, Exporting and Deleting Models

Use the right-click menu to delete or export a selected model. Models can also be imported using this menu.



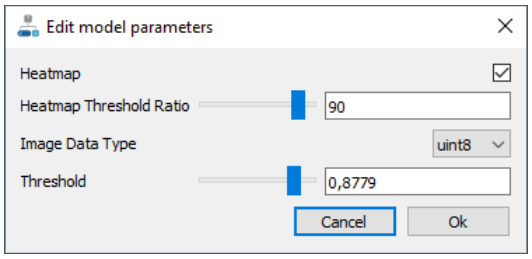
When importing models, the model name can be modified:



Name: Scratches

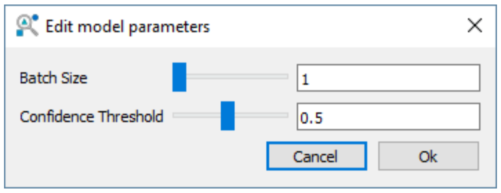
Exported models can be deployed using Teledyne DALSA's Sapera Processing SDK to analyze and process target images.

For anomaly detection models, exporting requires specifying the required anomaly threshold, as determined appropriate by model testing. The heatmap threshold must also be provided which is expressed as a ratio of the anomaly threshold. For older models, the Input Data Type parameter is also available.



Heatmap ✓
Heatmap Threshold Ratio 90
Image Data Type uint8
Threshold 0,8779

For object detection, exporting requires setting an appropriate confidence threshold, as determined by model testing. The batch size to use at inference is also configurable. To do so, click **Edit** to specify the confidence threshold and click **OK**:



Batch Size 1
Confidence Threshold 0.5

Hyper Parameters

Hyper parameters control the training process. For all Astrocyte modules, the Training tab includes a Hyper parameters section which organizes hyper parameters into Manual and Auto- train tabs.

Guidelines on Adjusting Hyper Parameters

The following are general guidelines for adjusting the main hyper parameters when trying to improve the overall performance of the model.

Number of Epochs

The number of epochs defines the number of times that the learning algorithm will work through the entire training dataset. To choose the right number of epochs for the training step, the metric to monitor is the validation loss. In general, train the model for as many epochs as long as the validation loss keeps decreasing. If unsure as to what is an appropriate number of epochs enable the Early Stopping hyper parameter option to automatically stop training when learning ceases.

Learning Rate

The learning rate controls how much to change the model in response to the estimated error each time the model weights are updated. If the learning rate is excessively smaller than optimal values, it will take a much longer time (hundreds or thousands of epochs) to reach an ideal state. On the other hand, if the learning rate is much larger than optimal value, then it would overshoot the ideal state and the algorithm might not converge.

Batch size

The batch size defines the number of samples to work through before updating the internal model parameters (neuron weights). A larger batch size allows computational boosts that utilizes matrix multiplication in the training calculations. However, it comes at the expense of needing more memory for the training process. A smaller batch size induces more noise in error calculations, often more useful in preventing the training process from stopping at local minima.

In general, it is recommended to start with a relatively small batch size and monitor memory usage to find an ideal value that fits within the training constraints (hardware, model architecture, image size, and so forth).

Commonly Observed Symptoms

The most often observed symptoms are:

- low accuracy model
- out-of-memory
- model not converging.

The following guidelines provide actions to take in such cases.

Poor Performance Metrics

Validation loss is usually a proxy for the metrics (for example, f1, mAP, IoU) the model is seeking to optimize. When metrics are poor it means the validation loss is also not good, therefore a generalization problem (overfitting) usually exists. One possible approach is first, make sure the test images are similar to training images. If this is the case, collect more images (data) is required.

Out of Memory

It is possible either to make the image resolution or batch size smaller. Smaller batch size means more noise during the training phase and slower convergence. Smaller image resolution means less information from the input. There should be a balance between these two.

Training does not Converge

If the training loss is high the model is not powerful enough to solve the problem. One possible solution is to make the model deeper (select another larger model architecture from the advanced hyper parameters). If the training loss is slowly getting decreased, using a greater learning rate or batch size can help increase the convergence.

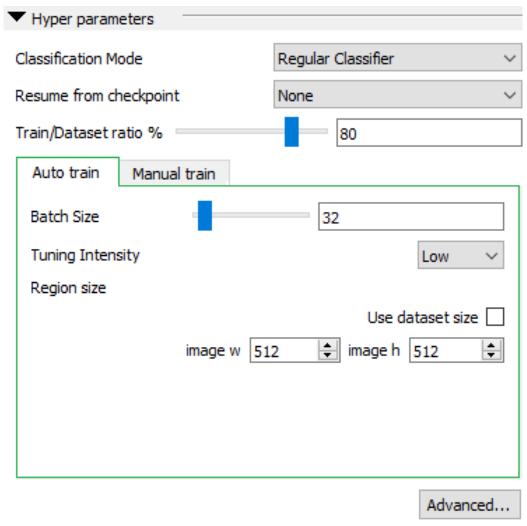
Manual train vs Auto train

Hyper parameters can be set using one of two methods:

- Manual train
- Auto train

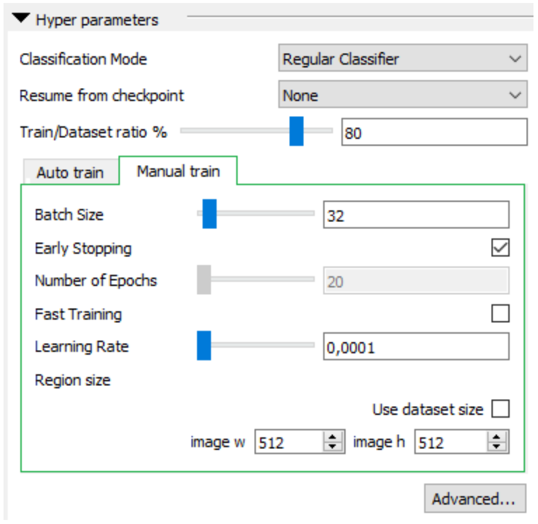
Selecting (enabling) a tab determines how hyper parameters are set for the training operation.

Auto-train Enabled



Hyper parameters
Classification Mode: Regular Classifier
Resume from checkpoint: None
Train/Dataset ratio %: 80
Auto train | Manual train
Batch Size: 32
Tuning Intensity: Low
Region size
Use dataset size ☐
image w 512 image h 512
Advanced...

Manual Enabled



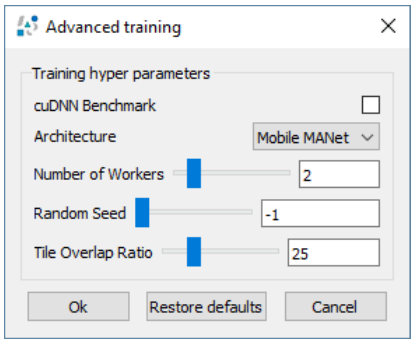
Hyper parameters
Classification Mode: Regular Classifier
Resume from checkpoint: None
Train/Dataset ratio %: 80
Auto train | Manual train
Batch Size: 32
Early Stopping ☑
Number of Epochs: 20
Fast Training ☐
Learning Rate: 0,0001
Region size
Use dataset size ☐
image w 512 image h 512
Advanced...

Auto Training

The **Auto train** option trains models using different “Trials”. For each trial, Astrocyte automatically adjusts hyper parameters and executes the training for a number of epochs. With each trial, hyper parameters are adjusted given past results. At the end of the entire training cycle, results for each trial are compared and the result with the best result are kept.

The **Tuning Intensity** determines the optimization intensity (Low, Medium or High); the number or trials increases with the Tuning Intensity setting. For example, for Segmentation, the following advanced hyper parameters are available (which is similar to other modules):

When using the Auto Train option, the available Advanced hyper parameter set is reduced compared to the Manual training set since these hyper parameters are being adjusted during the training trials.

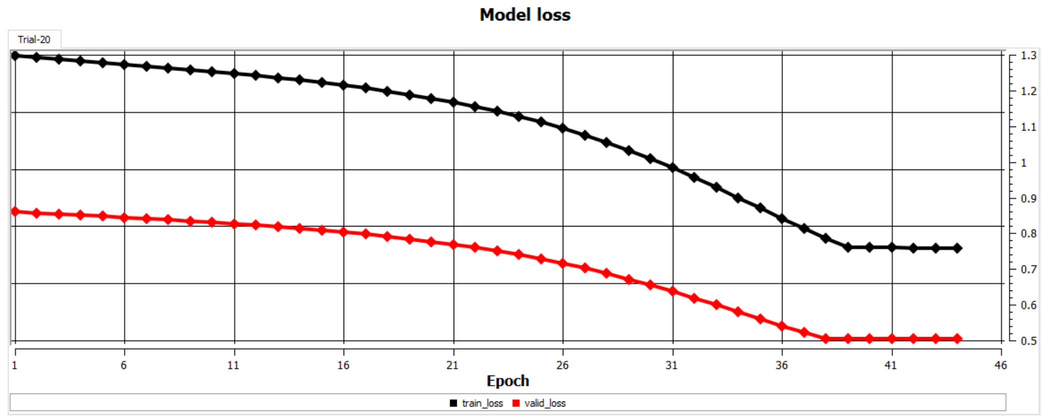


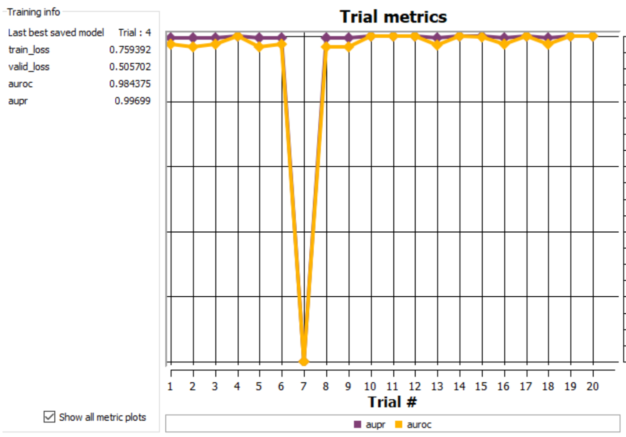
Advanced training

Training hyper parameters

cuDNN Benchmark	☐
Architecture	Mobile MANet
Number of Workers	2
Random Seed	-1
Tile Overlap Ratio	25

The Auto train option also provides an additional graph for Trial metrics.

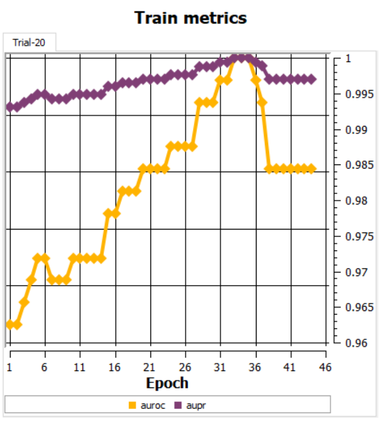




Training info

Last best saved model	Trial : 4
train_loss	0.759392
valid_loss	0.505702
auroc	0.984375
aupr	0.99699

☑ Show all metric plots



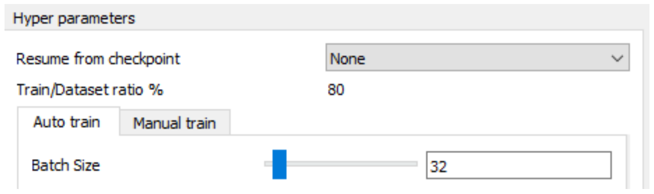
Basic Hyper Parameters

Basic hyper parameters for all modules, for both Auto and Manual training options, are:

Resume from checkpoint: This allows model training to begin from a previously trained model. This can be useful since if a model has been previously trained on the same or similar dataset, model parameters have already been tuned. For image-based learning, such as that used by Astrocyte, this can leverage previous training cycles since identifying basic image characteristics such as edges and corners are common to most applications.

Train/Dataset ratio %: Displays the Train/Dataset ratio % set in for the selected dataset. This is the percentage of images assigned to the Train portion (and inversely, the Validation portion).

Batch Size: The number of images the learning algorithm processes in a single pass before updating the model's internal parameters. A larger batch size reduces the number of times the model's weights are adjusted: it increases the number of data points usedfor fitting the cost function and uses more GPU resources as the images must be loaded into memory resulting in faster training. For example, a batch size of 512 reduces the number of iterations compared to a batch size of 32 by a factor of 16. However, if a dataset contains an insufficient number of images or too many categories, a large batch size can result in poorer accuracy. The batch size limit is determined by the image size, the available memory resources of the GPU and other hardware considerations.

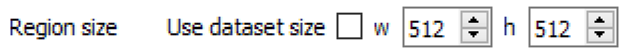


Resume from checkpoint: None
Train/Dataset ratio %: 80
Batch Size: 32

Region Size

The Region Size parameter is available for all modules. The Region size determines the input image size to the neural network during training and can be smaller than the original dataset image size.

This allows using a single high-resolution dataset to train on lower resolutions to determine the best input size for optimal speed and performance for inference operations.



Region size Use dataset size ☐ w 512 h 512

When Tiling is disabled, images are automatically resized during training to the region size. Non-square region sizes are supported. When Tiling is enabled, images are subdivided into tiles of dimensions corresponding to the region size.

Note: For Object Detection YOLOX architectures the region size must be a multiple of 32, with a minimum region size of 224x224.

.

Manual Hyper Parameters

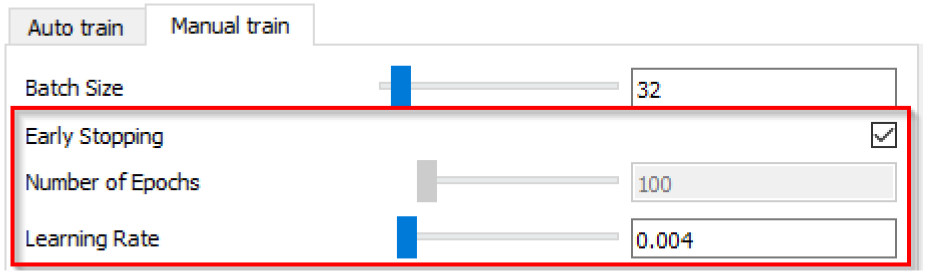
Hyper parameters available for all modules in the Manual train tab that can be set by the user before training include:

Early Stopping: Early stopping is a form of regularization used to avoid overfitting. When enabled, it will stop training when the model has stopped improving during the validation phase of training; the number of epochs is automatically determined (the Number of Epochs hyperparameter setting is ignored). When early stopping is enabled the Remaining and ETA times are not available. The Advanced hyper parameters Patience and Warmup provide additional control over Early Stopping performance.

Number of Epochs: The number of times that the learning algorithm works through the entire training dataset. One epoch means that each image in the training dataset has had an opportunity to update the internal model parameters.

Learning Rate: The learning rate determines how fast the model learns. It adjusts how much weight change is applied when updating internal model parameters after processing each batch of images in the dataset. If weights change too quickly, the model overcorrects and the training diverges, resulting in a poor-quality model with bad

performance. The learning rate value is a small real value such as 0.1, 0.001 or 0.0001 (valid range is 1 to 0.00000001). Different values for the specific datasets can provide optimal results; you may try different values to see which works best.

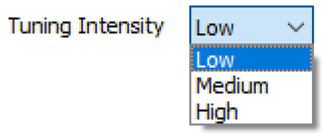


Batch Size: 32
Early Stopping: ✓
Number of Epochs: 100
Learning Rate: 0.004

Auto Train Hyper Parameters

When enabled, the **Auto train** option trains multiple times in succession; the model with the best performance is returned as the final model result. Following each training session or “trial”, Astrocyte automatically adjusts various hyper parameters given the previous trial results to improve performance. The number of trials to perform is controlled by the Tuning Intensity parameter:

Tuning Intensity: Sets the number of trials to perform when automatically tuning hyper parameters. A trial is a complete training operation that creates a model. Available settings include:



Low: 20 trials

Medium: 40 trials

High: 60 trials

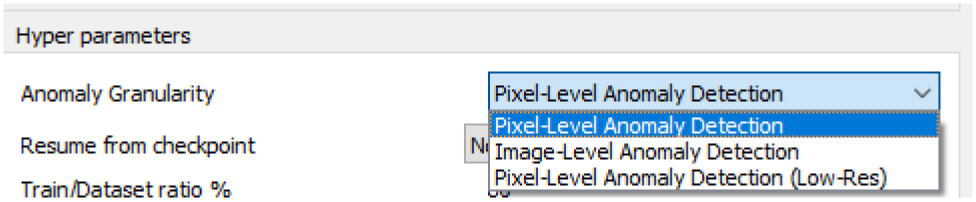
Module-Specific Hyper Parameters

Anomaly Detection

Anomaly Detection module-specific hyper parameters include:

Anomaly Granularity: Determines the anomaly detection model type. 3 options are available:

- *Pixel-Level Anomaly Detection:* This method aims to precisely segment anomalous regions in the image to determine its whether it is considered an anomaly. It is faster and more robust than the original *Image-Level Anomaly Detection* type and also includes support for generation of heat-maps during inference.
- *Pixel-Level Anomaly Detection (Low Res):* Low resolution version of pixel level anomaly detection for images where anomalies are present as larger percentages of the overall image and higher model resolution (to detect relatively small anomalies) is not required.
- *Image-Level Anomaly Detection*: This method is available for backwards compatibility; it is recommended that new models use the *Pixel-Level Anomaly Detection* type.



Hyper parameters

Anomaly Granularity: Pixel-Level Anomaly Detection

- Pixel-Level Anomaly Detection
- Image-Level Anomaly Detection
- Pixel-Level Anomaly Detection (Low-Res)

Resume from checkpoint

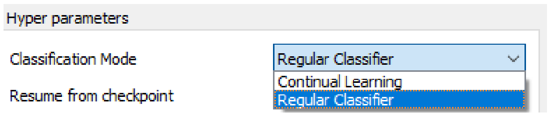
Train/Dataset ratio %

Classification

Classification module-specific basic hyper parameters include:

Classification Mode: Determines the classification model type. 2 options are available:

- *Continual Learning*: Extends the regular classifier for continual learning. A continual learning model has the ability to learn from new data samples at runtime without the need to retrain from scratch.
- *Regular Classifier:* Classifier based on training dataset with fixed set of category labels.



Hyper parameters

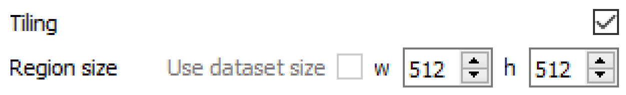
Classification Mode: Regular Classifier

- Continual Learning
- Regular Classifier

Resume from checkpoint

Tiling

The [Tiling]option is available for the Anomaly Detection, Object Detection and Segmentation modules. When enables the images are subdivided into tiles of dimensions corresponding to the region size instead of being resized.

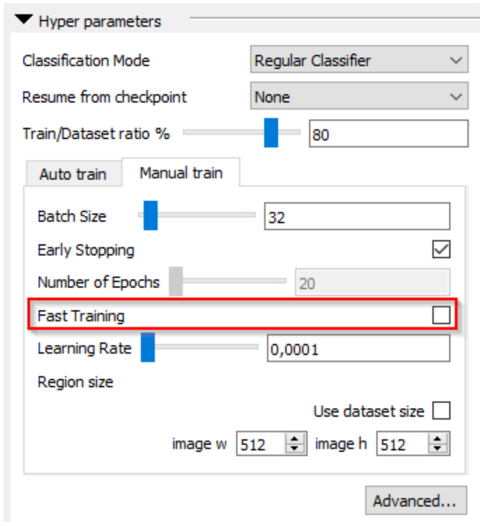


Tiling ✓

Region size: Use dataset size ☐ w 512 h 512

Fast Training

The Fast Training option is available (in the Manual train tab). For the Classification, Object Detection and Segmentation modules (all modules except Anomaly Detection). Fast Training accelerates the training process by only modifying the last layer of the model, keeping the backbone pre-trained layer unchanged. Fast Training can be effective for image datasets that are similar to the dataset on which the model is pretrained (ImageNet for classification and PASCAL VOC for object detection and segmentation) and can provide a quick proof-of-concept model.



Hyper parameters

Classification Mode: Regular Classifier
Resume from checkpoint: None
Train/Dataset ratio %: 80

Auto train | Manual train

Batch Size: 32
Early Stopping: ☑
Number of Epochs: 20
Fast Training: ☐
Learning Rate: 0,0001
Region size
Use dataset size ☐
image w 512 image h 512

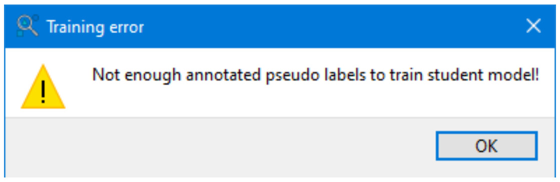
Self-Training Object Detection

For the Object Detection module, a Semi-Supervised Object Detection (SSOD) Self-Training option is available. Self-Training automatically generates bounding boxes and corresponding labels (pseudo labels) for unlabelled images in a dataset.

To enable Self-Training the number of unlabelled images must be greater than ~50% of the total number of images in the dataset. Self-Training produces acceptable results when sufficient labelled images with bounding boxes are available in the dataset for each category; the minimum number of labelled images is dependent the image dataset and number of categories and the amount of variation in the images.

Self-Training first trains a standard object detection model on the labelled images in the dataset that contain user-defined ground truths to create a “Teacher” model. This Teacher model is then applied to the unlabelled images to identify objects and create bounding boxes and labels in the unlabelled images. To generate pseudo labels, objects must be identified by the Teacher model with a confidence score of at least 50%.

For Self-Training to proceed, the number of images with pseudo labels generated must equal that of the number of images with user-defined ground truths otherwise an error message is generated.



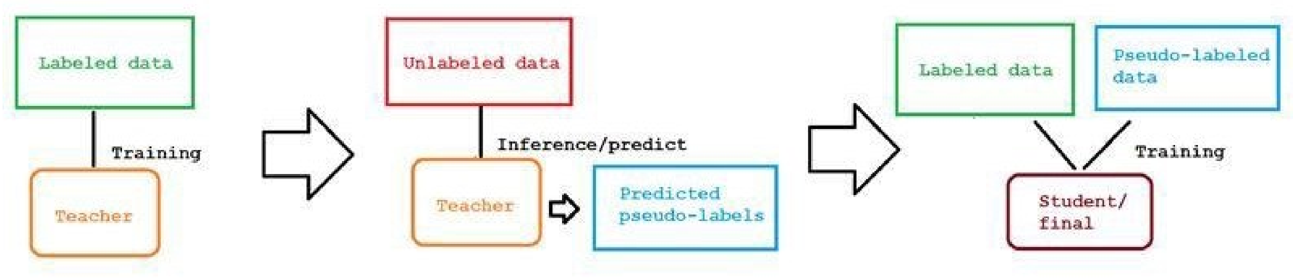
Training error

Not enough annotated pseudo labels to train student model!

