
Mech-DLK

Mech-Mind

Dec 02, 2022

INTRODUCTION

1	Mech-DLK V2.2.1 Release Notes	6
2	Installation	7
3	Train the First Model	12
4	Fast Positioning	18
5	Defect Segmentation	25
6	Classification	42
7	Object Detection	60
8	Instance Segmentation	77
9	Module Cascading	104
10	Running Mode	116
11	Image Preprocessing Tool	121
12	Data Augmentation	123
13	Keyboard Shortcuts	126
14	About Mech-DLK SDK	128
15	Getting Started with SDK	130
16	API Reference Guide	143
17	Prerequisites for Using Mech-Mind Software	149
18	Terminology	162
19	Compatibility	163
20	Support	166
21	FAQ	185

Mech-DLK is a machine vision deep learning software independently developed by Mech-Mind Robotics. With a variety of built-in industry-leading deep learning algorithms, it can solve many problems that traditional machine vision cannot handle, such as highly difficult segmentation, positioning, classification, etc.

Through intuitive and simple UI interactions, even without programming or specialized deep learning knowledge, users can quickly implement model training and verification with Mech-DLK.



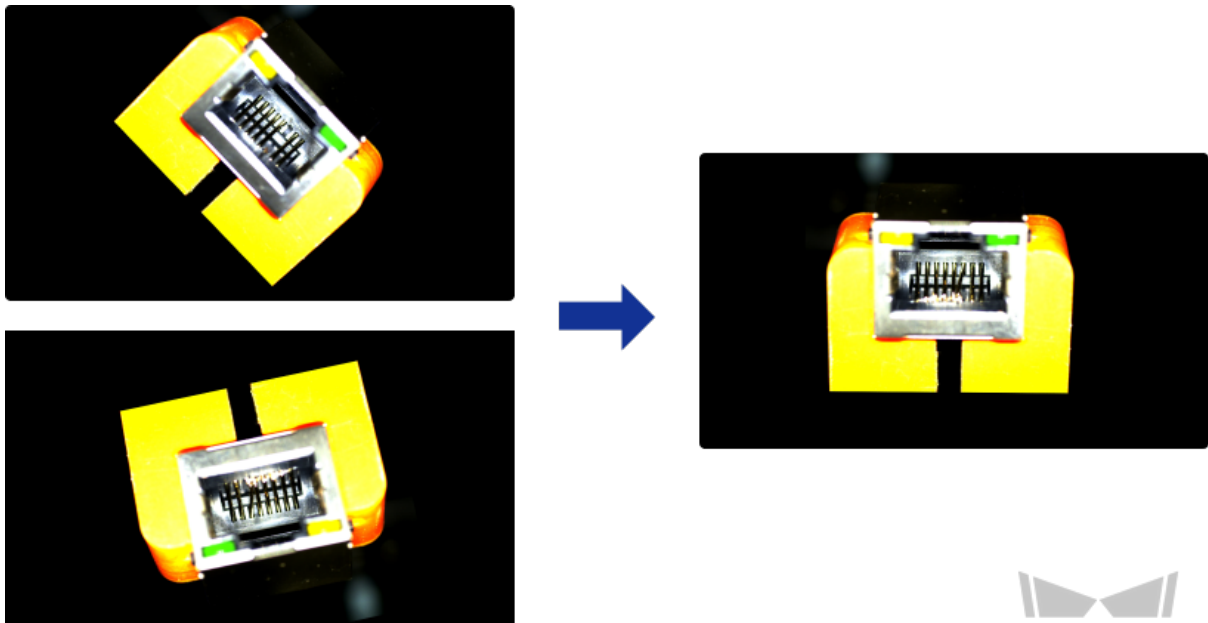
The software includes five algorithm modules: Fast Positioning, Defect Segmentation, Classification, Object Detection, and Instance Segmentation.

Fast Positioning

Recognize the object orientation in an image and correct the image based on the recognition result.

The module is used to correct object image orientations. It runs fast and is usually used as a preceding module for other algorithm modules.

- Recognize workpiece orientations in images and rotate the images to a specified orientation.



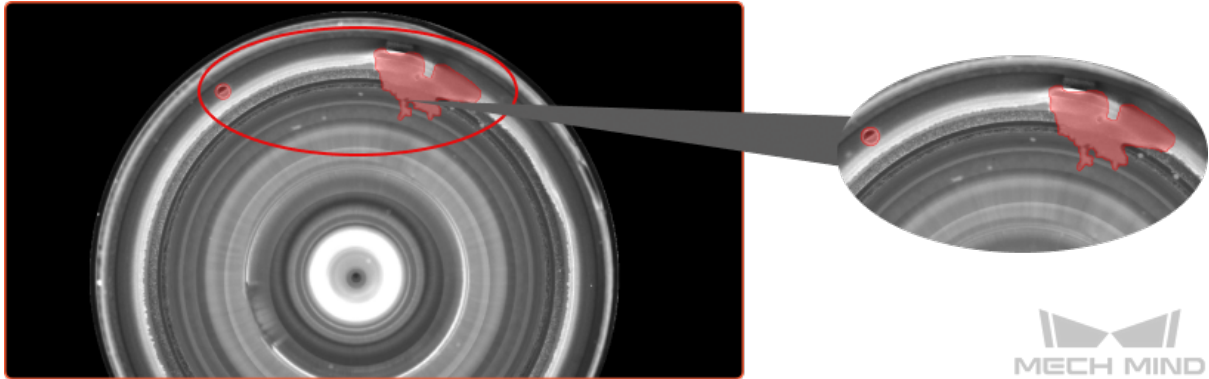


Defect Segmentation

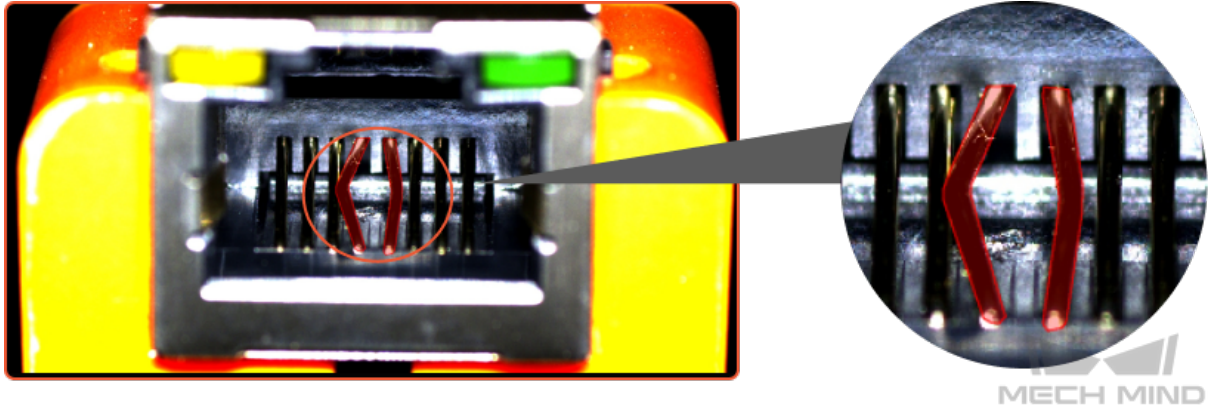
Determine whether an image is OK or NG. If it is NG, segment the defect region(s).

The module is used to detect various types of defects, including surface defects such as stains, bubbles, scratches, etc., positional defects such as bending, abnormal shape, absence, etc. It can be applied in complex situations such as small defects, complex backgrounds, and unstable workpiece positions.

- Detect air bubbles and glue spill defects on the lens surface.



- Detect bending defects of workpieces.

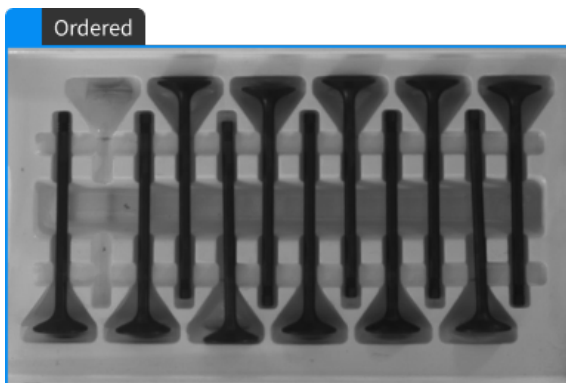


Classification

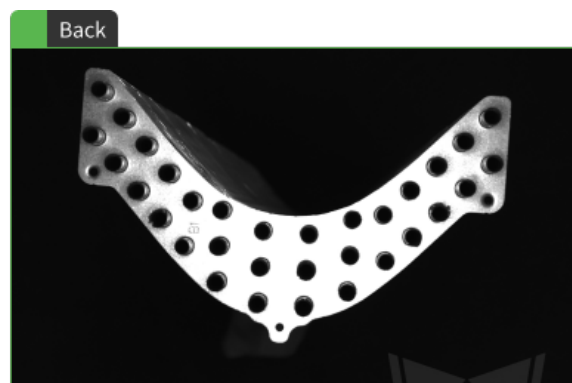
Recognize object classes in images.

The module is used to recognize workpiece front and back faces, workpiece orientations, and defect types, and to recognize whether objects are missing, or whether objects are neatly arranged.

- Recognize whether workpieces are neatly arranged or scattered.



- Recognize the fronts and backs of workpieces.

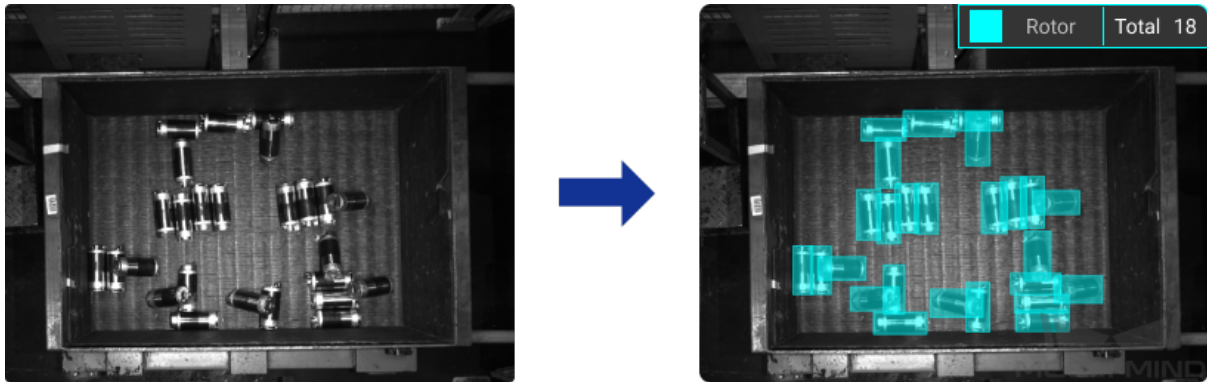


Object Detection

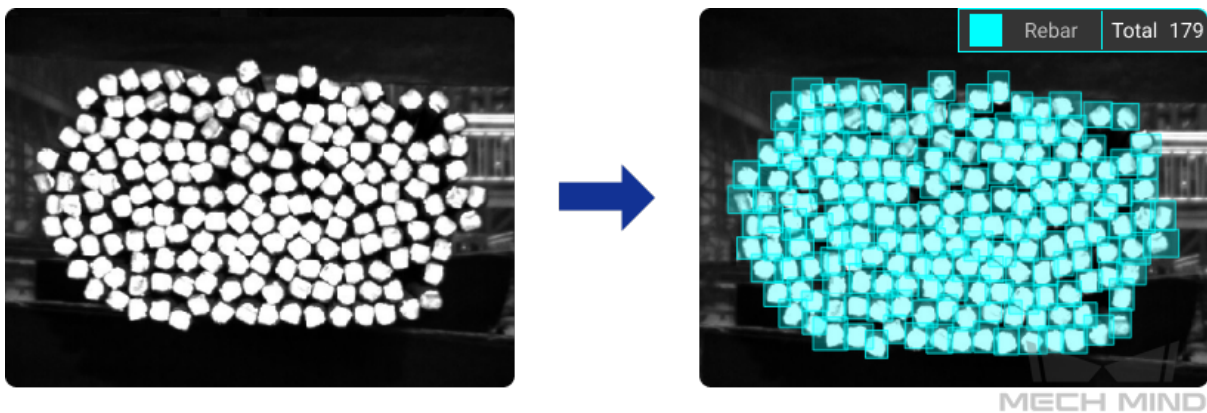
Detect the positions of all target objects and recognize their categories at the same time.

This module is used to detect the absence of workpieces of fixed position, such as missing components in a PCB; it can also be used for object counting. Even for hundreds or thousands of objects, the module can quickly perform locating and counting.

- Detect rotors that overlap each other.



- Count all rebars.



Instance Segmentation

Recognize the contour and class of each target object.

This module can produce more refined segmentation results than the module “Object Detection”. The module can recognize single or multi-class objects and segment the corresponding contours.

It is used for depalletizing, machine tending, piece picking, etc., and it cooperates with Mech-Vision and Mech-Viz to complete object picking.

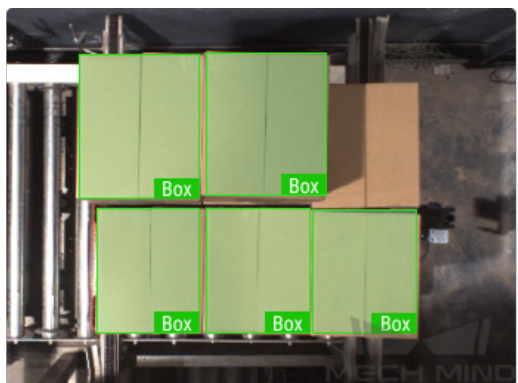
- Segment blocks of various types.



- Segment scattered and overlapping chain links.



- Segment cartons placed tightly and parallelly together.



MECH-DLK V2.2.1 RELEASE NOTES

1.1 New Features

1.1.1 Added the Function of Showing the Class Activation Maps for Module “Classification”

After the model is trained, click *Generate CAM*. The class activation maps show the weights of the features in the form of heat maps; the model classifies an image into its class according to these features. Image regions with warmer colors have higher weights for classifying the image into its class.

1.1.2 Verification and Export of CPU models

- **Classification, Object Detection:** After training is complete, select the deployment device as CPU or GPU before exporting the model.
- **Instance segmentation:** Before training the model, set the training parameters. When exporting a model, select the deployment device as CPU/GPU:
 - *CPU lightweight model* : Before training the model, set the training parameter **Model type** to **Lite (better with CPU deployment)**. When exporting the model for deployment, set **Deployment device** to **CPU** or **GPU**.
 - *GPU standard model* : Before training the model, set the training parameter **Model type** to **Normal (better with GPU deployment)**. When exporting the model for deployment, it is recommended to set **Deployment device** to **GPU**.

INSTALLATION

A Mech-Mind software environment should be installed to guarantee the proper functioning of Mech-DLK. This section covers the prerequisites for running Mech-DLK and how to install the Mech-Mind software environment and Mech-DLK.

2.1 Device Prerequisites

	Mech-DLK Pro-Run	Mech-DLK Pro-Train/Standard
Operating system	Windows 10 or above	
CPU	Intel® Core™ i5 or above	Intel® Core™ i7 or above
RAM	8 GB or above	16 GB or above
Graphics card	GeForce GTX 1650 (4GB) or above	GeForce GTX 2070 (8GB) or above
Graphics card driver	471.68 or above	

2.1.1 Requirements for the Graphics Card

- The computer graphics card's computation capacity should be at least that of Nvidia GeForce 6.1.
- [Click here](#) to check the compute capability for your GPU.

GeForce and TITAN Products

GPU	Compute Capability
GeForce RTX 3060 Ti	8.6
GeForce RTX 3060	8.6
GeForce RTX 3090	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
GeForce GTX 1650 Ti	7.5
NVIDIA TITAN RTX	7.5
GeForce RTX 2080 Ti	7.5
GeForce RTX 2080	7.5
GeForce RTX 2070	7.5
GeForce RTX 2060	7.5
NVIDIA TITAN V	7.0
NVIDIA TITAN Xp	6.1
NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1

GeForce Notebook Products

GPU	Compute Capability
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
GeForce RTX 3060	8.6
GeForce RTX 3050 Ti	8.6
GeForce RTX 3050	8.6
GeForce RTX 2080	7.5
GeForce RTX 2070	7.5
GeForce RTX 2060	7.5
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 980	5.2
GeForce GTX 980M	5.2
GeForce GTX 970M	5.2
GeForce GTX 965M	5.2

2.2 Install the Mech-Mind Software Environment

- If you are using Mech-DLK for inference only, then you do not need to install the Mech-Mind Software Environment.
- If you are using Mech-DLK for both model training and inference, please install the Mech-Mind Software Environment according to the following instructions.
 1. Download the file *Mech-Mind_software_environment_installer*.
 2. Right-click on the file and go to *CRC SHA* → *CRC-32* to check if the file has been corrupted or not. The CRC32 should be the same as the number in the installer name.
 3. Extract the file with any extraction tools.
 4. Double click on the *Mech_Mind_software_environment_installer.exe* file and install according to the instructions.

Mech-Mind_software_environment_installer(2.2.0)

required_resource_packs

Mech_Mind_software_environment_installer.exe

readme.txt

2.3 Install Mech-DLK

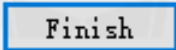
1. Download the file *Mech_DLK_installer.exe*.
2. Double click on the file and install it according to the instructions. A window as shown below will appear if the software has been installed successfully.



Mech-DLK Setup

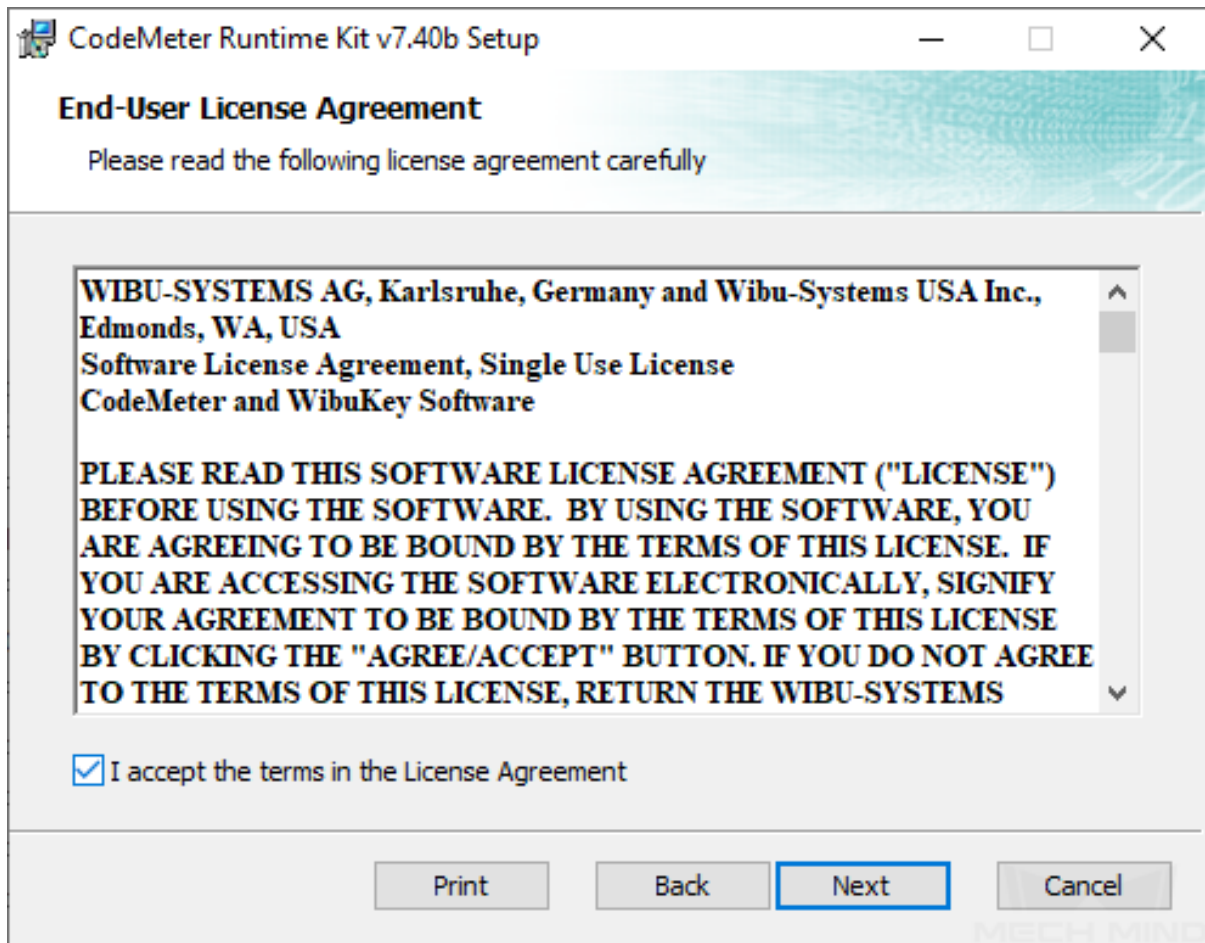
Completing the Mech-DLK Wizard

Click Finish to exit the Mech-DLK Wizard.

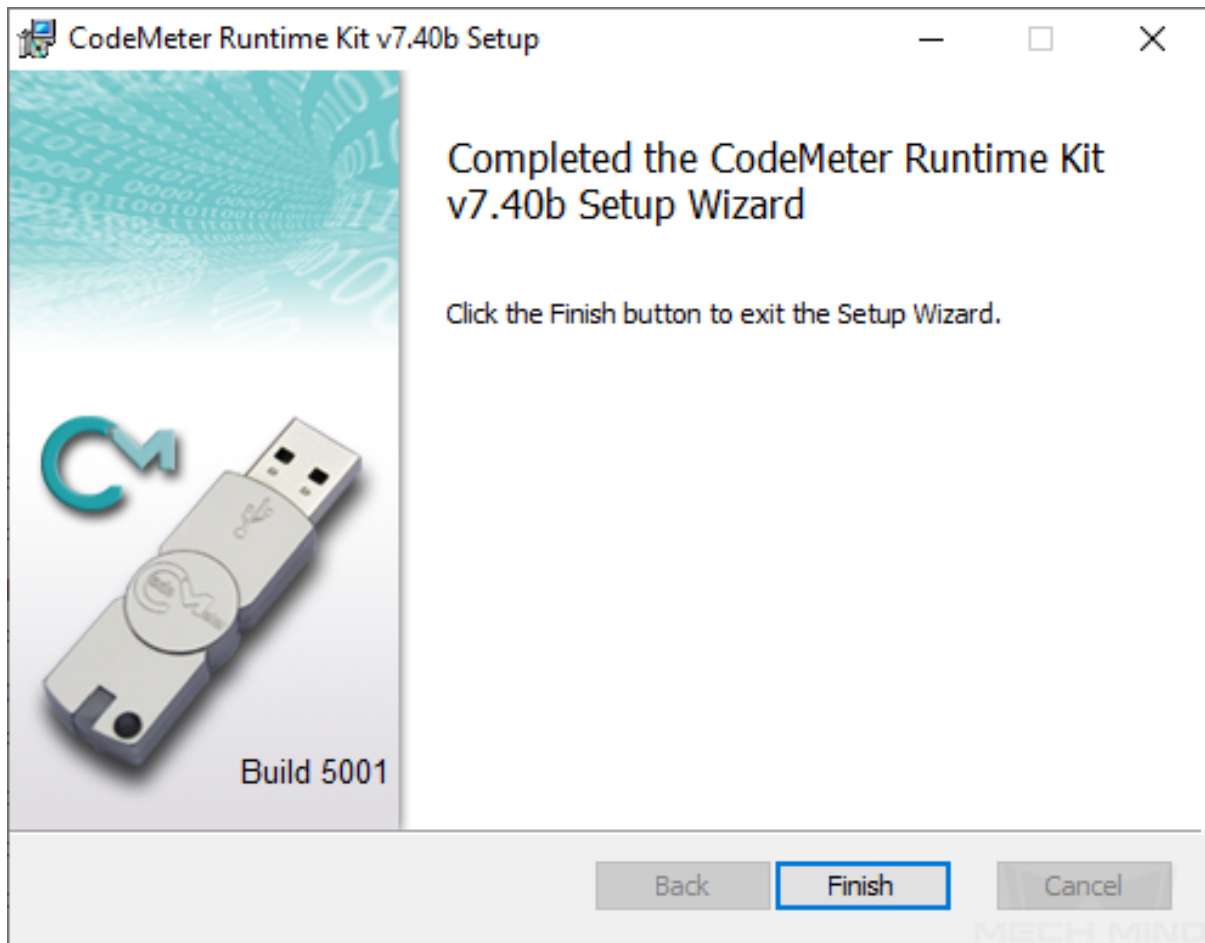
 Finish

2.4 Install the Dongle Driver

1. Run the CodeMeter installer received from Mech-Mind to install CodeMeter. Check the option as shown below, and then select *Next* to complete the installation.



2. A window as shown below will appear if the setting-up has been completed successfully.



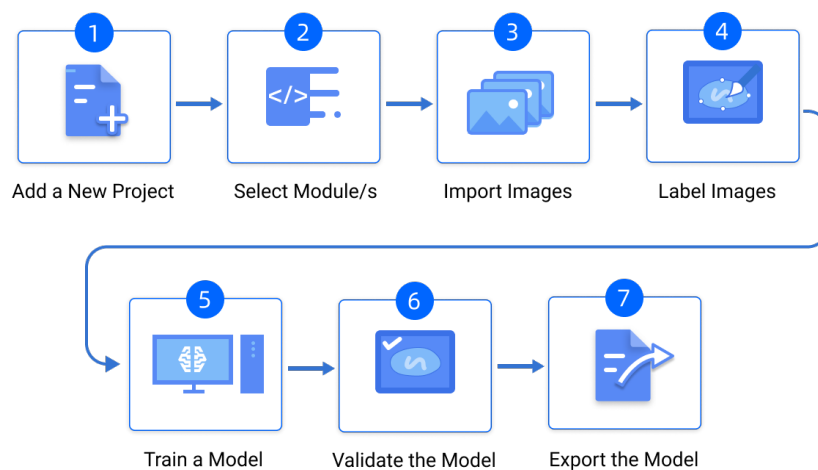
TRAIN THE FIRST MODEL

This section shows how to train and export an example model that can be used for defect segmentation. The data used for training is from an image dataset of connectors.

Preparation

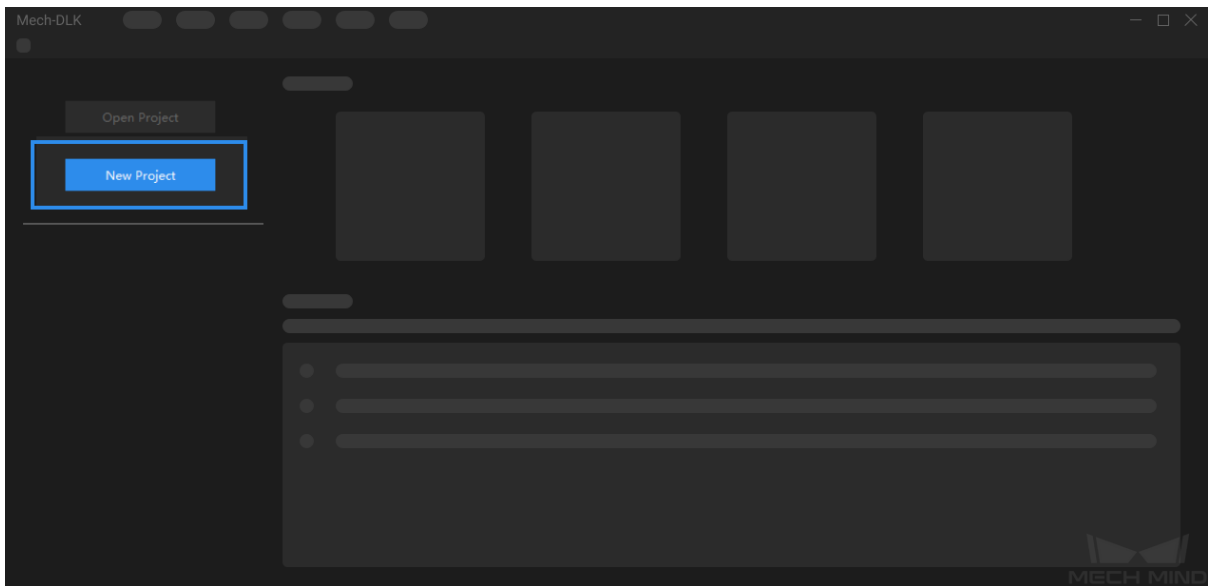
- Please make sure that you have *installed the Mech-Mind software environment and Mech-DLK* successfully.
- Click [here](#) to download the image dataset and decompress the file.