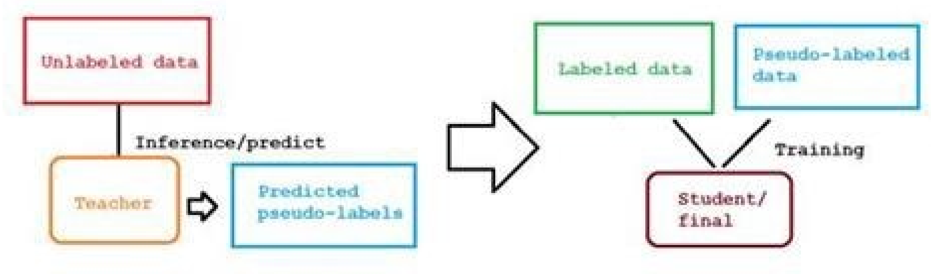
To solve this issue either increase the number of labelled images (keeping below the 50% total threshold, otherwise images may need to be added to the dataset) or try increasing the number of training epochs.

Note: any pseudo-labels generating during this aborted training are saved to the dataset.

If this threshold is met, training proceeds with a “Student” model that trains on the images with both the ground truth and pseudo labels, using different weights for each type. The final resulting trained model is equivalent to a standard object detection model.



When using the **Resume from checkpoint** option, the previously trained object detection model selected is used as the Teacher model to identify pseudo labels; the Self-Training step to create a Teacher model is not performed and the training time is therefore faster. The previously trained model must have trained on a similar image dataset and have the same number of categories as the current image dataset.

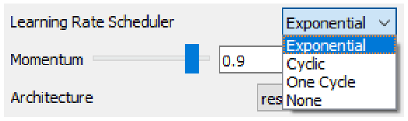


The image dataset is automatically updated and saved on the Cortex server with the generated pseudo-labels. In the Dataset, images with pseudo labels are displayed in pink and can be filtered using the Pseudo label checkbox. If pseudo label bounding boxes or category labels are modified, and the image dataset saved, they are considered as ground truth and no longer identified as pseudo labels.

Advanced Hyper Parameters

The available advanced hyper parameters depend on the selected training mode, when using the Auto train option, the available hyper parameters set is limited. When using the Manual train option, advanced hyper parameters for most Astrocyte modules include:

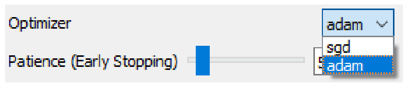
- **CUDNN Benchmark**: When enabled, the convolution algorithm used by the GPU is optimized resulting in faster processing times. This option increases GPU memory usage so may not be suitable for all image datasets if memory usage is already approaching upper limits.
- **Learning rate scheduler**: The learning rate scheduler allows the learning rate to be adjusted at specified epochs as the training proceeds. This can improve performance of model training by lowering the learning rate as the training progresses and the model becomes increasingly fine-tuned; at these later stages a higher learning rate may be counterproductive by over-correcting the model.:
    - *Exponential*: Learning rate is decreased at an exponential rate over training
    - *Cyclic*: Learning rate cycles between higher and lower rates for each epoch, gradually decreasing the rates with each epoch.



    - *One Cycle*: Learning rate cycles between higher and lower rates over the entire training

Object Detection YOLOX architectures use the **LR Warmup** parameter instead.

- **Momentum**: Momentum allows the update direction of the cost function to continue in the same direction towards the optimal global minimum, minimizing the effect of small variations. However, too large a momentum may create oscillations that grow in magnitude.
- **Number of Workers**: denotes the number of processes that generate batches in parallel. A high enough number of workers assures that CPU computations for data generation (loading images from RAM to GPU) are efficiently managed. That is, the GPU is not waiting for image batches to be loaded.
- **Optimizer**: determines how the learning rate is applied and model parameters are updated after each batch. Available options are:
    - *sgd*: Stochastic gradient descent. Maintains a single learning rate for all weight updates.
    - *adam*: Computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.



- **Patience (Early Stopping)**: When Early Stopping is enabled, patience is the number of epochs without improvement to wait before stopping training.
- **Random Seed**: A random seed sets the initial values of the neural network nodes; each random seed value results in the same initial values, allowing for the repeatability of training from the same starting point. The default value -1 picks a random seed instead of forcing one explicitly.

- **Warmup (Early Stopping)**: When Early Stopping is enabled, the warmup period is the number of epochs to wait before enabling the Early Stopping mechanism. This allows the training to stabilize to avoid premature stopping due to possible wider variations in validation during the initial training stages.
- **Weight Decay**: Weight decay causes weights to exponentially decay to zero if no weight update is applied during iterations. In practice this penalizes large weights; it causes the weight to decay in proportion to its size.
- **Tile Overlap Ratio**: The overlap between tiles is used to preserve content along the tile boundaries and to prevent any misses due to image partitioning. The overlap value in percentage is based on the shortest tile side (width, height). In object detection, the overlap ratio is automatically selected based on the size of the annotated objects and the provided value. More specifically, it is the minimum between the provided value and the maximum object-to-tile ratio computed.

For all modules, the model Architecture Type can also be specified; for more information on the available architecture types for each module, refer to Supported Model Architectures.

Anomaly Detection

Anomaly Detection module-specific advanced hyper parameters for Image-Level Anomaly Detection type models include (Pixel-Level Anomaly Detection model types do not use these):

- **Outlier exposure**: Outlier exposure is a collection of random images that are likely not normal and are utilized to improve unsupervised methods. Random images are constructed by manipulation of the original images (for example, adding noise to the original image).

Object Detection

Object Detection has the following module-specific advanced hyper parameters.

- **LR Warmup**: Learning rate warm-up period in epochs. This is the number of epochs where before the learning rate schedule is applied.

Segmentation

Segmentation module specific-advanced hyper parameters include:

Class Imbalance: Available with the Encoder-Decoder (Classic) Segmentation Module. When enabled the learning algorithm is modified to adapt to a dataset wherein the number of pixels representing segmented objects only represent a small percentage of the overall image compared to background pixels.

Supported Model Architectures

The model architecture type determines the size and number of layers in the convolutional neural network. Each architecture class (anomaly detection, classification, object detection and segmentation) supports several different model architectures.

Model architectures can vary the number of hidden layers and other model parameters; the greater the number of model parameters results in a larger model size.

Depending on the dataset, larger model architectures do not necessarily perform better than smaller architectures; it is recommended to experiment with different model architecture types to determine the best fit for the dataset and application constraints. In general, it is recommended to start with smaller architectures, evaluate performance; if performance is poor, then try larger model architectures.

Residual neural network (Resnet) architectures are pretrained. Using a pretrained network allows the transfer of learning to identify common image elements which are found in most classification efforts and is much more efficient than training a network from scratch.

For all modules except object detection the input image size is determined by the maximum image size in the image dataset. For object detection, the input image size is determined by the selected model architecture.

Anomaly Detection Model Architectures

Available anomaly detection model architectures are:

Model Architecture	Description
resnet18 (Pixel-Level only)	18 layer deep, 44MB network with 11 700 000 model parameters. The resnet18 architecture is the only option available for Pixel-Level
vgg16 (Image-Level only)	VGG-16 is a 16-layer convolutional neural network that has been pretrained on the ImageNet dataset. Developed by the Visual Geometry Group (VGG) at University of Oxford.
Alexnet (Image-Level only)	AlexNet is an 8-layer convolutional neural network that has been pretrained on the ImageNet dataset.

Classification Model Architectures

Available classification model architectures, for both Classification Modes (Regular Classifier and Continual Learning), are:

Model Architecture	Description
resnet18	18 layer deep, 44MB network with 11 700 000 model parameters.
resnet50	50 layer deep, 96MB network with 25 600 000 model parameters.
resnet101	101 layer deep, 167MB network with 44 600 000 model parameters.

squeezenet 1.1	Architecture using global average pooling instead of fully connected layers and reduced number of model parameters that results in small model size This type of architecture is useful for deployment in applications where memory resources are limited. Note, for Astrocyte implementation model size is ~5.8MB (model compression is not applied). For more information, refer to Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size (Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer).

Object Detection Model Architectures

YOLOX (You Only Look Once) includes several architecture configurations with a speed/accuracy trade-off. Architectures increase in size from YOLOX-Nano to YOLOX-L.

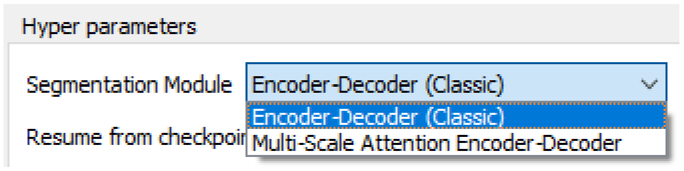
Available object detection model architectures are:

Model Architecture	**Description**
YOLOX-Nano	YOLOX-Nano: (You Only Look Once) single-stage object detector that produces small models (~4 MB) for use with embedded or other resource limited applications where a GPU is typically not available.
YOLOX-S	YOLOX-Small: Default architecture that provides high-performance with a typical model size of ~ 35 MB
YOLOX-Large	YOLOX-Large: Large architecture for datasets with many categories where object sizes and environments vary considerably. Produces large models (~ 200 MB) which increases inference time compared to YOLOX- Small models; recommended only for applications where YOLOX- Small does not perform well comparatively or fails to locate target objects

YOLOX supports non-square input image sizes of with minimum height/widths of 224 pixels, in increments of 32.

Segmentation Model Architectures

Two choices of Segmentation Module are available for training. Depending on the chosen module, different architectures are available in the Advanced training dialog.


Hyper parameters

Segmentation Module Encoder-Decoder (Classic)

Resume from checkpoi

Encoder-Decoder (Classic)
Multi-Scale Attention Encoder-Decoder

For the Segmentation Module *Multi-Scale Attention Encoder-Decoder* the available segmentation model architecture is:

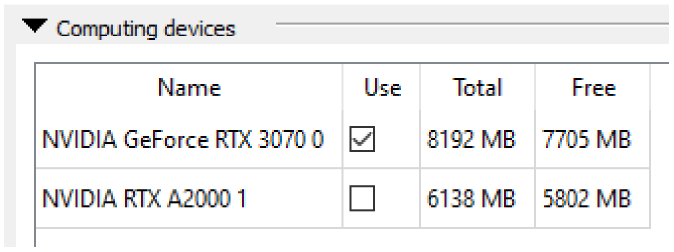
Model Architecture	Description
Mobile MANet	MANet: Multi-scale Attention Net.

For the Segmentation Module Encoder-Decoder (Classic) available segmentation model architectures include:

Model Architecture	Description
DeepLabV3_Resnet50	DeepLab version 3 semantic segmentation model designed and open-sourced by Google, using a pretrained resnet50 architecture.
DeepLabV3_Resnet101	DeepLab version 3 semantic segmentation model designed and open-sourced by Google, using a pretrained resnet101 architecture pretrained.
Unet Resnet50	U-Net convolutional network originally designed for biomedical image segmentation using a resnet50 architecture. Works with fewer training images and is tolerant to touching object categories.
Unet Resnet18	U-Net convolutional network originally designed for biomedical image segmentation using a resnet18 architecture. Works with fewer training images and is tolerant to touching object categories.

Computing Devices

The Computing devices section allows users to select which GPU(s) to use to perform training. All available GPUs are listed. When training in **Auto Train** mode you can select multiple devices. In this case, training trials will be dispatched in parallel on all available devices. When training in **Manual Train** mode you can only select one device at a time.



Name	Use	Total	Free
NVIDIA GeForce RTX 3070 0	☑	8192 MB	7705 MB
NVIDIA RTX A2000 1	☐	6138 MB	5802 MB

Image Augmentation

During model training, image augmentation applies random transformations to the image dataset. This allows the image dataset to be effectively enlarged; augmented images provide the model with additional views of the target object present in the image, but altered slightly.

Instead of presenting the model the same images for each epoch, small random transformations alter the image, changing its pixels values, such that it is still recognizable. The type and probability of applying image augmentation can be specified.

Each mode has image augmentation sets available; depending on the mode, certain image augmentations are supported.

Augmentation	Supported Modes			
	Anomaly Detection	Classification	Object Detection	Segmentation
Mixup			✓	
Random Brightness	✓	✓		✓
Random Contrastt	✓	✓		✓
Random Expand			✓	
Random Hue	✓	✓	✓	✓
Random Horizontal Flip	✓	✓	✓	✓
Random Lighting Noise			✓	
Random Photometric Distortion			✓	
Random Resized Crop	✓	✓	✓	✓
Random Rotate	✓	✓	✓	✓

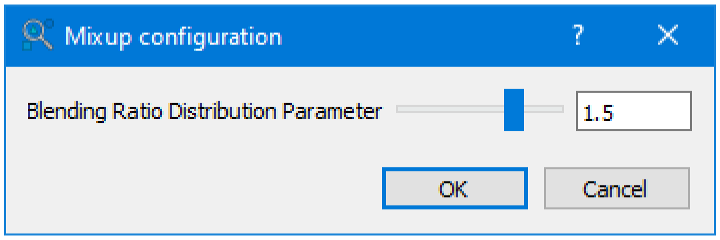
Probability of applying an image augmentation function ranges from 0-1, with 1 = 100% (default values are either 50% or 75%, depending on the function).

⚠	**Note**: Image augmentation transforms may not be suitable for all model training. For example, for character recognition, horizontally flipping a training image can train the model on characters which do not exist, thereby increasing the possibility of false positives.

Mixup

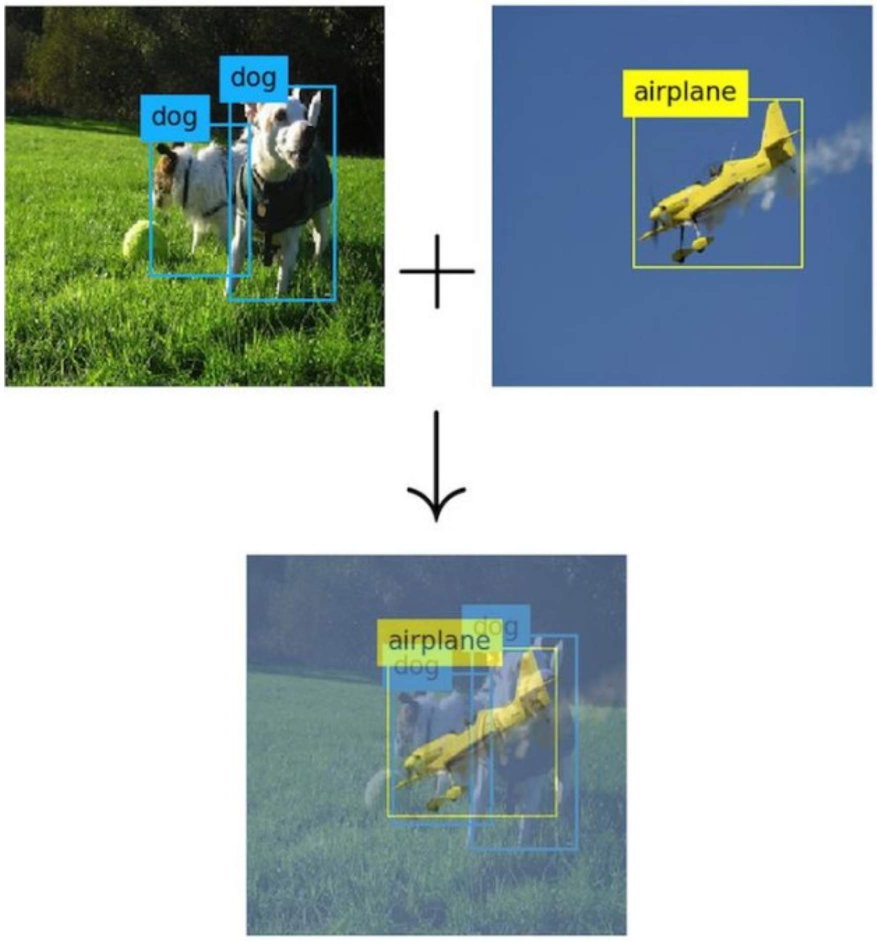
Mixup constructs virtual images by linearly combining sets of training images. This transformation also applies to the ground truth labels that are combined for each set of images.

The Blending Ratio Distribution parameter determines the strength of the image interpolation.



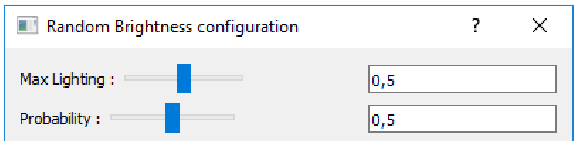
Blending Ratio Distribution Parameter 1.5

For example, for illustrative purposes, combining the following two image results in a virtually contructed “mixup” image that is a blend of the original images, with bounding boxes and labels added from both.



Random Brightness

Performs a random brightness adjustment of the image. The Max Lighting setting determines the range of brightness values to apply, from [-value, value]. For example, a Max Lighting of 0.5 provides a brightness range of -0.5 to 0.5. A Max Lighting of 0 will not transform the original image, while a value of 1 will allow the augmented image to go from completely black to completely white.

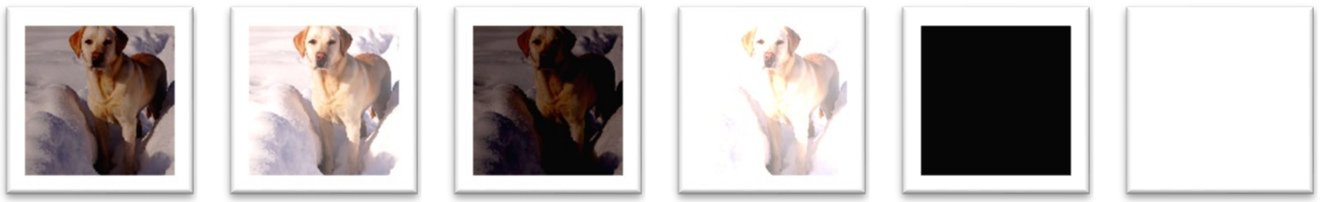

Random Brightness configuration

Max Lighting : 0,5

Probability : 0,5

The following images show the effect of different brightness settings.

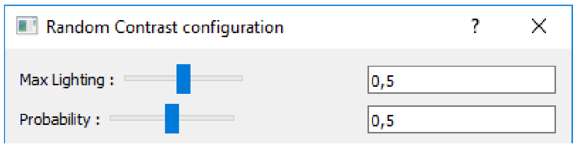
-0.2 **0.2** **-0.5** **0.5** **-1** **1**



The original image:

Random Contrast

Performs a random contrast adjustment of the image. The Max Lighting setting determines the range of contrast values to apply, from [-value, value]. For example, a Max Lighting of 0.5 provides a contrast range of -0.5 to 0.5. A contrast of 0 will not transform the original image; 1 will allow the augmented image to have a gain going from 0 to 2 and therefore going from completely black to twice the original contrast.

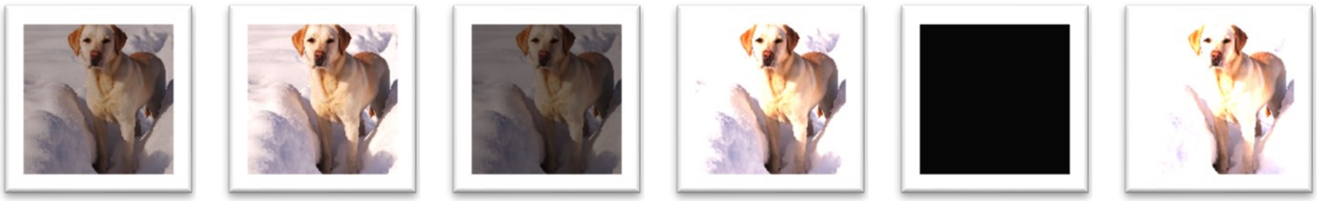

Random Contrast configuration

Max Lighting : 0,5

Probability : 0,5

The following images show the effect of different contrast settings.

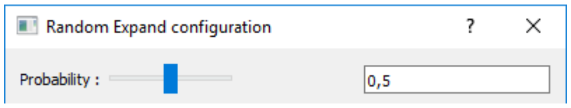
-0.2 **0.2** **-0.5** **0.5** **-1** **1**



The original image:

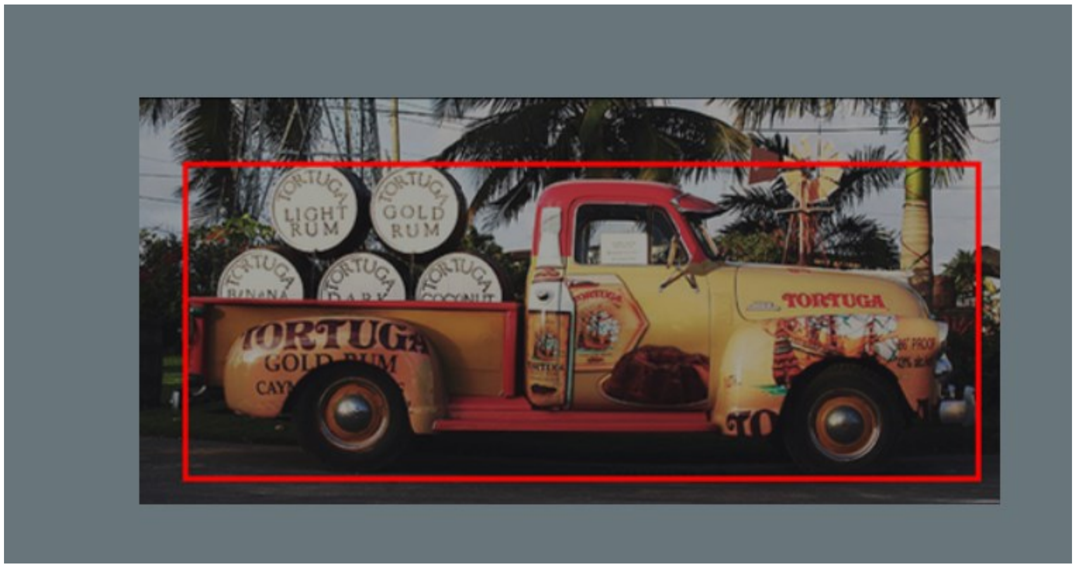
Random Expand

Performs a random expansion of the image size, maintaining the aspect ratio. The expanded image is filled with a fixed pixel value (mean value of 8-bit image). Width and height of the expanded image is calculated by multiplying the width and height of the original image with a randomly selected expansion ratio. Ground truth bounding boxes are translated in the coordinate system of the expanded image.


Random Expand configuration

Probability : 0,5

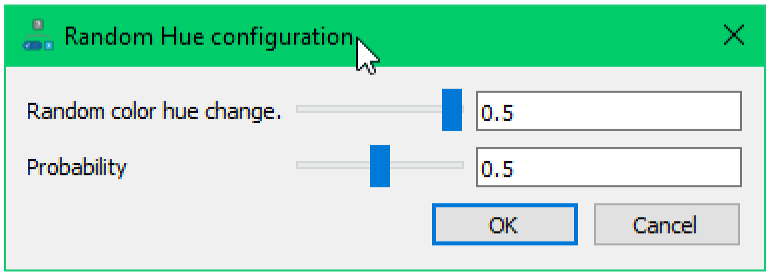
The following image shows the effect of a random expand operation.



Random Hue

Performs a random hue adjustment of the image. Adjust hue of an image. The image hue is adjusted by converting the image to HSV and cyclically shifting the intensities in the hue channel (H). The image is then converted back to original image mode.

The *Random color hue change* factor is the amount of shift in H channel and must be in the interval [0-0.5]. 0.5 gives a complete reversal of hue channel in HSV space in positive direction. 0 means no shift. Therefore, 0.5 will give an image with complementary colors while 0 gives the original image.

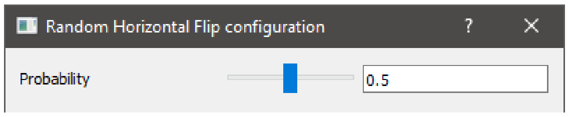

Random Hue configuration

Random color hue change. 0.5

Probability 0.5

Random Horizontal Flip

Performs a random horizontal flip of the image.



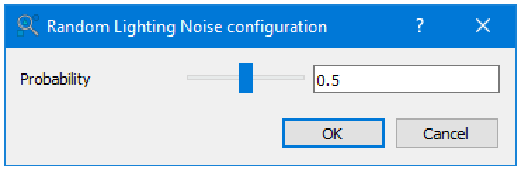
Random Horizontal Flip configuration

Probability 0.5

The following images show the effect of a horizontal flip.

Random Lighting Noise

Lighting noise involves random color channel swaps; for example, swapping the blue and green channels. The probability of applying the transform can be configured.

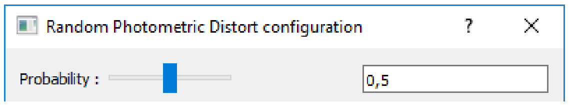


Random Lighting Noise configuration

Probability 0.5

Random Photometric Distortion

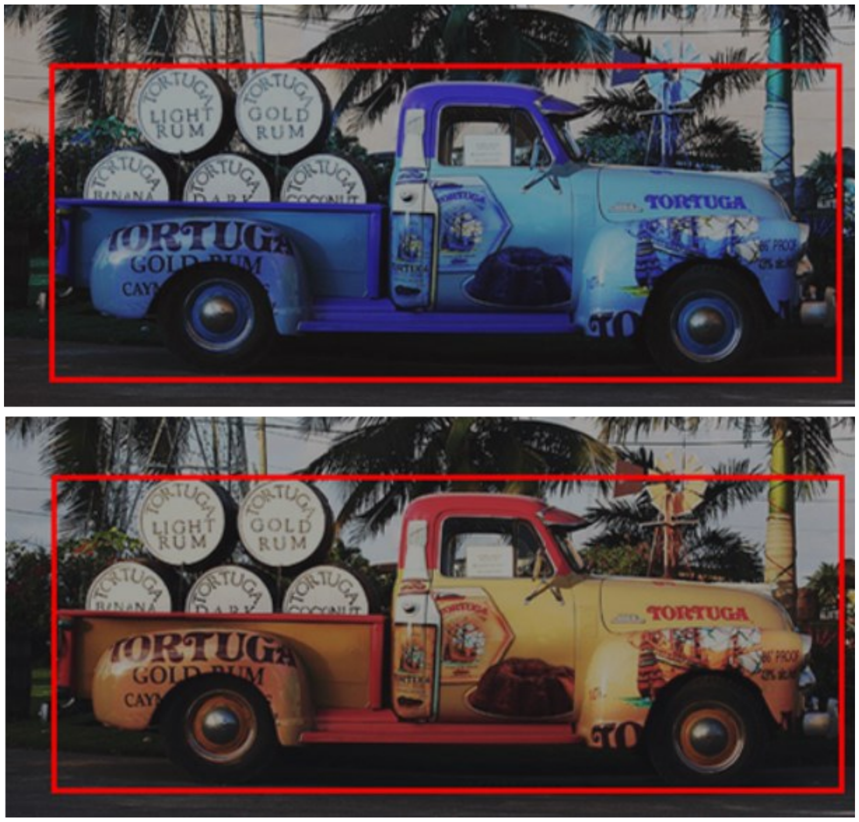
Performs a random photometric distortion of the image. This function combines brightness, contrast, hue, saturation and lighting noise adjustments. Each distortion is applied with the specified probability. Lighting noise involves random color channel swaps; for example, swapping the blue and green channels.



Random Photometric Distort configuration

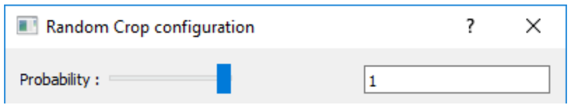
Probability : 0,5

The following image shows the effect of a photometric distortion, in particular, a random color channel swap.



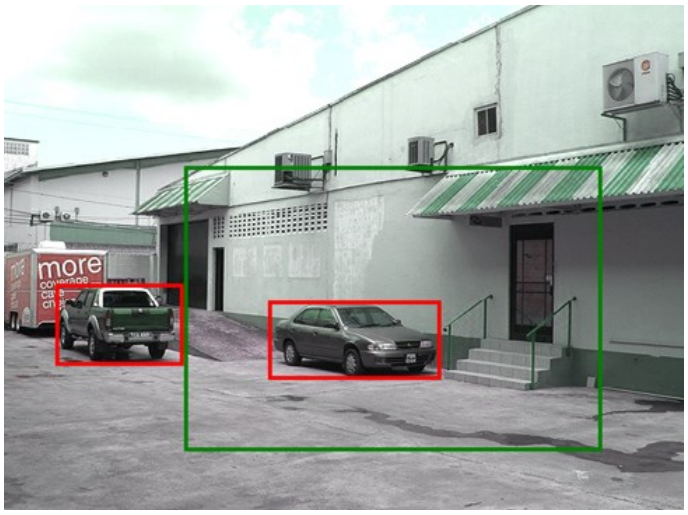
Random Resized Crop

Performs a random crop of the image. The cropped region overlaps the majority of at least one ground truth bounding box contained in the original image.


Random Crop configuration

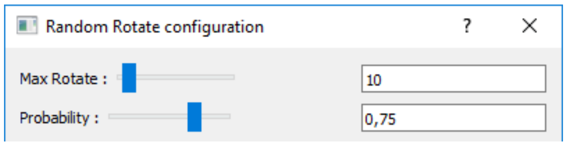
Probability : 1

The following image shows an example of a random crop (image boundaries shown in green).



Random Rotate

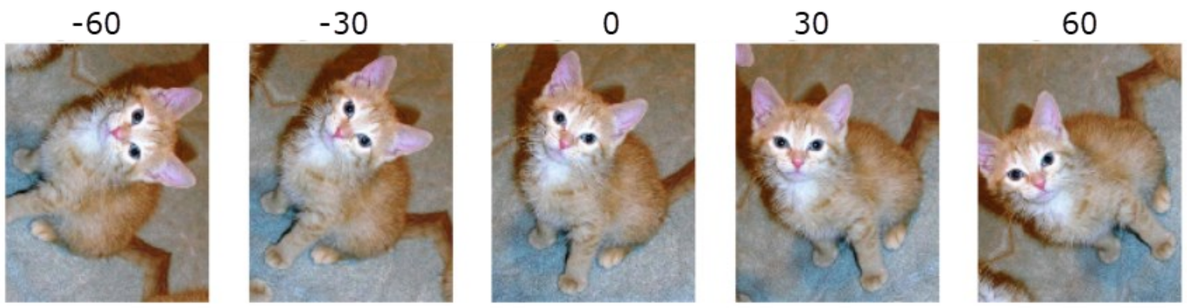
Performs a random rotation of the image; the Max Rotate parameter determines the maximum rotation angle (in degrees), in both the clockwise or counter-clockwise direction.


Random Rotate configuration

Max Rotate : 10

Probability : 0,75

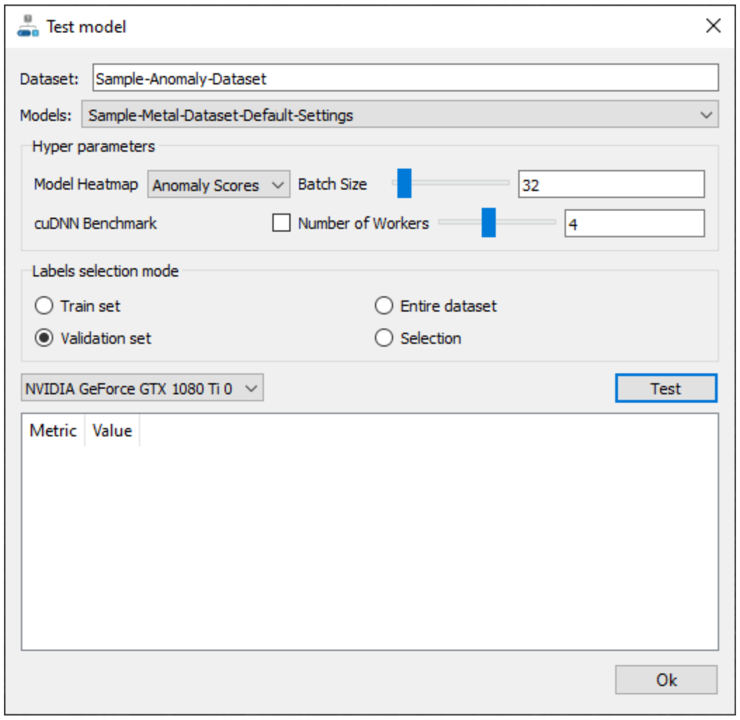
The following images show examples of random rotation:



Testing Models

For all Astrocyte modules, when model training is complete its performance can be tested against an image dataset. Ideally, the model should be tested on images which it has never seen and are not part of the training dataset. Astrocyte allows model testing against the training, validation, entire dataset or a selection of images.

In general, the model testing options are the same for all modules.



Dataset: Sample-Anomaly-Dataset

Models: Sample-Metal-Dataset-Default-Settings

Model Heatmap: Anomaly Scores

Batch Size: 32

cuDNN Benchmark: ☐

Number of Workers: 4

Train set [✓ Validation set] Entire dataset Selection

NVIDIA GeForce GTX 1080 Ti 0

Metric	Value

Testing a model trained on a vastly different dataset than the test dataset might generate less than satisfactory results. If there is no overlap between the model’s categories and those in the test dataset an error message is displayed and the test cannot be conducted.

To test a model:

1. In the Dataset, with the required dataset loaded, click **Test model…**:

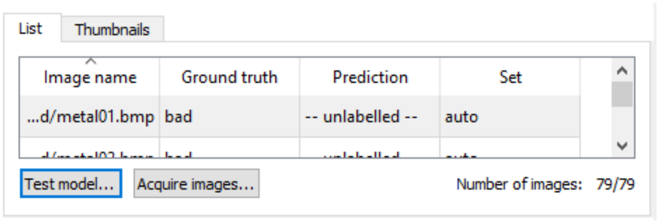
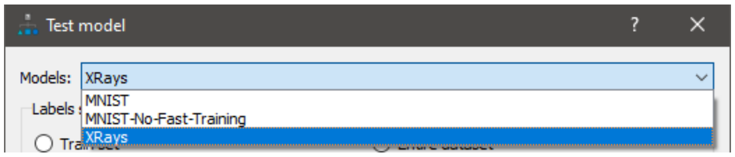


Image name	Ground truth	Prediction	Set
...d/metal01.bmp	bad	-- unlabelled --	auto

Number of images: 79/79

2. Use the Models drop-down list to select a previously trained model:



All models that have been trained, regardless of the selected image dataset, are available. This allows models trained on different image datasets to be tested against any selected image dataset (with at least a subset of the same category labels).

Select the dataset on which to test:

- **Train set**: Subset of training image dataset on which the model was trained.
- **Validation set**: Subset of training image dataset used for validation during model training.
- **Entire dataset**: All images in the training dataset (both validation and training subsets)

Selection: Selected images chosen in the Dataset.

3. Click **Test**; the status bar indicates the test progress. The test execution time is

dependent on the number of images in the selected dataset, the batch size and, in particular, if heatmaps are generated.

Other optional testing options include:

Batch size: The batch size is the number of images processed at one time in the GPU for calculating results; a larger batch size increases the speed of the testing operation. The initial batch size is that used to train the model. The maximum batch size is limited by the memory resources available in the GPU. The batch size does not affect the quality of results, only the amount of time to compute; a larger batch size simply allows processing more images (in a batch) at a time depending of the GPU memory available (the more memory the bigger the possible batch size, the shorter the test time).

CUDNN Benchmark: When enabled, the convolution algorithm used by the GPU is optimized resulting in faster processing times. This option increases GPU memory usage so may not be suitable for all image datasets if memory usage is already approaching upper limits.

Number of Workers: denotes the number of processes that generate batches in parallel. A high enough number of workers assures that CPU computations for data generation (loading images from RAM to GPU) are efficiently managed. That is, the GPU is not waiting for image batches to be loaded.

Model Heatmap: An additional option to include heatmaps for results is also available for all modules.

Object Detection Specific Options

Confidence Threshold: For object detection, the confidence threshold determines the minimum confidence score required for an object to be considered a valid result. A higher threshold may result in less objects being detected but with higher probability.

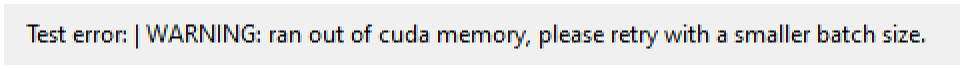
Overlap Threshold: For object detection, the overlap threshold determines the maximum percentage of area overlapping between two adjacent bounding boxes. A low threshold eliminates overlapping bounding boxes. By default, the overlap threshold only applies to bounding boxes of the same class. Enabling *Class-agnostic Overlap Threshold* causes the Overlap Threshold to be applied to any overlapping bounding boxes of different categories as well.

Max Detections: For object detection, this parameter limits the number of predictions at the output per image. It is set to 200 by default but can be increased up to 64K. In practice it is limited to 5000 as a restriction of the GPU optimization library.

Class-agnostic Overlap Threshold: When enabled, overlap-based filtering is performed independent of the classes. By default, it is applied to each class separately. The filtering is based on non-maximum suppression (NMS) which is a post-processing technique used in object detection to eliminate duplicate detections and select the most relevant detected objects. This helps reduce false positives and the computational complexity of the detection algorithm. Requires the Overlap Threshold parameter to be enabled.

Out-of-Memory Error during Testing

Testing models can require more memory than the training phase for models, particularly when generating heatmaps. This can generate out-of-memory errors during the testing phase.


```
Test error: | WARNING: ran out of cuda memory, please retry with a smaller batch size.
```

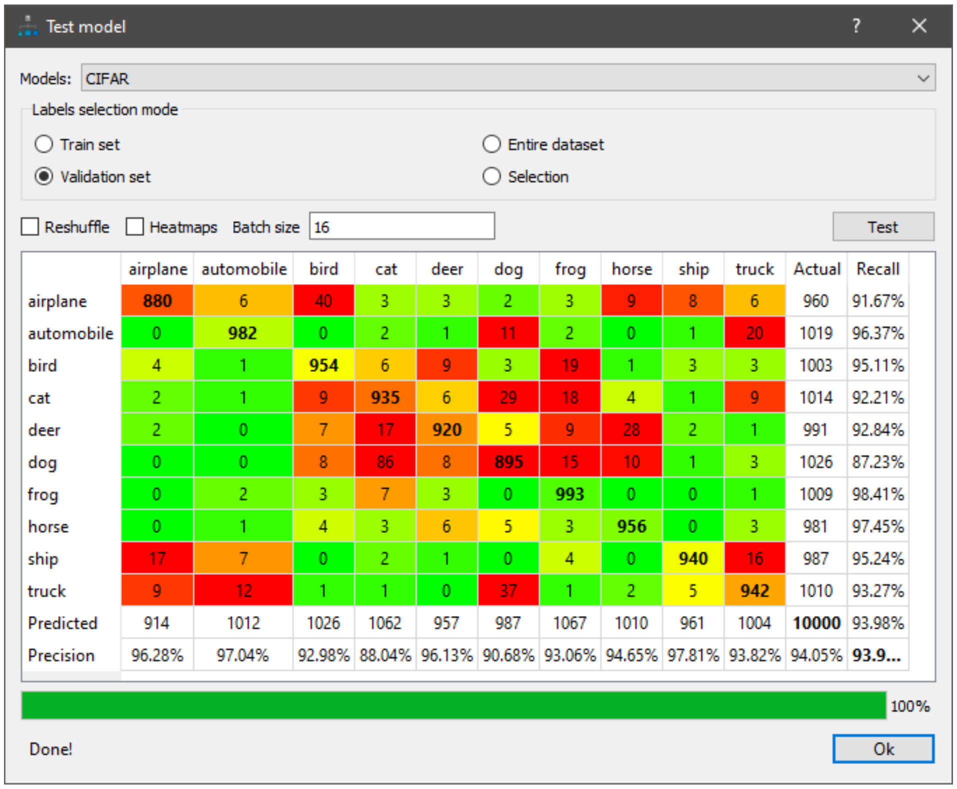
To correct testing out-of-memory problems:

Reduce the batch size. The batch size may need to be smaller than that used during training.

1. If reducing the batch size to the minimum of 2 still fails, reduce the dataset image size. For example, if using the maximum image size of 1024x1024, consider trying a smaller image size such as 512x512. The performance benefits of a larger image size may be negligible for many types of image datasets.
2. Disable generation of heatmaps. Heatmap generation requires significant memory resources but are very useful for determining the effectiveness of a model by ensuring that it is correctly identifying relevant features in the image (and not focusing on unrelated image details), therefore it is recommended to avoid disabling heatmaps if possible.

Classification Model Test Results

For a classification model, the test result is shown as a confusion matrix. Color coding is used to represent the precision/recall success of the model, with green indicating a rate exceeding 90%.



Test model

Models: CIFAR

Labels selection mode

Train set [✓ Validation set] Entire dataset Selection

☐ Reshuffle ☐ Heatmaps Batch size 16

Test

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck	Actual	Recall
airplane	**880**	6	40	3	3	2	3	9	8	6	960	91.67%
automobile	0	**982**	0	2	1	11	2	0	1	20	1019	96.37%
bird	4	1	**954**	6	9	3	19	1	3	3	1003	95.11%
cat	2	1	9	**935**	6	29	18	4	1	9	1014	92.21%
deer	2	0	7	17	**920**	5	9	28	2	1	991	92.84%
dog	0	0	8	86	8	**895**	15	10	1	3	1026	87.23%
frog	0	2	3	7	3	0	**993**	0	0	1	1009	98.41%
horse	0	1	4	3	6	5	3	**956**	0	3	981	97.45%
ship	17	7	0	2	1	0	4	0	**940**	16	987	95.24%
truck	9	12	1	1	0	37	1	2	5	**942**	1010	93.27%
Predicted	914	1012	1026	1062	957	987	1067	1010	961	1004	**10000**	93.98%
Precision	96.28%	97.04%	92.98%	88.04%	96.13%	90.68%	93.06%	94.65%	97.81%	93.82%	94.05%	**93.9...**

100%

Done!

Ok

Double-clicking on result fields opens the associated images in the Dataset for further investigation.

False Positives vs. False Negatives

Classification test results use the notion of False Positives (FP) and False Negatives (FN) to evaluate model performance.

To clarify the difference between a FP or FN result, consider the question “Am I pregnant?”:

A FP result indicates a positive result for pregnancy (baby present), though the ground truth is “Not pregnant” (no baby).

A FN result indicates a negative result for pregnancy (no baby), though the ground truth is “Pregnant” (baby present).

Precision vs. Recall

Precision can be seen as a measure of quality and recall as a measure of quantity. Higher precision means that when a model predicts a label it is correct, though it does not account for not labelling images that may contain that label (false negatives).

High recall means that the model correctly identifies the label among instances that should have that label.

Precision

Precision measures how often an instance that is predicted as category A is actually category A. That is, what proportion of positive identifications are actually correct.

Precision is calculated as follows:

$$Precision = \frac{TP}{TP + FP}$$

where *TP* is true positive and *FP* is false positive (another category is incorrectly identified as A).

A model that produces no false positives therefore has a precision of 1.0.

Recall

Recall measures how often a category A instance in the dataset is predicted as category A by the classifier. That is, what proportion of actual positives are identified correctly. Recall is calculated as follows:

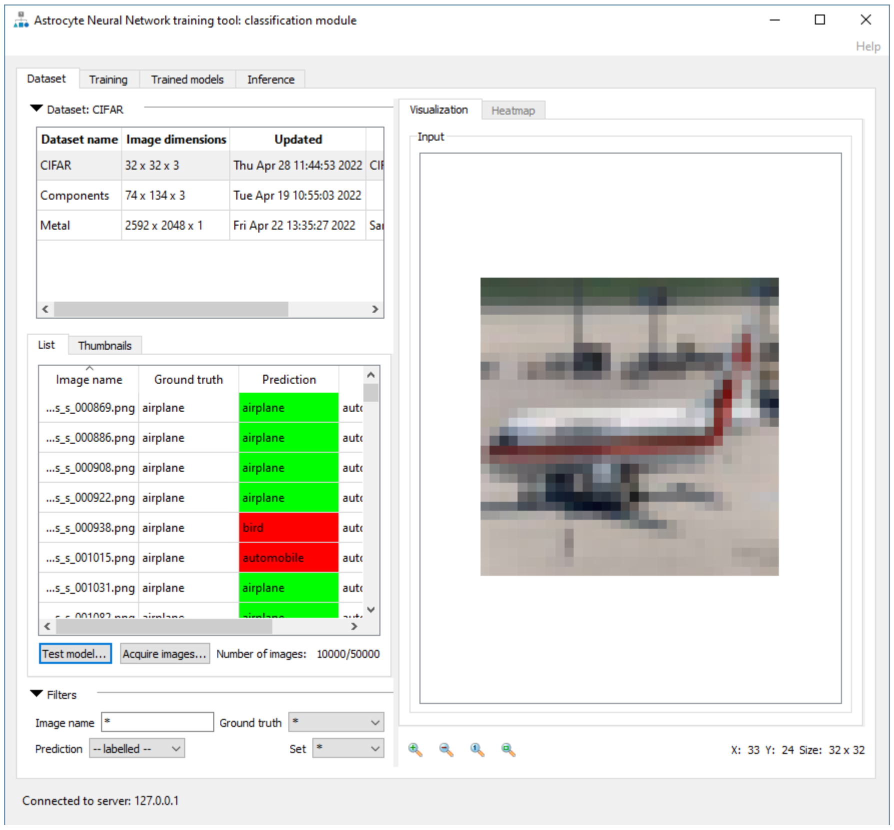
$$Recall = \frac{TP}{TP + FN}$$

where *TP* is true positive and *FN* is false negative (category A is identified as another category).