Training Process




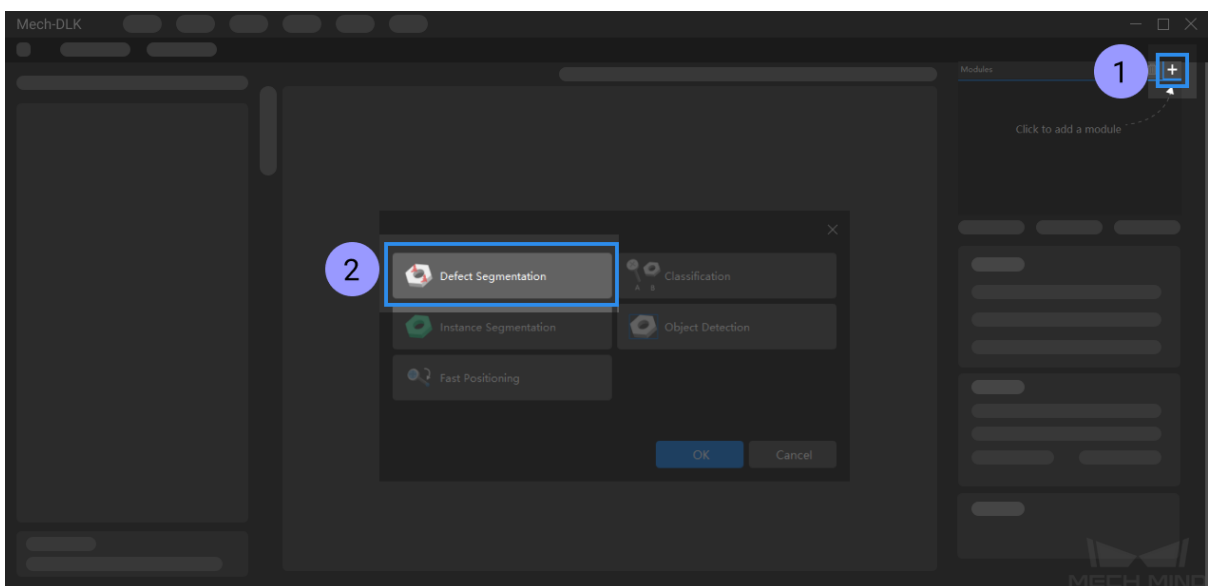
1. Create a New Project

Click on *New Project* in the interface, name the project, and select a directory to save the project.



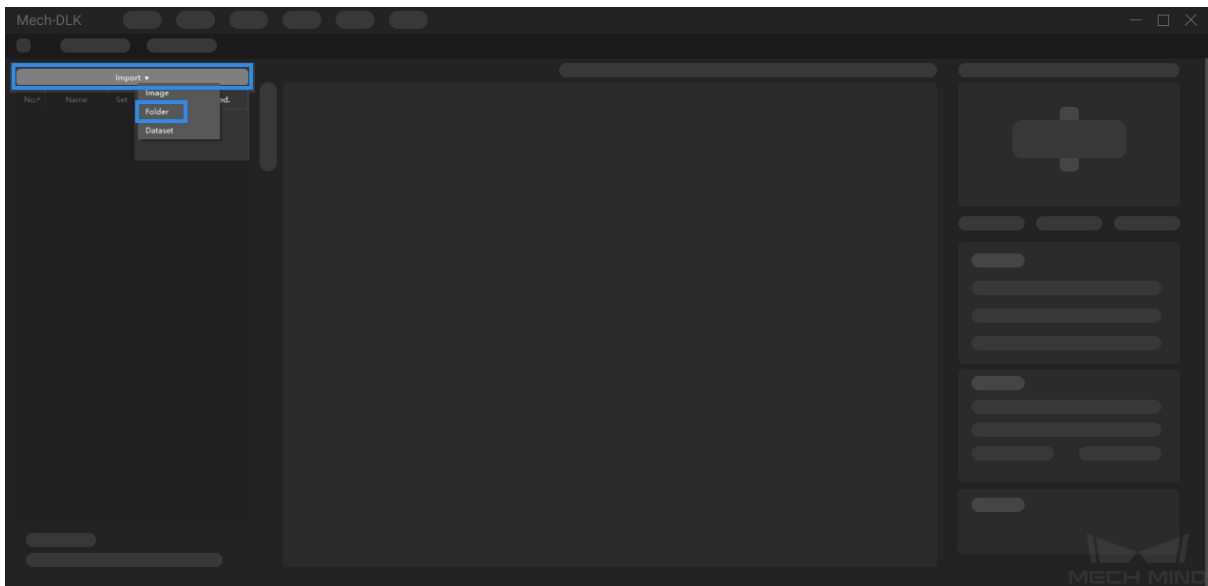
2. Add the Defect Segmentation Module

Click on  in the upper right corner of the **Modules** panel to add a module. Select *Defect Segmentation* and then click on *OK*.



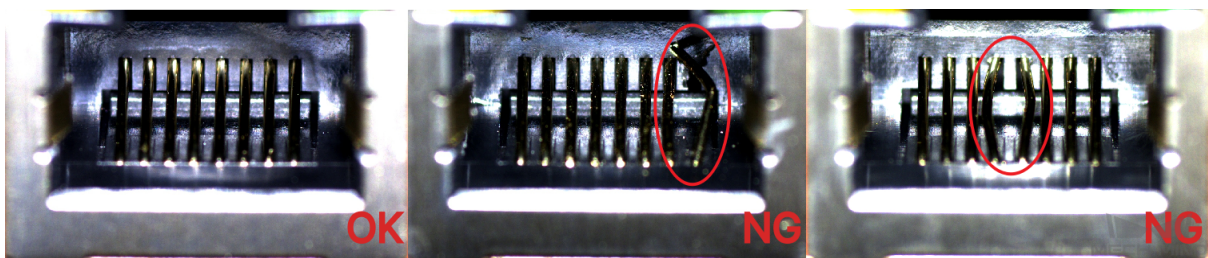
3. Import Data



Click on the *Import* button in the upper left corner, select **Folder** and import the image dataset you have downloaded.

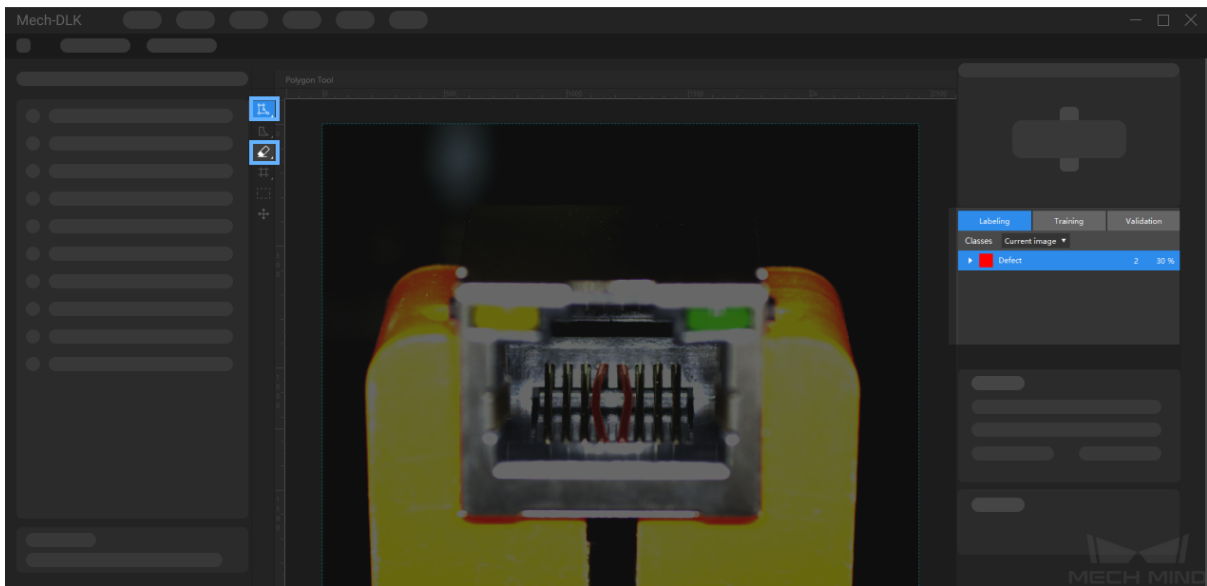


4. Labeling

In this example, you will need to label the OK images and NG images in each dataset. OK means that the connectors meet quality requirements and NG means that there are defects such as deformations and fractures on the connectors.

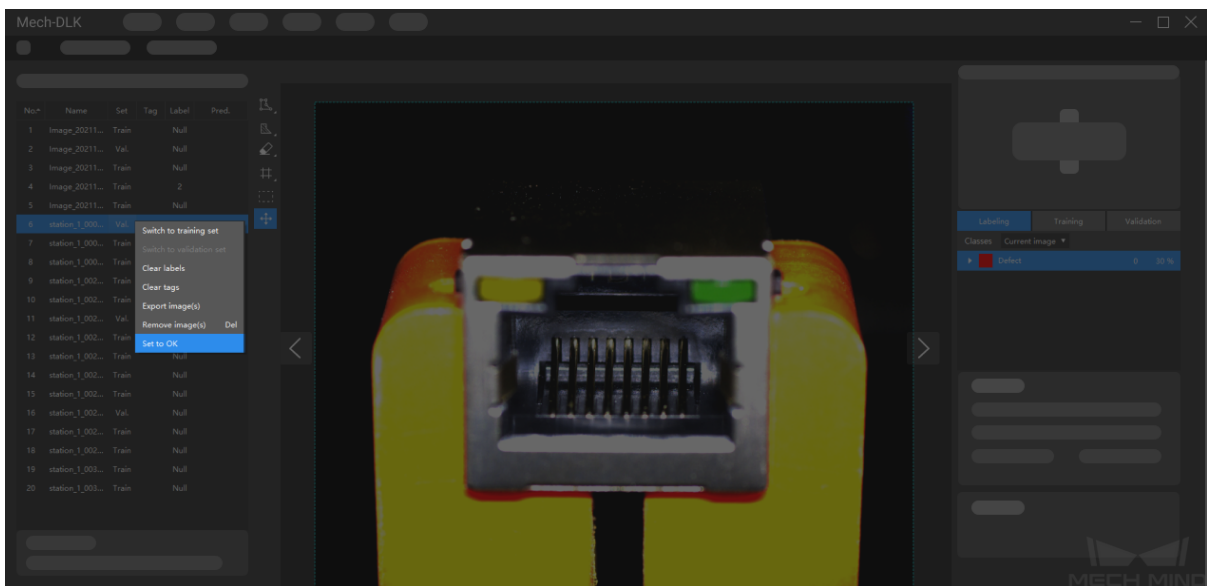


For NG images, click on  and then hold the left mouse button to select the area with defects. Click on  to use the eraser tool to remove the labeled area.



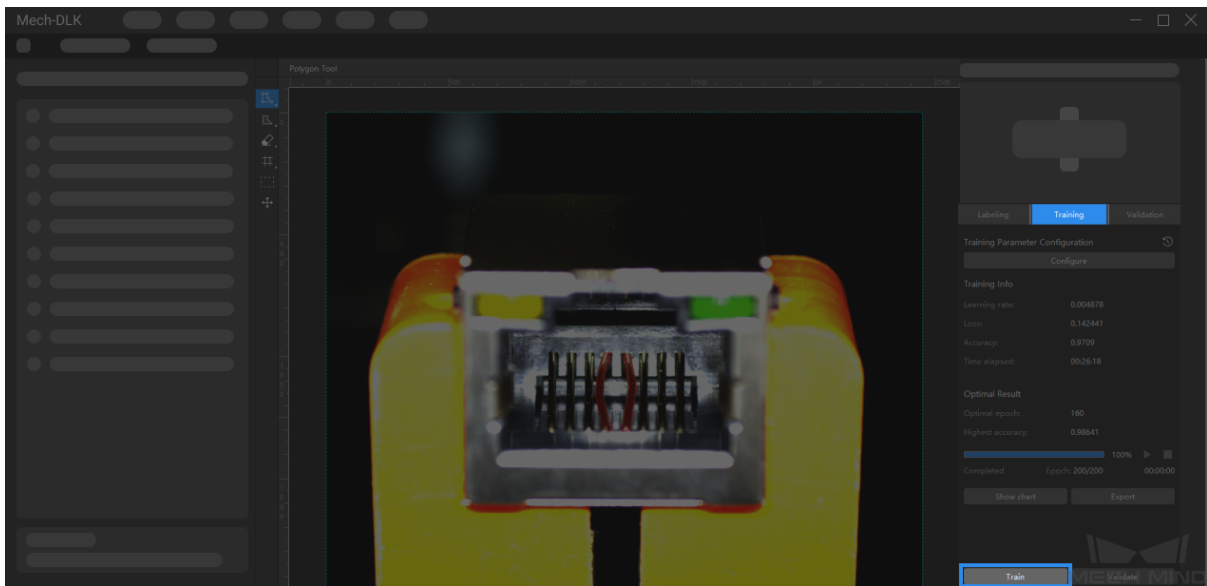
Hint: After selecting the area with defects, right-click to confirm the selection and exit the polygon tool.

For OK images that do not contain any defect, please select the image and then right-click and select *Set to OK* in the context menu. Please make sure that there is at least one OK image in each dataset.



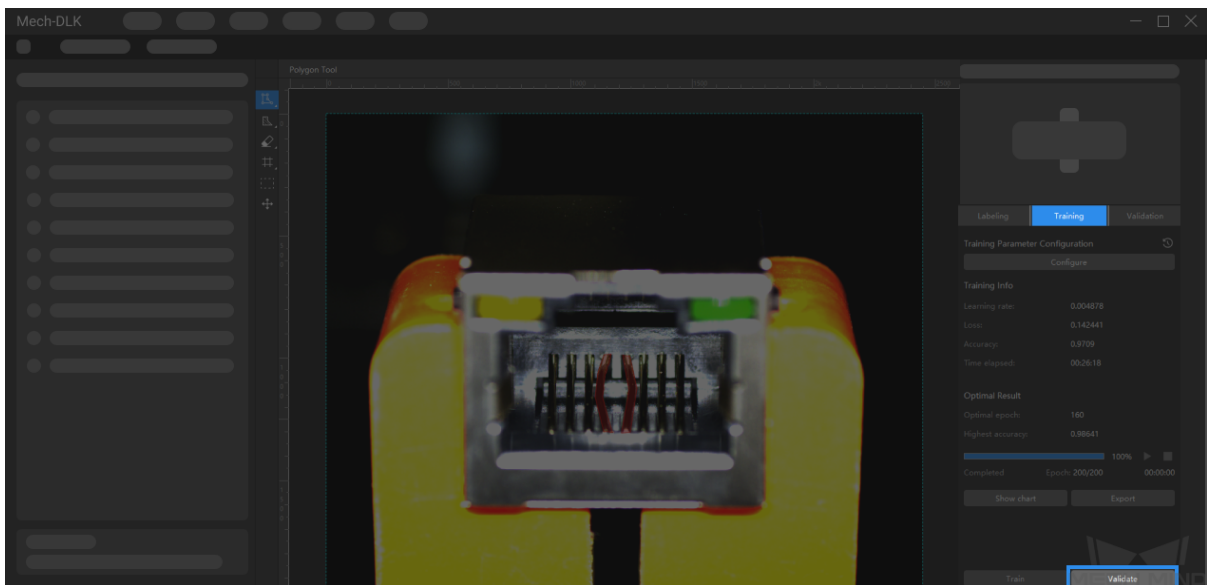
5. Train the Model

Select *Training* and then click on *Train* in the lower right corner of the interface to start training the model.



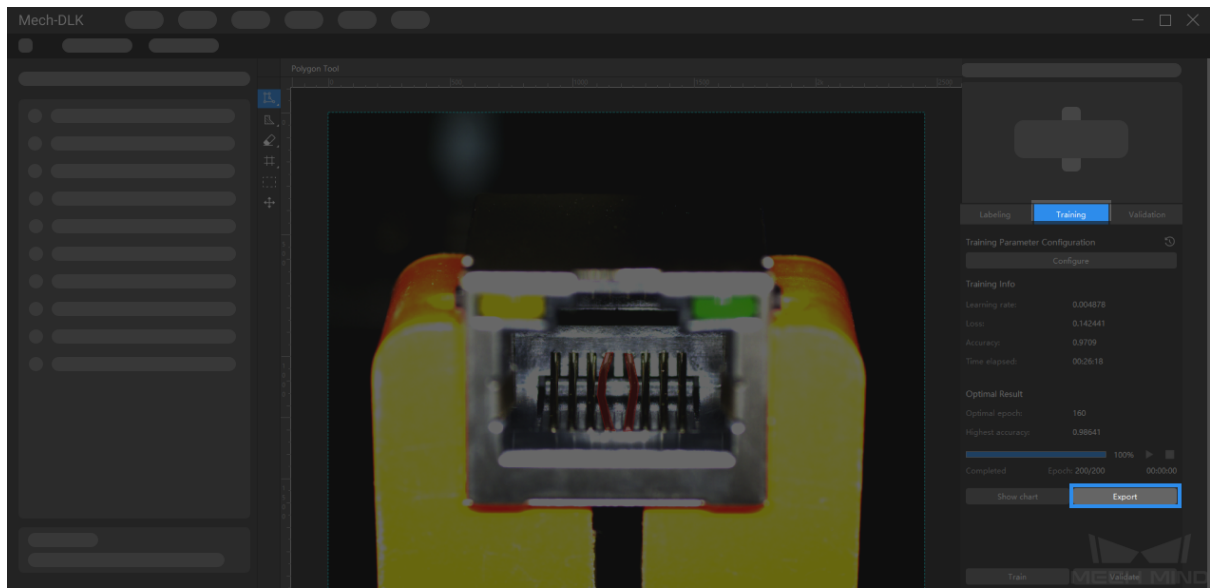
6. Validate the Model

After the training is completed, click on *Validate* to validate the model and check the results.



7. Export the Model

Click on *Export* and select a directory to save the exported model (with file extension dlkpack). Then you can deploy the model according to actual needs.



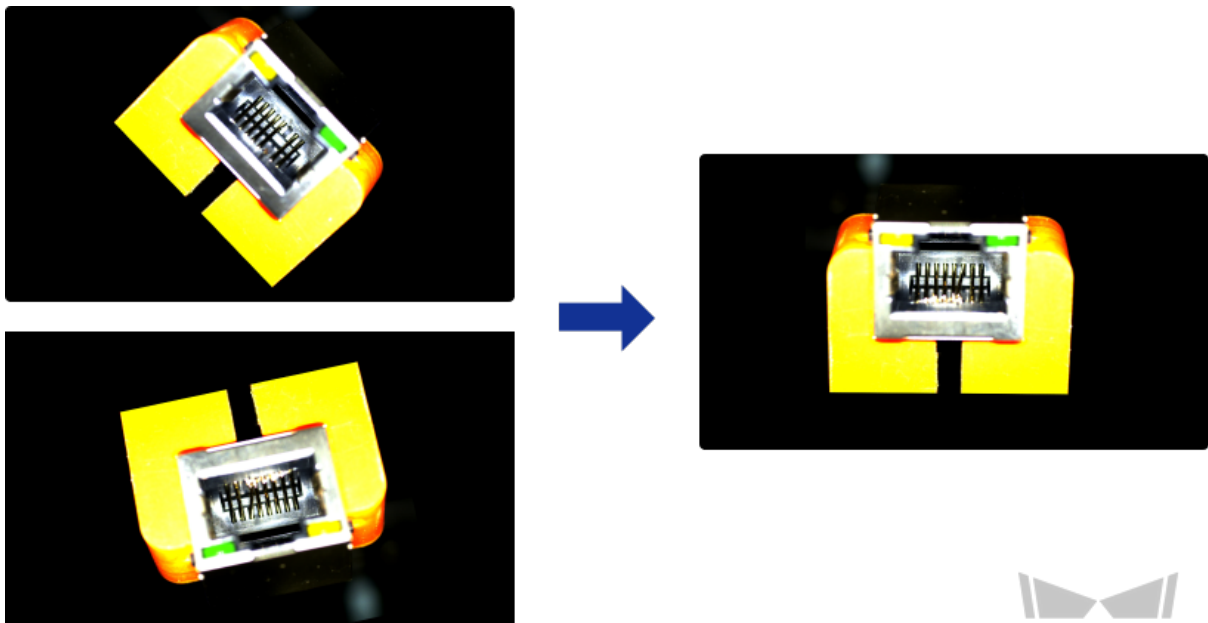
FAST POSITIONING

4.1 Introduction

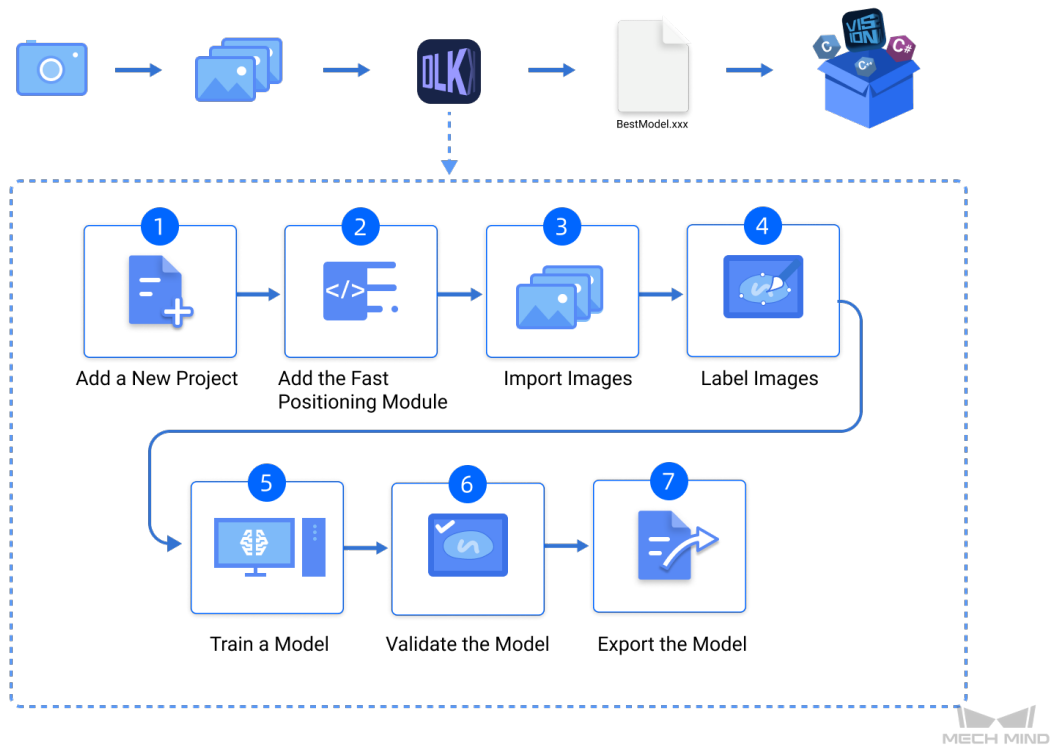
This module is used to detect objects in images and rotate the images to a specified orientation.

4.1.1 Applicable Scenarios

Electronics manufacturing: Detect the electronic components with different placement orientations in the images and adjust the orientations to a specified one.




4.1.2 General Workflow

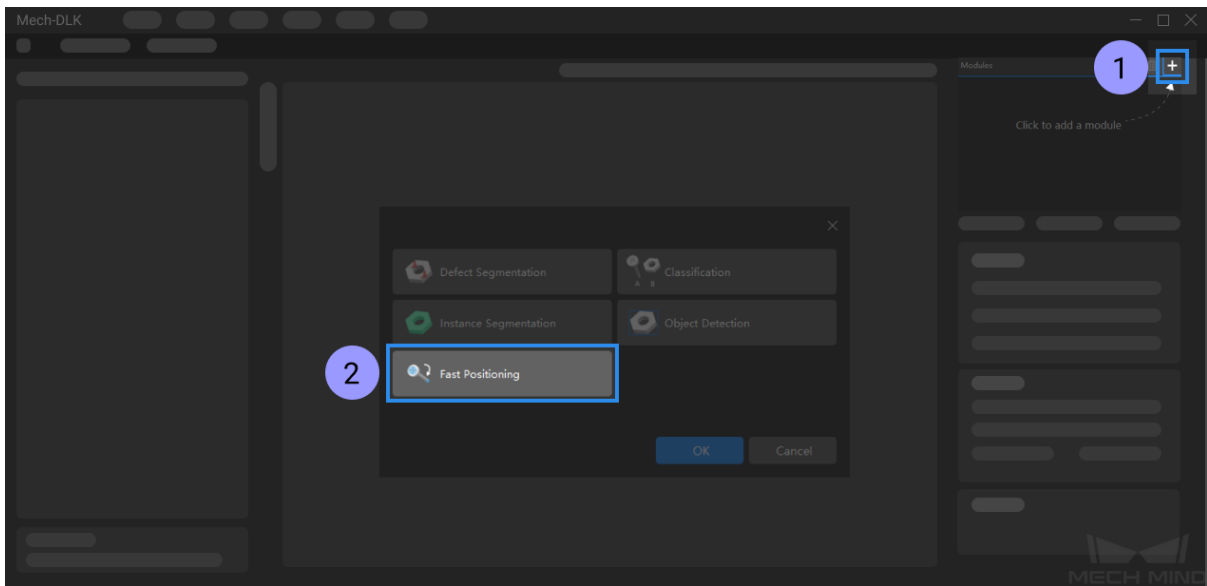


4.2 Start Using the “Fast Positioning” Module

Please click [here](#) to download an image dataset of connectors. In this section, we will use a **Fast Positioning** module and train a model to rotate the connectors in the images to a specified orientation.