A model that produces no false negatives therefore has a precision of 1.0.

Predictions

The Dataset Predicted column displays correctly labelled predictions in green and prediction errors in red.



Dataset name	Image dimensions	Updated	
CIFAR	32 x 32 x 3	Thu Apr 28 11:44:53 2022	CIF
Components	74 x 134 x 3	Tue Apr 19 10:55:03 2022	
Metal	2592 x 2048 x 1	Fri Apr 22 13:35:27 2022	Sai

Image name	Ground truth	Prediction	
...s_s_000869.png	airplane	airplane	autc
...s_s_000886.png	airplane	airplane	autc
...s_s_000908.png	airplane	airplane	autc
...s_s_000922.png	airplane	airplane	autc
...s_s_000938.png	airplane	bird	autc
...s_s_001015.png	airplane	automobile	autc
...s_s_001031.png	airplane	airplane	autc

Number of images: 10000/50000

Image name *　Ground truth *
Prediction -- labelled --　Set *

X: 33 Y: 24 Size: 32 x 32

Connected to server: 127.0.0.1

Anomaly Detection Model Test Results

For an anomaly detection model, the test results are displayed as a TPR/FPR (True Positive Rate/ False Positive Rate) graph, also known as AUROC. Testing should be performed on the entire dataset to evaluate how well the model identifies anomalies compared to normal images.

True Positive Rate (TPR): probability of correctly identifying positives.

False Positive Rate (FPR): probability of falsely identifying negatives as positives (also known as the "false alarm rate").

The True Positive Rate and False Positive Rate are calculated as:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Where TP = True Positives, FN = False Negatives, FP = False Positives and TN = True Negatives

AUPR and AUROC

Result metrics are:

AUPR: Area Under Precision Recall. It is the area under the curve where x is recall and y is precision. AUPR is a relative measure of how precisely a model predicts true positives and the true positive rate. In general, for evaluating performance of an imbalanced dataset, AUPR is more appropriate than AUROC

AUROC: Area Under Receiver Operating Characteristic. It is the area under the curve where x is **false positive rate** (FPR) and y is **true positive rate** (TPR). The greater the AUROC the better the model performance. AUROC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUROC of 0.0; one whose predictions are 100% correct has an AUROC of 1.0.

AUPR is a useful metric with imbalanced datasets, for example when good samples highly outnumber bad samples or anomalies.

For example, given an imbalanced dataset with 100 anomalies and 10000 good samples, if model A identifies 100 anomalies (90 TP and 10 FP) and model B identifies 2000 anomalies (90 TP and 1910 FP), clearly model A is preferable since they both come back with the same number of relevant results, but model B includes a high number of false positives.

Model A: **100** anomalies, **90** correct

Model B: **2000** anomalies, **90** correct

The AUROC measures of TPR and FPR will reflect that, but since the number of good samples significantly outnumbers anomalies, the difference is mostly lost:

Model A: 0.9 TPR, 0.001 FPR

Model B: 0.9 TPR, 0.161 FPR (difference of 0.16)

Precision and recall, are not affected by the relative imbalance:

Model B: 0.9 recall, 0.9 precision

Model B: 0.9 recall, 0.045 precision (difference of 0.855)

To compare both methods, using AUROC, the FPR has a difference of 0.16, which is very small. However, using AUPR, we see that the Precision has a difference of 0.855 which is much more pronounced.

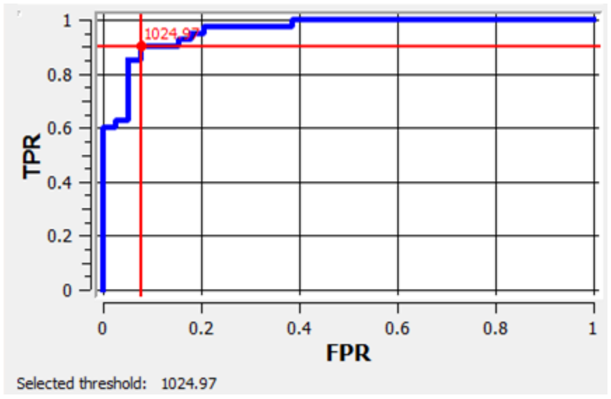
AUPR is much better at illustrating the differences between models with greatly imbalanced datasets.

Anomaly Confidence Score

When testing the model, each image is assigned an anomaly confidence score with relatively higher values indicating an anomaly compared to normal images. The anomaly confidence score is an arbitrary number generated by the model that is determined by the image dataset; different image datasets will have different ranges of anomaly confidence scores depending on the image data.

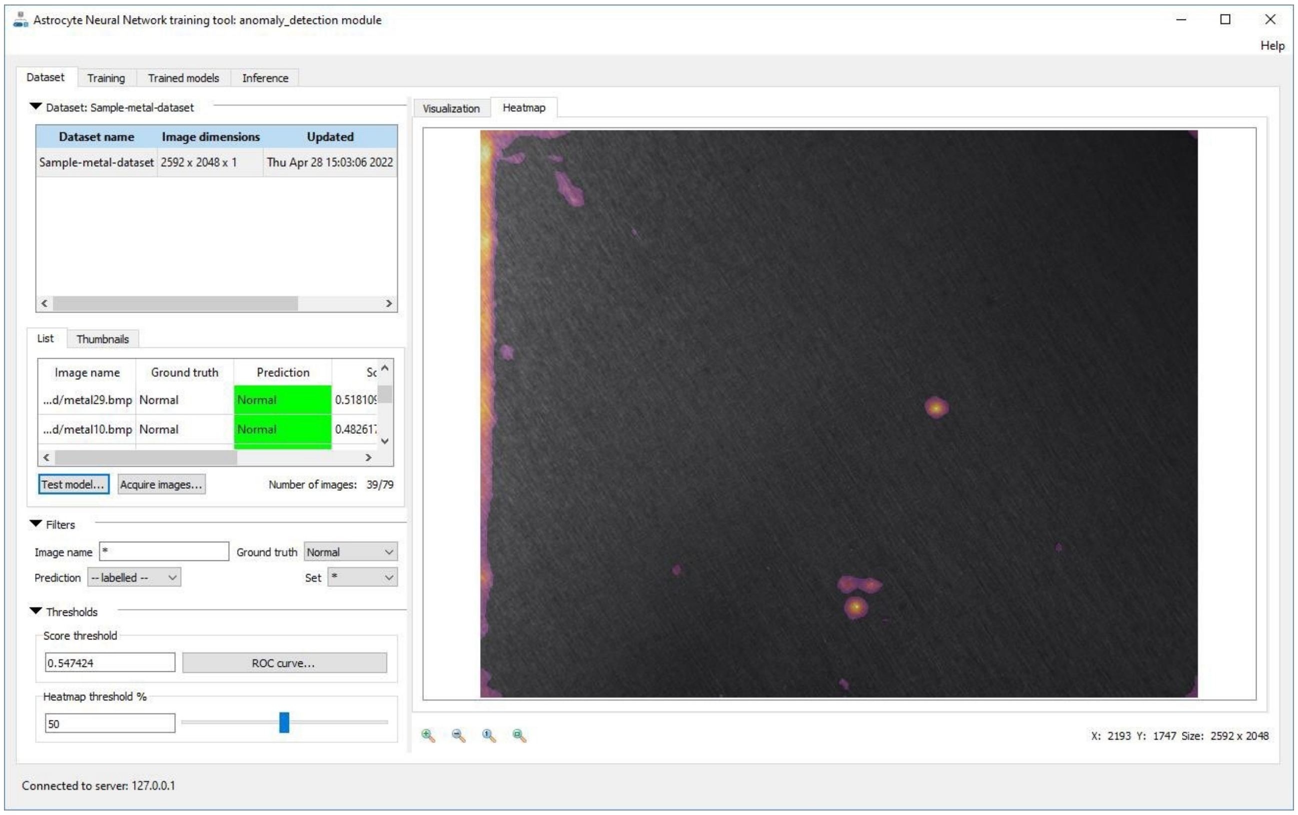
The anomaly confidence score represents the level of anomaly; the higher the score the greater the anomaly compared to what has been learned as good. The specified anomaly threshold determines the minimum anomaly confidence score above which images are considered anomalies.

To select the anomaly threshold to apply, click on the AUROC graph at a corner point; in general, the best performance is achieved at the anomaly score associated with the point closest to the upper-left corner as this will result in the highest TPR with the lowest corresponding FPR; that is, the highest precision with the least amount of false positives (a perfect result is a precision of 1 (all predictions correct) with no false positives).



Selected threshold: 1024.97

Results are shown in the Dataset tab, with incorrect predictions shown in red.



Dataset name	Image dimensions	Updated
Sample-metal-dataset	2592 x 2048 x 1	Thu Apr 28 15:03:06 2022

Image name	Ground truth	Prediction	Sc
...d/metal29.bmp	Normal	Normal	0.51810
...d/metal10.bmp	Normal	Normal	0.48261

Number of images: 39/79

Image name: *
Ground truth: Normal
Prediction: -- labelled --
Set: *

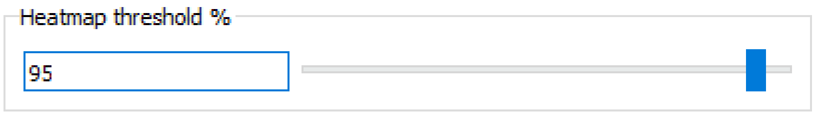
Score threshold: 0.547424

Heatmap threshold %: 50

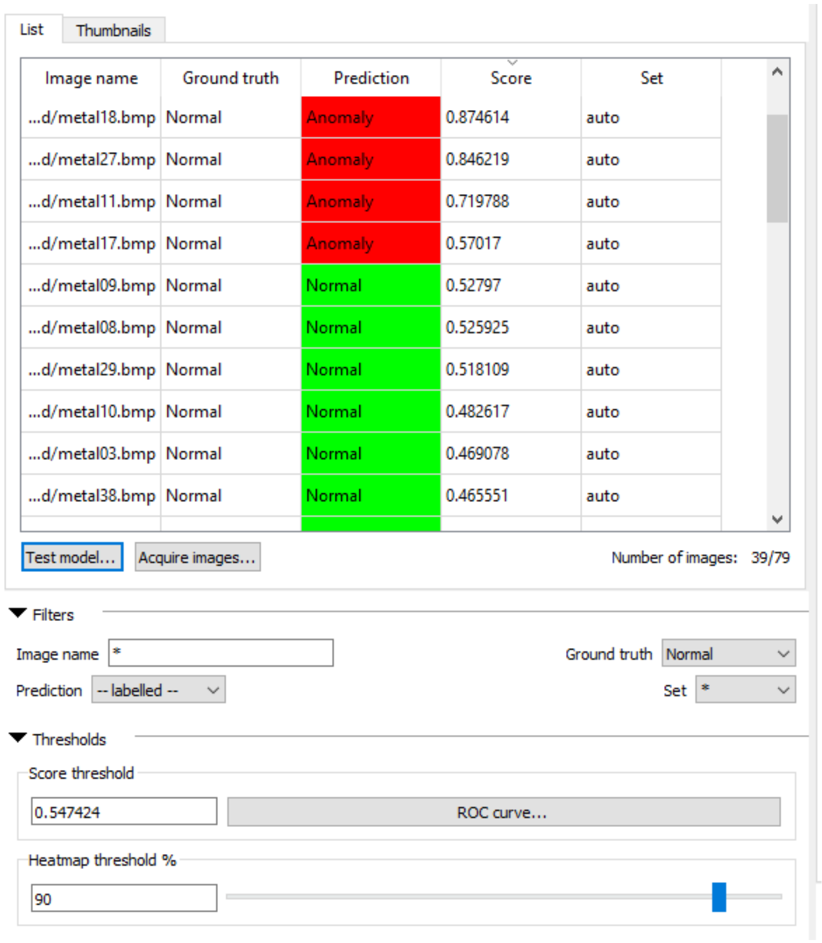
X: 2193 Y: 1747 Size: 2592 x 2048

Connected to server: 127.0.0.1

When heatmap generation is enabled, the heatmap visualizes each pixel’s activation, with brighter pixels reflecting higher scores (the score for each image indicates the highest pixel activation level in the image). The Heatmap Threshold determines the percentage of activated pixels to display (for example, with a threshold of 95% only the top 5% of activated pixels are shown).


Heatmap threshold %: 95

Results can be sorted according to the ground truth label to show the effect of the selected anomaly confidence score threshold. For example, displaying only the Normal ground truth labels identifies how many false positives are predicted for a given threshold.



List | Thumbnails

Image name	Ground truth	Prediction	Score	Set
...d/metal18.bmp	Normal	🟥 Anomaly	0.874614	auto
...d/metal27.bmp	Normal	🟥 Anomaly	0.846219	auto
...d/metal11.bmp	Normal	🟥 Anomaly	0.719788	auto
...d/metal17.bmp	Normal	🟥 Anomaly	0.57017	auto
...d/metal09.bmp	Normal	🟩 Normal	0.52797	auto
...d/metal08.bmp	Normal	🟩 Normal	0.525925	auto
...d/metal29.bmp	Normal	🟩 Normal	0.518109	auto
...d/metal10.bmp	Normal	🟩 Normal	0.482617	auto
...d/metal03.bmp	Normal	🟩 Normal	0.469078	auto
...d/metal38.bmp	Normal	🟩 Normal	0.465551	auto

Test model... Acquire images... Number of images: 39/79

Filters

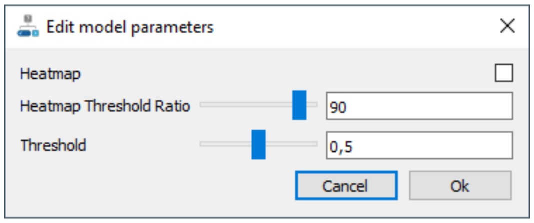
Image name: *
Ground truth: Normal
Prediction: -- labelled --
Set: *

Thresholds

Score threshold: 0.547424 ROC curve...

Heatmap threshold %: 90

When exporting an anomaly detection model heatmap generation can be enabled and thresholds assigned.



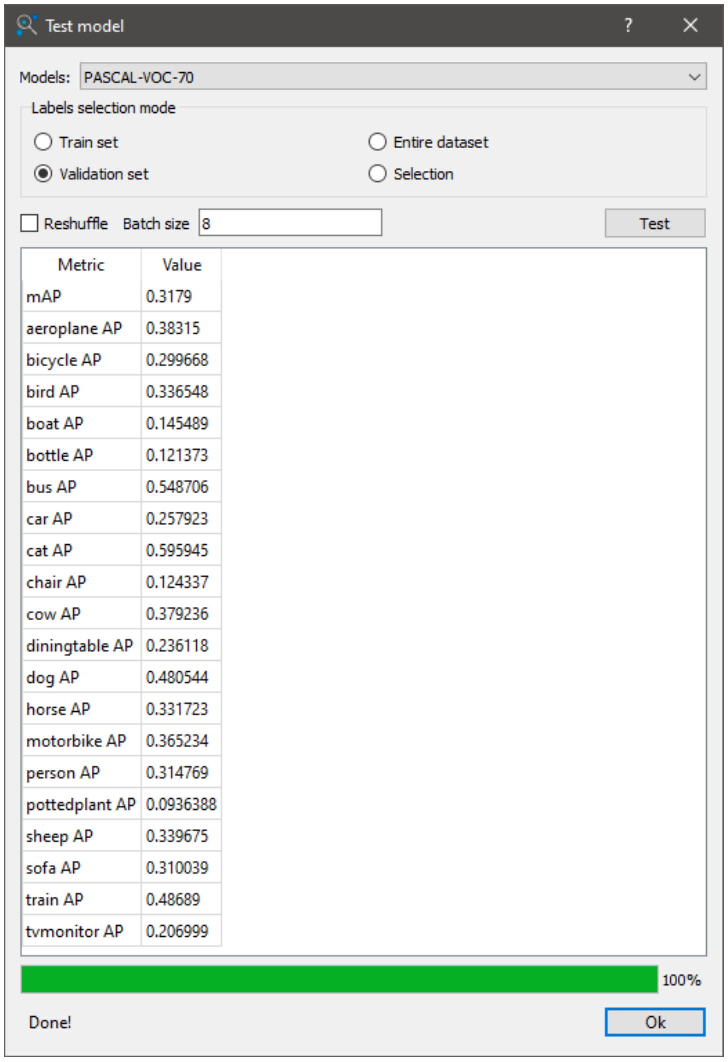
Edit model parameters

Heatmap: ☐
Heatmap Threshold Ratio: 90
Threshold: 0,5

Cancel Ok

Object Detection Model Test Results

Object detection allows for the location and detection of multiple categories of objects within the same image. The location of objects is returned as a bounding box. For an object detection model, test results are provided as the average precision (AP) for each category label, as well as the mean average precision (mAP) for the model overall (all categories).



Models: PASCAL-VOC-70

Labels selection mode: ( ) Train set [✓ Validation set] ( ) Entire dataset ( ) Selection

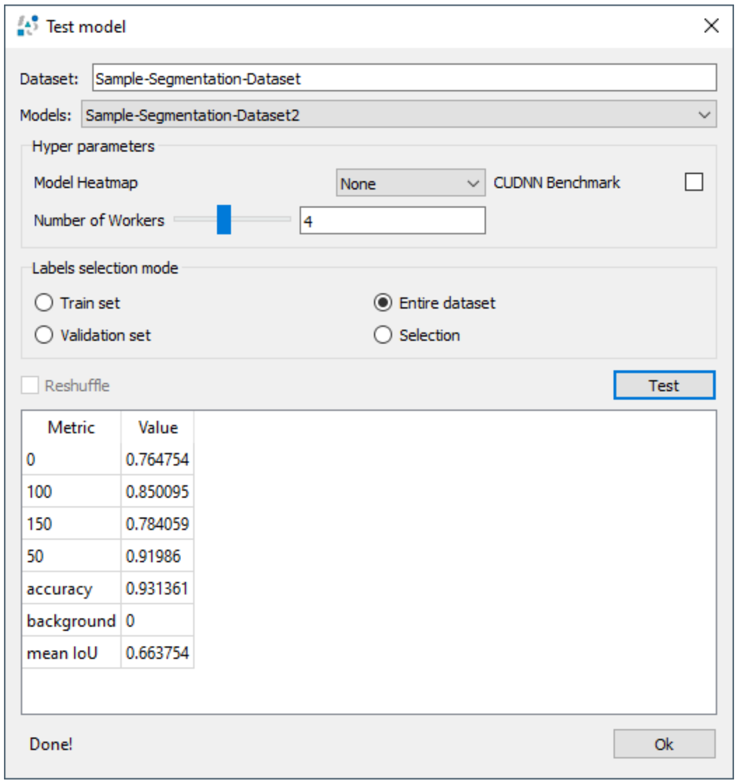
Reshuffle ☐ Batch size 8

Metric	Value
mAP	0.3179
aeroplane AP	0.38315
bicycle AP	0.299668
bird AP	0.336548
boat AP	0.145489
bottle AP	0.121373
bus AP	0.548706
car AP	0.257923
cat AP	0.595945
chair AP	0.124337
cow AP	0.379236
diningtable AP	0.236118
dog AP	0.480544
horse AP	0.331723
motorbike AP	0.365234
person AP	0.314769
pottedplant AP	0.0936388
sheep AP	0.339675
sofa AP	0.310039
train AP	0.48689
tvmonitor AP	0.206999

Done!

Segmentation Model Test Results

Segmentation allows for the location and detection of multiple categories of objects within the same image, identified by pixel value corresponding to the associated category. Test results include the IoU for each category, the mean IoU for all categories, the accuracy as well as the background IoU.



Dataset: Sample-Segmentation-Dataset
Models: Sample-Segmentation-Dataset2
Model Heatmap: None
CUDNN Benchmark: ☐
Number of Workers: 4
Labels selection mode: Train set / Validation set / [✓ Entire dataset] / Selection
Reshuffle: ☐

Metric	Value
0	0.764754
100	0.850095
150	0.784059
50	0.91986
accuracy	0.931361
background	0
mean IoU	0.663754

Done!

Accuracy is defined as the ability of the model to correctly predict pixels. For example, in an image with a large percentage of background pixels, a poor model which predicts all pixels as background can still have a high accuracy since most pixels are background and are correctly predicted; in this case, the mean IoU, which measures the average IoU for all labelled segments (including the background), is a better representation of the overall model performance.

An IoU of 0 means that images in the test set have ground truth pixels that belong to that category but there is no prediction to match against it.

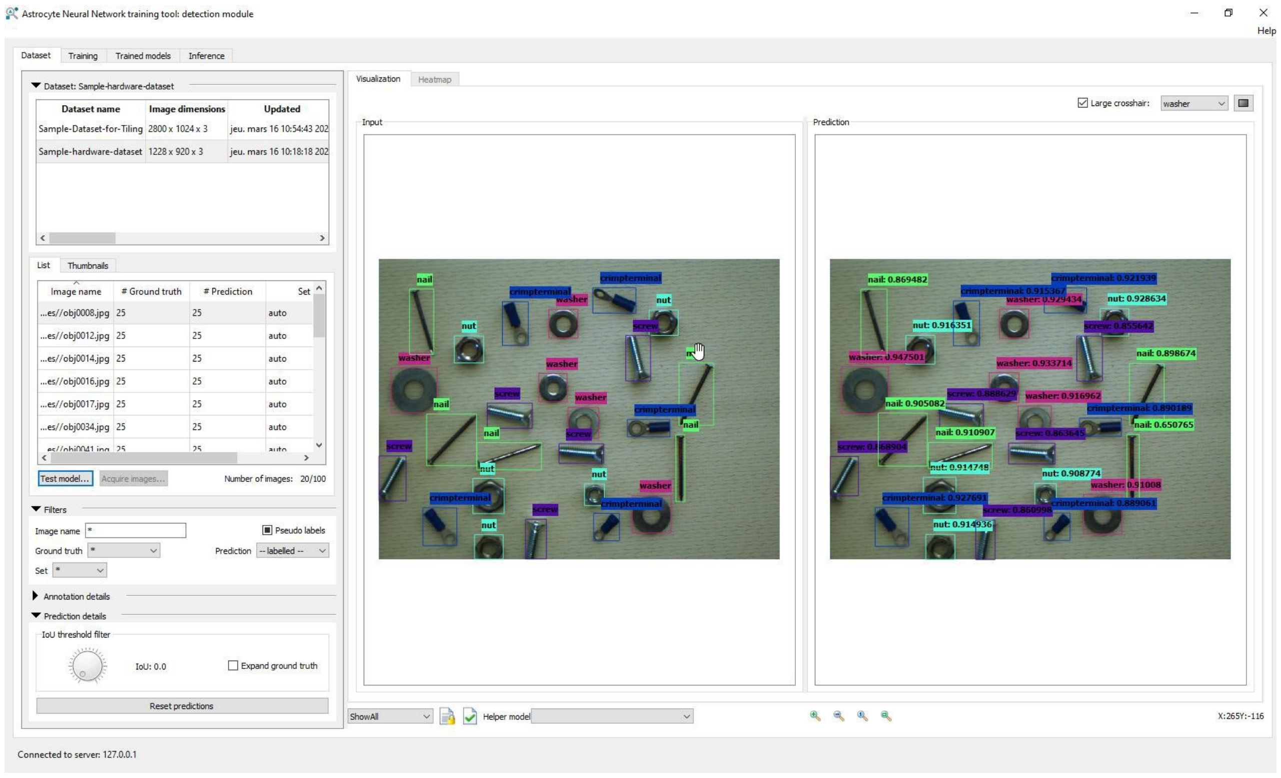
An IoU of -1 means the category does not exist as a ground truth in any of the images in the selected test set and the model did not make any predictions belonging to that category.

The background IoU measures the predicted background pixels against the ground truth background pixels in the selected test image set. The background IoU is typically very large because most images have a lot of background so whatever the model prediction is, the chances are the IoU between the real background and the predicted background is large .

For more information on how IoU is calculated, refer to the IoU Threshold Filter section.

Predicted Bounding Box & Segmentation Results

For the object detection and segmentation modules, predicted bounding boxes and segments and their corresponding labels are displayed in the Dataset tab.



Dataset: Sample-hardware-dataset

Dataset name	Image dimensions	Updated
Sample-Dataset-for-Tiling	2800 x 1024 x 3	jeu. mars 16 10:54:43 202
Sample-hardware-dataset	1228 x 920 x 3	jeu. mars 16 10:18:18 202

Image name	# Ground truth	# Prediction	Set
...es//obj0008.jpg	25	25	auto
...es//obj0012.jpg	25	25	auto
...es//obj0014.jpg	25	25	auto
...es//obj0016.jpg	25	25	auto
...es//obj0017.jpg	25	25	auto
...es//obj0034.jpg	25	25	auto
...es//obj0041.jpg	25	25	auto

Number of images: 20/100

Filters

Image name: *

Pseudo labels [✓]

Ground truth: *

Prediction: -- labelled --

Set: *

Prediction details

IoU threshold filter

IoU: 0.0

Expand ground truth [ ]

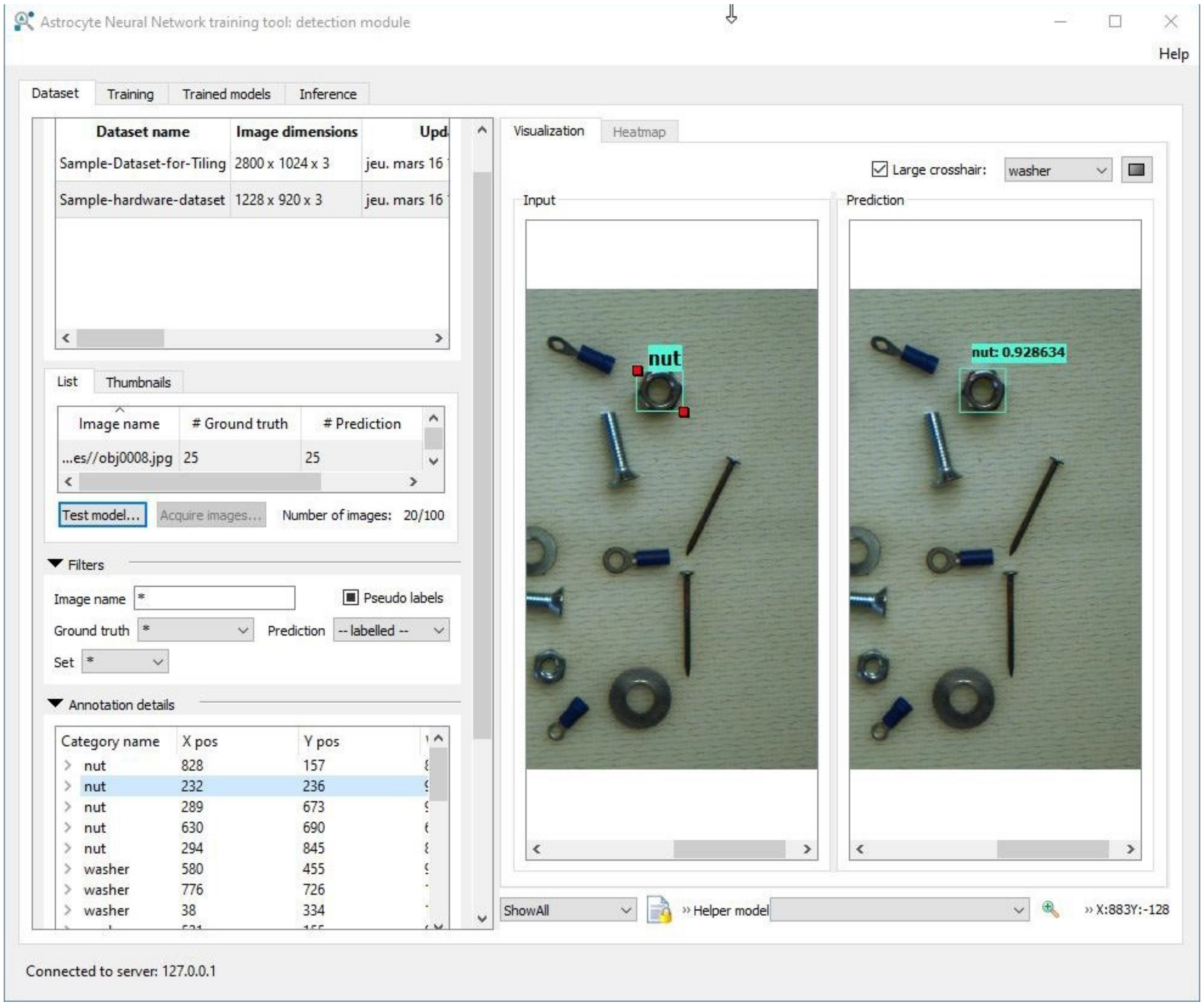
Large crosshair: [✓] washer

Connected to server: 127.0.0.1

Clicking on an entry selects the bounding box or segment.

Double-clicking on the x, y, width or height entries of a ground truth allows entering a numerical value.

Double-clicking on the category name singles out that annotation along with its prediction and makes all the other ground truth and annotation disappear from the image.



Dataset name	Image dimensions	Upd
Sample-Dataset-for-Tiling	2800 x 1024 x 3	jeu. mars 16
Sample-hardware-dataset	1228 x 920 x 3	jeu. mars 16

Image name	# Ground truth	# Prediction
...es//obj0008.jpg	25	25

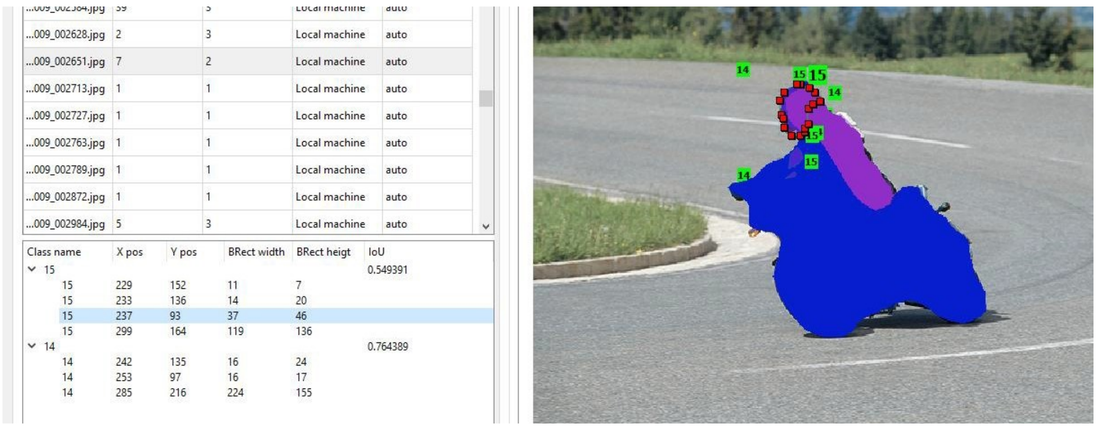
Number of images: 20/100

Image name: *
Pseudo labels
Ground truth: *
Prediction: -- labelled --
Set: *
Large crosshair: washer

Category name	X pos	Y pos
nut	828	157
nut	232	236
nut	289	673
nut	630	690
nut	294	845
washer	580	455
washer	776	726
washer	38	334

Connected to server: 127.0.0.1

For the segmentation module, prediction segments are shown as solid color by category and the ground truth bounding polygons can be selected and their handles shown.

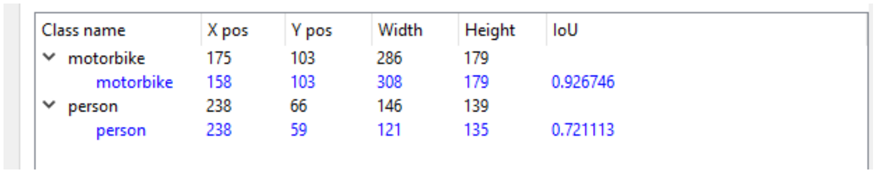


...009_002628.jpg	2	3	Local machine	auto
...009_002651.jpg	7	2	Local machine	auto
...009_002713.jpg	1	1	Local machine	auto
...009_002727.jpg	1	1	Local machine	auto
...009_002763.jpg	1	1	Local machine	auto
...009_002789.jpg	1	1	Local machine	auto
...009_002872.jpg	1	1	Local machine	auto
...009_002984.jpg	5	3	Local machine	auto

Class name	X pos	Y pos	BRect width	BRect heigt	IoU
15					0.549391
15	229	152	11	7	
15	233	136	14	20	
15	237	93	37	46	
15	299	164	119	136	
14					0.764389
14	242	135	16	24	
14	253	97	16	17	
14	285	216	224	155	

Expand Ground Truth

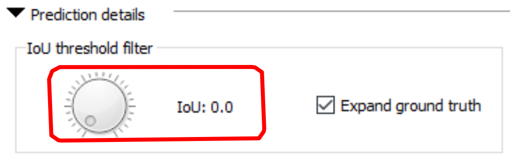
When the Expand ground truth checkbox is enabled, the annotation details window is expanded to show the predicted bounding coordinates and IoU percentage for model test results.



Class name	X pos	Y pos	Width	Height	IoU
motorbike	175	103	286	179	
motorbike	158	103	308	179	0.926746
person	238	66	146	139	
person	238	59	121	135	0.721113

IoU Threshold Filter

Intersection-over-Union (IoU) is an evaluation metric used to measure the accuracy of the object detection or segmentation models on a particular dataset. To filter results using IoU, specify the minimum acceptable IoU using the dial button.

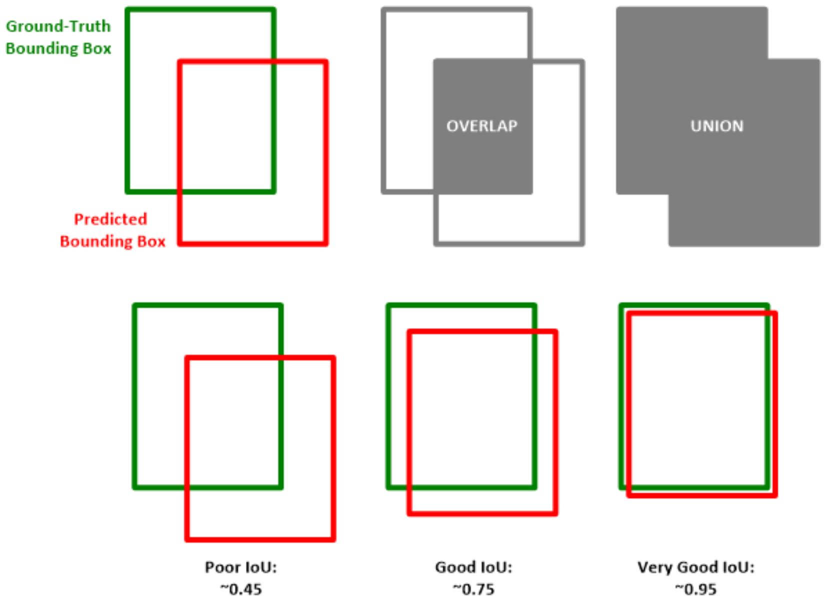


The IoU metric evaluates how well the model's predicted bounding box or segment for the object fits the ground-truth bounding boxes (the labelled bounding boxes that identify the object or "ground truth") or segmentation mask. When testing a model, the IoU threshold determines the minimum acceptable overlap of the predicted bounding box or segment and the ground-truth bounding box or segmentation mask, removing predictions which do not meet this threshold.

IoU is calculated as the ratio of overlap/union of the predicted and actual bounding boxes or segmentation mask:

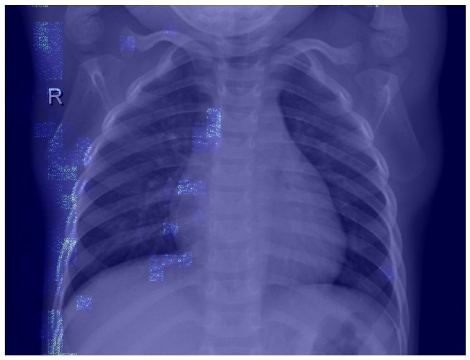
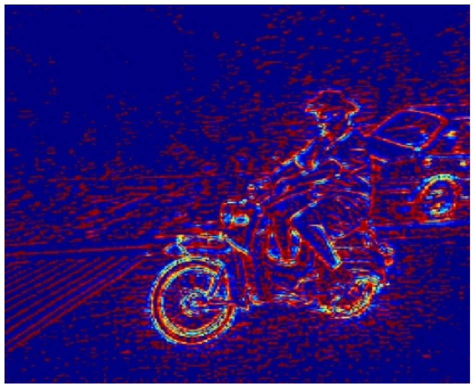
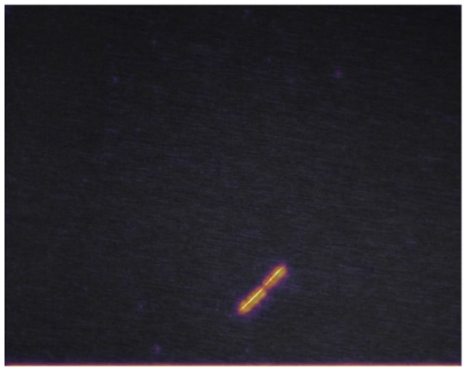
IoU=AoO / AoU

where AoO = Area of Overlap and AoU = Area of Union.



Heatmaps

Heatmaps provide insight into what aspects of an input image a convolutional neural network is using to make a prediction. The heatmap graphically displays which parts of the image are activating the model neurons. Heatmaps can be useful for gaining insight into a model's misclassifications. For example, a heatmap can reveal that a model is focusing on an image detail or feature that has no relevance to the desired analysis of the target scene or object in the image. Depending on the Astrocyte module, different types of heatmap generation algorithms are available:

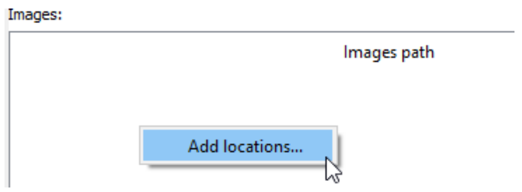
Heatmap	Description	Example
Guided GradCAM	Gradient-weighted Class Activation Mapping (Grad-CAM). Measures the relative importance of input features by calculating the gradient of the output decision with respect to those input features.	
Guided Backprop	Guided back propagation that suppresses the flow of gradients through neurons wherein either of input or incoming gradients were negative.	
Saliency	Saliency heatmaps highlight input image pixels that most caused the output classification.	

Note: When testing a model, generating heatmaps can significantly increase the amount of time to compute results

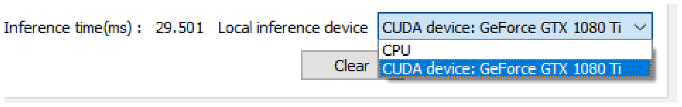
Inference

The Inference tab allows saved models to be deployed to analyze and process unlabelled target images. Inference processing time is dependent on the available GPU hardware and performance.

Right-click in the Images panel to specify the location of the target images. Astrocyte also supports drag-and-drop of images from Windows Filer Explorer.

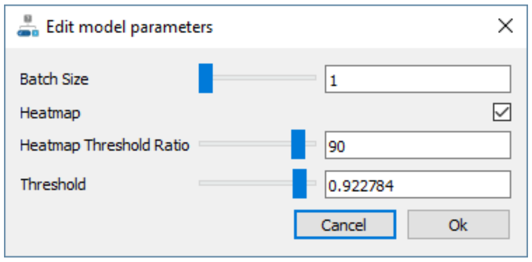


Use the drop-down list to select the inference (computing) device to use:


Inference time(ms) : 29.501 Local inference device CUDA device: GeForce GTX 1080 Ti

Use the model drop-down list to select the model from among those previously saved.

Certain model types have adjustable parameters which can be changed when loading.


Edit model parameters

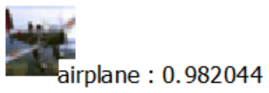
Batch Size 1
Heatmap ✓
Heatmap Threshold Ratio 90
Threshold 0.922784

Note: Batch Size is only available if the model has been trained with tiling enabled.

Selecting an image analyzes and processes the image using the selected model. The inference (processing) time is displayed.

Classification Inference

For classification, the category with the highest confidence score is returned:

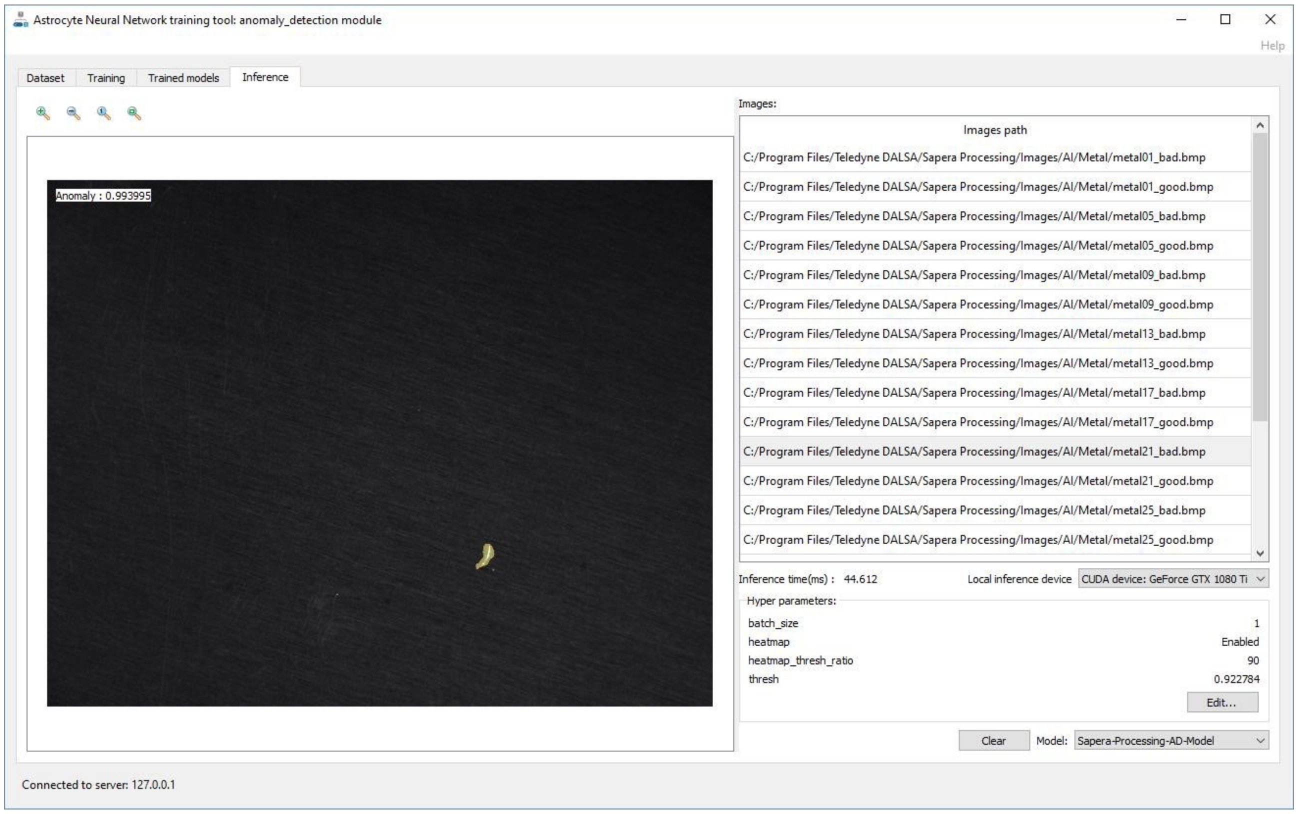

airplane : 0.982044

For Continual Learning type models, if an image is presented that differs significantly from existing categories in the model it is labelled as “Undefined”. These images may be good candidates to add as new categories to the model.

Anomaly Detection Inference

For anomaly detection, inference requires setting an appropriate anomaly threshold, as determined by model testing. To do so, click **Edit** to specify the anomaly threshold and click **OK**. For Pixel-Level Anomaly Detection models, saliency heatmaps can be output to review the image features that the model identified as anomalies (Image-Level Anomaly Detection model types do not support heatmap generation during inference (available when testing only). When heatmaps are enabled, the Heatmap Threshold Ratio (a percentage of the anomaly threshold) can be set to adjust the amount of heatmap pixels displayed.

The anomaly score and corresponding Normal or Anomaly label are displayed as an overlay in the image.


Astrocyte Neural Network training tool: anomaly_detection module
Help
Dataset
Training
Trained models
Inference
Anomaly : 0.993995
Images:
Clear
Model: Sapera-Processing-AD-Model
Edit...