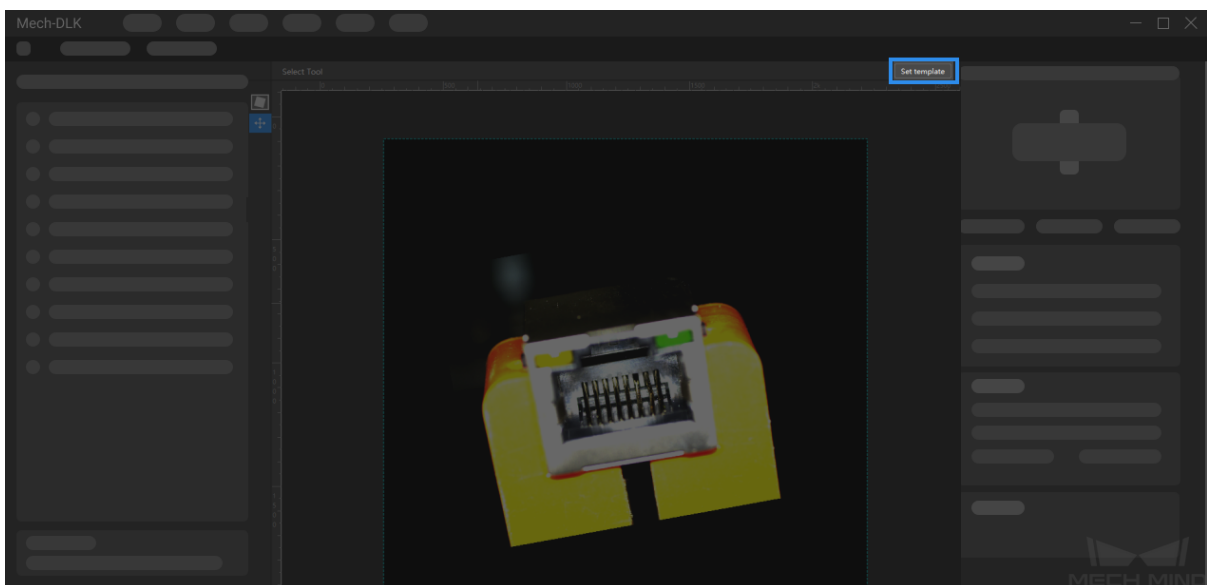
1. Create a new project and add the fast positioning module

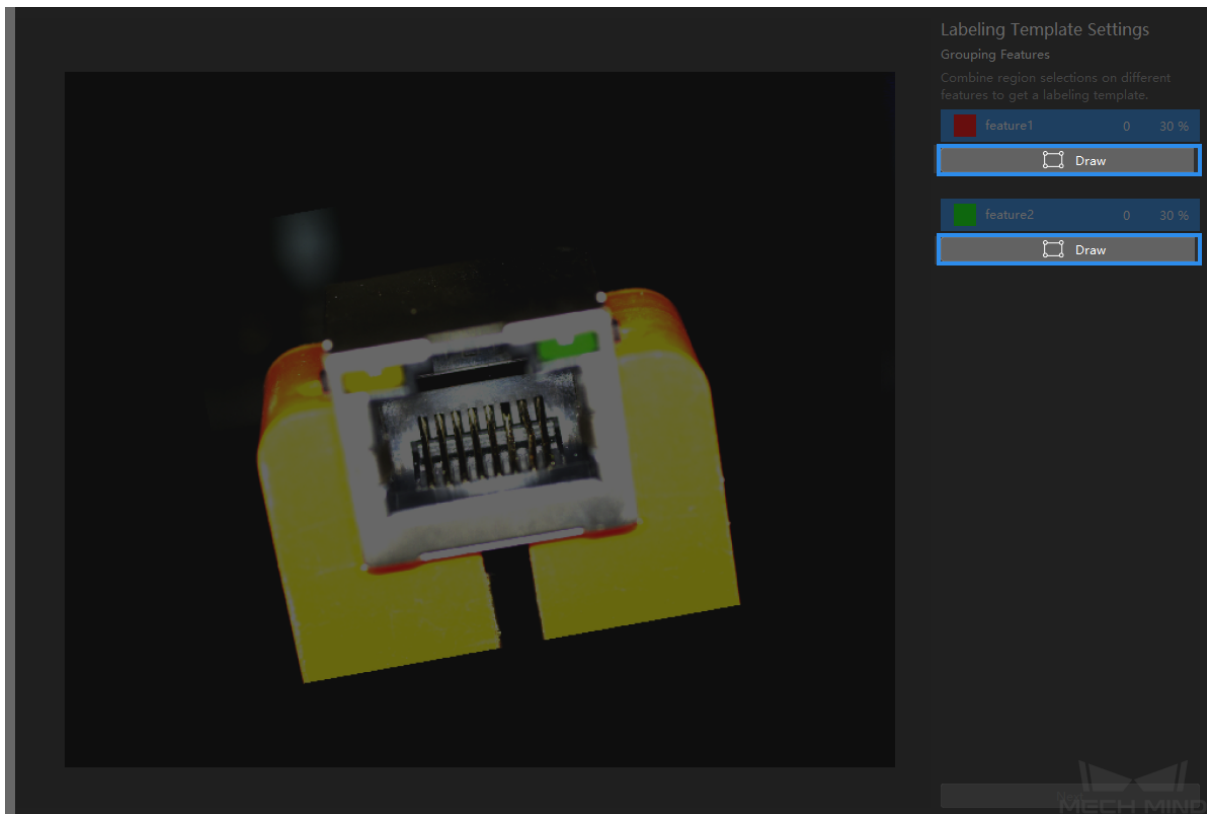
Click on *New Project* in the interface, name the project, and select a directory to save the project. Click on  in the upper right corner of the **Modules** panel and add the **Fast Positioning** module.





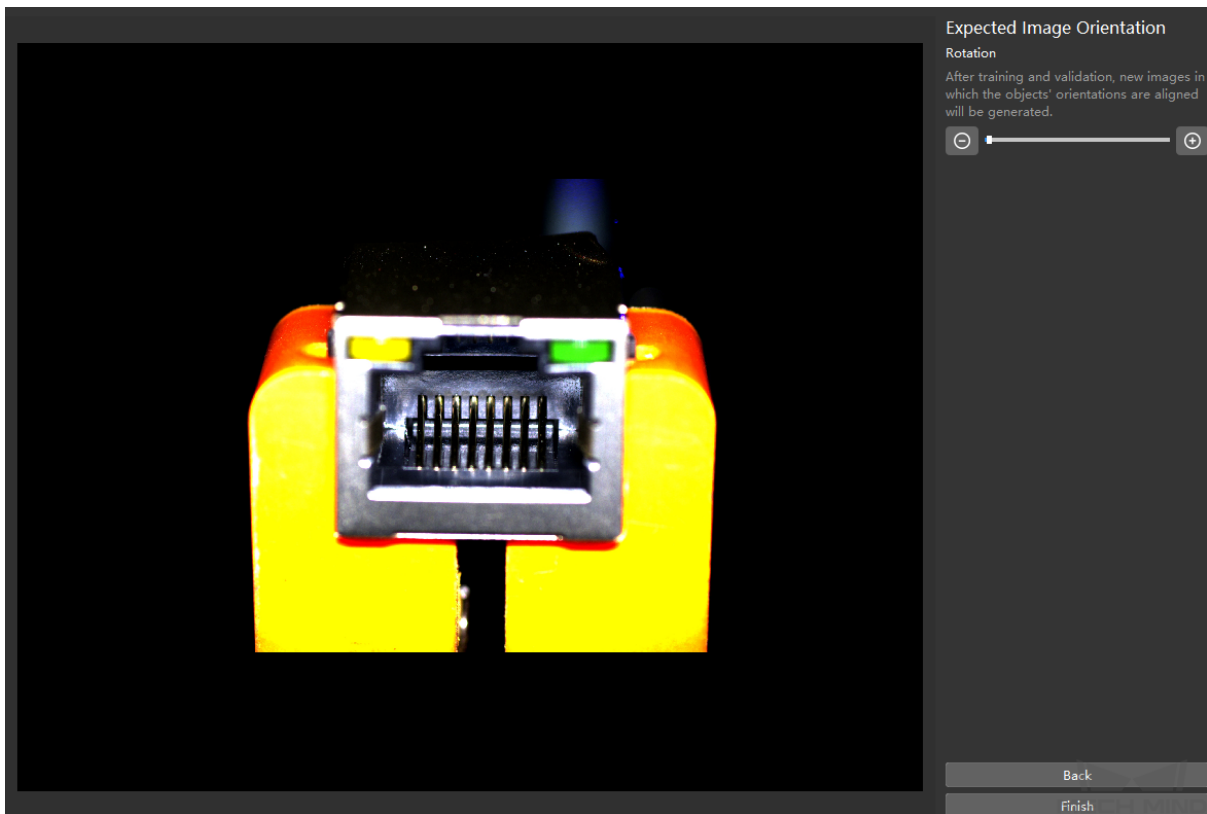
2. Set a template

Click on *Set Template* and select two areas that contain features. Click on *Draw* under **feature1** to select the first feature and then click on *Draw* under **feature2** to select the second feature. Then click on *Next* to adjust the expected image orientation.




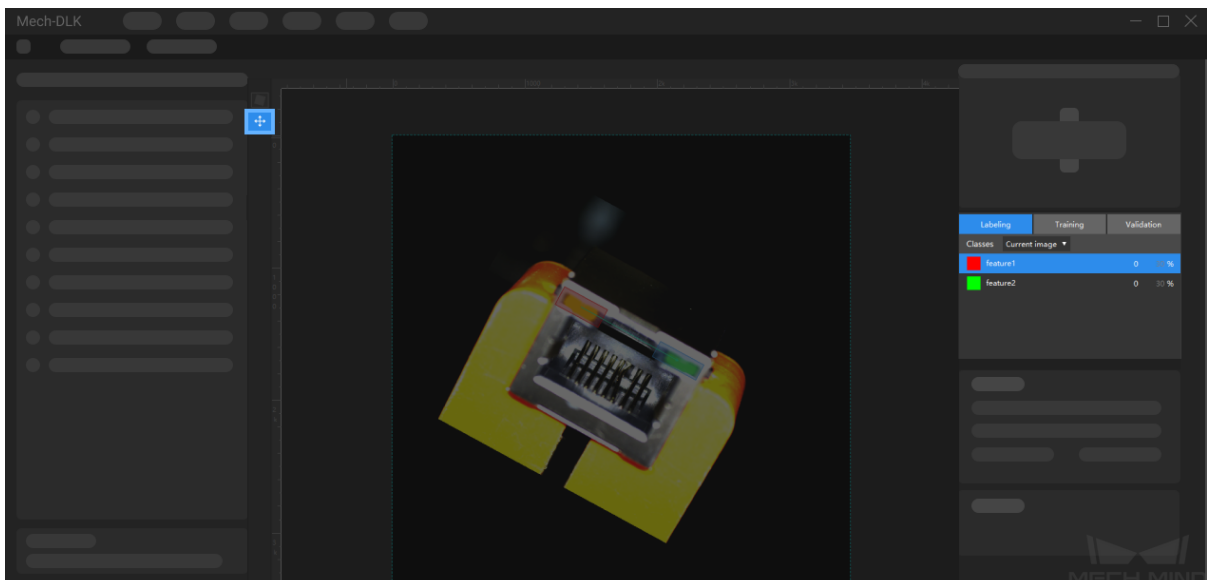


Drag the slider or click on  and  to adjust the image to an expected orientation, and click on *Finish* to confirm settings.



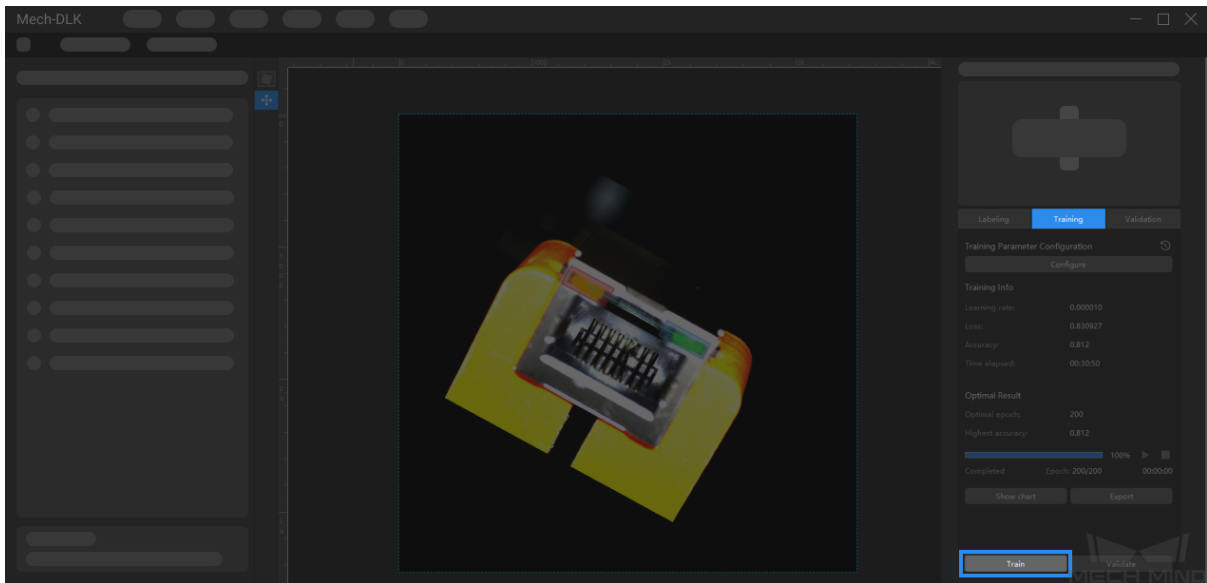
3. Modify the labeling

Click on  and then go to *Set template*, the labeled image will appear in the window. You can drag the feature frames to adjust the feature regions and re-adjust the expected orientation in the same way that you set it the first time.



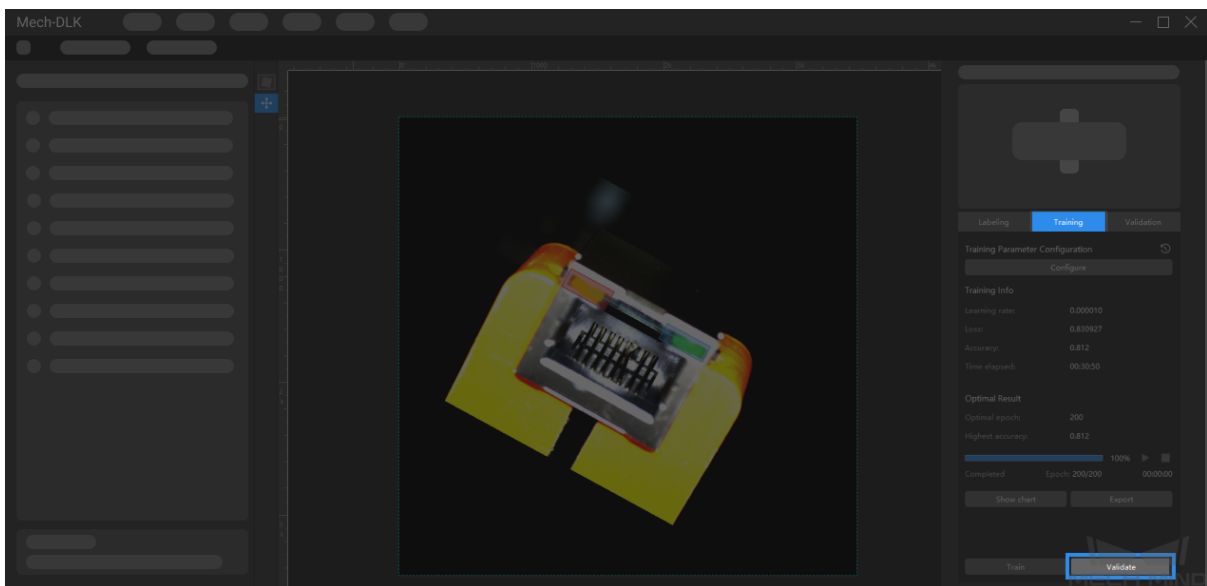
4. Train the model

Keep the default training parameter settings and click on *Train* to start training the model.



5. Validate the model

After the training is completed, click on *Validate* to validate the model and check the results.



6. Export the model

Click on *Export* and select a directory to save the exported model. You can deploy the model according to actual needs.



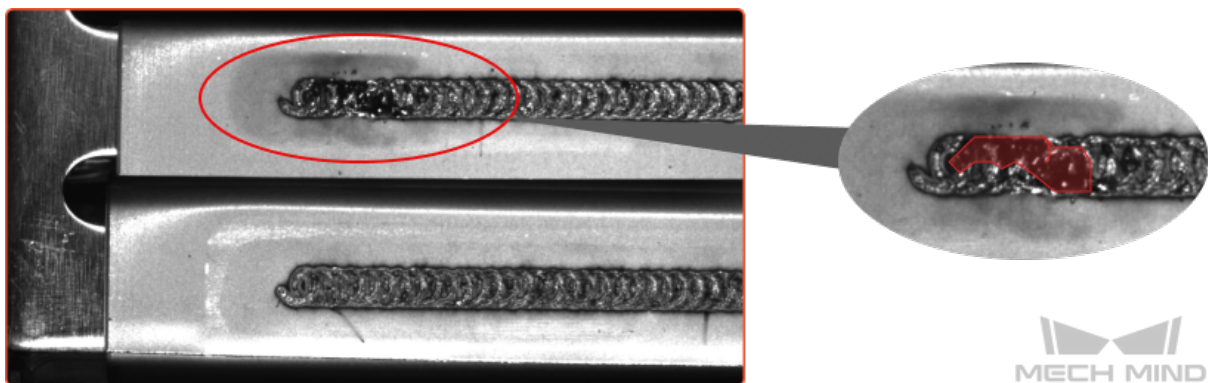
DEFECT SEGMENTATION

5.1 Introduction

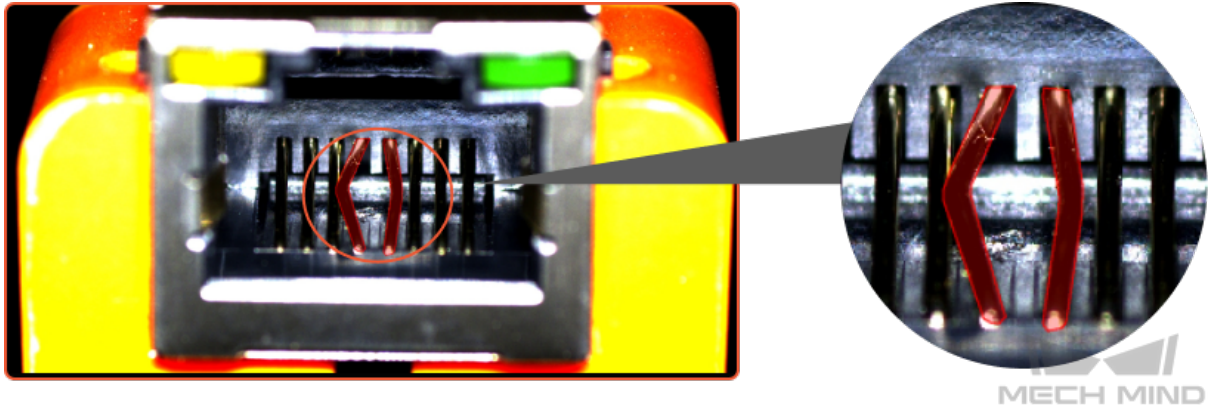
This module is used to detect and segment the defect areas in the image.

5.1.1 Applicable Scenarios

Renewable energy: Suitable for detecting various types of defects. This module is capable of working under complicated situations, such as when the defects are tiny, the background is complex, or the positions of the workpieces are not fixed, etc. For example, this module can be used for weld seam inspection or appearance inspection of lithium batteries.



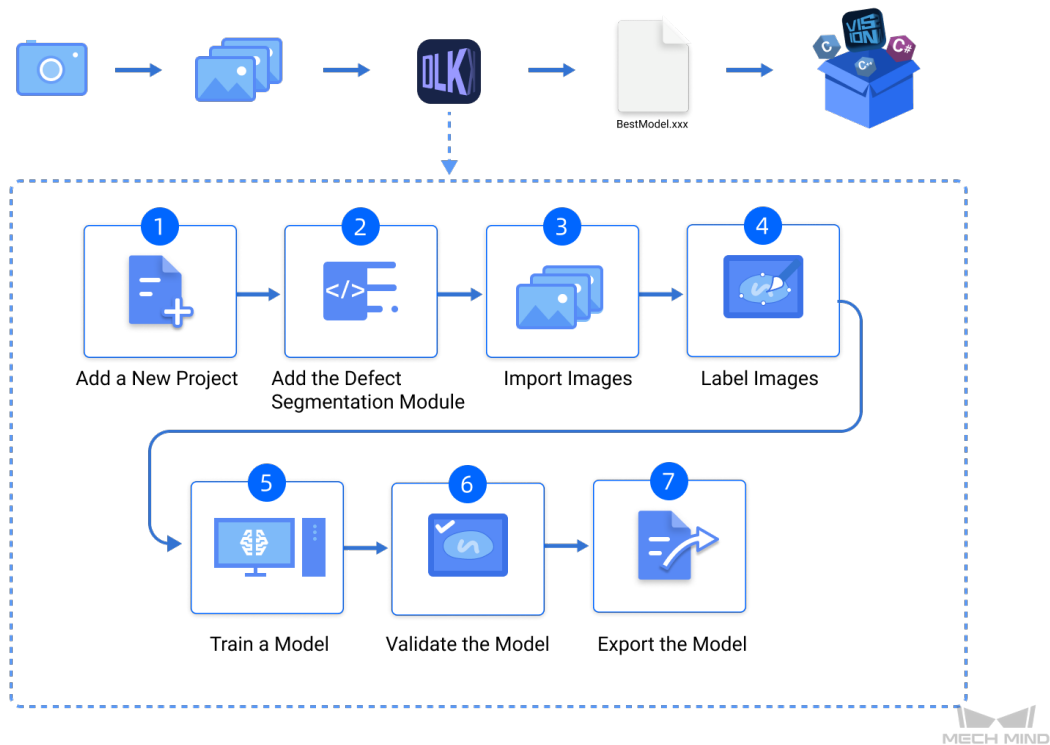
Electronics manufacturing: Suitable for detecting the surface defects, such as stains, bubbles, scratches, etc., of functional modules and electronic components.



PCB manufacturing, printing, daily necessities manufacturing, and other industries: Suitable for detecting surface defects, such as scratches or foreign object debris, of PCB, connectors, printing products, daily necessities, and other objects.



5.1.2 General Workflow



5.1.3 Tips


The performance of the trained model depends on the followings.

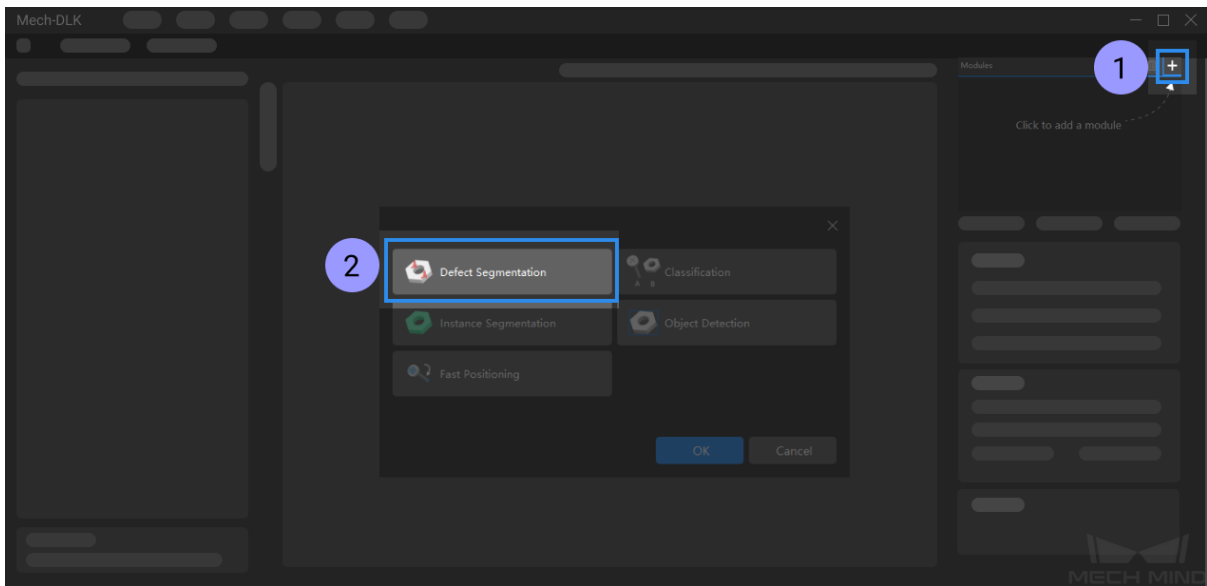
1. *The quality of labeling.*
2. *Selecting a proper ROI.*
3. *Selecting a suitable dataset.*

5.2 Start Using the “Defect Segmentation” Module

Please click [here](#) to download an image dataset of connectors. In this section, we will use a **Defect Segmentation** module and train a model to detect the defects such as deformations and fractures on the connectors.

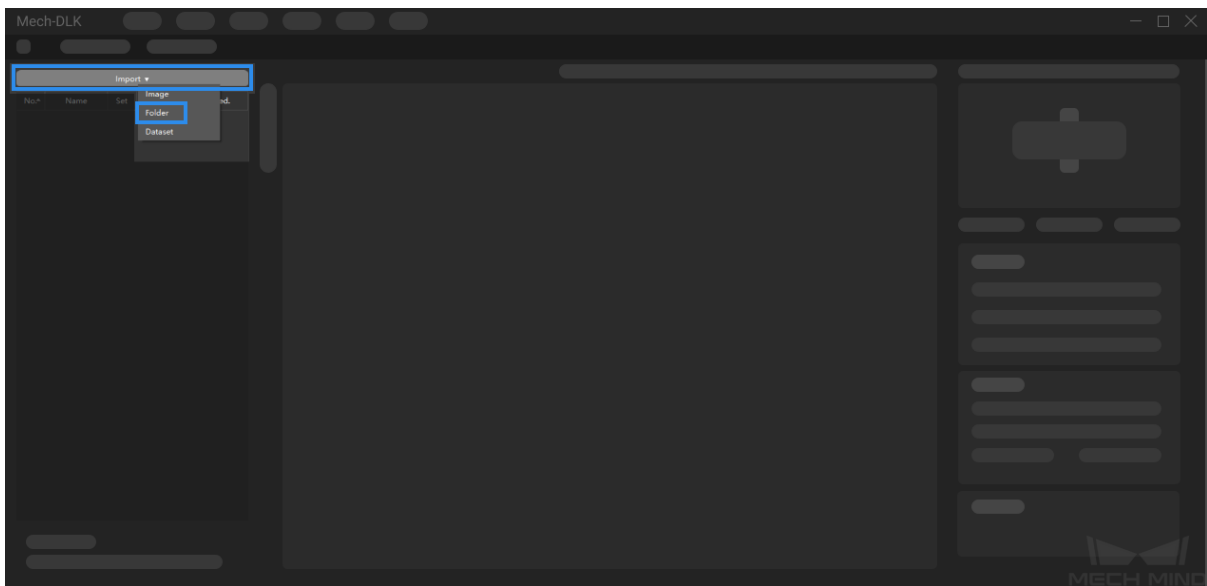
1. Create a new project and add the defect segmentation module

Click on *New Project* in the interface, name the project, and select a directory to save the project. Click on  in the upper right corner of the **Modules** panel and add the **Defect Segmentation** module.




2. Import the image dataset of connectors

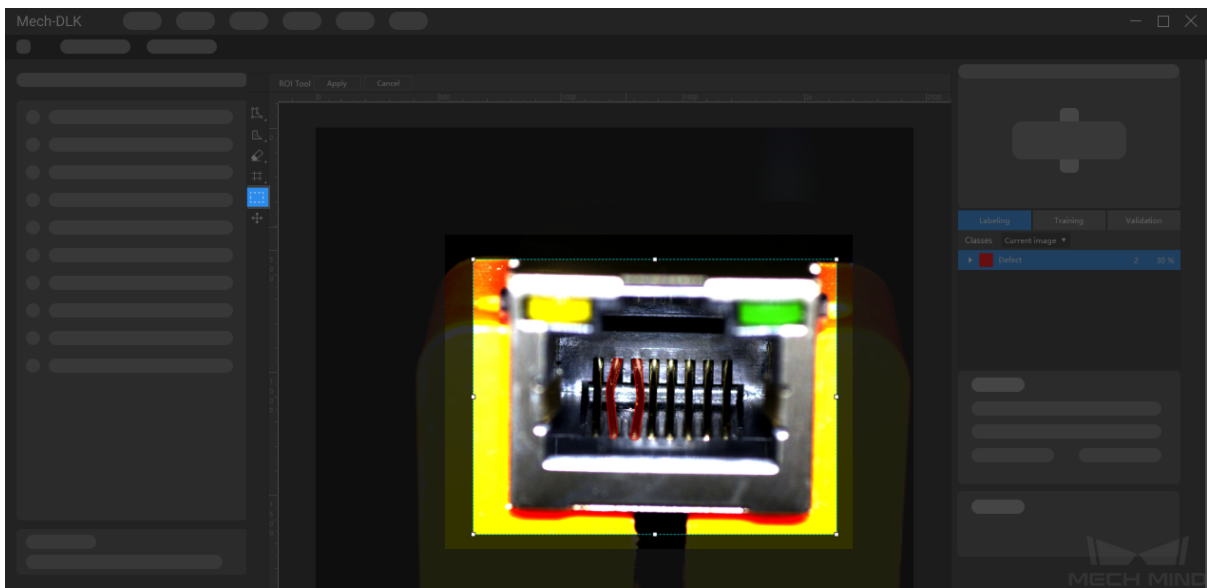
Decompress the downloaded dataset file. Click on the *Import* button in the upper left corner, select **Folder**, and import the image dataset. The connector pins in the images can be deformed, fractured, or intact.