Images path
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal01_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal01_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal05_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal05_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal09_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal09_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal13_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal13_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal17_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal17_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal21_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal21_good.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal25_bad.bmp
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal25_good.bmp

Inference time(ms) : 44.612 Local inference device CUDA device: GeForce GTX 1080 Ti

Hyper parameters:

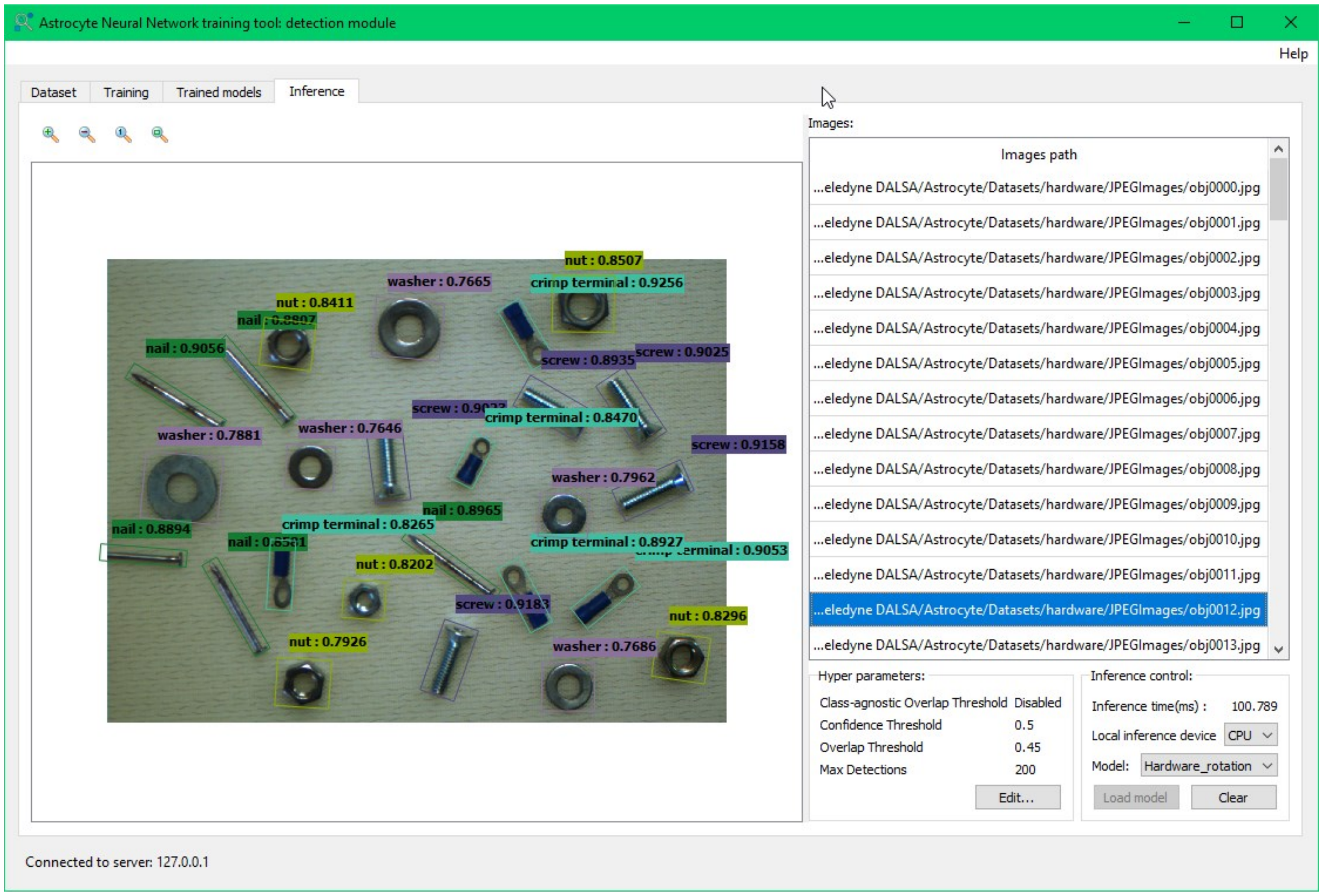
batch_size	1
heatmap	Enabled
heatmap_thresh_ratio	90
thresh	0.922784

Connected to server: 127.0.0.1

Object Detection Inference

For object detection, images are annotated with located bounding boxes, labels and confidence scores. Confidence score is the probability that a predicted bounding box contains an object. At prediction time, the model generates scores for the presence of each object category in each predicted bounding box where an object is detected and produces adjustments to the box to better match the object shape.

A detection is considered a true positive (TP) only if it satisfies three conditions:confidence score > threshold; the predicted category matches the category of a ground truth; the predicted bounding box has an IoU greater than a threshold (for example., 0.5) with the ground-truth.



...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0000.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0001.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0002.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0003.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0004.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0005.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0006.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0007.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0008.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0009.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0010.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0011.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0012.jpg
...eledyne DALSA/Astrocyte/Datasets/hardware/JPEGImages/obj0013.jpg

Hyper parameters:

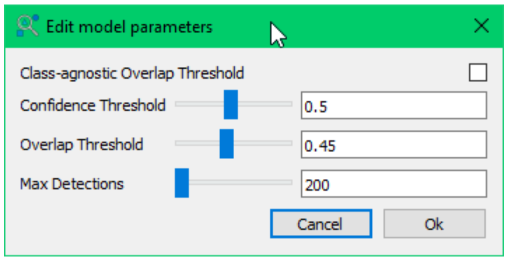
Class-agnostic Overlap Threshold	Disabled
Confidence Threshold	0.5
Overlap Threshold	0.45
Max Detections	200

Inference control:

Inference time(ms) :	100.789
Local inference device	CPU
Model:	Hardware_rotation

Connected to server: 127.0.0.1

For object detection, inference requires setting an appropriate confidence threshold, as determined by model testing. Other settings including the Overlap Threshold, Max Detectionsand Class-agnostic Ovelap Threshold can be enabled. To do so, click **Edit** and set the required values and click **OK**.

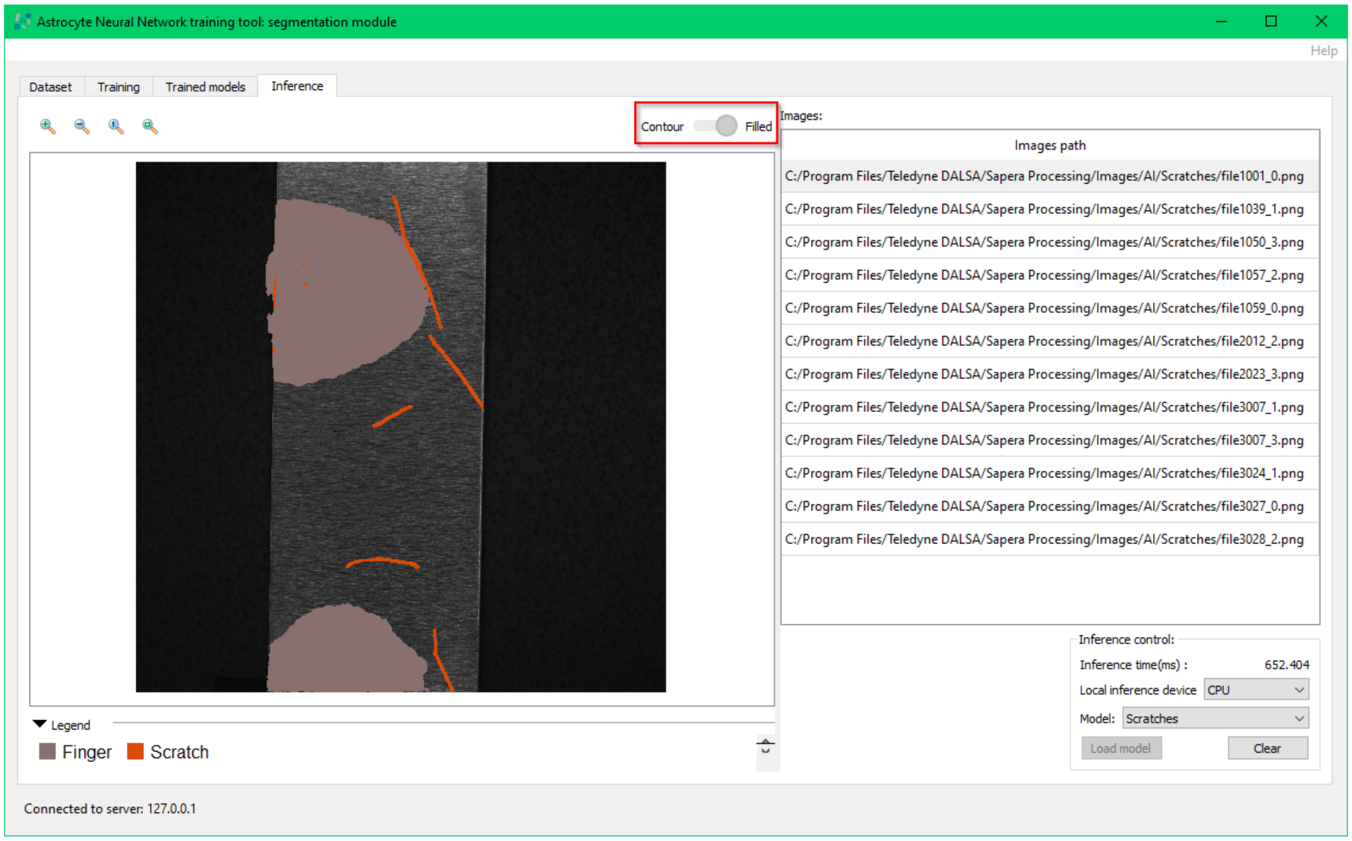


Class-agnostic Overlap Threshold	☐
Confidence Threshold	0.5
Overlap Threshold	0.45
Max Detections	200

Segmentation Inference

For segmentation, the original image is displayed at top with the segmentation mask image just below. Category names are shown at the bottom of the segmentation mask image in the Legend section.

Use the Counter/Filled slider to change the annotation style.



Astrocyte Neural Network training tool: segmentation module

C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file1001_0.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file1039_1.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file1050_3.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file1057_2.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file1059_0.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file2012_2.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file2023_3.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file3007_1.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file3007_3.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file3024_1.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file3027_0.png
C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Scratches/file3028_2.png

🟫 Finger 🟧 Scratch

Inference control:

Inference time(ms) : 652.404

Local inference device CPU

Model: Scratches

Connected to server: 127.0.0.1

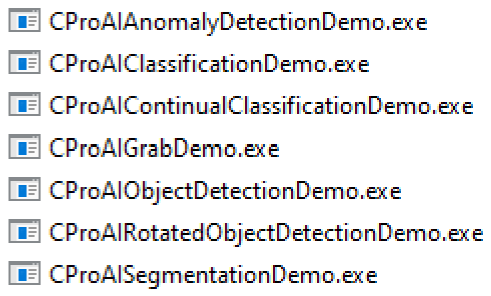
Deploying Models with Sapera Processing

Astrocyte models can be deployed using Teledyne DALSA's Sapera Processing SDK to analyze and process target images. Models must first be exported from the Astrocyte AI Trainer to use with the Sapera Processing SDK.

The Sapera Processing SDK includes demo applications for each type of model demonstrating how to use the Sapera Processing AI Inference classes to load an Astrocyte model, analyze and process unlabelled target images. Images are preprocessed to the input format of the model.

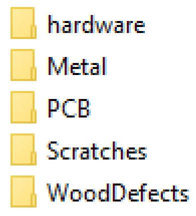
Compiled demo applications are available in the following directory:

<installation directory>\Teledyne DALSA\Sapera Processing\Demos\Executables

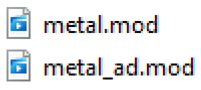

CProAIAnomalyDetectionDemo.exe
CProAIClassificationDemo.exe
CProAIContinualClassificationDemo.exe
CProAIGrabDemo.exe
CProAIObjectDetectionDemo.exe
CProAIRotatedObjectDetectionDemo.exe
CProAISegmentationDemo.exe

The compiled demo applications use sample models and images which are available in the following directory:

<installation directory>\Teledyne DALSA\Sapera Processing\Images\AI


hardware
Metal
PCB
Scratches
WoodDefects

For example, the Metal directory contains models for anomaly detection and classification:

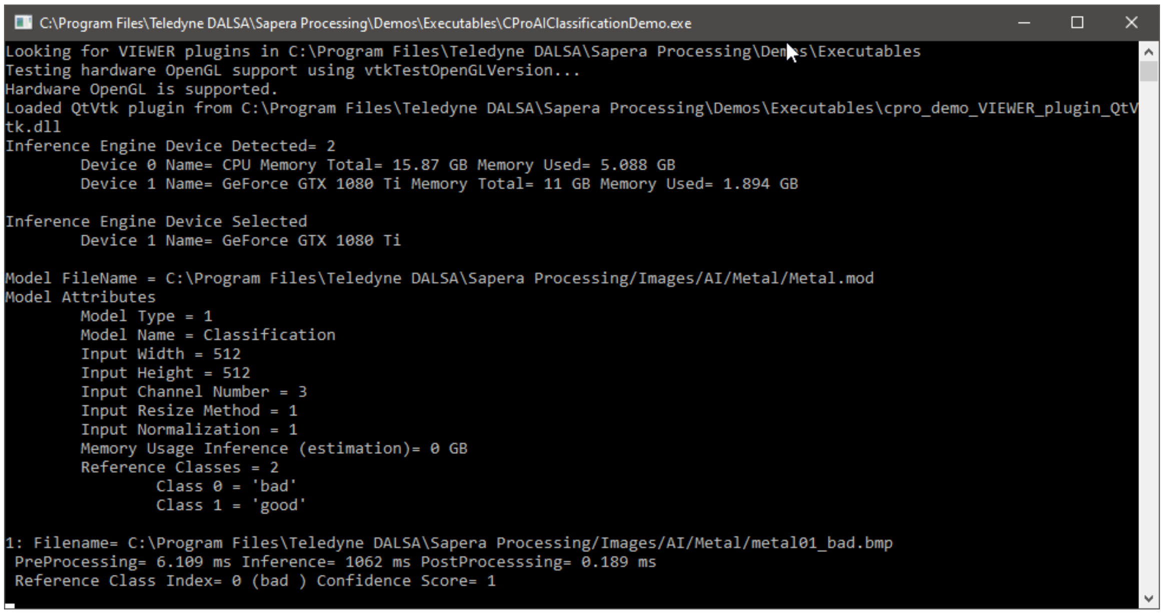

metal.mod
metal_ad.mod

Complete source code and solutions for Microsoft Visual Studio is included; users can adapt this code for applications to use their own models and target images.

The CProAIGrab demo shows how to acquire and perform inference on images from an acquisition device.

For emulating acquisition from a device using saved images, demo programs use an *.ini* file to configure details such as the path to the image folder and simulated acquisition frame rate.

Demos include processing statistics and image display.

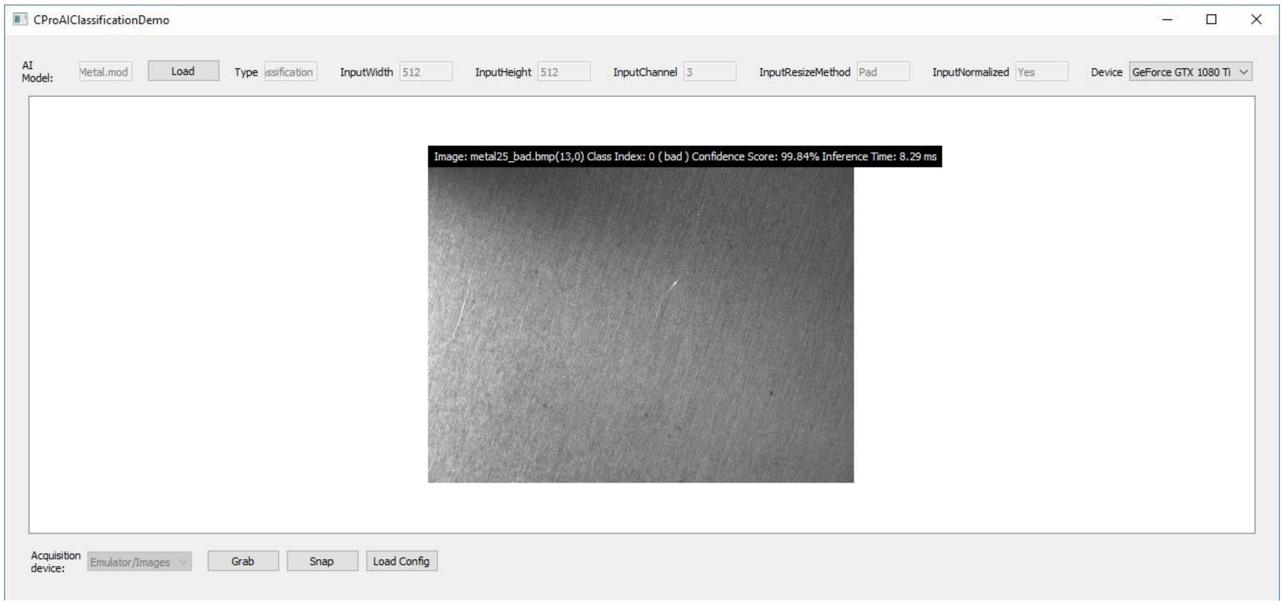


C:\Program Files\Teledyne DALSA\Sapera Processing\Demos\Executables\CProAIClassificationDemo.exe

```
Looking for VIEWER plugins in C:\Program Files\Teledyne DALSA\Sapera Processing\Demos\Executables
Testing hardware OpenGL support using vtkTestOpenGLVersion...
Hardware OpenGL is supported.
Loaded QtVtk plugin from C:\Program Files\Teledyne DALSA\Sapera Processing\Demos\Executables\cpro_demo_VIEWER_plugin_QtV
tk.dll
Inference Engine Device Detected= 2
        Device 0 Name= CPU Memory Total= 15.87 GB Memory Used= 5.088 GB
        Device 1 Name= GeForce GTX 1080 Ti Memory Total= 11 GB Memory Used= 1.894 GB

Inference Engine Device Selected
        Device 1 Name= GeForce GTX 1080 Ti

Model FileName = C:\Program Files\Teledyne DALSA\Sapera Processing/Images/AI/Metal/Metal.mod
Model Attributes
        Model Type = 1
        Model Name = Classification
        Input Width = 512
        Input Height = 512
        Input Channel Number = 3
        Input Resize Method = 1
        Input Normalization = 1
        Memory Usage Inference (estimation)= 0 GB
        Reference Classes = 2
                Class 0 = 'bad'
                Class 1 = 'good'

1: Filename= C:\Program Files\Teledyne DALSA\Sapera Processing/Images/AI/Metal/metal01_bad.bmp
 PreProcessing= 6.109 ms Inference= 1062 ms PostProcesssing= 0.189 ms
 Reference Class Index= 0 (bad ) Confidence Score= 1
```

Use the Load button to load your own model exported from Astrocyte. Use the Grab and Snap buttons to emulate image acquisition.

Use the Load Config button to load a configuration file (.ini) for the emulated acquisition.



AI Model: Metal.mod Type ıssification InputWidth 512 InputHeight 512 InputChannel 3 InputResizeMethod Pad InputNormalized Yes Device GeForce GTX 1080 Ti

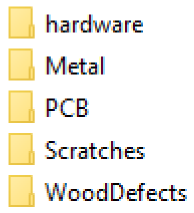
Image: metal25_bad.bmp(13,0) Class Index: 0 (bad) Confidence Score: 99.84% Inference Time: 8.29 ms

Acquisition device: Emulator/Images

Sample Models Included with Sapera Processing

Sapera Processing includes sample models trained with Astrocyte for each of the modules for use with the demo programs. The sample models are available in each of the sub-directories contained in:

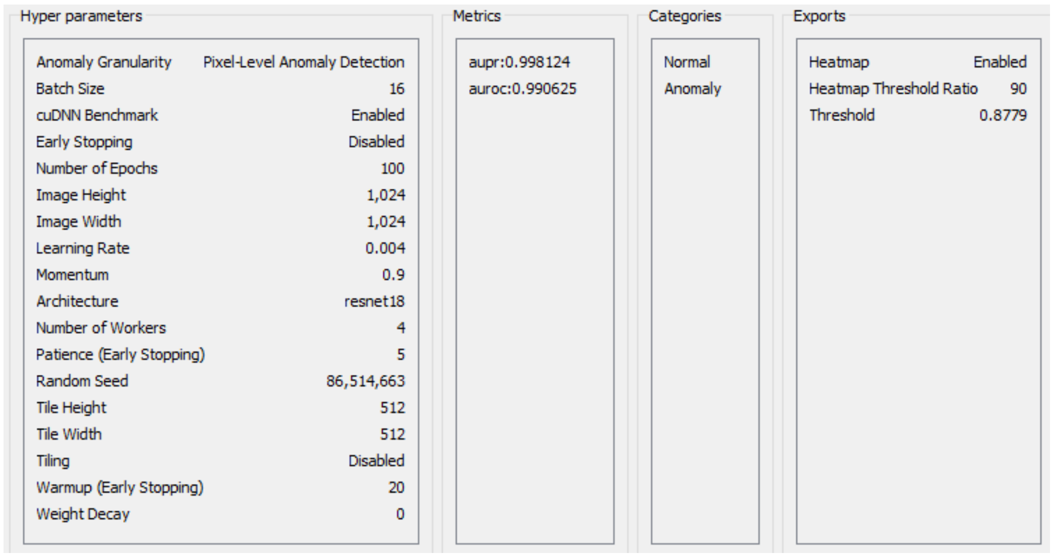
<install directory>\Teledyne DALSA\Sapera Processing\Images\AI



- hardware
- Metal
- PCB
- Scratches
- WoodDefects

Sample model details are available when importing models into Astrocyte.