3. Select an ROI

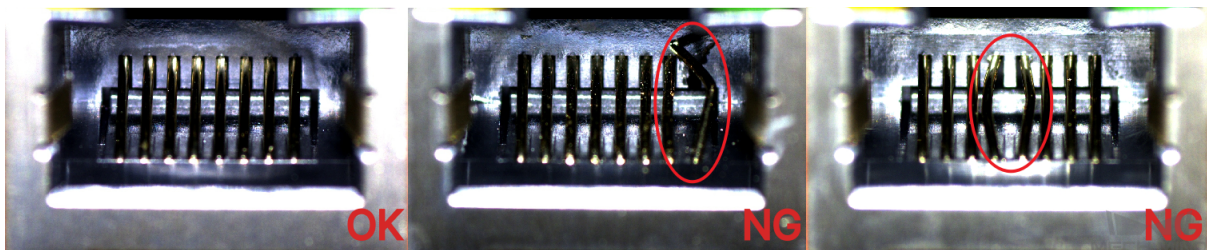
Click on the **ROI Tool** button  and adjust the frame to select the pins in the image as an ROI, and click on *Apply* to save the settings. Setting the ROI can avoid

interferences from the background and reduce processing time.



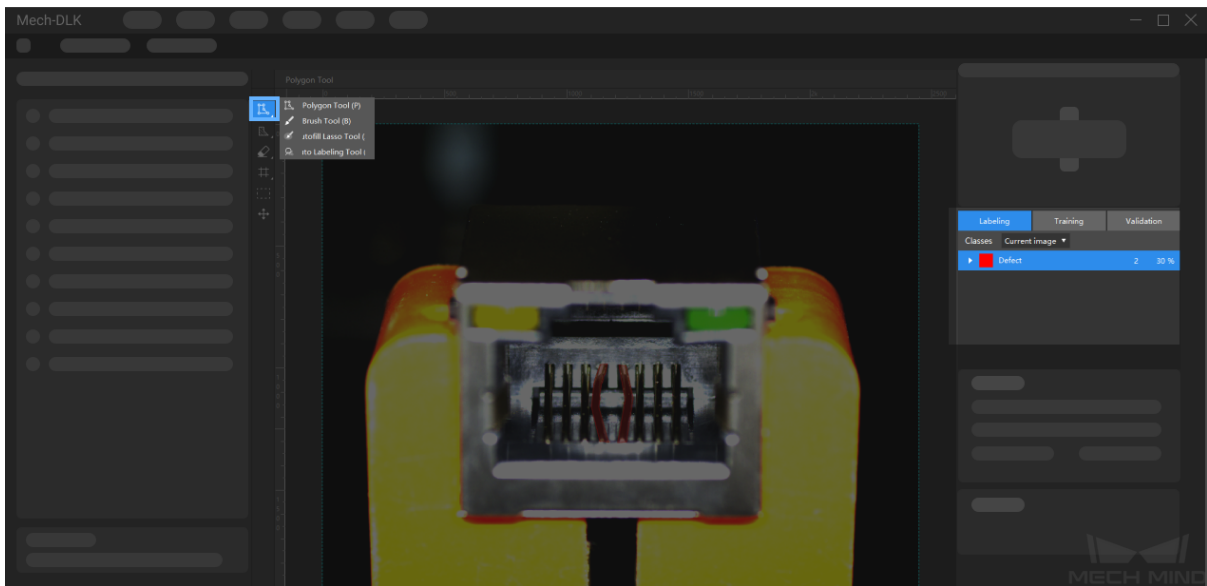
4. Label images

In this section, you will need to label the OK images and NG images in each dataset. OK means that the connectors meet quality requirements and NG means that there are defects such as deformations and fractures on the connectors.



For NG images, Right-click on the **Polygon Tool** and select a proper tool to select the defect area. Please select the defect as precisely as possible with the tools, and avoid including irrelevant regions.

Hint: The quality of labeling is the most important factor affecting the performance of the model. Please refer to *Ensure Labeling Quality* for detailed instructions on image labeling.

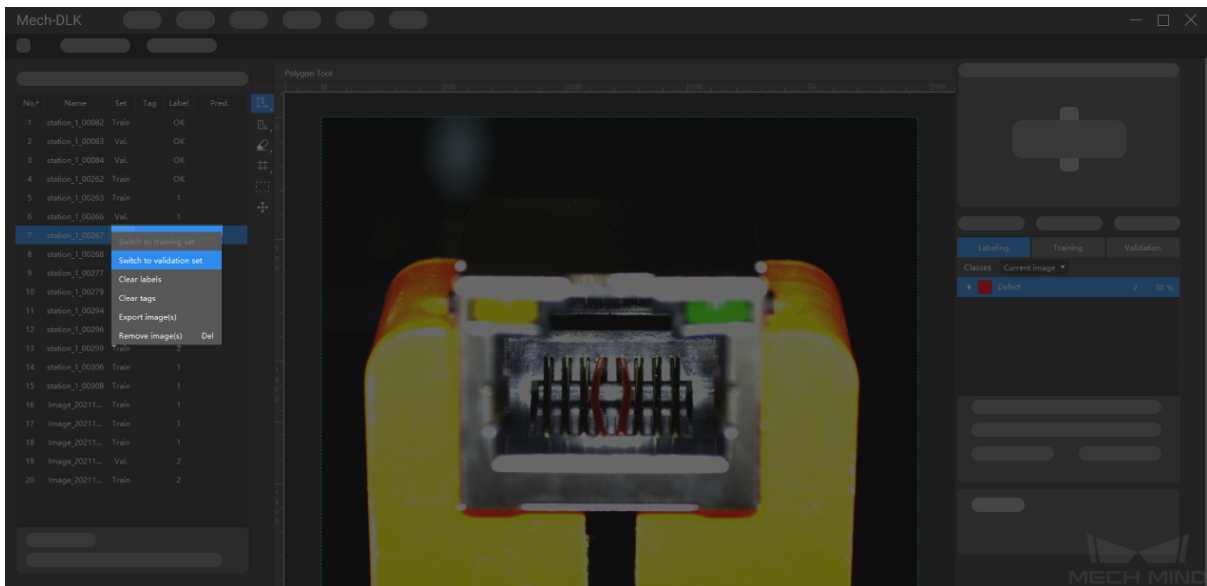


For OK images that do not contain any defect, select the image and then right-click and select *Set to OK* in the context menu.



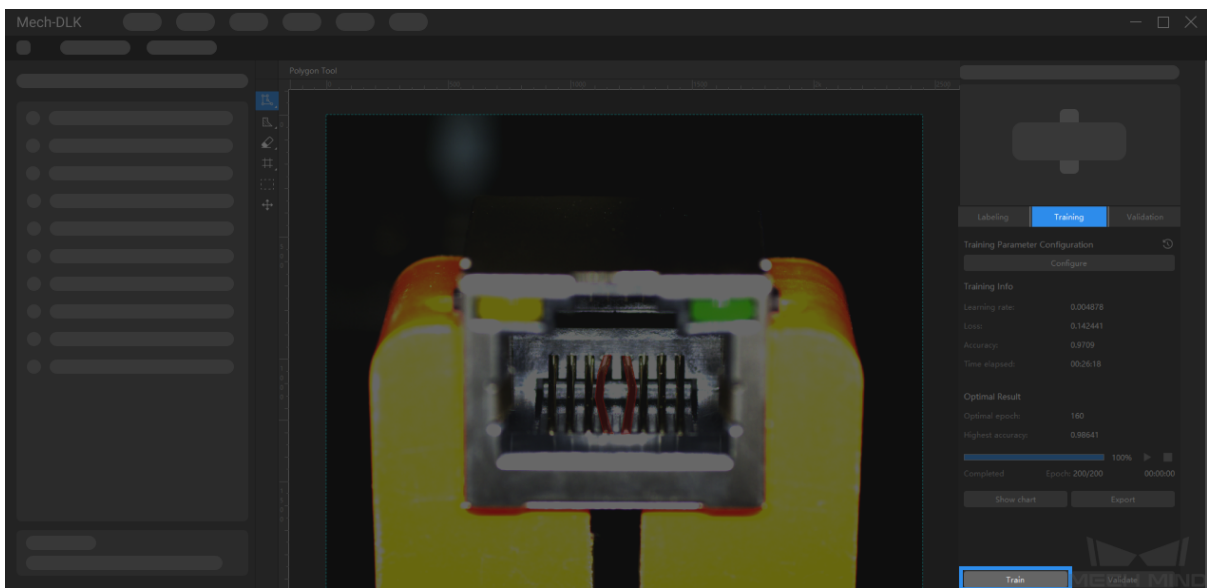
5. Split the dataset into the training set and validation set

Please make sure that both the training set and validation set include **images with all types of defects**, which will guarantee that the algorithm can learn all different types of defects and validate the images with different defects properly. If the default training set and validation set cannot meet this requirement, please right-click the individual image and switch it to the training/validation set manually.



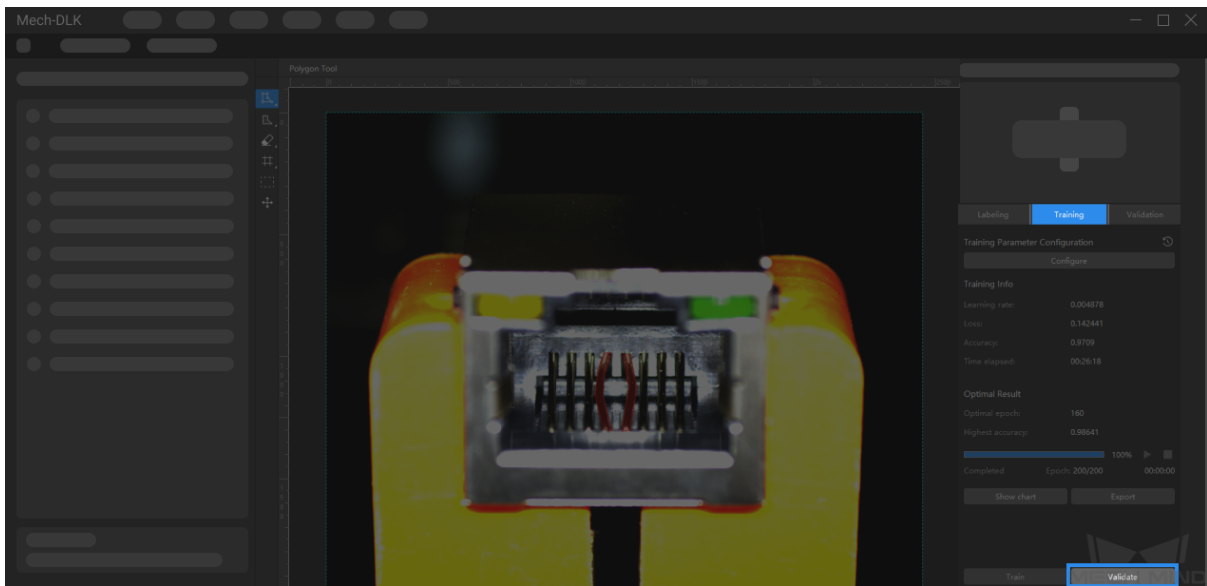
6. Train the model

Keep the default training parameter settings and click on *Train* to start training the model.



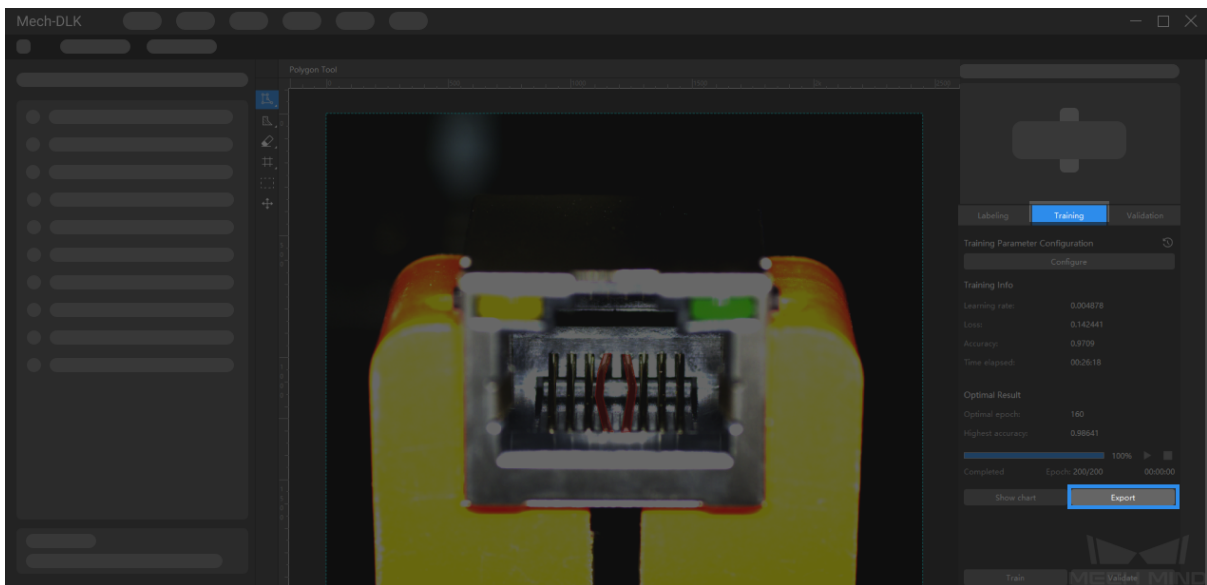
7. Validate the model

After the training is completed, click on *Validate* to validate the model and check the results. You can also use the *Set Defect Judgment Rules* to adjust the criteria for evaluating defects.



8. Export the model

Click on *Export* and select a directory to save the exported model. You can deploy the model according to actual needs.



5.3 Train a High-Quality Model

Industrial quality inspections usually have strict limits on false negative and false positive rates.

Therefore, the quality of the “Defect Segmentation” model is very important.

This section introduces several factors that most affect the model quality and how to train high-quality “Defect Segmentation” models.

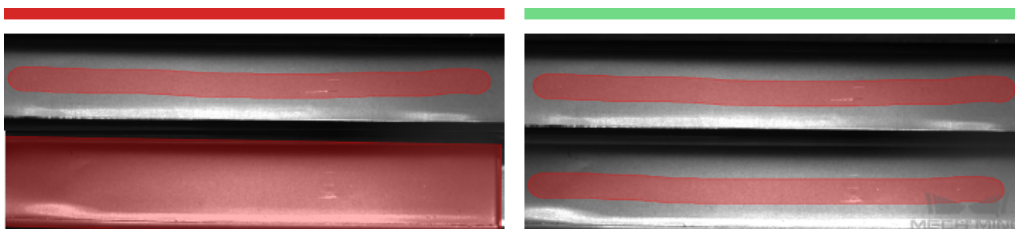
- *Ensure labeling quality*
- *Set the proper region of interest (ROI)*
- *Select the right dataset*

5.3.1 Ensure Labeling Quality

Labeling quality is the most significant factor affecting model performance. In actual projects, low labeling quality accounts for the reasons for more than 90% of poor model performance cases. Therefore, if the model is not performing well, solving labeling quality issues should be prioritized.

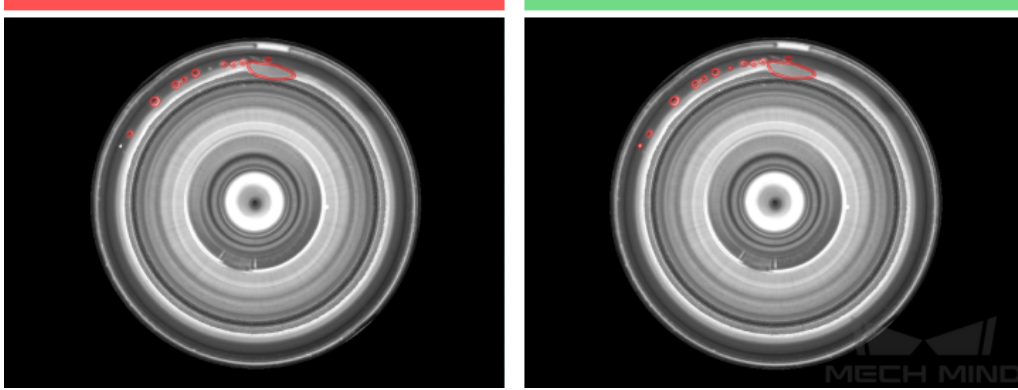
Labeling quality involves consistency, completeness, and accuracy:

1. **Consistency:** Ensure the consistency of defect labeling methods, and avoid using different labeling methods for the same type of defects.
 - **Left image, bad example:** Label defects of the same type in different ways.
 - **Right image, good example:** Label defects of the same type in a consistent way.
2. **Completeness:** Ensure that all regions that should be considered as defect regions according to the user-defined standard are selected, and avoid any missed selections.
 - **Left image, bad example:** Omit the regions that should be labeled.
 - **Right image, good example:** Label all necessary regions.



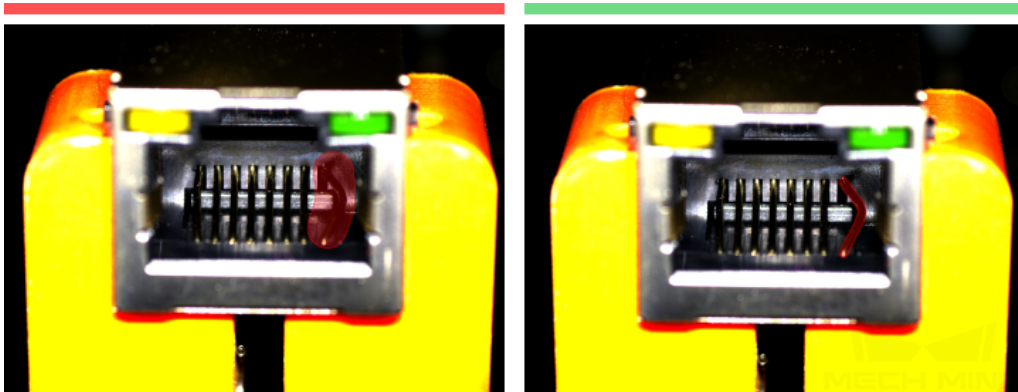
The defects are missing parts in the welding areas. Although both ways of labeling are correct, when labeling, please stick to one way.

3. **Accuracy:** Make the region selection as fine as possible to ensure the selected regions’ contours fit the actual defects’ contours and avoid bluntly covering the defects with coarse large selections.
 - **Left image, bad example:** Cover defects with a coarse large selection.
 - **Right image, good example:** Make the contour of the selection fit the defect’ s contour.



Some bubbles in the example on the left were omitted.

4. **Certainty:** For ambiguous defects, when it is impossible to judge whether the defect judgment criteria are met, the mask polygon tool can be used to cover the defect area.
- **Left image, good example:** Mask out regions containing ambiguous defects.
 - **Right image, mediocre example:** Leave regions in which whether there are defects is hard to determine unprocessed and exposed to the model.



The contours of the selection and the actual defect should be the same.

Attention: When there are multiple defects in the image, if it is impossible to judge whether each defect meets the defect judgment criteria, you can delete the current image to avoid affecting the model training effect.

5.3.2 Set the Proper Region of Interest (ROI)

Setting the ROI can effectively eliminate the interference of the background, and the ROI boundary should be as close to the outer contours of the objects as possible.



Hint: The same ROI setting will be applied to all images, so it is necessary to ensure that objects in all images are located within the ROI, especially in scenarios where the object positions/sizes are not fixed.

5.3.3 Select the Right Dataset

1. Control dataset image quantities

For the first-time model building of the “Classification” module, capturing 20 to 30 images is recommended.

It is not true that the larger the number of images the better. Adding a large number of inadequate images in the early stage is not conducive to the later model improvement, and will make the training time longer.

2. Collect representative data

The datasets should contain NG images covering all the defect types with all defect features, in terms of shape, background, color, size, etc. When the features in OK images do not differ across images, the number of OK images can be relatively small.

3. Balance data proportion

The number of images of different conditions/object classes in the datasets should be proportioned according to the actual project; otherwise, the training effect will be affected.

4. Dataset images should be consistent with those from the application site

The factors that need to be consistent include lighting conditions, object features, background, field of view, etc.

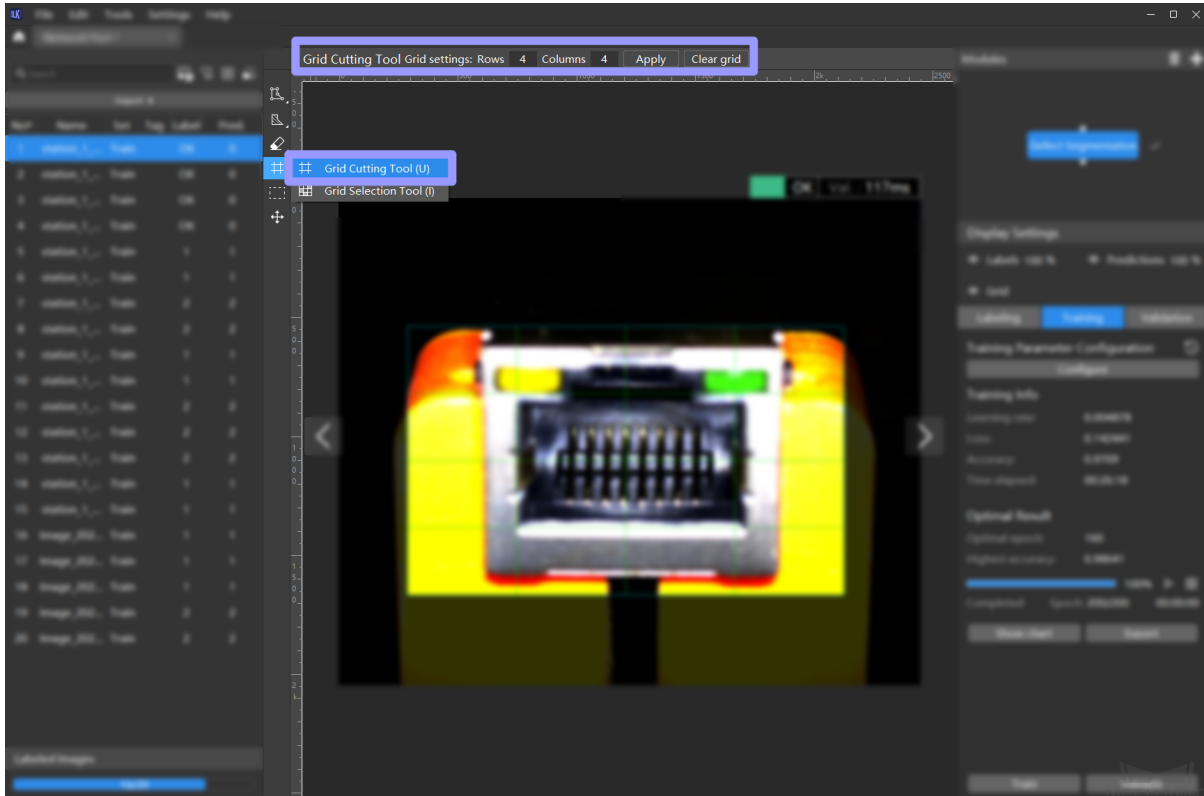
5.4 Solve Difficult Problems

5.4.1 Grid Cutting Tool

In industrial inspection scenarios, if the image size from the camera is large, smaller defects may be inconspicuous. In this case, the grid cutting tool in the “Defect Segmentation” module can be used to cut a larger image into small images of the same size, which makes it easier for detecting smaller defects.

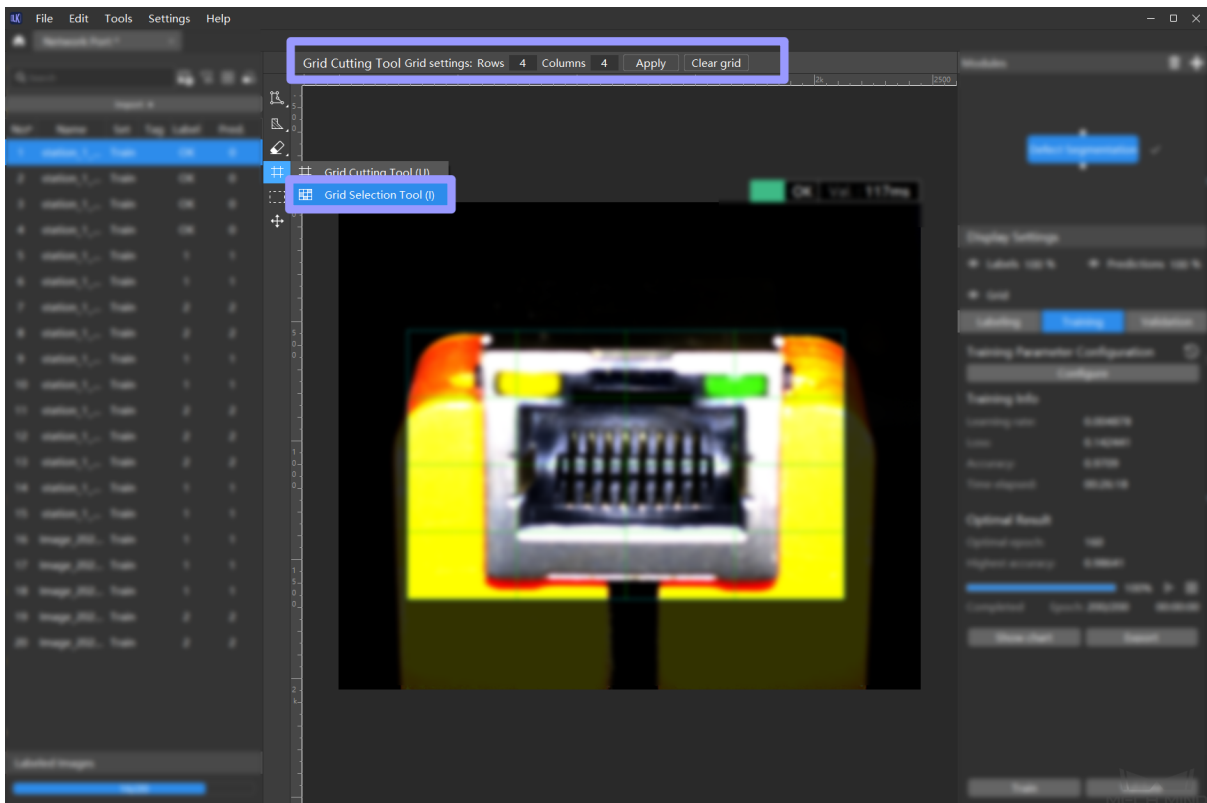
This feature includes the following two tools:

1. Grid cutting tool: Gridize large images, and the number of grid rows and columns can be set by the user.



Attention: The number of grid rows and columns should not be too large. Otherwise, the number of small images produced will be large, which will lead to lower inference speed.

2. Grid selection tool: Select NG and OK images among the produced small images to put into each dataset.

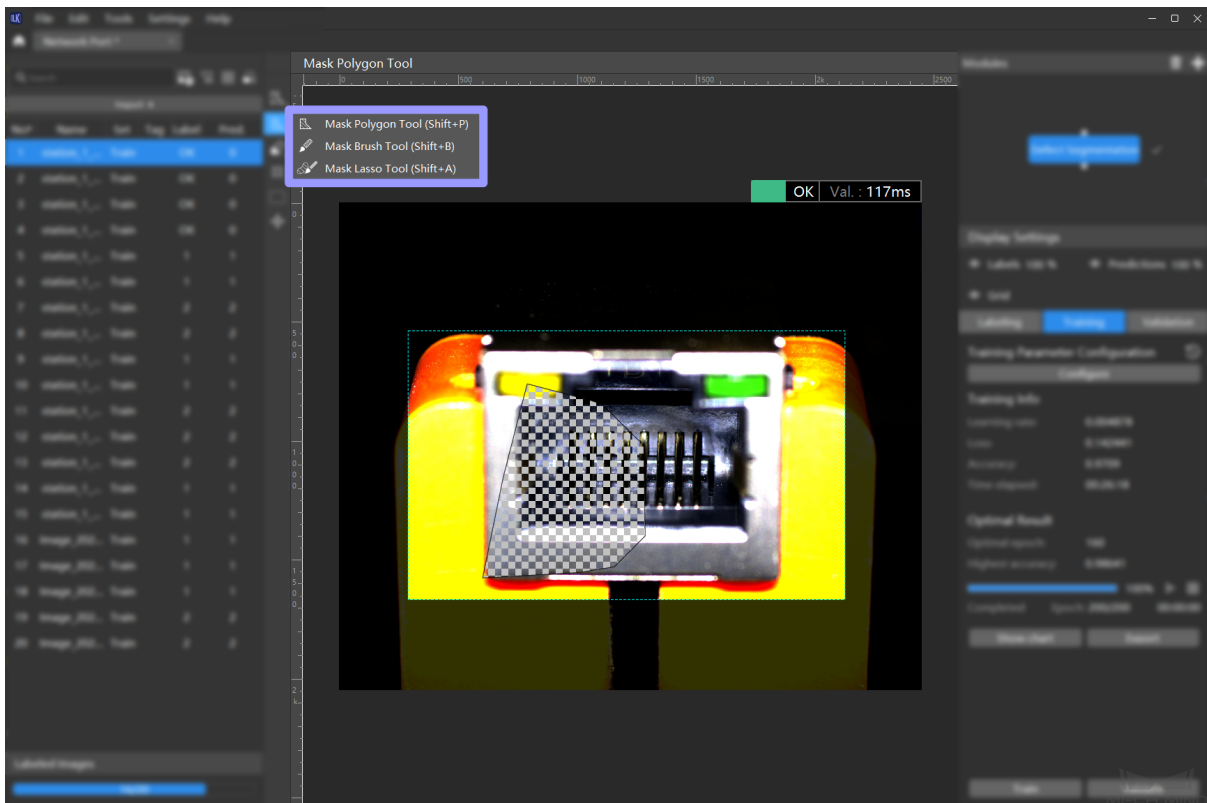


Attention: If the grid selection tool is not used after using the grid cutting tool, then all the small images containing defects in NG images and all the small images in the OK images will be put into the training set by default. This may affect the training effect because there would be too many similar NG and OK images.