Anomaly Detection Sample Model Details



Hyper parameters

Anomaly Granularity	Pixel-Level Anomaly Detection
Batch Size	16
cuDNN Benchmark	Enabled
Early Stopping	Disabled
Number of Epochs	100
Image Height	1,024
Image Width	1,024
Learning Rate	0.004
Momentum	0.9
Architecture	resnet18
Number of Workers	4
Patience (Early Stopping)	5
Random Seed	86,514,663
Tile Height	512
Tile Width	512
Tiling	Disabled
Warmup (Early Stopping)	20
Weight Decay	0

Metrics

aupr:0.998124
auroc:0.990625

Categories

Normal
Anomaly

Exports

Heatmap	Enabled
Heatmap Threshold Ratio	90
Threshold	0.8779

Classification Sample Model Details

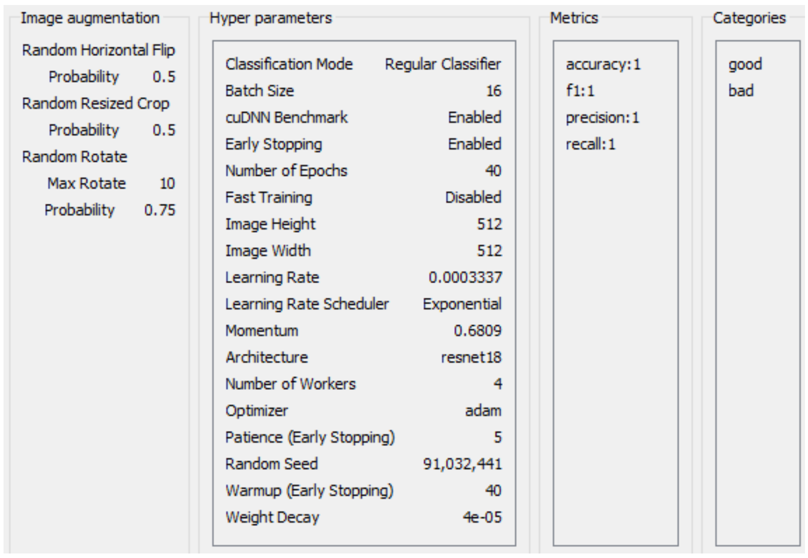


Image augmentation

Random Horizontal Flip	
Probability	0.5
Random Resized Crop	
Probability	0.5
Random Rotate	
Max Rotate	10
Probability	0.75

Hyper parameters

Classification Mode	Regular Classifier
Batch Size	16
cuDNN Benchmark	Enabled
Early Stopping	Enabled
Number of Epochs	40
Fast Training	Disabled
Image Height	512
Image Width	512
Learning Rate	0.0003337
Learning Rate Scheduler	Exponential
Momentum	0.6809
Architecture	resnet18
Number of Workers	4
Optimizer	adam
Patience (Early Stopping)	5
Random Seed	91,032,441
Warmup (Early Stopping)	40
Weight Decay	4e-05

Metrics

accuracy:1
f1:1
precision:1
recall:1

Categories

good
bad

Object Detection Sample Model Details

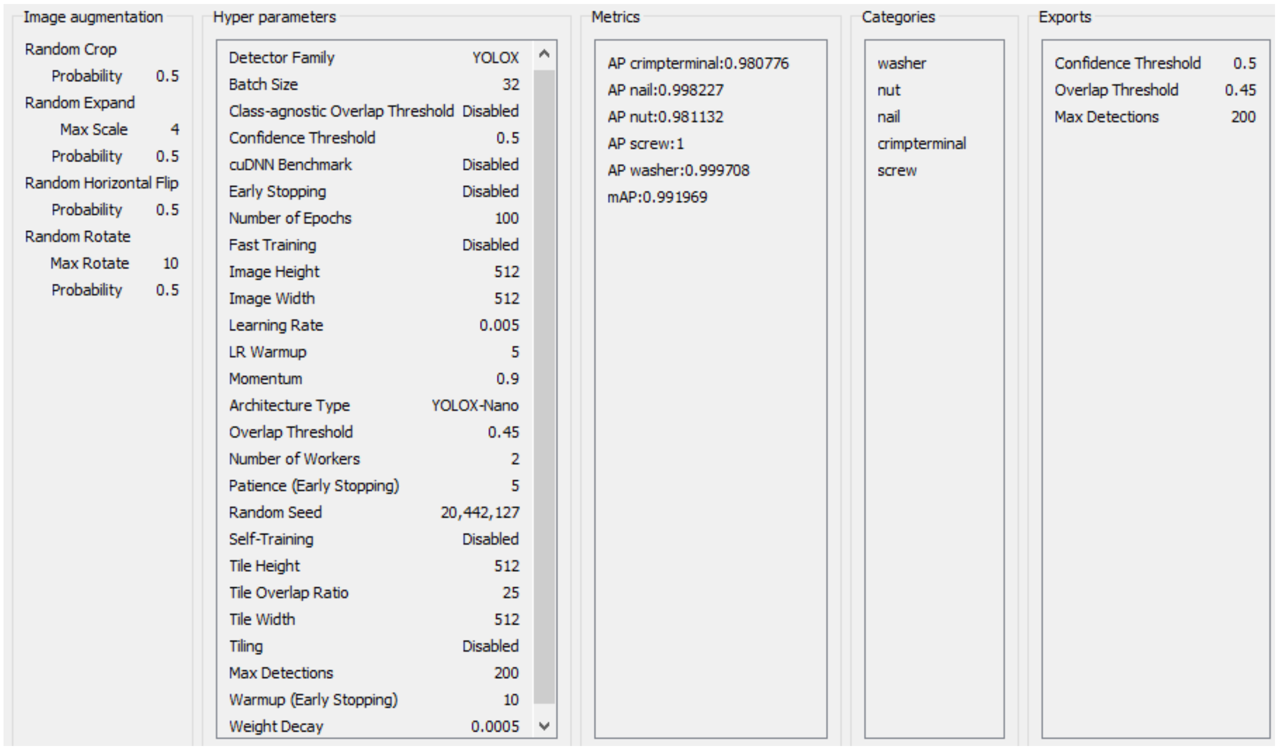


Image augmentation

Random Crop	
Probability	0.5
Random Expand	
Max Scale	4
Probability	0.5
Random Horizontal Flip	
Probability	0.5
Random Rotate	
Max Rotate	10
Probability	0.5

Hyper parameters

Detector Family	YOLOX
Batch Size	32
Class-agnostic Overlap Threshold	Disabled
Confidence Threshold	0.5
cuDNN Benchmark	Disabled
Early Stopping	Disabled
Number of Epochs	100
Fast Training	Disabled
Image Height	512
Image Width	512
Learning Rate	0.005
LR Warmup	5
Momentum	0.9
Architecture Type	YOLOX-Nano
Overlap Threshold	0.45
Number of Workers	2
Patience (Early Stopping)	5
Random Seed	20,442,127
Self-Training	Disabled
Tile Height	512
Tile Overlap Ratio	25
Tile Width	512
Tiling	Disabled
Max Detections	200
Warmup (Early Stopping)	10
Weight Decay	0.0005

Metrics

AP crimpterminal:0.980776
AP nail:0.998227
AP nut:0.981132
AP screw:1
AP washer:0.999708
mAP:0.991969

Categories

washer
nut
nail
crimpterminal
screw

Exports

Confidence Threshold	0.5
Overlap Threshold	0.45
Max Detections	200

Tiling Example Model Details

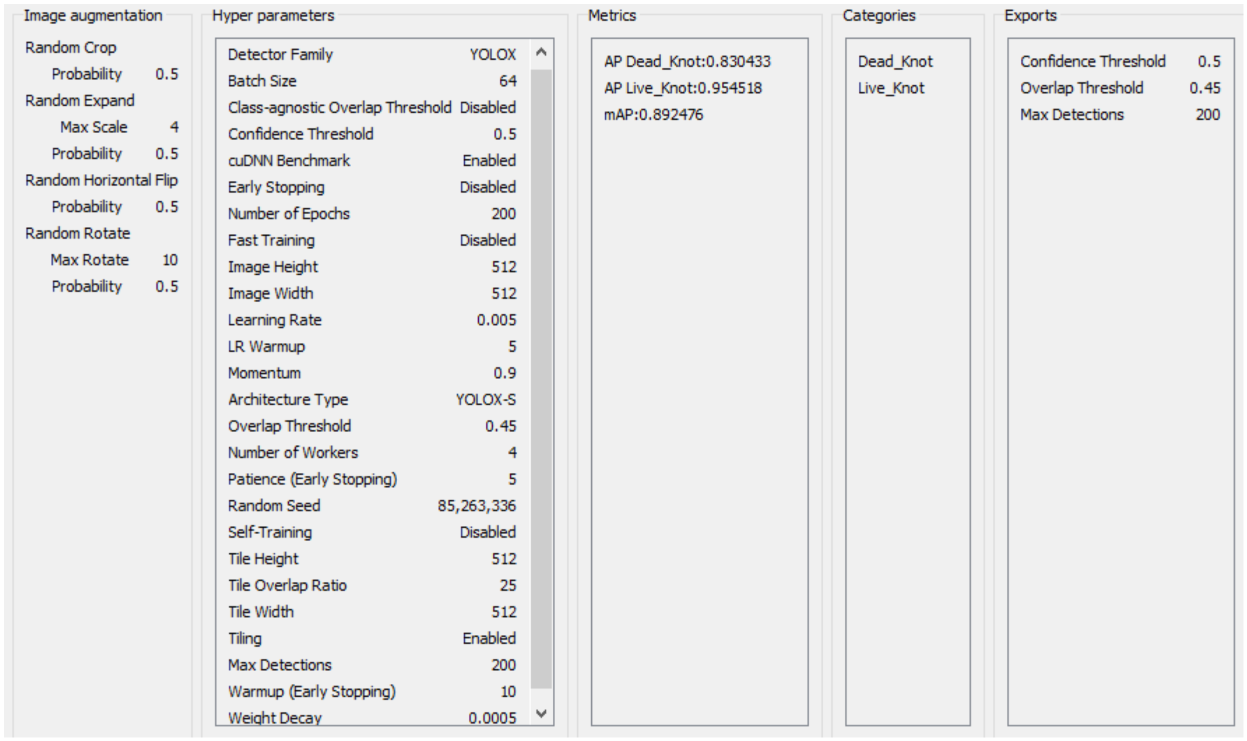


Image augmentation

Random Crop	
Probability	0.5
Random Expand	
Max Scale	4
Probability	0.5
Random Horizontal Flip	
Probability	0.5
Random Rotate	
Max Rotate	10
Probability	0.5

Hyper parameters

Detector Family	YOLOX
Batch Size	64
Class-agnostic Overlap Threshold	Disabled
Confidence Threshold	0.5
cuDNN Benchmark	Enabled
Early Stopping	Disabled
Number of Epochs	200
Fast Training	Disabled
Image Height	512
Image Width	512
Learning Rate	0.005
LR Warmup	5
Momentum	0.9
Architecture Type	YOLOX-S
Overlap Threshold	0.45
Number of Workers	4
Patience (Early Stopping)	5
Random Seed	85,263,336
Self-Training	Disabled
Tile Height	512
Tile Overlap Ratio	25
Tile Width	512
Tiling	Enabled
Max Detections	200
Warmup (Early Stopping)	10
Weight Decay	0.0005

Metrics

AP Dead_Knot:0.830433
AP Live_Knot:0.954518
mAP:0.892476

Categories

Dead_Knot
Live_Knot

Exports

Confidence Threshold	0.5
Overlap Threshold	0.45
Max Detections	200

Object Detection Sample Model Details

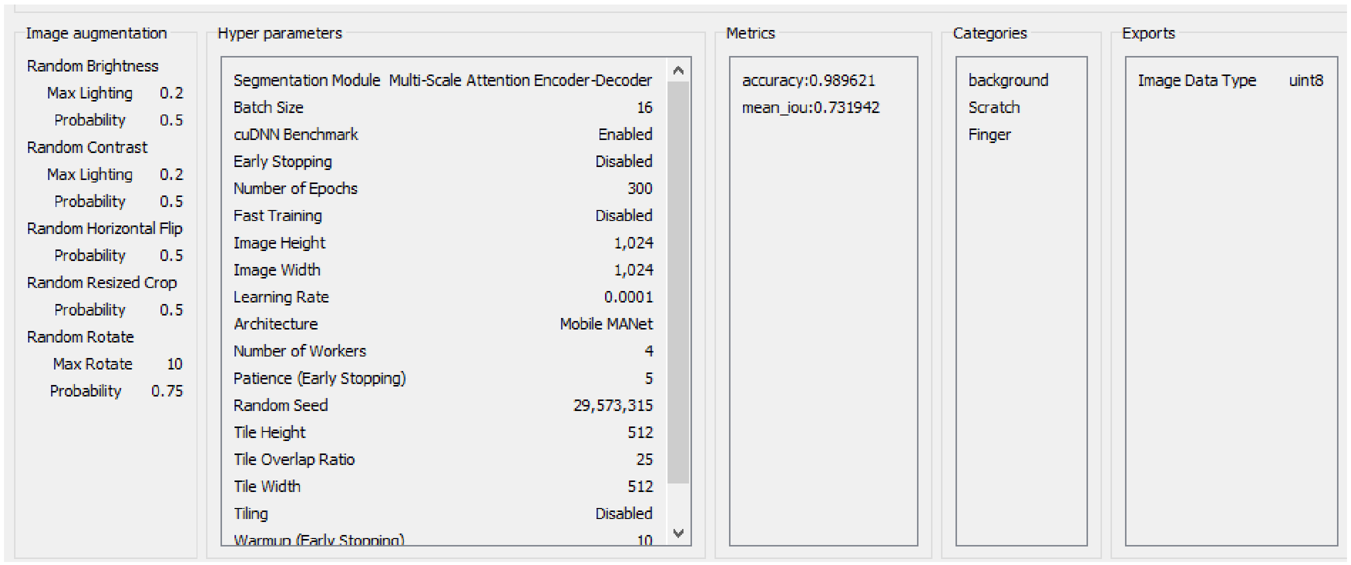


Image augmentation

Random Brightness	
Max Lighting	0.2
Probability	0.5
Random Contrast	
Max Lighting	0.2
Probability	0.5
Random Horizontal Flip	
Probability	0.5
Random Resized Crop	
Probability	0.5
Random Rotate	
Max Rotate	10
Probability	0.75

Hyper parameters

Segmentation Module	Multi-Scale Attention Encoder-Decoder
Batch Size	16
cuDNN Benchmark	Enabled
Early Stopping	Disabled
Number of Epochs	300
Fast Training	Disabled
Image Height	1,024
Image Width	1,024
Learning Rate	0.0001
Architecture	Mobile MANet
Number of Workers	4
Patience (Early Stopping)	5
Random Seed	29,573,315
Tile Height	512
Tile Overlap Ratio	25
Tile Width	512
Tiling	Disabled
Warmup (Early Stopping)	10

Metrics

accuracy:0.989621
mean_iou:0.731942

Categories

background
Scratch
Finger

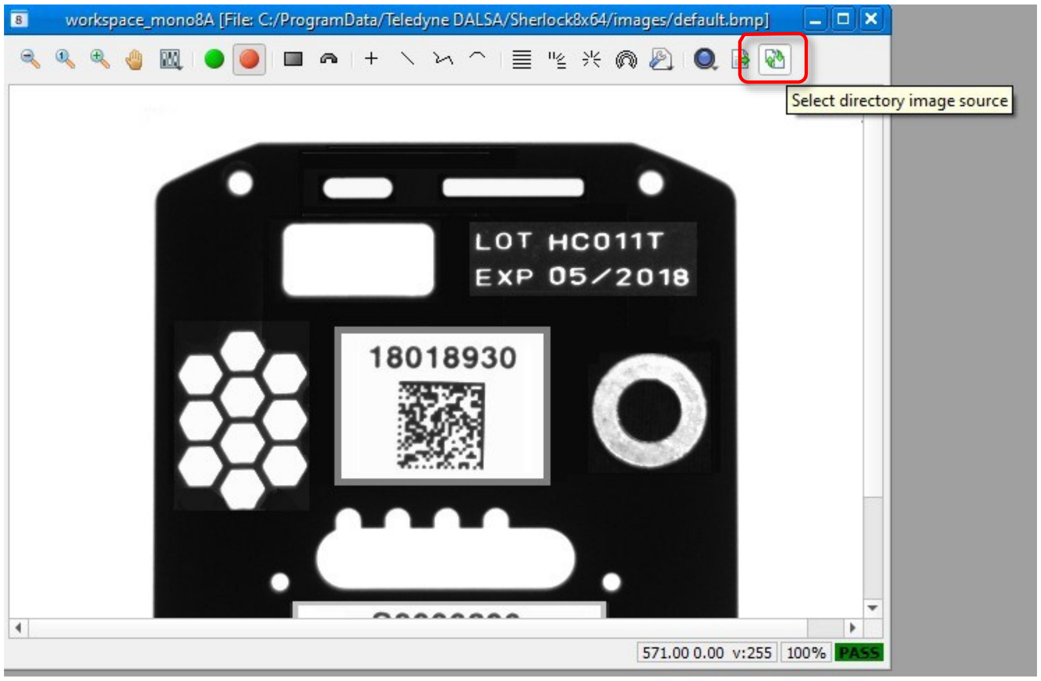
Exports

Image Data Type	uint8

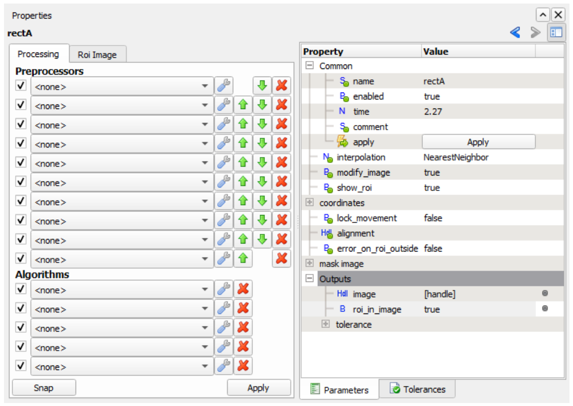
Deploying Models with Sherlock

Astrocyte models can be deployed using Teledyne DALSA's Sherlock application to analyze and process target images. Models must first be exported from the Astrocyte AI Trainer to use with the Sherlock.

Image directories (for example those included with Sapera Processing) can be loaded using the **Select directory image source** button.


workspace_mono8A [File: C:/ProgramData/Teledyne DALSA/Sherlock8x64/images/default.bmp]
Select directory image source
LOT HC011T
EXP 05/2018
18018930
571.00 0.00 v:255
100%
PASS

Double-clicking on an ROI in the image opens the Properties dialog.



Preprocessors

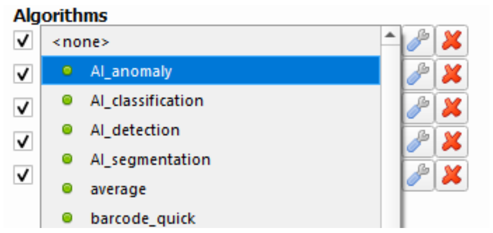
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>

Algorithms

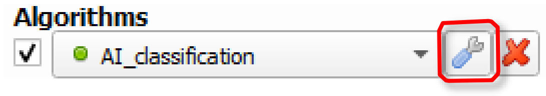
✓ <none>
✓ <none>
✓ <none>
✓ <none>
✓ <none>

Property	Value
Common	
name	rectA
enabled	true
time	2.27
comment	
apply	Apply
interpolation	NearestNeighbor
modify_image	true
show_roi	true
coordinates	
lock_movement	false
alignment	
error_on_roi_outside	false
mask image	
Outputs	
image	[handle]
roi_in_image	true
tolerance	

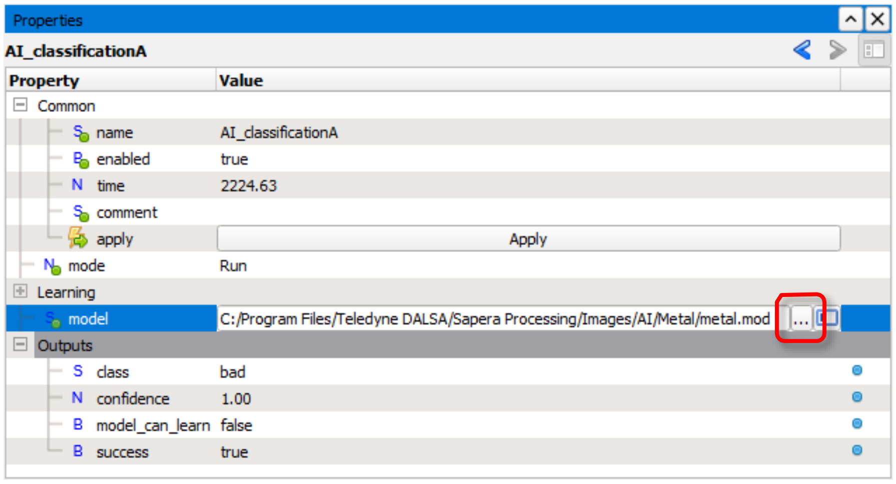
The Anomaly Detection, Classification, Object Detection and Segmentation modules are available in the Algorithms panel.



To load a model, open the Properties page for the selected module using the associated **Properties** button.



In the **model** parameter click on the Value field and load the *.mod* file using the **…** button.



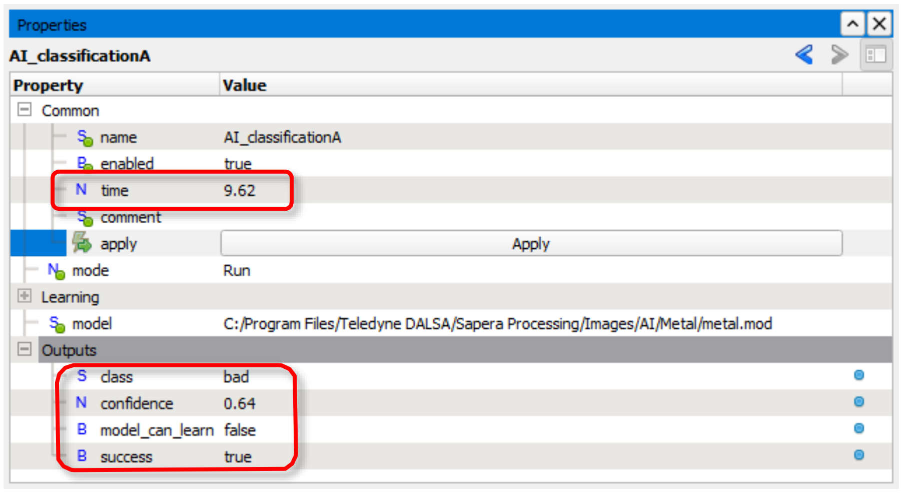
Properties

AI_classificationA

Property	Value
Common	
name	AI_classificationA
enabled	true
time	2224.63
comment	
apply	Apply
mode	Run
Learning	
model	C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal.mod ...
Outputs	
class	bad
confidence	1.00
model_can_learn	false
success	true

The **Run once** and **Run continuously** buttons can be used to perform the inference operation on the images.

Results are displayed in the Outputs and the execution time for the inference operation is displayed in the Time variable. For example, for Classification models the class and confidence score are displayed.



Properties

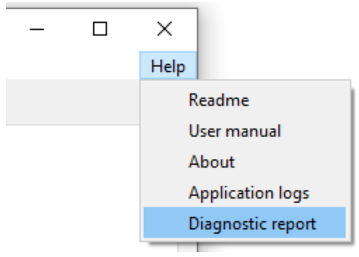
AI_classificationA

Property	Value
Common	
name	AI_classificationA
enabled	true
time	9.62
comment	
apply	Apply
mode	Run
Learning	
model	C:/Program Files/Teledyne DALSA/Sapera Processing/Images/AI/Metal/metal.mod
Outputs	
class	bad
confidence	0.64
model_can_learn	false
success	true

Troubleshooting and Maintenance

Help Menu

The Help button provides access to the Readme, User Manual, About, Application logs and Diagnostic report.

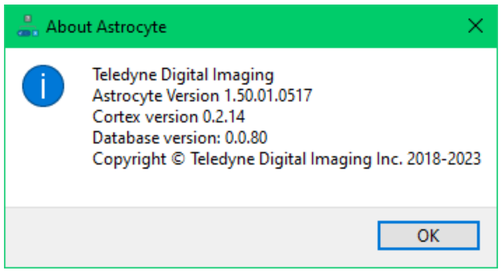


Readme File

The Readme file provides information about the current release.