5.4.2 Mask Polygon Tool

When labeling object defects, if the background of the object to be inspected or the surface features of the object itself are similar to the defects, the subsequent training and even the quality of the model will be affected.

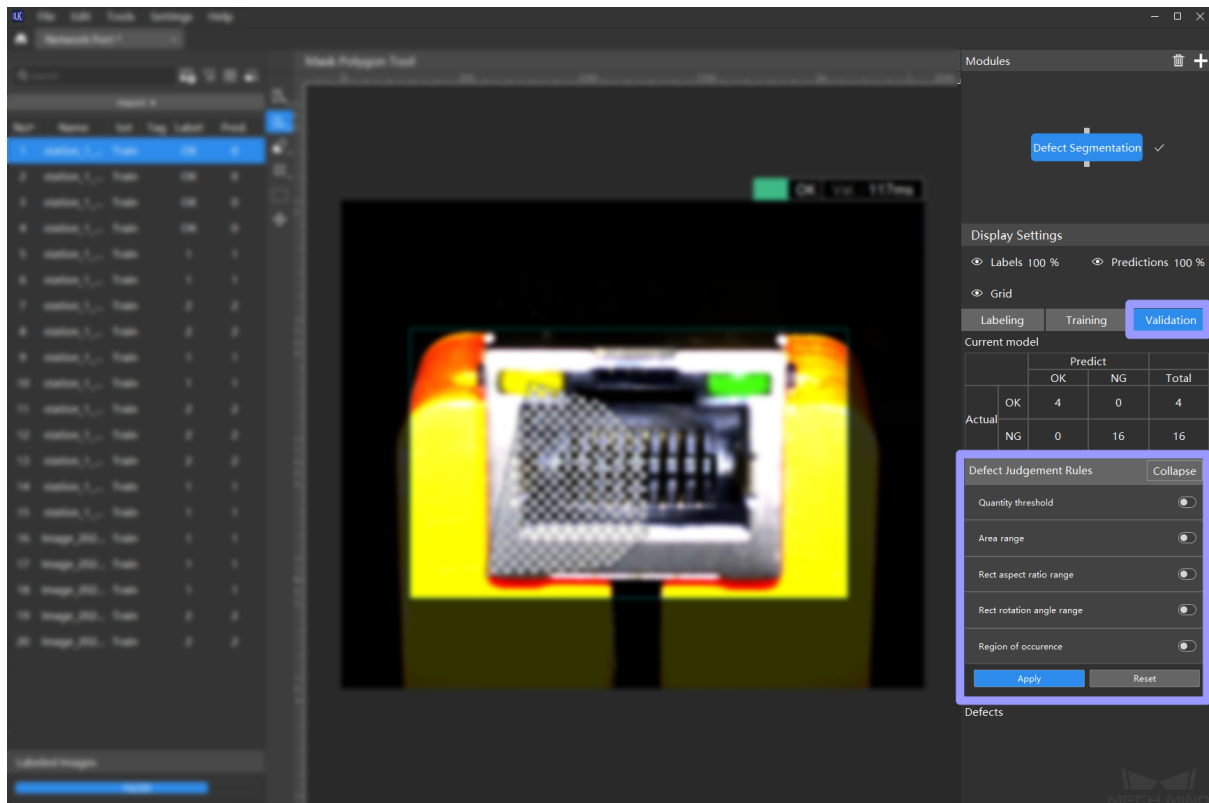
To avoid the above issue, you can use the mask polygon tool to mask out regions that may cause a negative influence on model training.



Hint: The usage of the mask polygon tool is similar to that of the polygon tool for labeling. The masked regions will not be used for learning by the model.

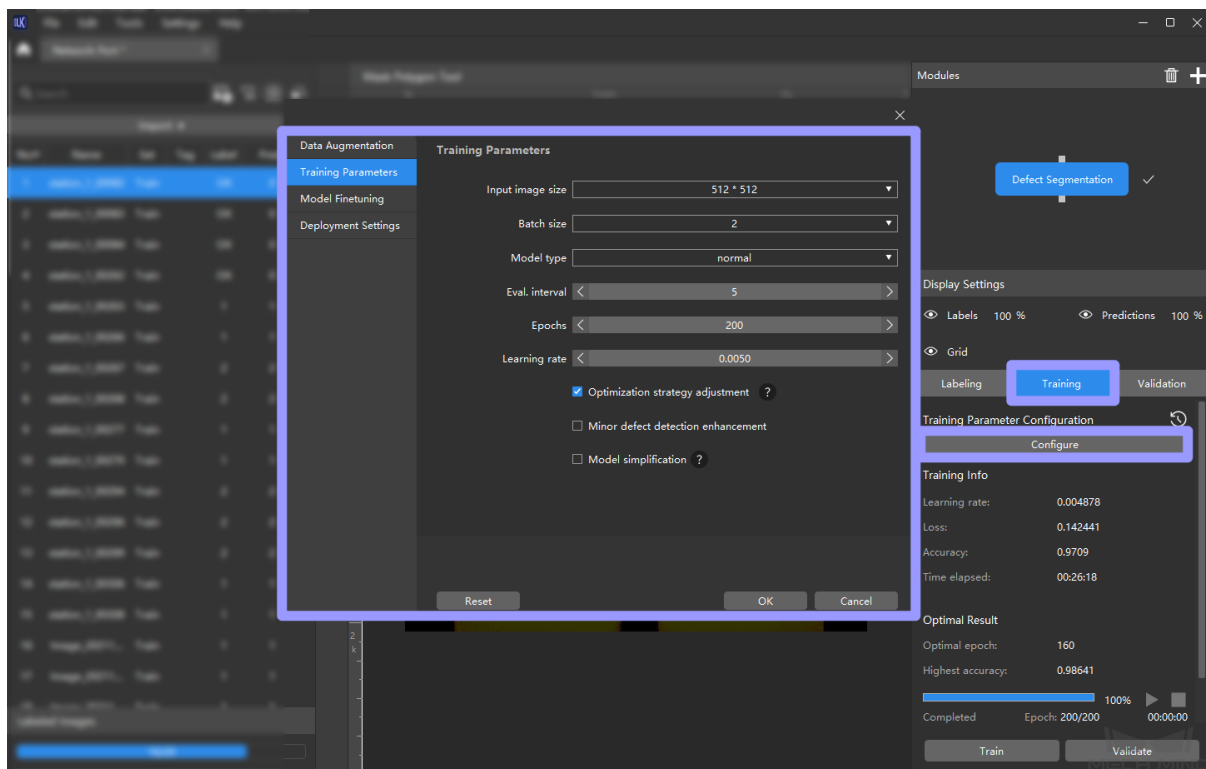
5.4.3 Set Defect Judgment Rules

In defect segmentation scenarios, if you need to refer to the user-defined defect definition criteria to further recognize defects, or you want to re-adjust the defect ratio according to the yield rate, you can filter defects by adjusting the quantity threshold, area range, minimum circumscribed rectangle rotation angle range, and region of occurrence.



5.4.4 Adjust Training Parameters

When there is a need to speed up inference and improve model accuracy during training, the training parameters can be adjusted.



Input image size

Please set the image size according to the project requirements.

Model type

It has two options: “normal” and “enhanced”. In regular cases, keeping the default selection “normal” will suffice. In complex cases, or when there are higher requirements on model accuracy, “enhanced” can be selected, but selecting “enhanced” may lead to a longer training time.

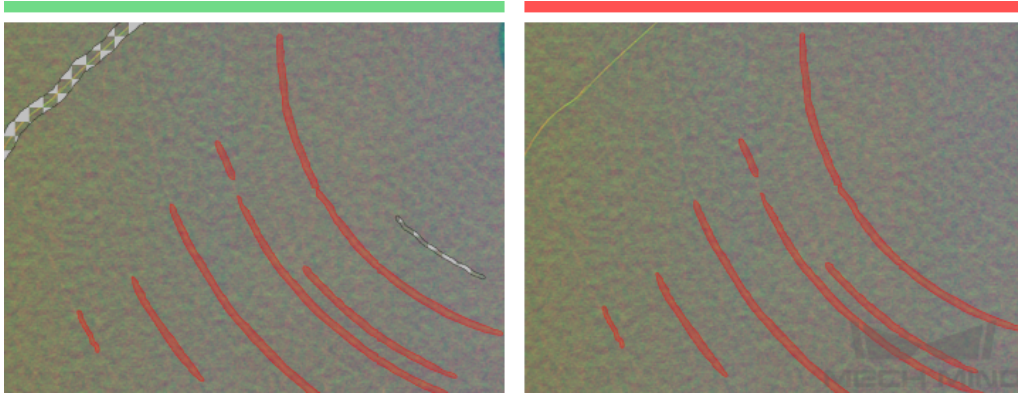
Eval. interval

This parameter sets the number of training epochs in the interval of two evaluations. The larger the evaluation interval, the fewer the performed evaluations and the faster the training.

Epochs

Usually, the default setting suffices. If the image features to be recognized are complex, it is necessary to appropriately increase the number of epochs to improve the convergence of the model, but it will make training slower.

Attention: It is not true that the larger the number of epochs the better. Excessive emphasis on convergence may lead to overfitting.



You can use the mask polygon tool to mask out the regions containing ambiguous defects.

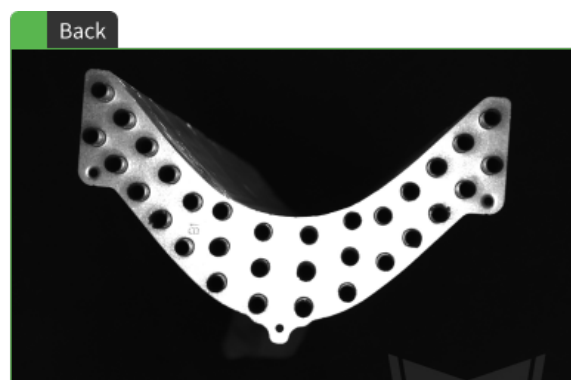
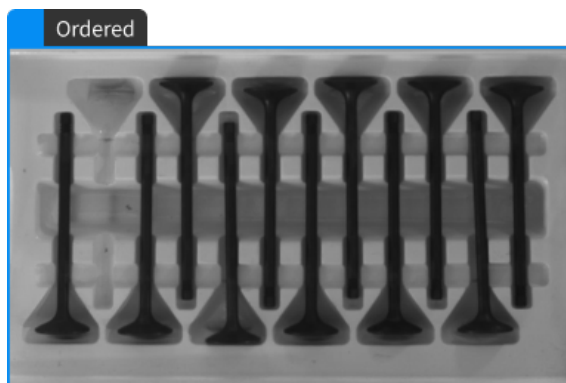
CLASSIFICATION

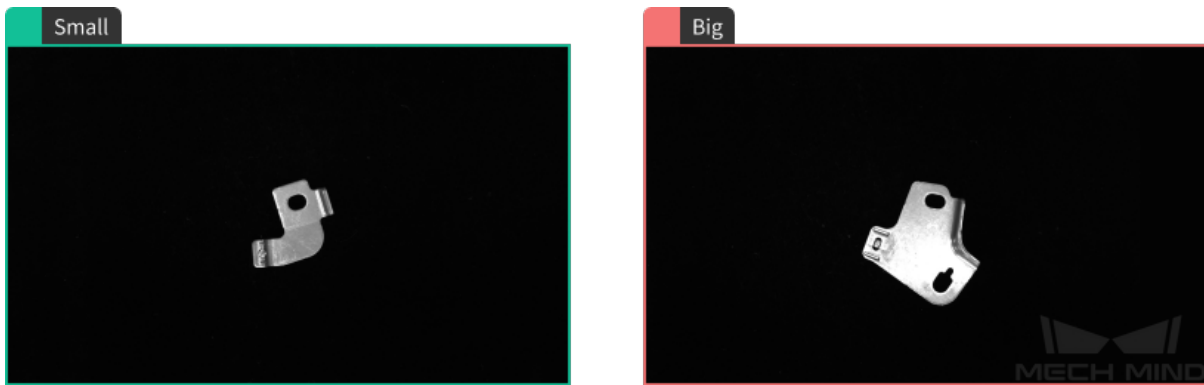
6.1 Introduction

This module is used to classify different images.

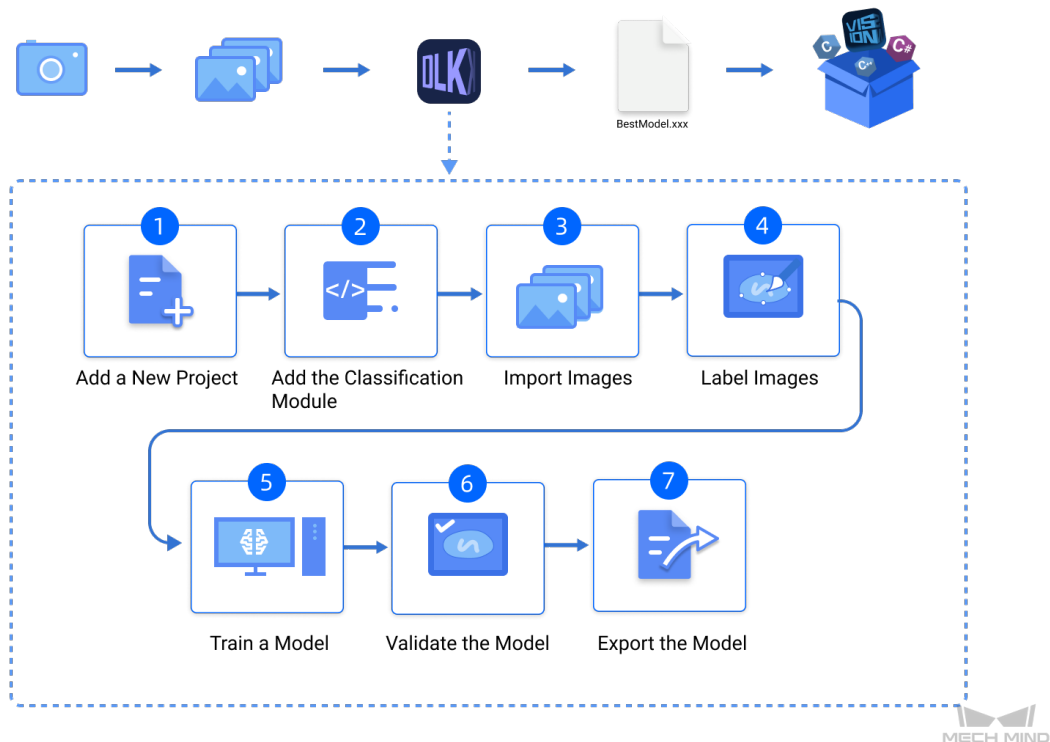
6.1.1 Applicable Scenarios

Machine tending: Suitable for classifying the front and back side, positions, types, or other properties of the workpieces in industries such as steel and machinery.





6.1.2 General Workflow



6.1.3 Tips

The performance of the trained model depends on the followings.

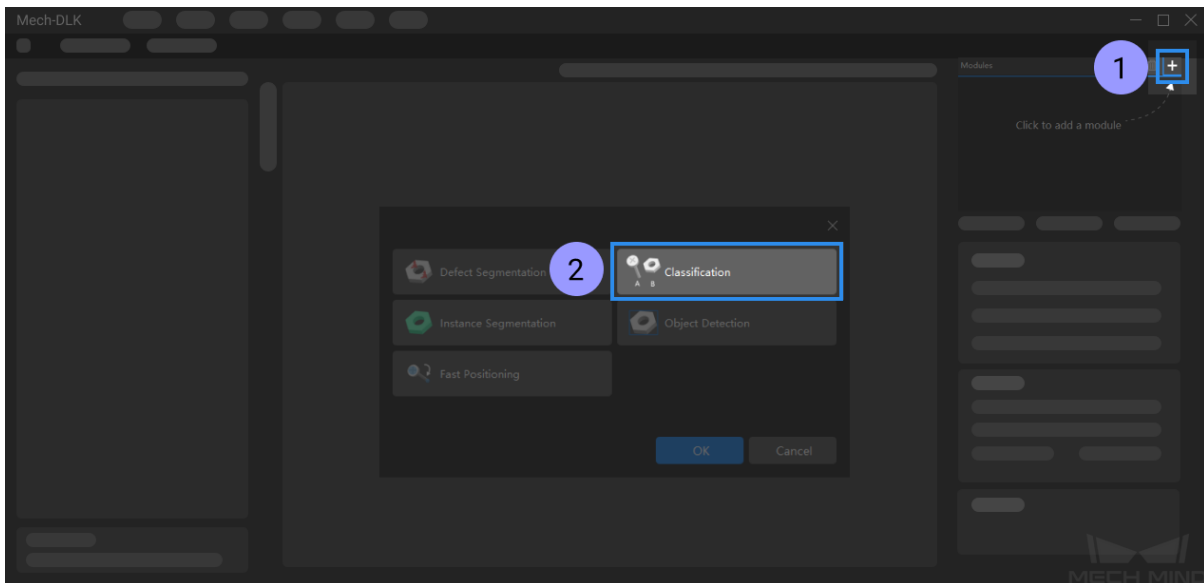
1. *The quality of the captured images.*
2. *The quality of the datasets.*
3. *The quality of labeling.*

6.2 Start Using the “Classification” Module

Please click [here](#) to download an image dataset of condensers. In this section, we will use a **Classification** module and train a model to distinguish between the front and back sides of the condenser.

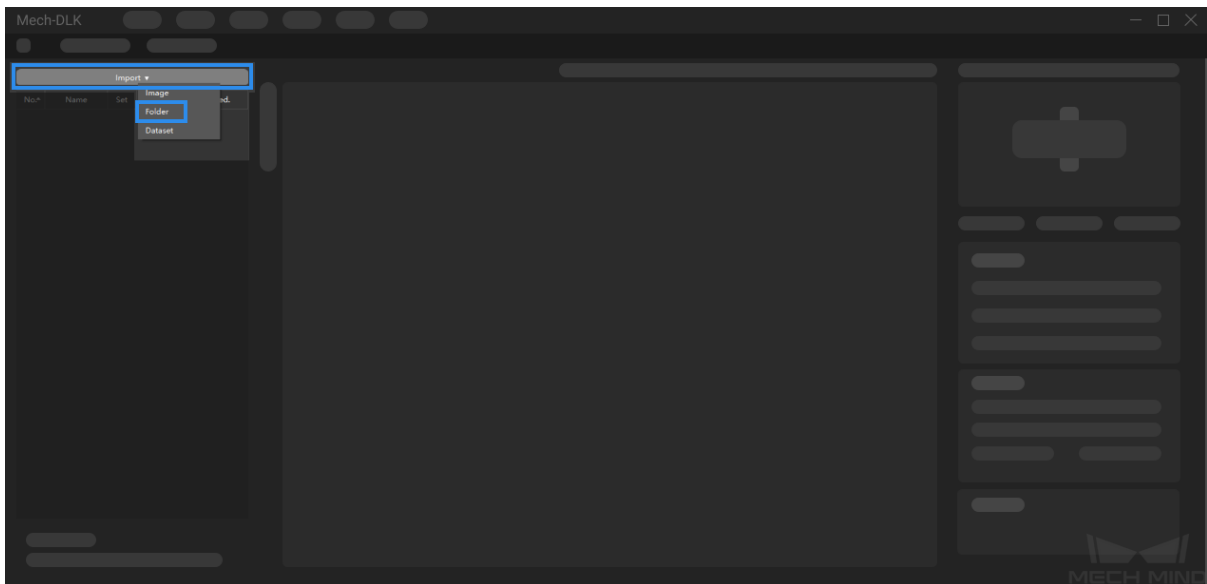
1. Create a new project and add the classification module

Click on *New Project* in the interface, name the project, and select a directory to save the project. Click on **+** in the upper right corner of the **Modules** panel and add the **Classification** module.



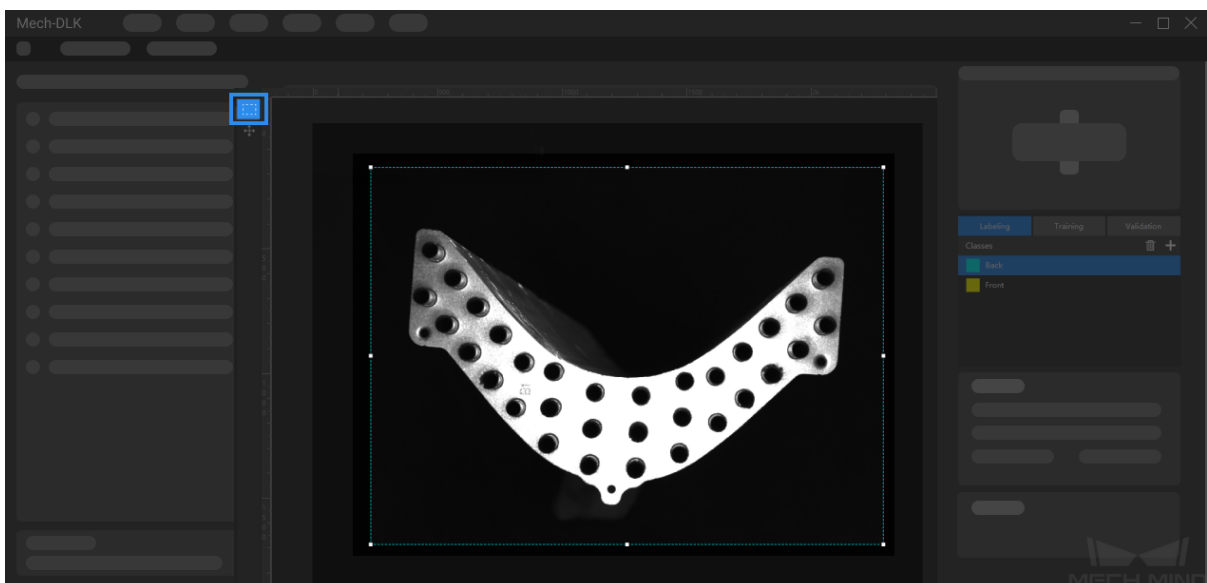
2. Import the image dataset of condensers

Decompress the downloaded dataset file. Click on the *Import* button in the upper left corner, select **Folder**, and import the image dataset.




3. Select an ROI

Click on the *ROI Tool* and adjust the frame to select the whole condenser as an ROI, and click on *Apply* to save the settings. Setting the ROI can avoid interferences from the background and reduce processing time.

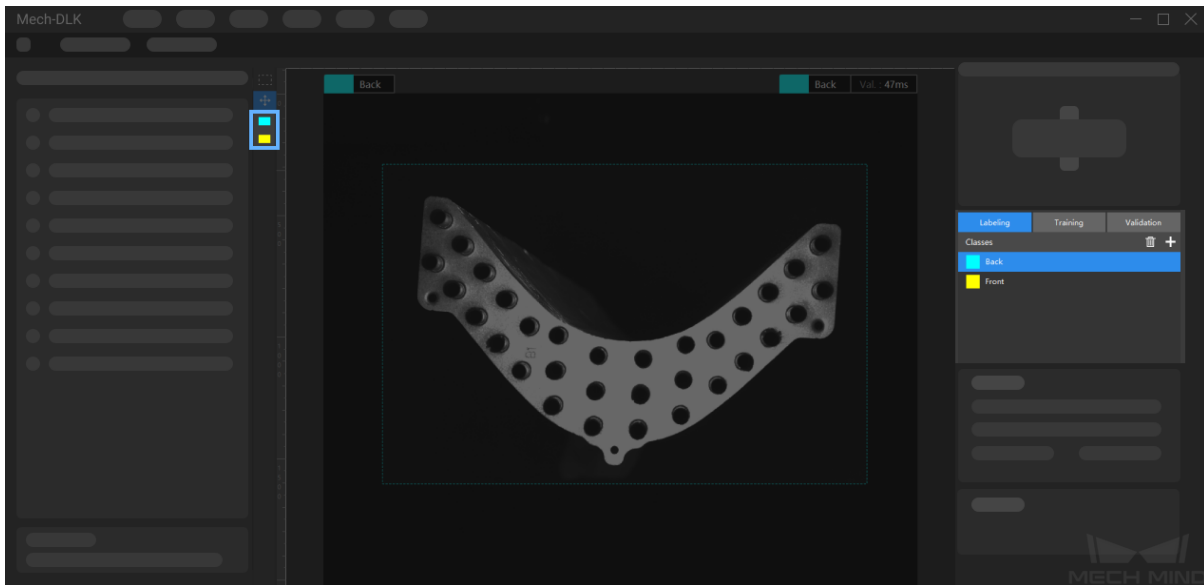


4. Create labels

Click on the  button in the **Classes** panel to create labels and name the labels based on the object names or their features. In this example, the labels are named **Front** and **Back** to distinguish between the front and back sides of the condenser.


5. Label images

Classify the images with corresponding labels. You can select multiple images and label them together. Please make sure that you have labeled the images accurately.

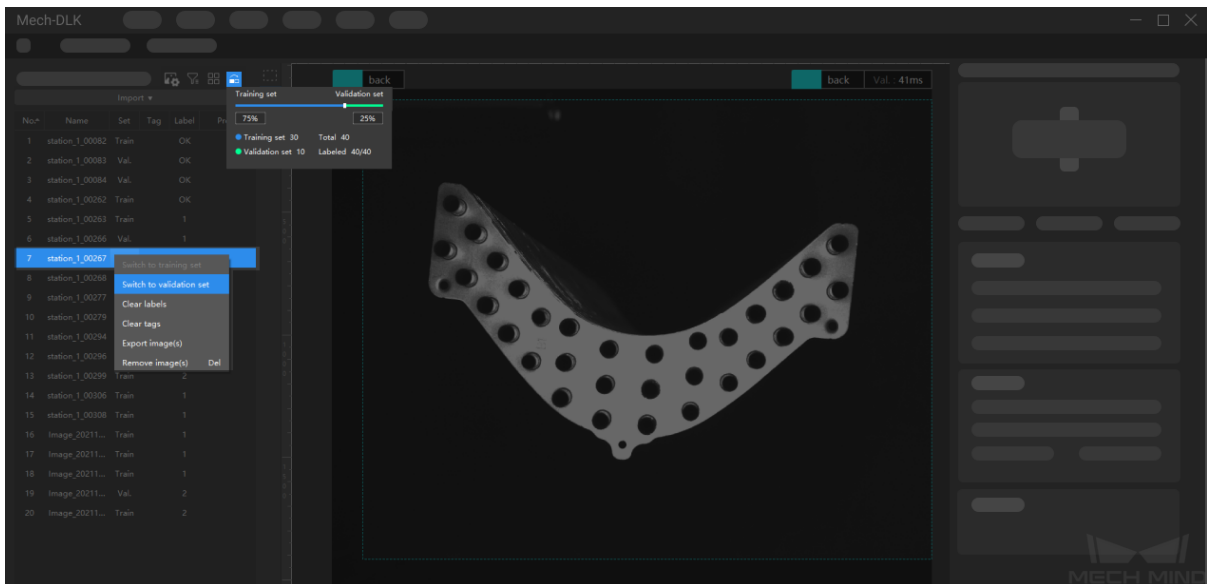


6. Split the dataset into the training set and validation set

By default, 80% of the images in the dataset will be split into the training set and the rest 20% will be split into the validation set. Please make sure that both the training set and validation set include **images in all different classes**, which will guarantee that the algorithm can learn all different features and validate the images of different classes properly.