About Dialog

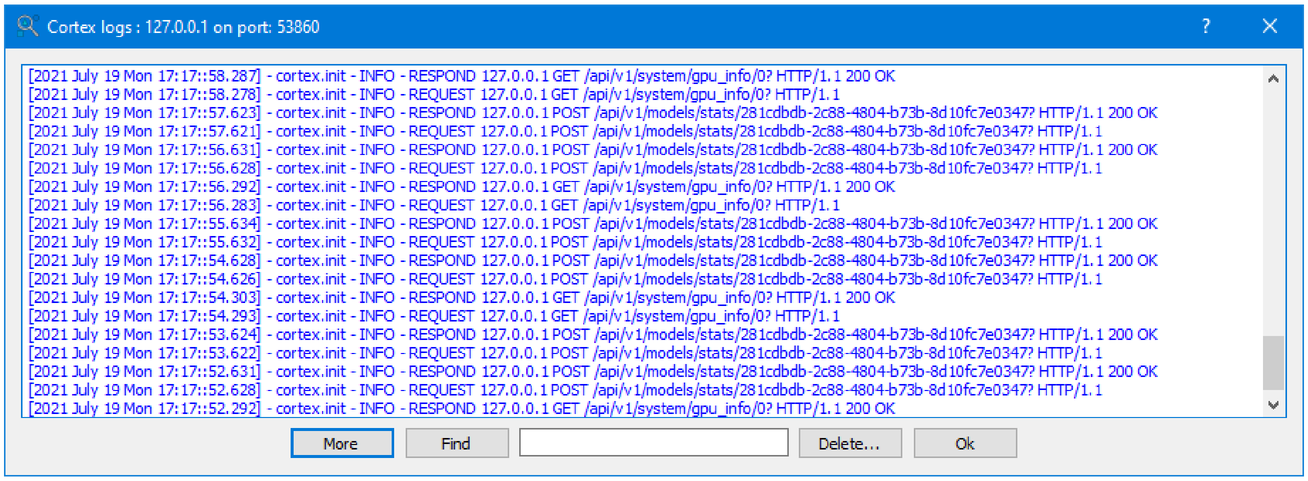
The About Astrocyte dialog displays the build and licensing information.



Teledyne Digital Imaging
Astrocyte Version 1.50.01.0517
Cortex version 0.2.14
Database version: 0.0.80
Copyright © Teledyne Digital Imaging Inc. 2018-2023

Application Logs

The application logs dialog provides detailed logging information that can be useful details for troubleshooting and debugging. The **Find** function allows for searching for text strings in the log to locate relevant entries. Use the **More** button to access additional available log files.

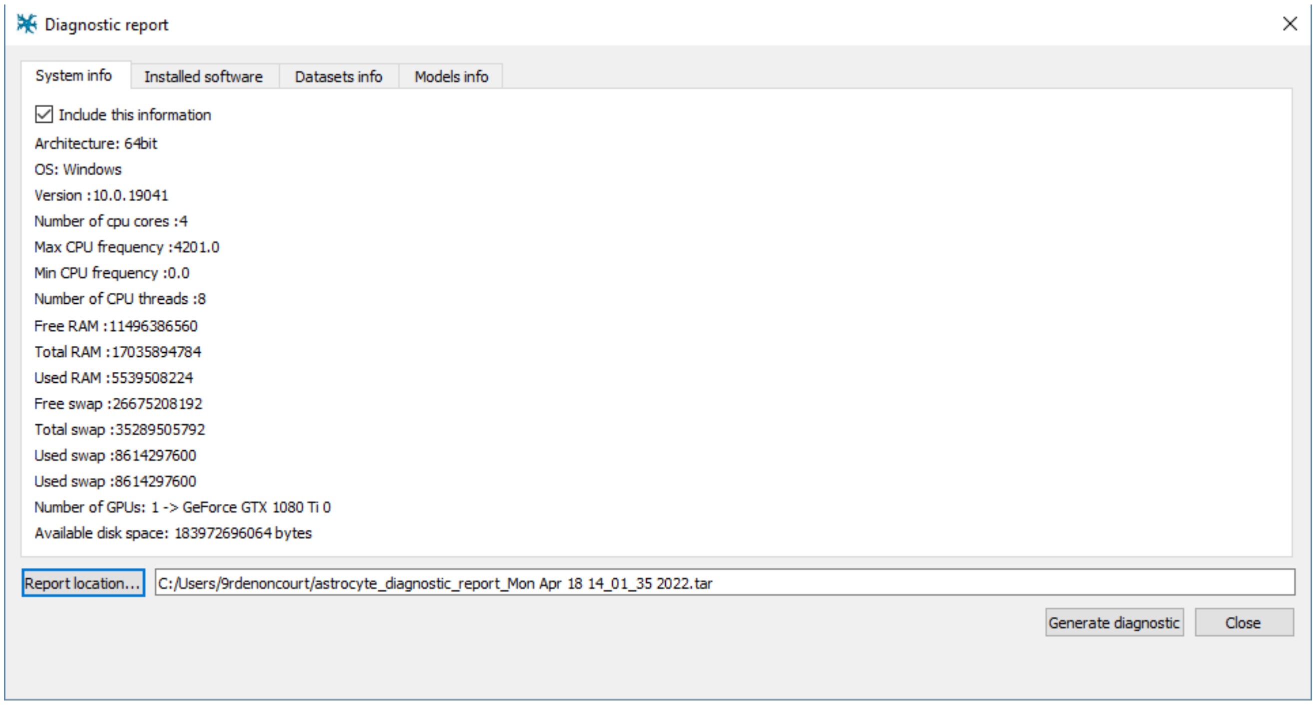


```
[2021 July 19 Mon 17:17::58.287] - cortex.init - INFO - RESPOND 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::58.278] - cortex.init - INFO - REQUEST 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1
[2021 July 19 Mon 17:17::57.623] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::57.621] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::56.631] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::56.628] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::56.292] - cortex.init - INFO - RESPOND 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::56.283] - cortex.init - INFO - REQUEST 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1
[2021 July 19 Mon 17:17::55.634] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::55.632] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::54.628] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::54.626] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::54.303] - cortex.init - INFO - RESPOND 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::54.293] - cortex.init - INFO - REQUEST 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1
[2021 July 19 Mon 17:17::53.624] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::53.622] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::52.631] - cortex.init - INFO - RESPOND 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1 200 OK
[2021 July 19 Mon 17:17::52.628] - cortex.init - INFO - REQUEST 127.0.0.1 POST /api/v1/models/stats/281cdbdb-2c88-4804-b73b-8d10fc7e0347? HTTP/1.1
[2021 July 19 Mon 17:17::52.292] - cortex.init - INFO - RESPOND 127.0.0.1 GET /api/v1/system/gpu_info/0? HTTP/1.1 200 OK
```

Diagnostic Report

For troubleshooting purposes or sending requests to Teledyne technical support, a diagnostic report can be generated that, in addition to the application logs can include system information, installed software, dataset information and model information; the user can select what type of information to include (dataset images can also be provided if appropriate).

Click the **Generate diagnostic** button to save all enabled information to the selected report location as a *.tar* archive file.



☑ Include this information

```
Architecture: 64bit
OS: Windows
Version :10.0.19041
Number of cpu cores :4
Max CPU frequency :4201.0
Min CPU frequency :0.0
Number of CPU threads :8
Free RAM :11496386560
Total RAM :17035894784
Used RAM :5539508224
Free swap :26675208192
Total swap :35289505792
Used swap :8614297600
Used swap :8614297600
Number of GPUs: 1 -> GeForce GTX 1080 Ti 0
Available disk space: 183972696064 bytes
```

Report location... C:/Users/9rdenoncourt/astrocyte_diagnostic_report_Mon Apr 18 14_01_35 2022.tar

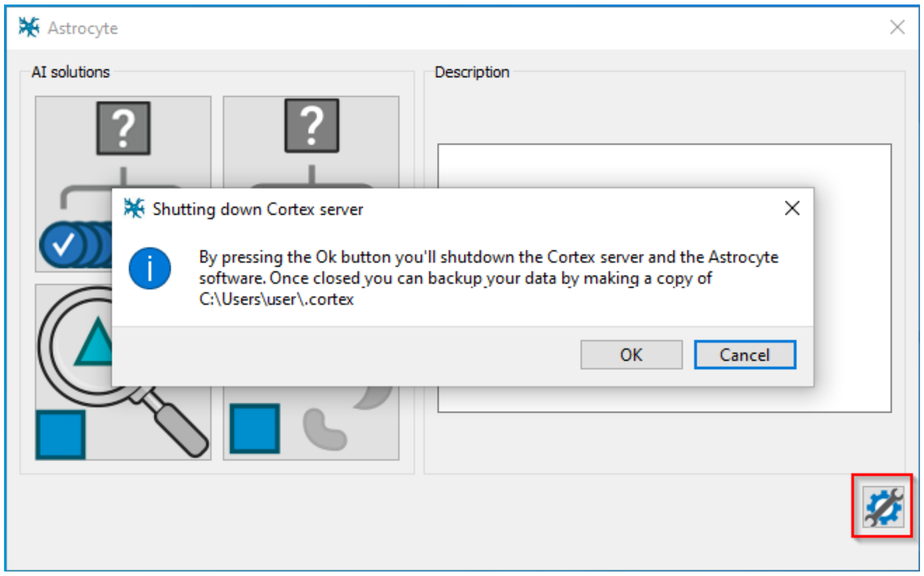
Backup of Astrocyte Database

The Astrocyte database contain all datasets and models you have created in Astrocyte since its first installation. The database persists after uninstalling Astrocyte and reinstalling another version later. The Astrocyte database is located in the following folder on your disk.

C:\Users\<your account>\.cortex

If you want to transfer the whole database to a different computer or if you want to create a backup to prevent a disk failure then you can do the following steps.

Step 1. Shutdown Astrocyte using the maintenance button located in the bottom-left corner of the startup screen. This ensures the Cortex server is fully closed before you can copy the database.



By pressing the Ok button you'll shutdown the Cortex server and the Astrocyte software. Once closed you can backup your data by making a copy of C:\Users\user\.cortex

Step 2. Copy the whole **.cortex** folder (with all subfolders) to the desired location for transfer or backup.

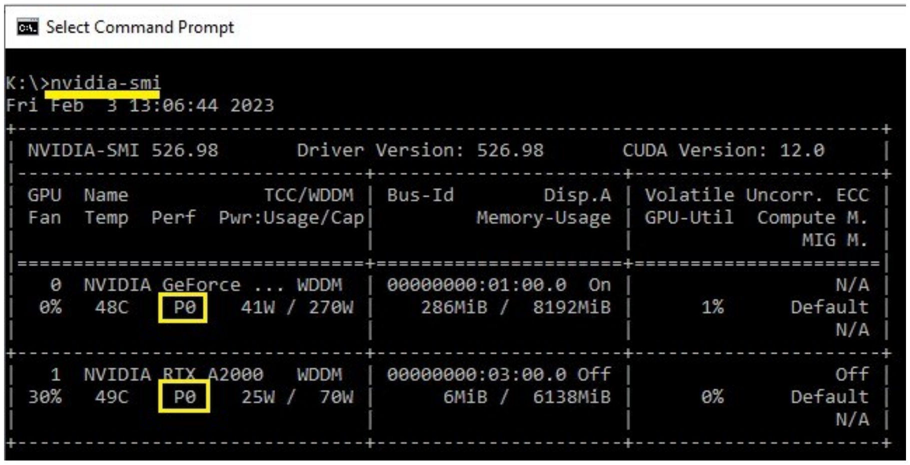
To restore the database redo Step 1 and then copy the whole **.cortex** server back to the same location.

Appendix A: NVIDIA Driver Settings for Maximum Performance

To obtain the optimal speed performance, the graphics card driver's Power Management Mode needs to be set to "**Prefer maximum performance"**.

Verifying the Current Mode

To check whether this setting is properly configured type, use the command **nvidia-smi** in a command prompt. Verify that the power management level is set to **P0 or P2** for each GPU card used for AI inference. This level ranges from P0 to P8 with P0/P2 for optimal performance and P8 for optimal power saving.



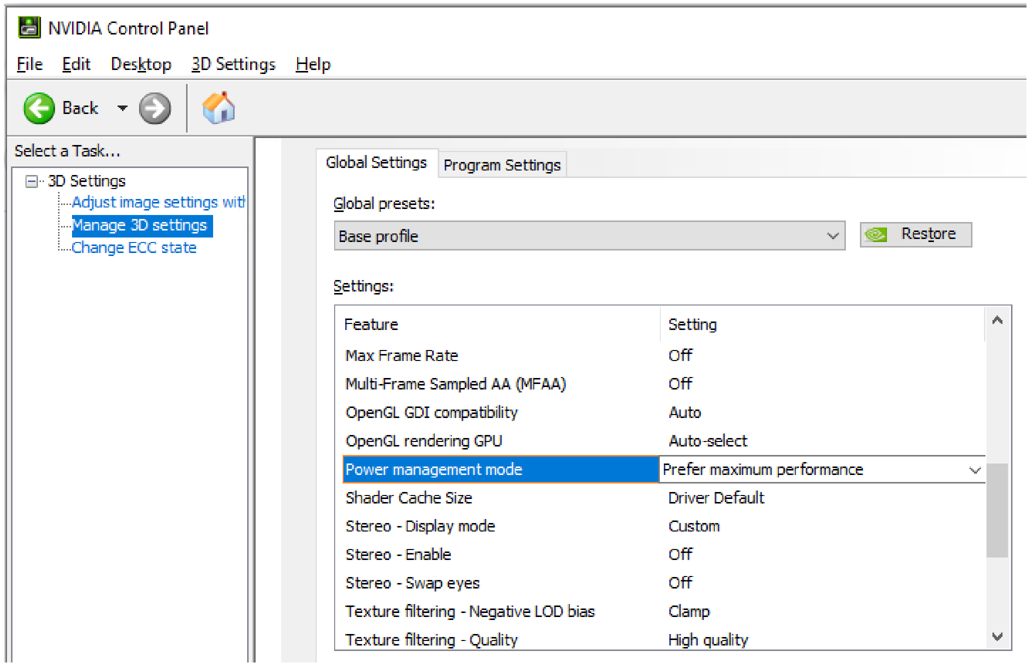
```
K:\>nvidia-smi
Fri Feb  3 13:06:44 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 526.98       Driver Version: 526.98       CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0  On |                  N/A |
|  0%   48C    P0    41W / 270W |    286MiB /  8192MiB |      1%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA RTX A2000    WDDM | 00000000:03:00.0 Off |                  Off |
| 30%   49C    P0    25W /  70W |      6MiB /  6138MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

If the power management level is not properly set to P0 or P2, see Changing GPU Power Saving Mode (Method A - Preferred).

Changing GPU Power Saving Mode (Method A - Preferred)

The *Power management mode* setting is set using the **NVIDIA Control Panel**.

IMPORTANT NOTE: *you need to reboot your system for the changes to take effect. NVIDIA Control Panel does not notify you to do so.*



Global presets: Base profile

Settings:

Feature	Setting
Max Frame Rate	Off
Multi-Frame Sampled AA (MFAA)	Off
OpenGL GDI compatibility	Auto
OpenGL rendering GPU	Auto-select
Power management mode	Prefer maximum performance
Shader Cache Size	Driver Default
Stereo - Display mode	Custom
Stereo - Enable	Off
Stereo - Swap eyes	Off
Texture filtering - Negative LOD bias	Clamp
Texture filtering - Quality	High quality

After the change is applied and the system rebooted, verify the power management level again with the **nvidia-smi** command.

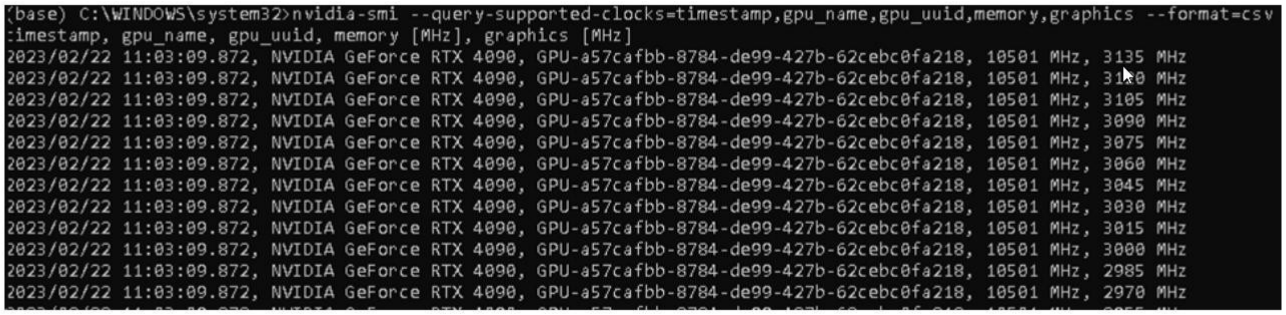
Changing GPU Power Saving Mode (Method B - Alternate)

If the *Power saving mode* could not be set properly with Method A then the following method should work.

Note: all the commands must be executing in an admin command prompt.

First list the supported combinations of memory and graphics clock with the following: nvidia-smi --query-supported- clocks=timestamp,gpu_name,gpu_uuid,memory,graphics --format=csv

Output is similar to this:



```
(base) C:\WINDOWS\system32>nvidia-smi --query-supported-clocks=timestamp,gpu_name,gpu_uuid,memory,graphics --format=csv
imestamp, gpu_name, gpu_uuid, memory [MHz], graphics [MHz]
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3135 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3120 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3105 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3090 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3075 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3060 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3045 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3030 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3015 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 3000 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 2985 MHz
2023/02/22 11:03:09.872, NVIDIA GeForce RTX 4090, GPU-a57cafbb-8784-de99-427b-62cebc0fa218, 10501 MHz, 2970 MHz
```

Only the first line matters as the goal is to lock the memory and GPU clocks to the highest values. In this example, these are 10501 and 3135, respectively.

Type the following commands:

nvidia-smi—lock-gpu-clocks=<your GPU clock value>

nvidia-smi—lock-memory-clocks=<your memory clock value>

In this example:

nvidia-smi --lock-gpu-clocks=3135 nvidia-smi --lock-memory-clocks=10501

You can run the nvidia-smi command again and check that we get the PO or P2 mode. Once done using GPU at maximum performance you can bring the driver back to its default state with the following commands:

nvidia-smi --reset-gpu-clocks nvidia-smi --reset-memory-clocks

which will cause the performance level to go back to P8 (power saving mode).

Glossary of Terms

AUPR

Area Under Precision Recall. It is the area under the curve where x is recall and y is precision.

AUROC

Area Under Receiver Operating Characteristic. It is the area under the curve where x is false positive rate (FPR) and y is true positive rate (TPR). The greater the AUROC the better the model performance.

Batch

The set of examples used in one iteration of model training.

Confusion Matrix

Table that describes the performance of a classification model.

Convergence

Refers to a state reached during training in which training loss and validation loss change very little or not at all with each iteration after a certain number of iterations. In other words, a model reaches convergence when additional training on the current data will not improve the model. In deep learning, loss values sometimes stay constant or nearly so for many iterations before finally descending, temporarily producing a false sense of convergence.

Epoch

A full training pass over the entire dataset such that each image has been seen once. Thus, an epoch represents N/batch size training iterations, where N is the total number of images.

Ground Truth

For supervised learning, ground truth refers to the labels (and bounding boxes if used) applied to the training set which designate the actual target object(s) in the image. Statistics related to the accuracy of the model's predictions are generated compared to the real-world ground truth.

Inference

Refers to the process of making predictions by applying the trained model to unlabelled images; that is, classifying data to derive a conclusion or infer a result.

Iteration

A single update of a model's weights during training. An iteration consists of computing the gradients of the parameters with respect to the loss on a single batch of data (images).

Loss Function

For each predication, there is an associated number which is the loss. For a true prediction, the loss will be small and for a totally wrong prediction the loss will be high. Also referred to as cost function.

mAP

Mean average precision. The mean value of the APs calculated for each category in the model. In practice, a higher mAP value indicates a better performance.

Model

A structure that stores a generalized, internal representation of a dataset for description or prediction. When a machine learning algorithm trains on a dataset, the output is a model.

Neural Network

Neural networks are made up of numerous interconnected conceptualized artificial neurons, which pass data between themselves, and which have associated weights which are tuned based upon the network's "experience." Neurons have activation thresholds which, if met by a combination of their associated weights and data passed to them, are fired; combinations of fired neurons result in "learning." - Liping Yang

Overfitting

Overfitting occurs when a model learns the training data too well and incorporates details and noise specific to the dataset. A model is overfitting when it performs very well on the training/validation set, but poorly on a test set (or new real-world data).

Precision

Measures the accuracy of a model's predictions; the percentage of predictions that are correct.

PSNR

Peak signal-to-noise ratio (PSNR) between images. The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. This ratio is used as a quality measurement between the original and a compressed image. The higher the PSNR, the better the quality of the compressed, or reconstructed image.

Recall

Measures the performance of the model in identifying true positives. That is, the number of correct results divided by the number of results that should have been returned; the fraction of true positives over the number of actual positives.

SSIM

Structural similarity (SSIM) index for measuring image quality. SSIM is used for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measurement or prediction of image quality is based on an initial uncompressed or distortion-free image as reference. SSIM is designed to improve on traditional methods such as peak signal-to-noise ratio (PSNR) and mean squared error (MSE).

Transfer Learning

Allows new model training that leverages the already existing neural network of a model trained on a similar dataset related to the current task or domain.

Weight Decay

Penalizes large artificial neuron weights using penalties or constraints on their squared or absolute values.

Contact Information

Sales Information

Visit our web site: https://www.teledynedalsa.com/
Email: mailto:info@teledynedalsa.com

Canadian Sales	Canadian Sales
Teledyne DALSA — Head office 605 McMurray Road Waterloo, Ontario N2V 2E9 Canada Tel: +1 519-886-6000 Fax: +1 519-886-8023	Teledyne DALSA — Montreal office 880 Rue McCaffrey Saint-Laurent, Quebec H4T 2C7 Canada Tel: +1 514-333-1301 Fax: +1 514-333-1388
USA Sales	**European Sales**
Teledyne DALSA — Billerica office 700 Technology Park Drive Billerica, MA 01821 USA Tel: +1 978-670-2000 Fax: +1 978-670-2010 sales.americas@teledynedalsa.com	Teledyne DALSA GMBH Lise-Meitner-Str. 7 82152 Krailling (Munich) Germany Tel: +49 89-89545730 Fax: +49 89-895457346 sales.europe@teledynedalsa.com
Asia Pacific Sales	**Asia Pacific Sales**
Teledyne DALSA Asia Pacific Ikebukuro East 6F 3-4-3 Higashi Ikebukuro, Toshima-ku Tokyo, 170-0013 Japan Tel: +81 3-5960-6353 Fax: +81 3-5960-6354 sales.asia@teledynedalsa.com	Teledyne DALSA Asia Pacific Room 904, Block C, Poly West Bund Center 275 Rui Ping Road Shanghai 200032 China Tel: +86 21-60131571 sales.asia@teledynedalsa.com

Technical Support

Submit any support question or request via our web site:
https://www.teledynedalsa.com/en/support/options/

Available support information includes:

Warranty Information	**Camera Accessories**	**Documentation**
Third-Party Components	**Calculators**	**Partners**
Legacy Products	**Compatible Products**	**Software Registration**