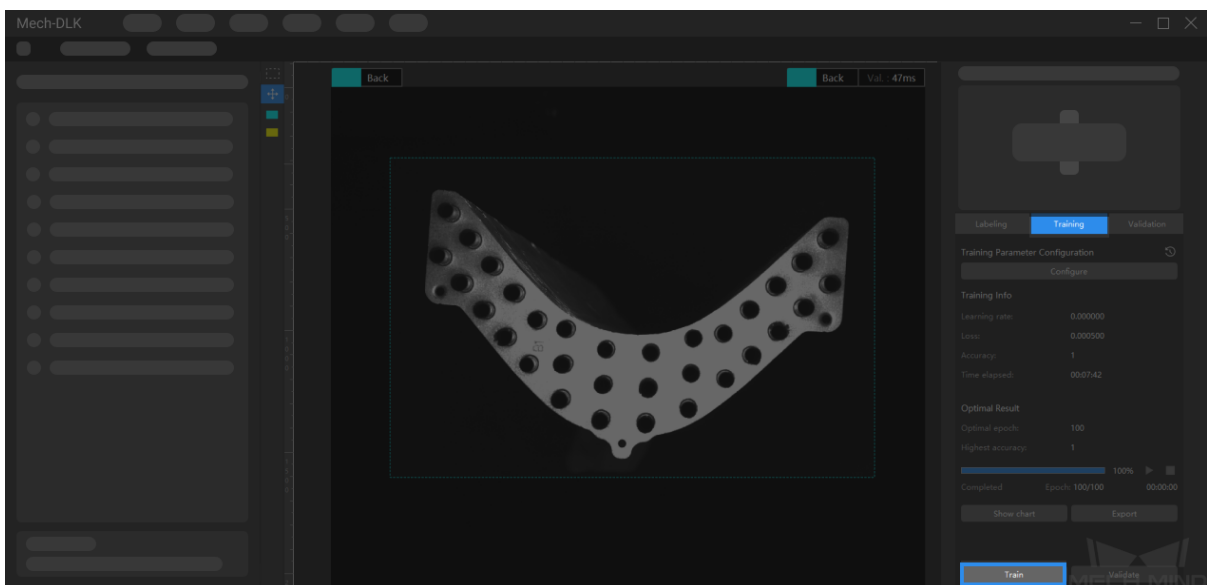
If the default training set and validation set cannot meet this requirement, please click on  and drag the slider to adjust the proportion.

You can also right-click the individual image and switch it to the training/validation set manually.



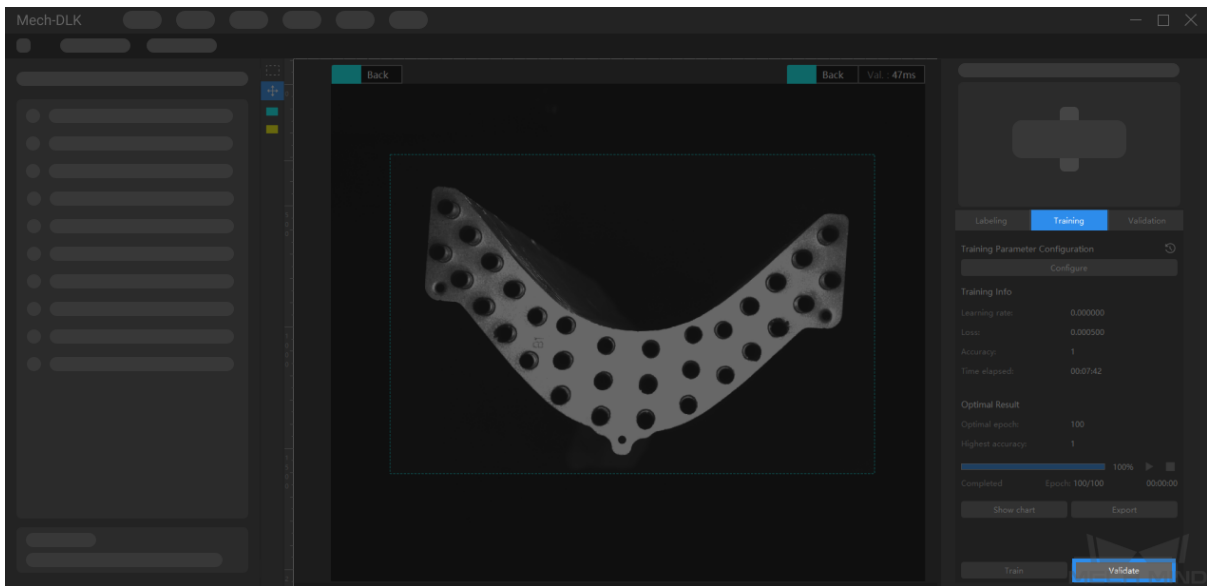
7. Train the model

Keep the default training parameter settings and click on *Train* to start training the model.



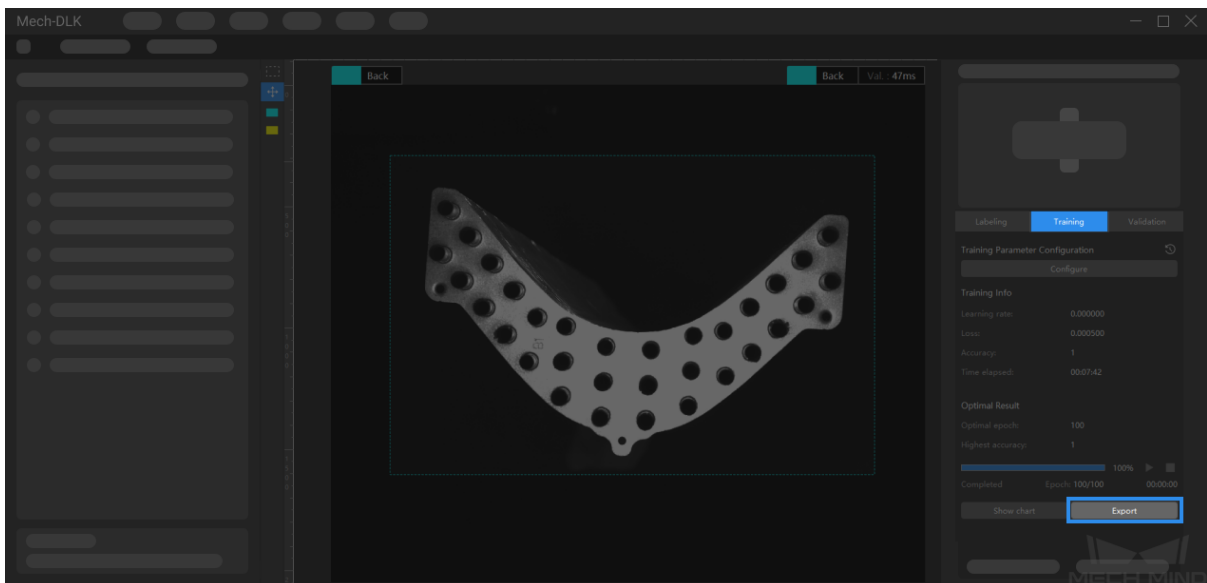
8. Validate the model

After the training is completed, click on *Validate* to validate the model and check the results.



9. Export the model

Click on *Export* and select a directory to save the exported model (with file extension dlkpack). You can deploy the model according to actual needs.



6.3 Train a High-Quality Model

This section introduces several factors that most affect the model quality and how to train a high-quality image classification model.

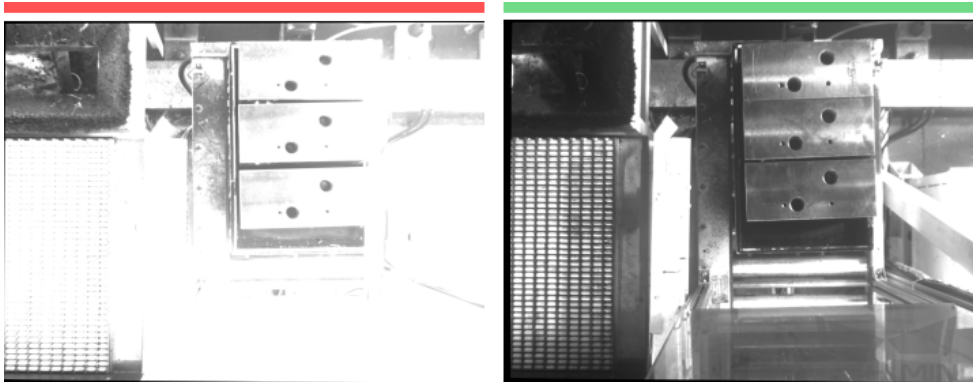
- *Ensure image quality*
- *Ensure dataset quality*
- *Ensure labeling quality*

6.3.1 Ensure Image Quality

1. Avoid **overexposure**, **dimming**, **color distortion**, **blur**, **occlusion**, etc. These conditions will lead to the loss of features that the deep learning model relies on, which will affect the model training effect.

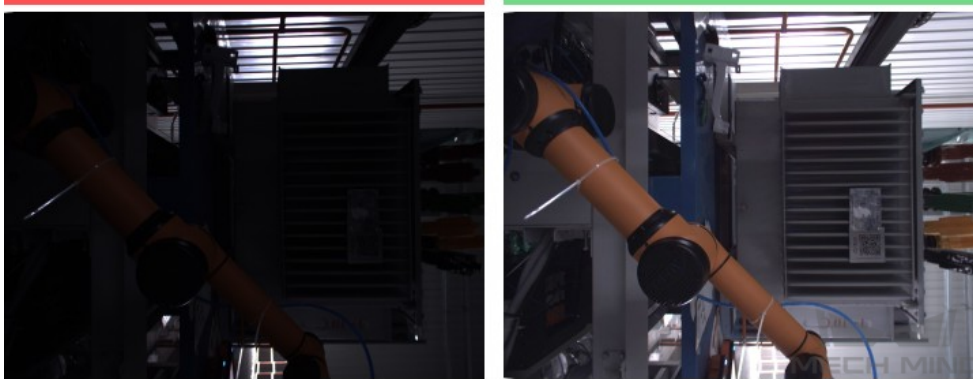
- **Left image, bad example:** Overexposure.
- **Right image, good example:** Adequate exposure.

- **Left image, bad example:** Dim image.
- **Right image, good example:** Adequate exposure.



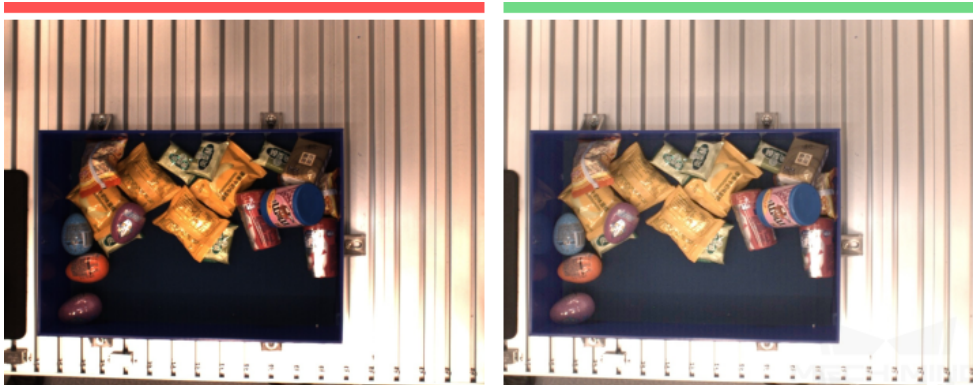
You can avoid overexposure by methods such as shading.

- **Left image, bad example:** Color distortion.
- **Right image, good example:** Normal color.



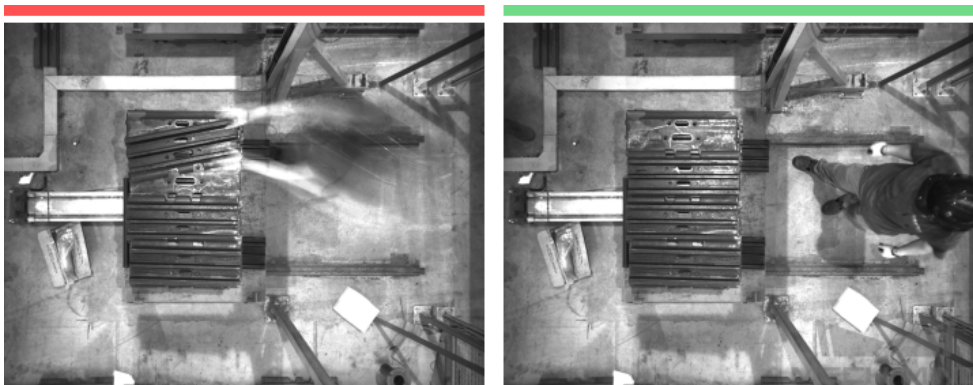
You can avoid dimming by methods such as supplementary light.

- **Left image, bad example:** Blur.
- **Right image, good example:** Clear.



Color distortion can be avoided by adjusting the white balance.

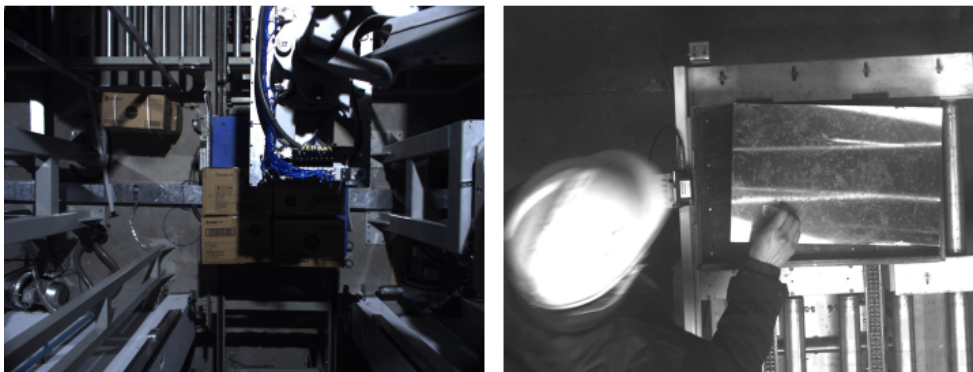
- **Left image, bad example:** Occluded by the robot arm.
- **Right image, good example:** Occluded by a human.



Please avoid capturing images when the camera or the objects are still moving.

2. Ensure that the **background, perspective, and height** of the image capturing process are consistent with the actual application. Any inconsistency will reduce the effect of deep learning in practical applications. In severe cases, data must be re-collected. Please confirm the conditions of the actual application in advance.

- **Bad example:** The background in the training data (left) is different from the background in the actual application (right).



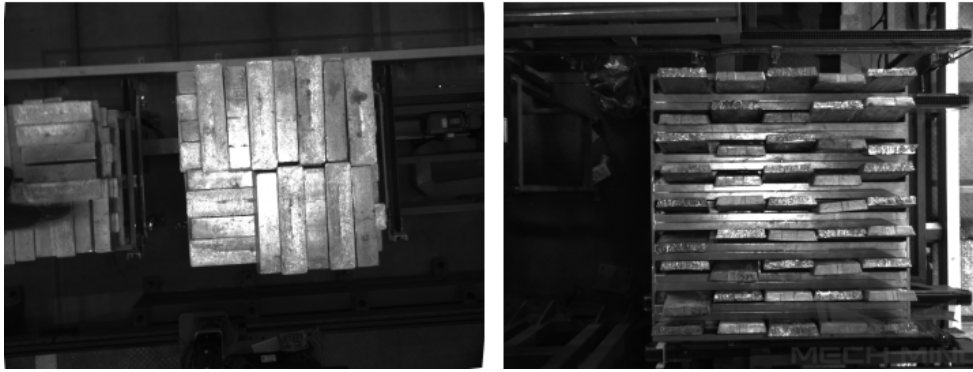
Please make sure there is no robot or human in the way from the camera to the objects.

- **Bad example:** The field of view and perspective in the training data (left) are different from that in the actual application (right).



Please make sure the background stays the same when capturing the training data and when deploying the project.

- **Bad example:** The camera height in the training data (left) is different from the background in the actual application (right).



Please make sure the field of view and perspective stay the same when capturing the training data and when deploying the project.

Attention: The quality of image classification is sensitive to lighting, and the lighting conditions need to be consistent during collection; if the light is inconsistent in the morning and evening, it needs to be collected separately according to the situation.

6.3.2 Ensure Dataset Quality

The “Classification” module obtains a model by learning the features of existing images and applies what is learned to the actual application.

Therefore, to train a high-quality model, the conditions of the collected and selected dataset must be consistent with those of the actual applications.

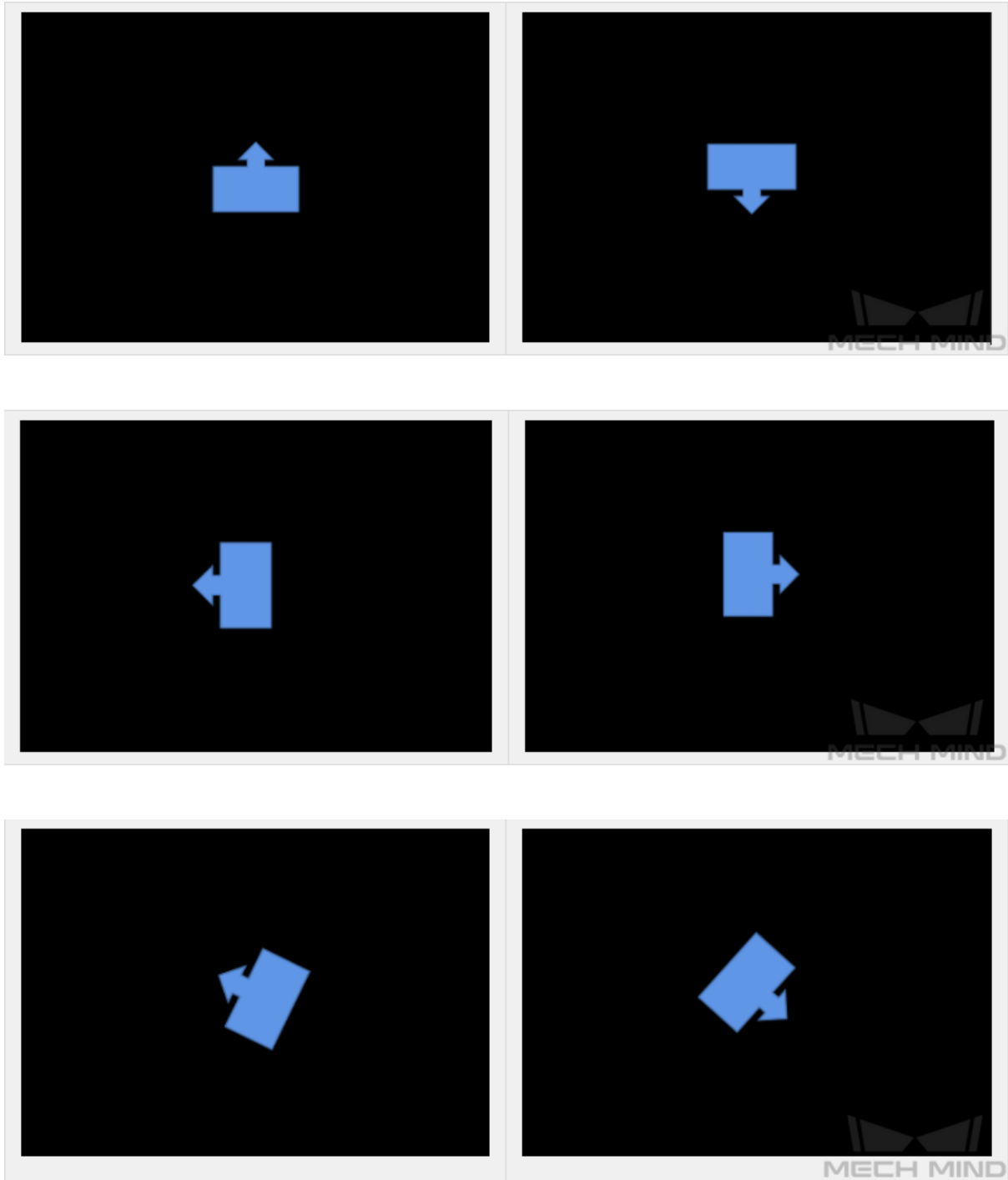
1. *Collect datasets*
2. *Select datasets*

Collect Datasets

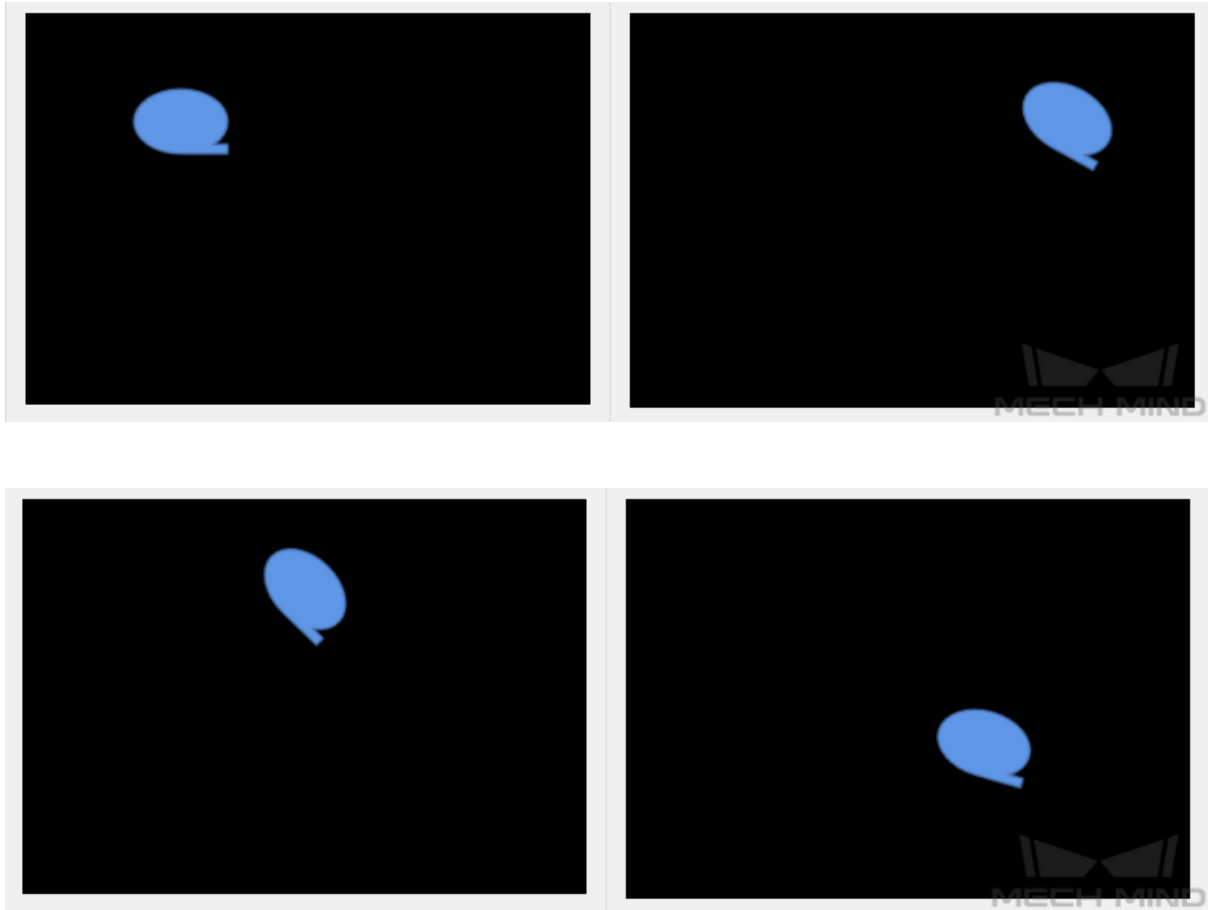
Various placement conditions need to be properly allocated. For example, if there are horizontal and vertical incoming materials in actual production, but only the data of horizontal incoming materials are collected for training, the classification effect of vertical incoming materials cannot be guaranteed.

Therefore, when collecting data, it is necessary to **consider various conditions** of the actual application, including the features present given different object placement **orientations** and **positions**.

1. **Different orientations**



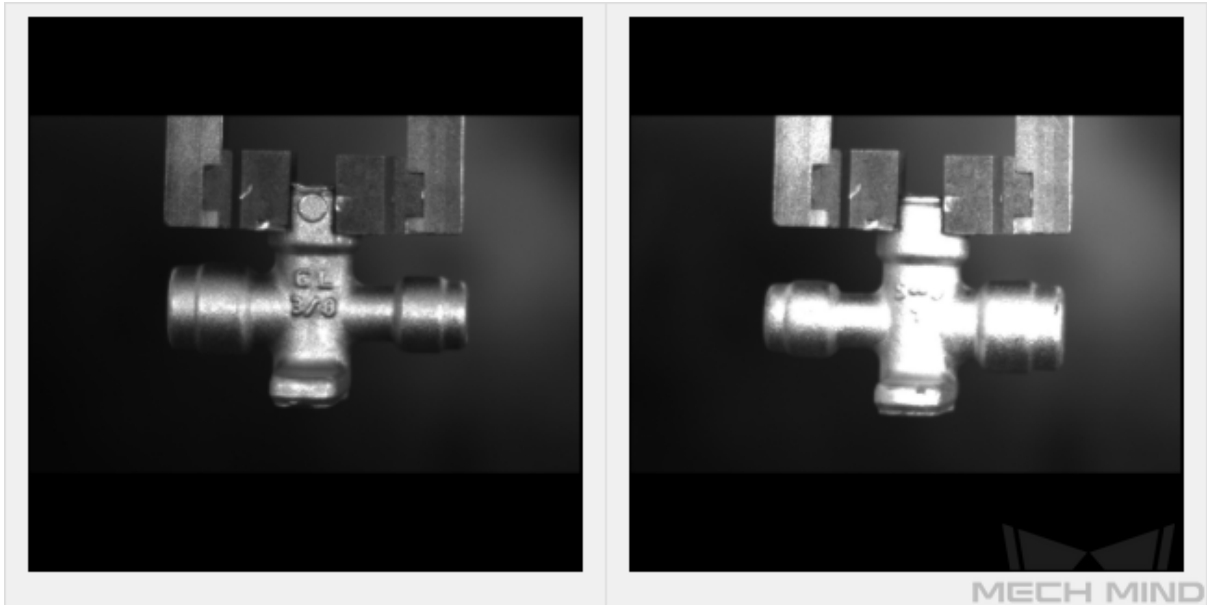
2. Different positions



Data Collection Examples

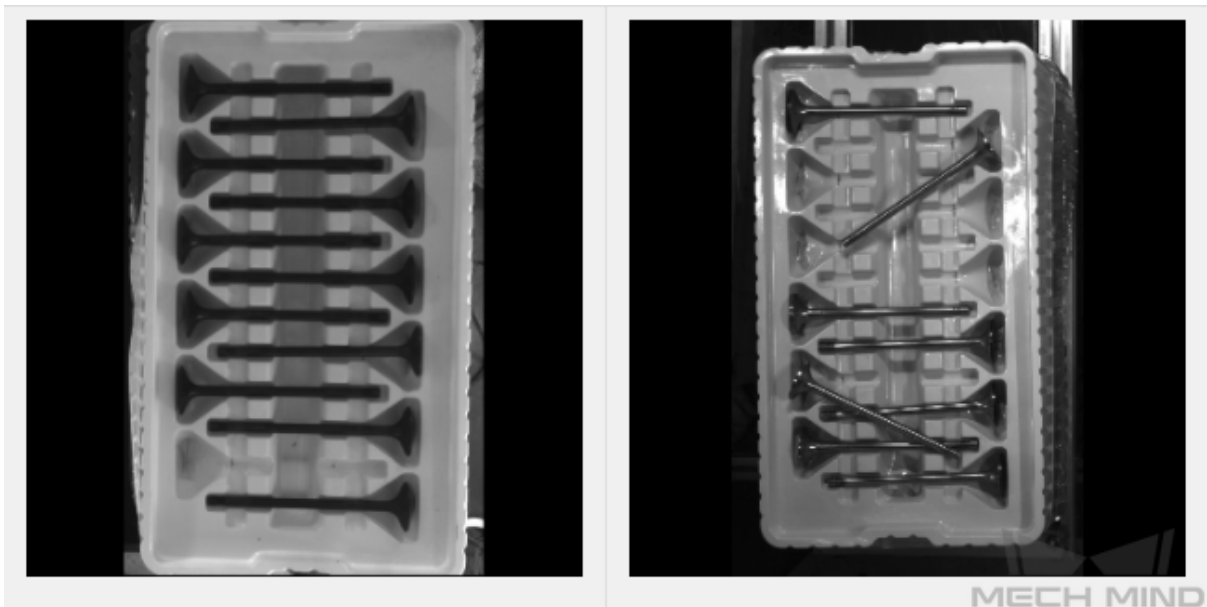
A valve tube project

- Single object class.
- Distinguishing between the front and back sides of the valve tubes is needed.
- Positions are generally fixed with small deviations.
- 15 images for the front and back sides each were collected.



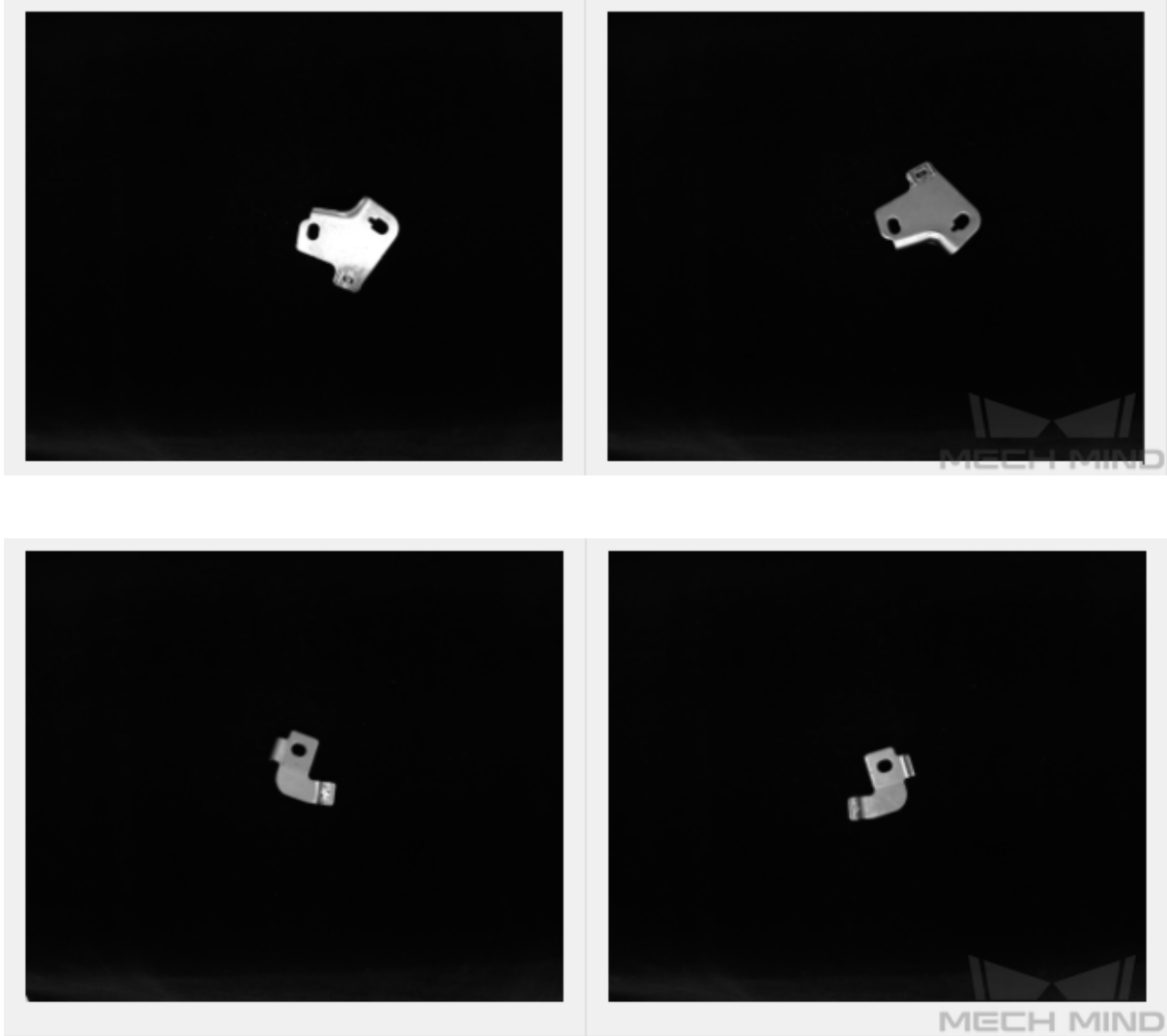
An engine valve assembly project

- Single object class.
- Determining whether the object is correctly placed in the slot is needed.
- Since outside the slot, the object may appear in various positions and orientations, it is necessary to consider different positions and orientations. 20 images were collected for objects outside the slot.
- In the slot, only the factor of different positions need to be considered, so 10 images were collected for objects inside the slot.



A sheet metal project

- Two object classes. Different object sizes need to be recognized.
- Objects may come in different positions and orientations.
- 20 images were collected for the front and back sides each.



Select the Right Dataset

1. Control dataset image quantities

For the first-time model building of the “Classification” module, capturing 30 images is recommended.

It is not true that the larger the number of images the better. Adding a large number of inadequate images in the early stage is not conducive to the later model improvement, and will make the training time longer.

2. Collect representative data

Dataset image capturing should consider all the conditions in terms of illumination, color, size, etc. of the objects to recognize.

- **Lighting:** Project sites usually have environmental lighting changes, and the datasets should contain images with different lighting conditions.
- **Color:** Objects may come in different colors, and the datasets should contain images of objects of all the colors.
- **Size:** Objects may come in different sizes, and the datasets should contain images of objects of all existing sizes.

Attention: If the actual on-site objects may be rotated, scaled in images, etc., and the corresponding image datasets cannot be collected, the datasets can be supplemented by adjusting the data augmentation training parameters to ensure that all on-site conditions are included in the datasets.

3. Balance data proportion

The number of images of different conditions/object classes in the datasets should be proportioned according to the actual project; otherwise, the training effect will be affected.

4. Dataset images should be consistent with those from the application site

The factors that need to be consistent include lighting conditions, object features, background, field of view, etc.

6.3.3 Ensure Labeling Quality

- Please ensure there are no missed or incorrect labelings.
 - **Left image, bad example:** Wrong label.
 - **Right image, good example:** Correct label.

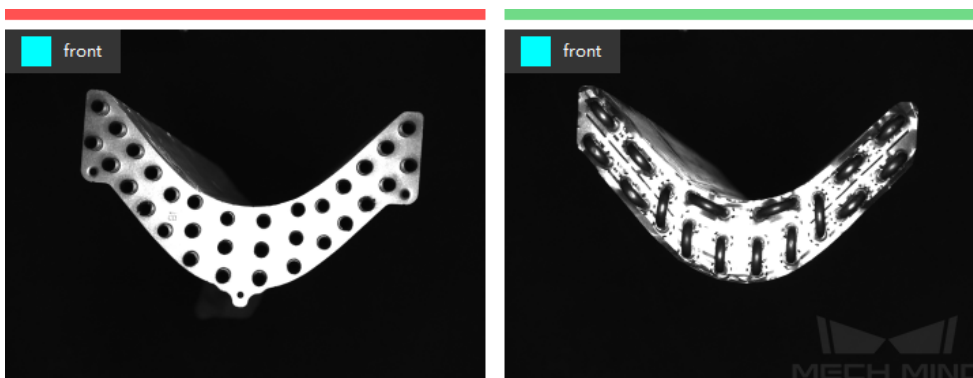


Please make sure the camera height stays the same when capturing the training data and when deploying the project.

6.3.4 Class Activation Maps

After the training of the image classification model is completed, click on *Generate CAM* to generate the class activation maps, and click on *Show class activation maps (CAM)*.

The class activation maps show the feature regions in the images that are paid attention to when training the model, and they help check the classification performance, thus providing references for optimizing the mode.

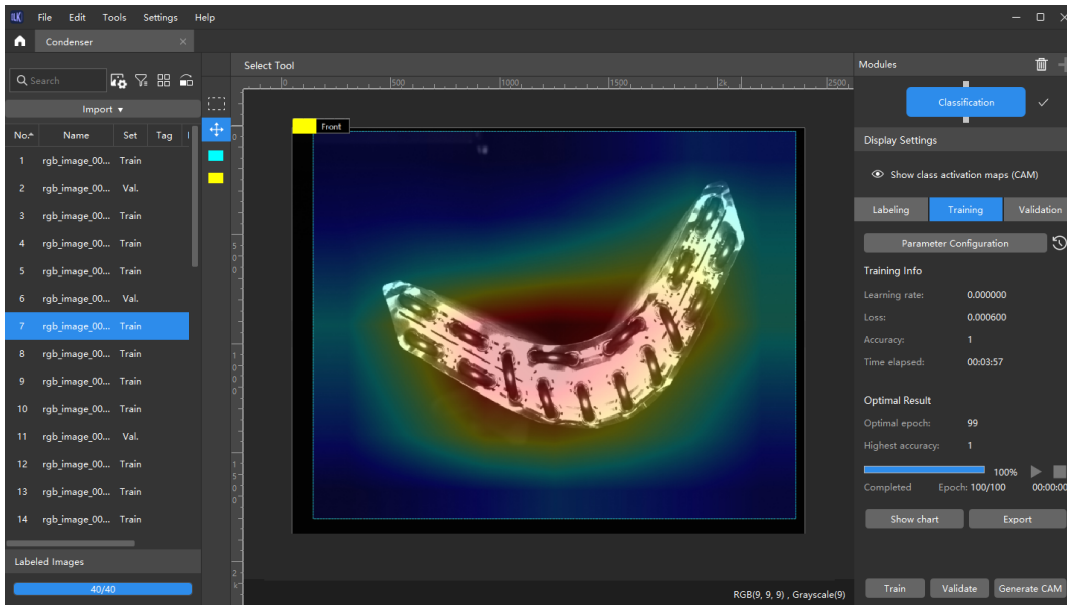


The left image is the workpiece front and the right image is the workpiece back.

6.4 CPU and GPU Model Deployment

6.4.1 CPU Model Deployment

If the model is to be deployed on a CPU device, please select **CPU** for **Deployment device**.

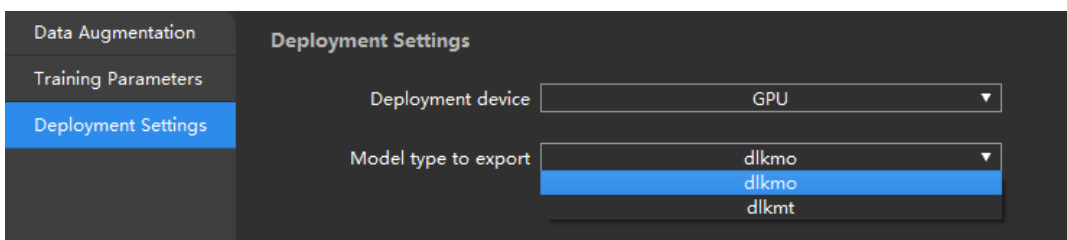
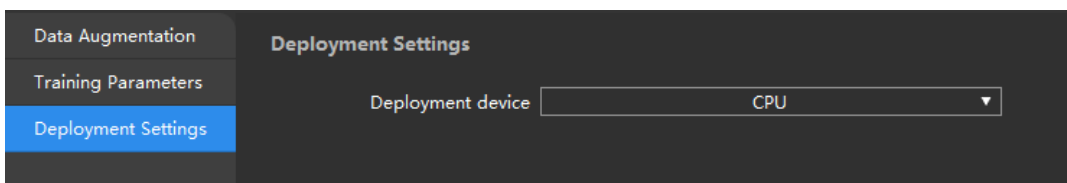


6.4.2 GPU Model Deployment

If the model is to be deployed on a GPU device, please select **GPU** for **Deployment device**.

For **Model type to export**, please select **dlkmt** only when the model training and deployment are done on GPU graphics cards of the same model.

Otherwise, please select **dlkmo** to avoid the problem that the model file cannot be used due to the difference in graphics card models for training and deployment.



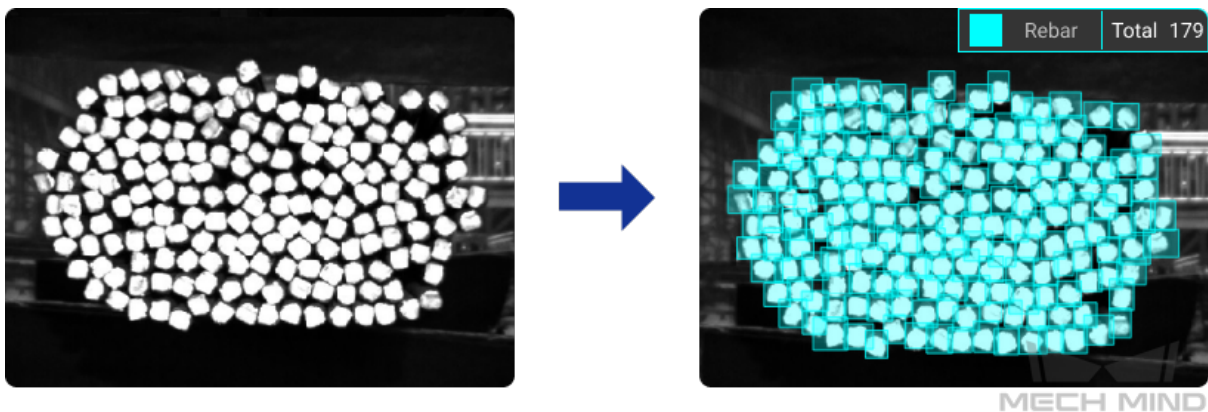
OBJECT DETECTION

7.1 Introduction

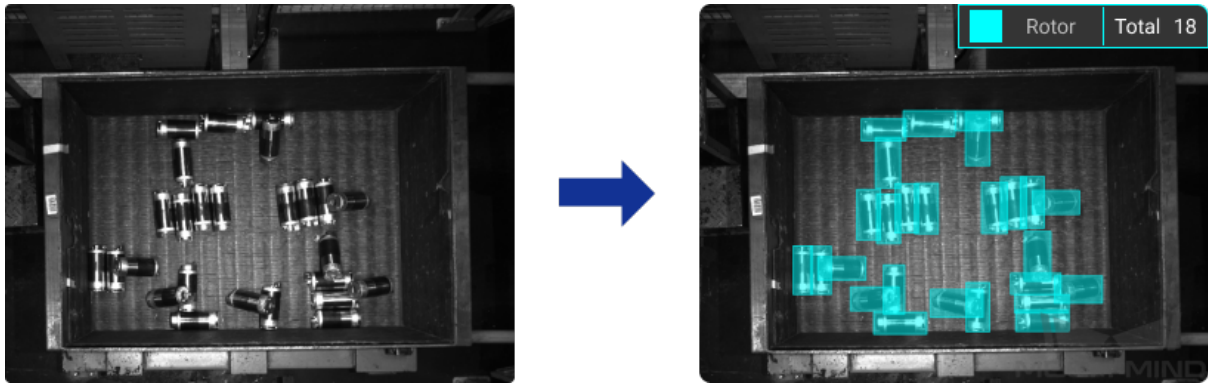
This module is used to detect the location of all target objects and estimate their classes.

7.1.1 Applicable Scenarios

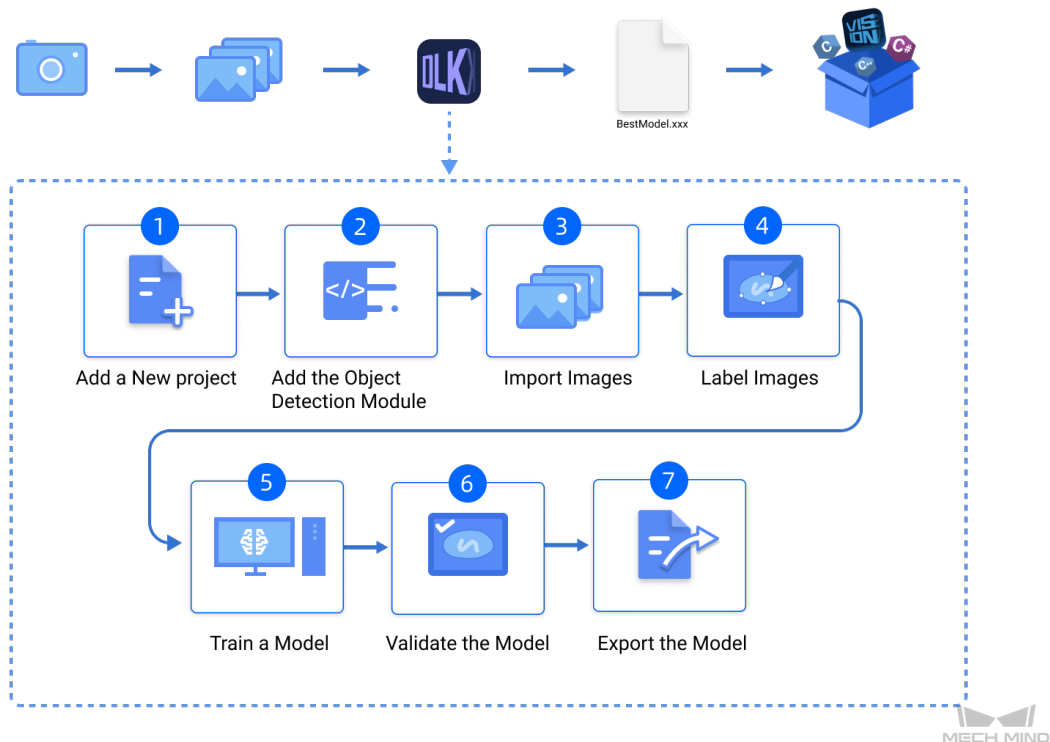
Counting workpieces: Suitable for counting bundles of steel bars, loose parts, and tiny parts in factories.



Detecting the location of workpieces: Suitable for detecting and locating the metal parts in factories or on assembly lines.



7.1.2 General Workflow



7.1.3 Tips

The performance of the trained model depends on the followings.


1. *The quality of the captured images.*
2. *The quality of the datasets.*
3. *The quality of labeling.*

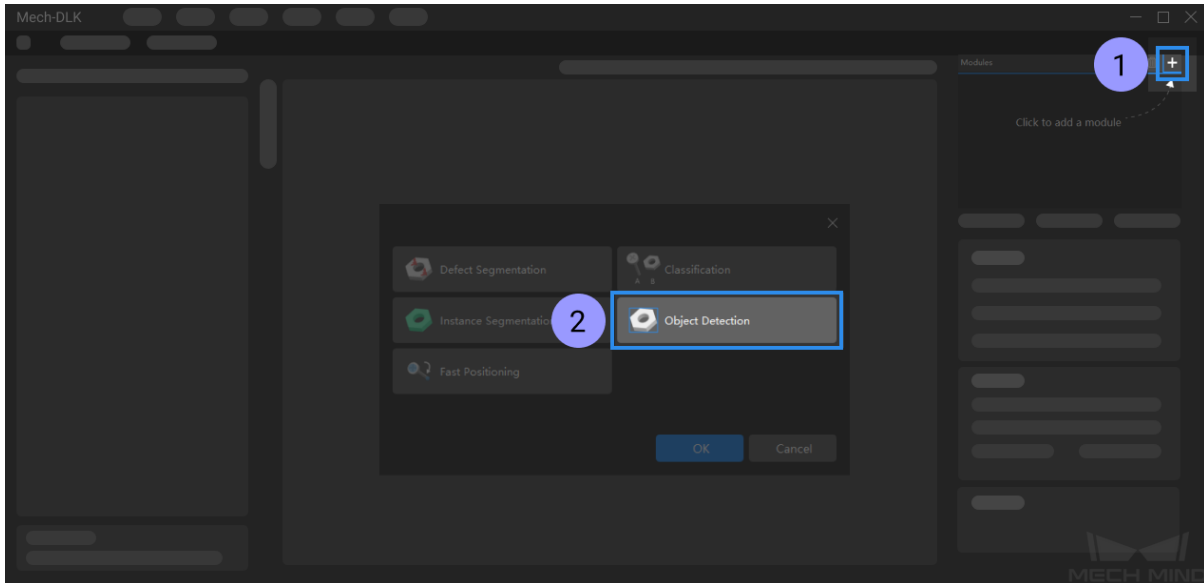
7.2 Start Using the “Object Detection” Module

Please click [here](#) to download an image dataset of rotors. In this section, we will use an **Object Detection** module and train a model to detect the rotors in the image and output the quantity.

1. Create a new project and add the object detection module

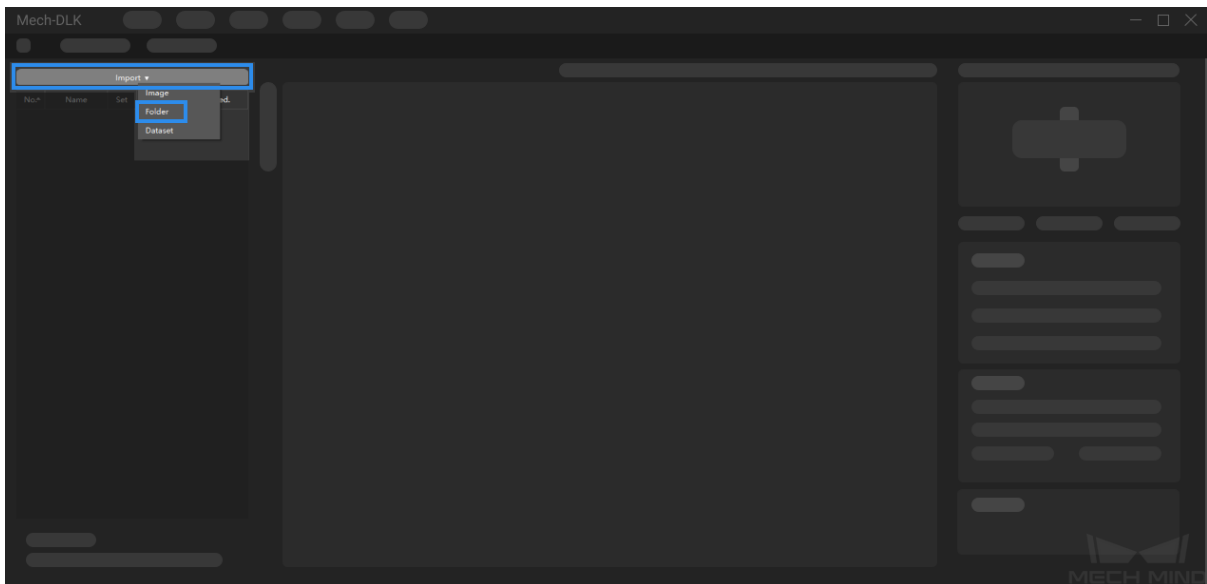
Click on *New Project* in the interface, name the project, and select a directory to save the project.

Click on  in the upper right corner of the **Modules** panel and add the **Object Detection** module.




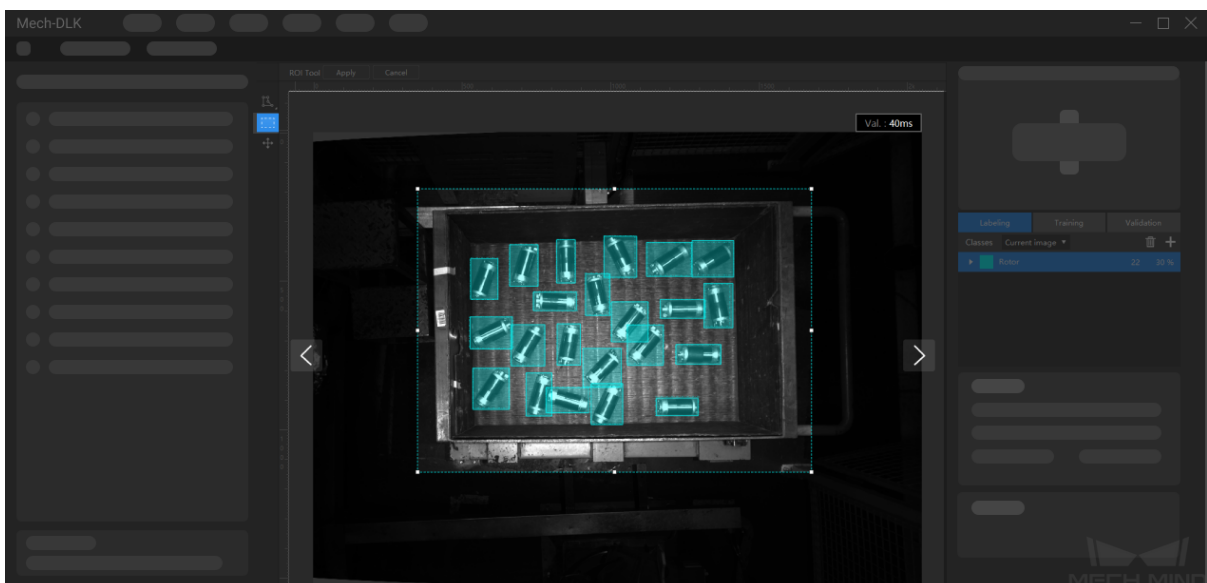
2. Import the image dataset of rotors

Decompress the downloaded dataset file. Click on the *Import* button in the upper left corner, select **Folder**, and import the image dataset.



3. Select an ROI


Click on the ROI Tool button  and adjust the frame to select the bin containing rotors in the image as an ROI, and click on *Apply* to save the settings. Setting the ROI can avoid interferences from the background and reduce processing time.



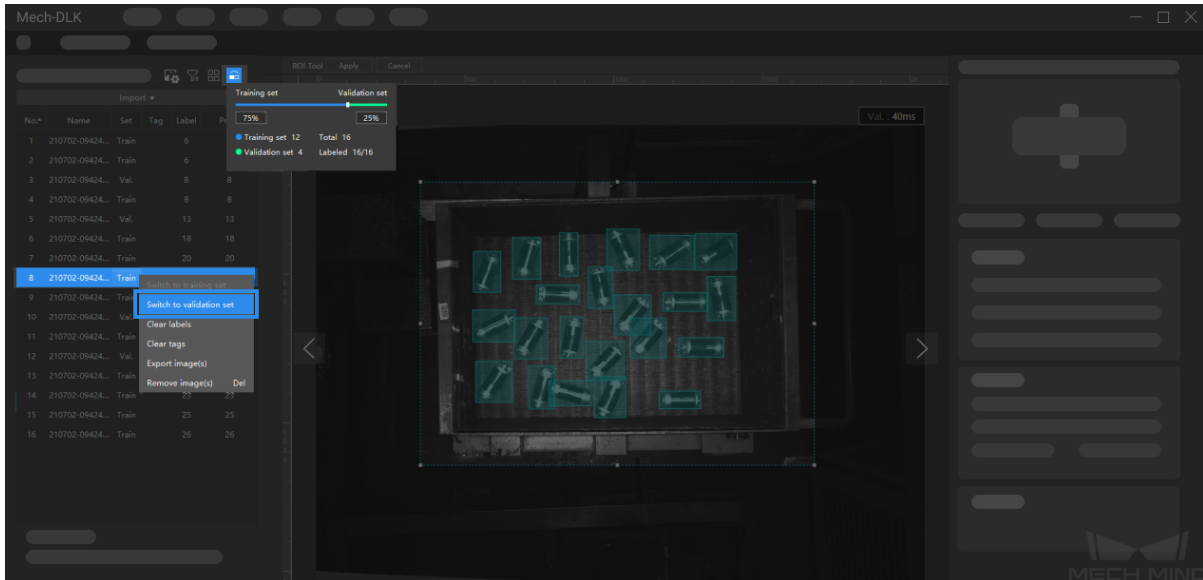
4. Split the dataset into the training set and validation set

By default, 80% of the images in the dataset will be split into the training set and the rest 20% will be split into the validation set. Please make sure that both the training set and validation set


include **objects of all classes to be detected**, which will guarantee that the algorithm can learn all different classes and validate the images properly.

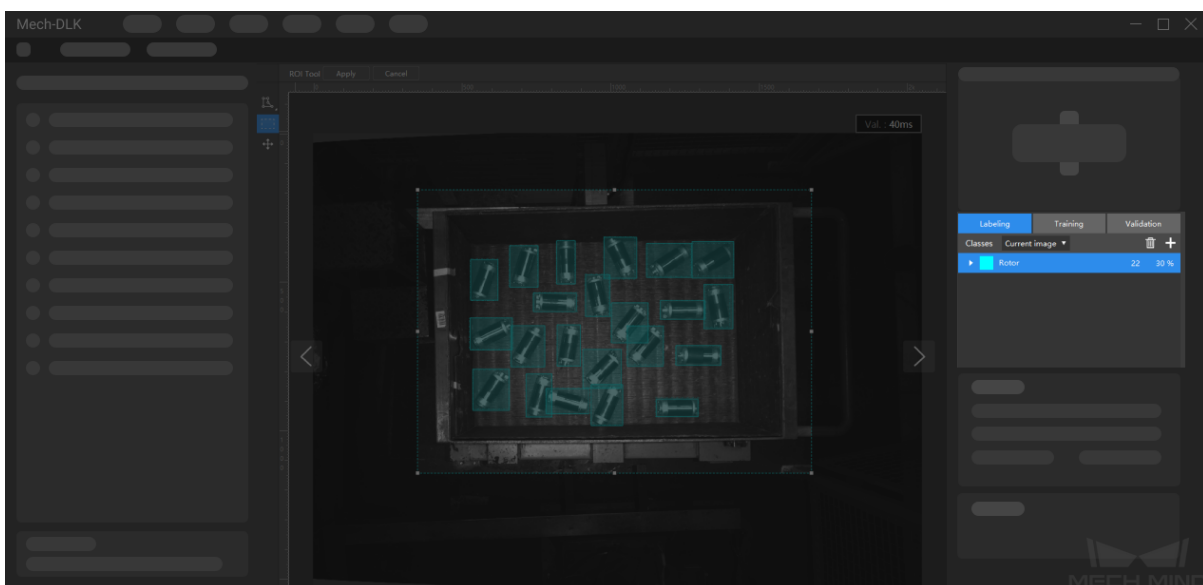
If the default training set and validation set cannot meet this requirement, please click on  and drag the slider to adjust the proportion.

You can also right-click the individual image and switch it to the training/validation set manually.




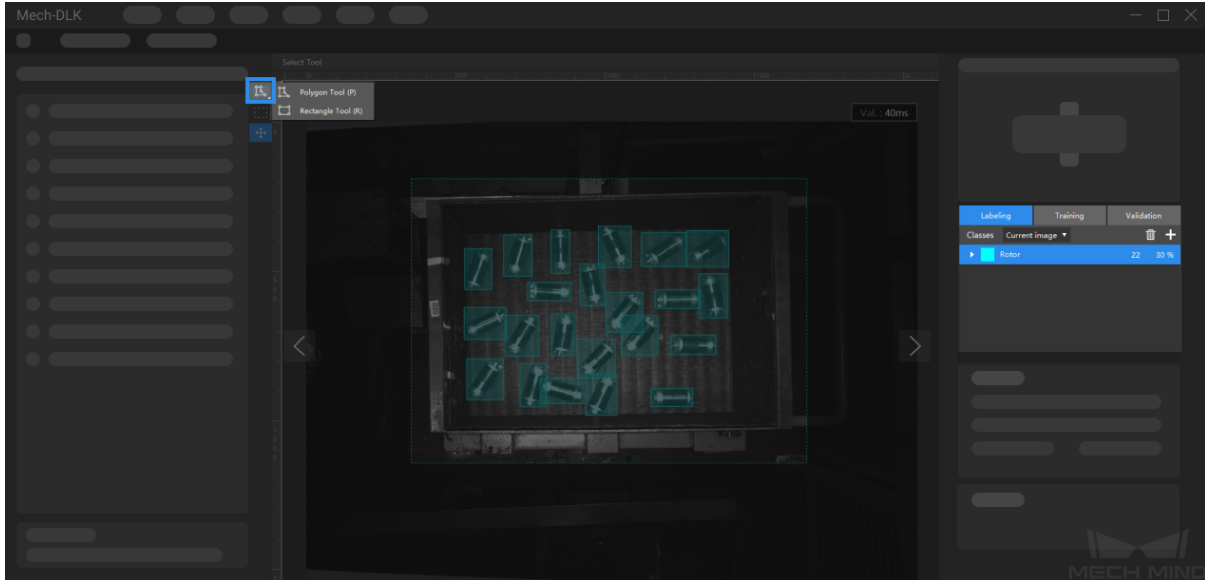
5. Create Labels

Select *Labeling* and click on the  button in the **Classes** panel to create labels based on the type or feature of different objects. In this example, the labels are named after the rotors.



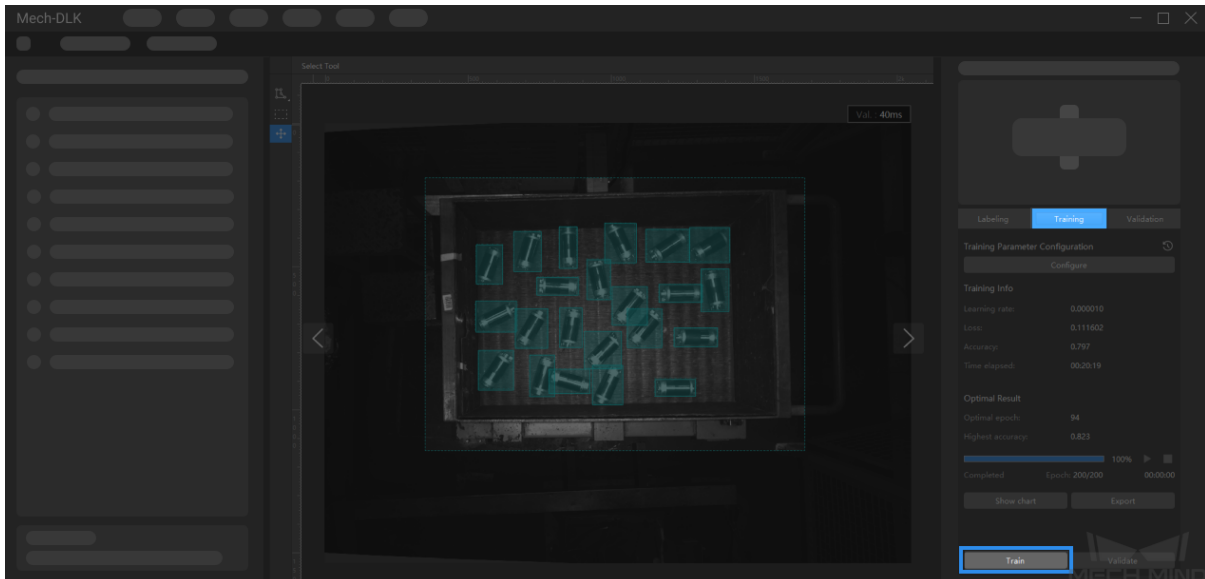
6. Label images

Right-click on the  button and select a proper tool to label the image. Please select the rotors as precisely as possible and avoid including irrelevant regions. Inaccurate labeling will affect the training result of the model.



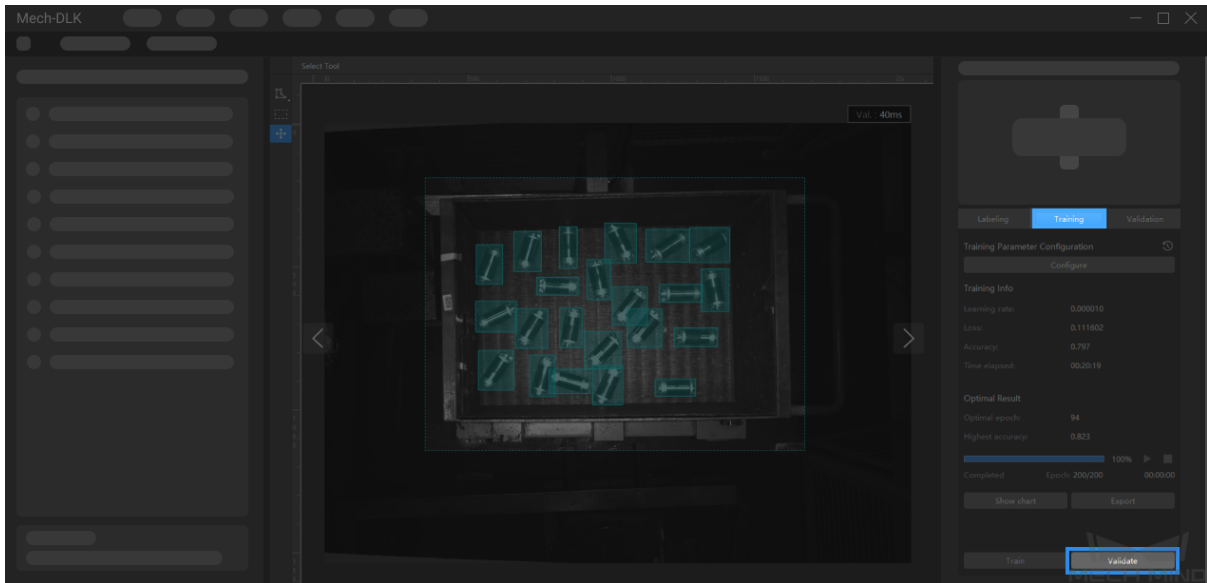
7. Train the model

Keep the default training parameter settings and click on *Train* to start training the model.



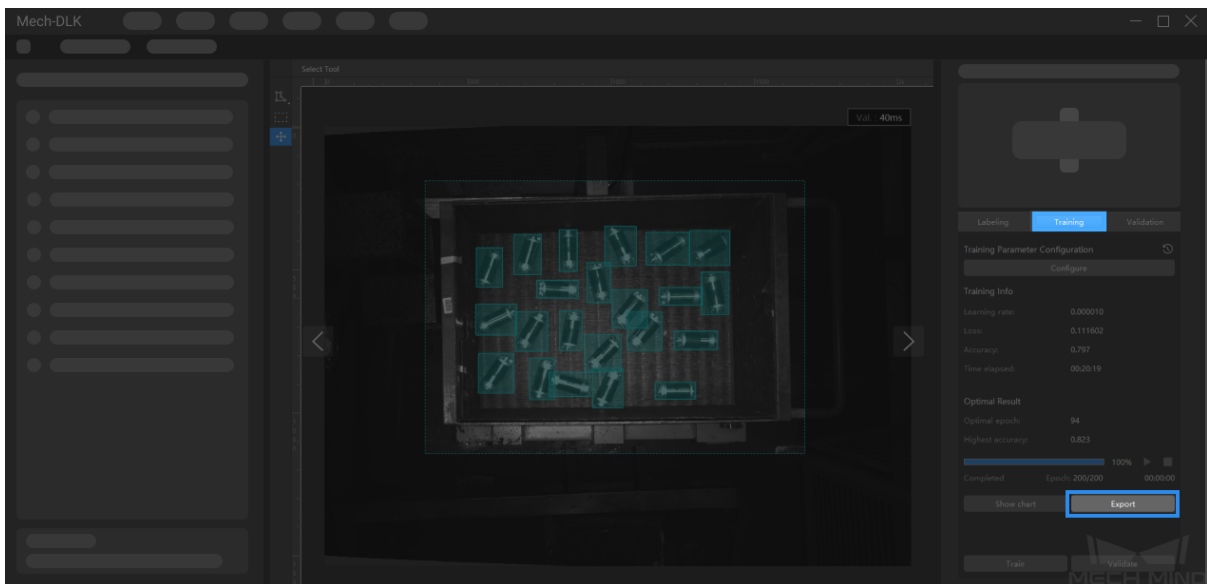
8. Validate the model

After the training is completed, click on *Validate* to validate the model and check the results.



9. Export the model

Click on *Export* and select a directory to save the exported model. You can deploy the model according to actual needs.



7.3 Train a High-Quality Model

This section introduces several factors that most affect the model quality and how to train a high-quality image classification model.

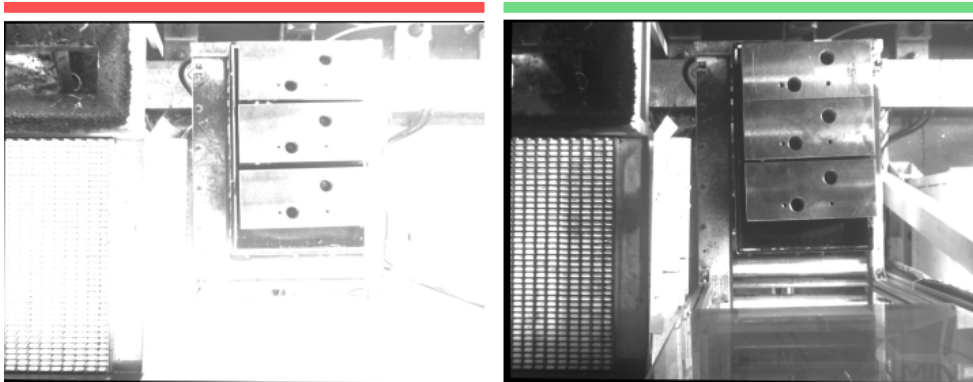
- *Ensure image quality*
- *Ensure dataset quality*
- *Ensure labeling quality*

7.3.1 Ensure Image Quality

1. Avoid **overexposure**, **dimming**, **color distortion**, **blur**, **occlusion**, etc. These conditions will lead to the loss of features that the deep learning model relies on, which will affect the model training effect.

- **Left image, bad example:** Overexposure.
- **Right image, good example:** Adequate exposure.

- **Left image, bad example:** Dim image.
- **Right image, good example:** Adequate exposure.



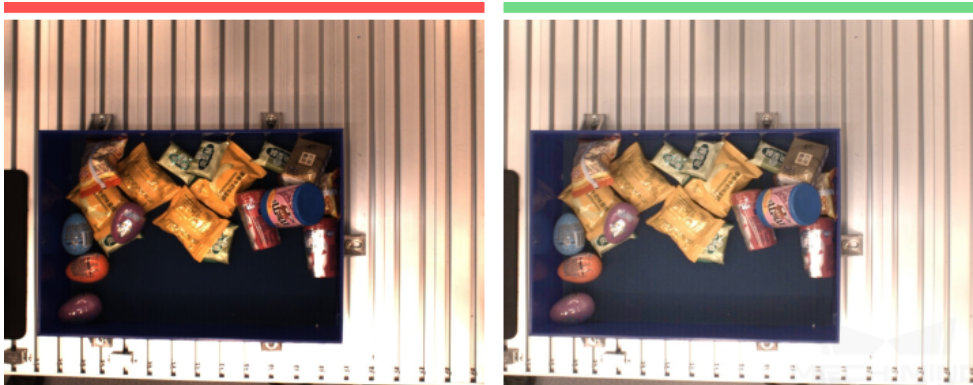
You can avoid overexposure by methods such as shading.

- **Left image, bad example:** Color distortion.
- **Right image, good example:** Normal color.



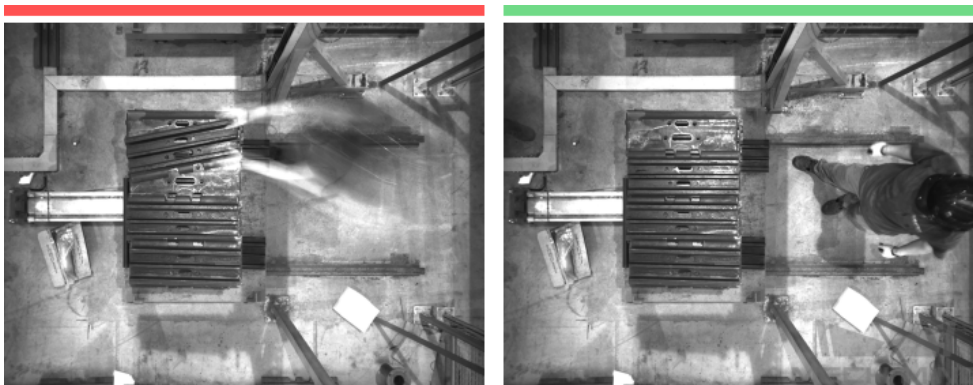
You can avoid dimming by methods such as supplementary light.

- **Left image, bad example:** Blur.
- **Right image, good example:** Clear.



Color distortion can be avoided by adjusting the white balance.

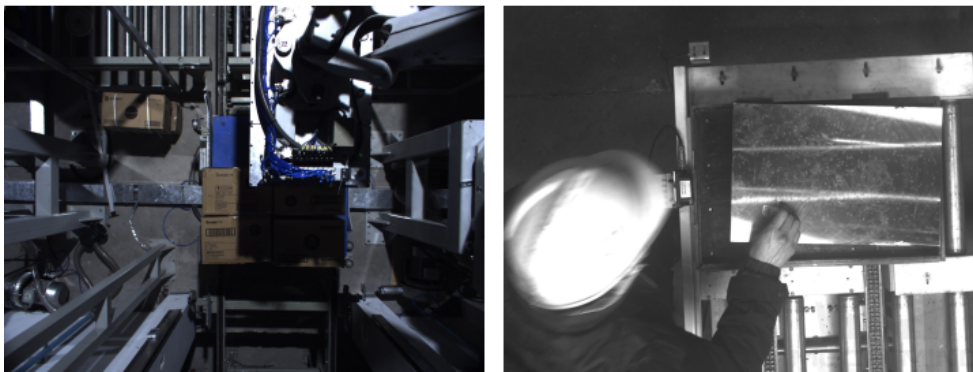
- **Left image, bad example:** Occluded by the robot arm.
- **Right image, good example:** Occluded by a human.



Please avoid capturing images when the camera or the objects are still moving.

2. Ensure that the **background, perspective, and height** of the image capturing process are consistent with the actual application. Any inconsistency will reduce the effect of deep learning in practical applications. In severe cases, data must be re-collected. Please confirm the conditions of the actual application in advance.

- **Bad example:** The background in the training data (left) is different from the background in the actual application (right).



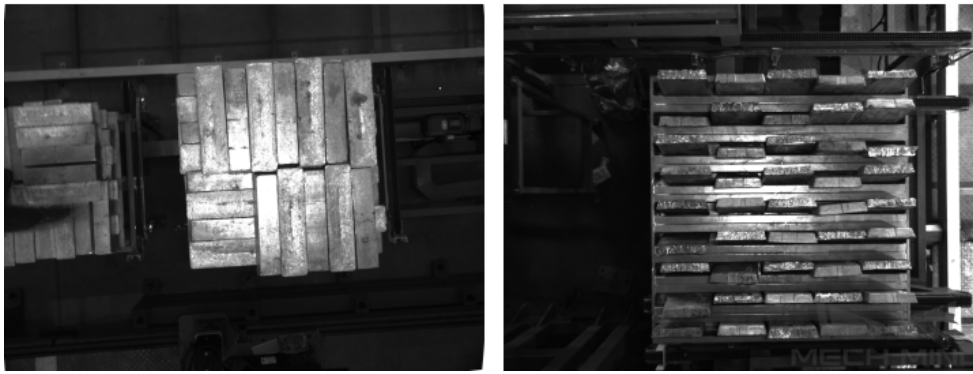
Please make sure there is no robot or human in the way from the camera to the objects.

- **Bad example:** The field of view and perspective in the training data (left) are different from that in the actual application (right).



Please make sure the background stays the same when capturing the training data and when deploying the project.

- **Bad example:** The camera height in the training data (left) is different from the background in the actual application (right).



Please make sure the field of view and perspective stay the same when capturing the training data and when deploying the project.

7.3.2 Ensure Dataset Quality

An object detection model is trained by learning the features of the objects in the image. Then the model applies what is learned in the actual applications.

Therefore, to train a high-quality model, the conditions of the collected and selected dataset must be consistent with those of the actual applications.

1. *Collect Datasets*
2. *Select Datasets*

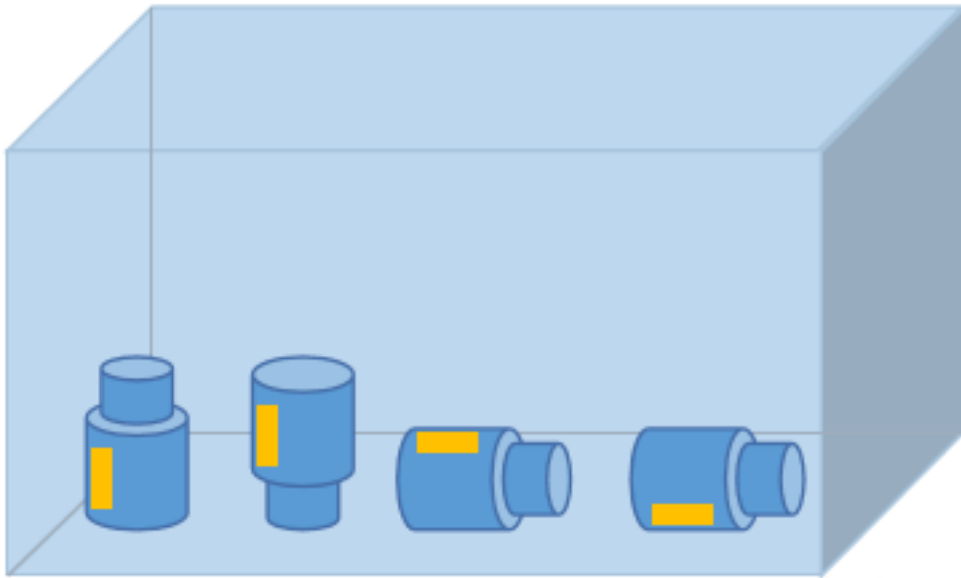
Collect Datasets

Various placement conditions need to be properly allocated. For example, if there are horizontal and vertical incoming materials in actual production, but only the data of horizontal incoming materials are collected for training, the classification effect of vertical incoming materials cannot be guaranteed.

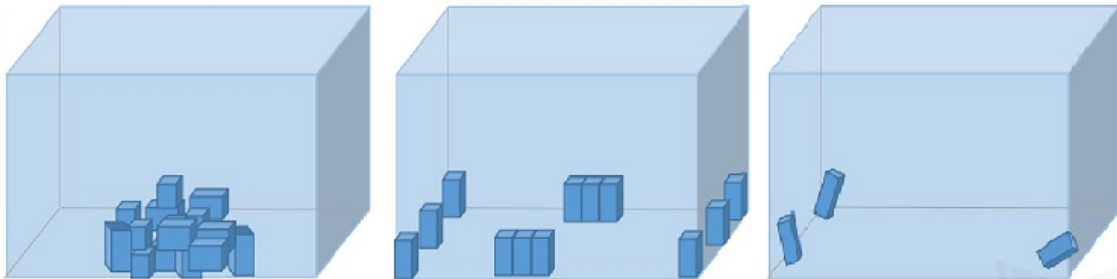
Therefore, when collecting data, it is necessary to **consider various conditions** of the actual application, including the features present given different object placement **orientations, positions, and positional relationships between objects**.

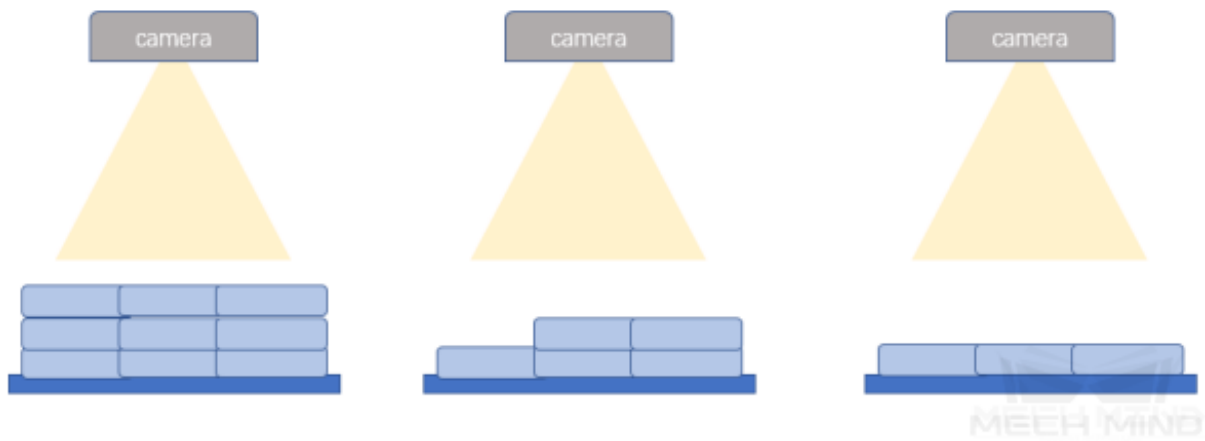
Attention: If some situations are not in the datasets, the deep learning model will not go through inadequate learning on the corresponding features, which will cause the model to be unable to effectively make recognitions given such conditions. In this case, data on such conditions must be collected and added to reduce the errors.

Orientations

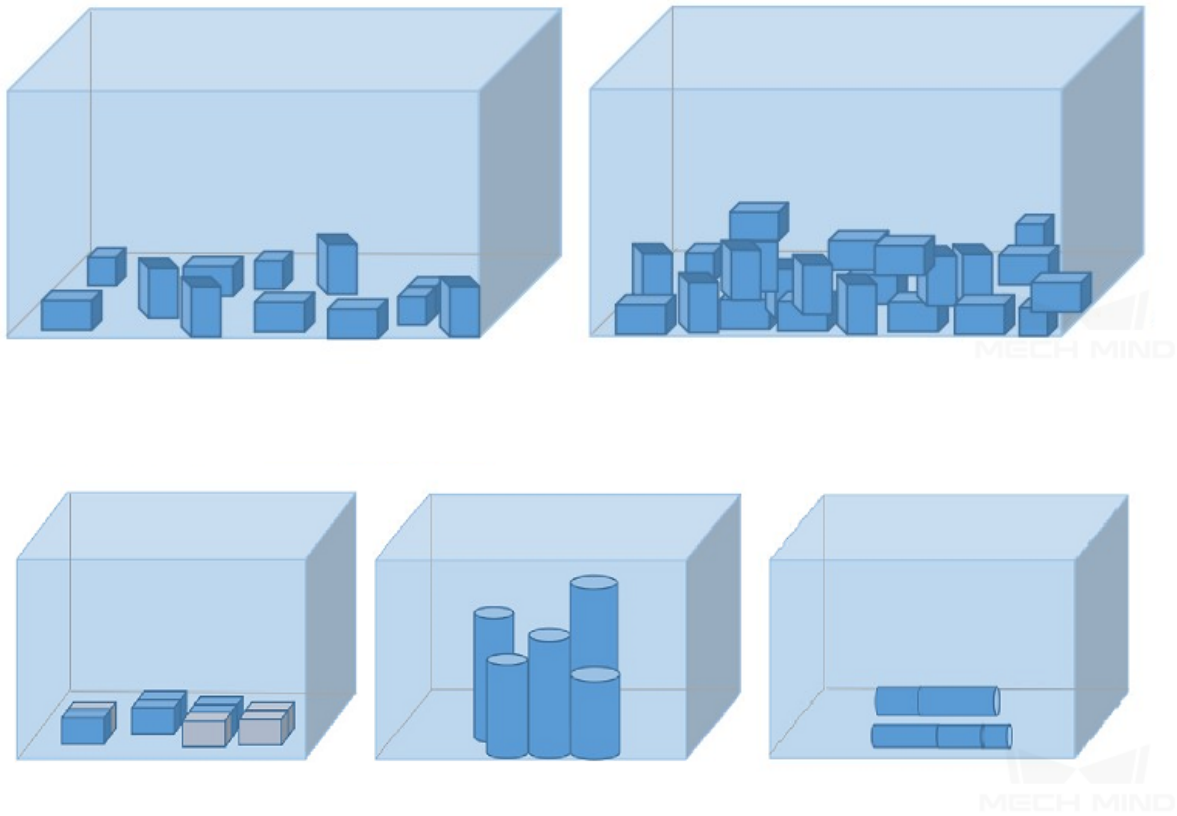


Positions





Positional relationships between objects



Data Collection Examples

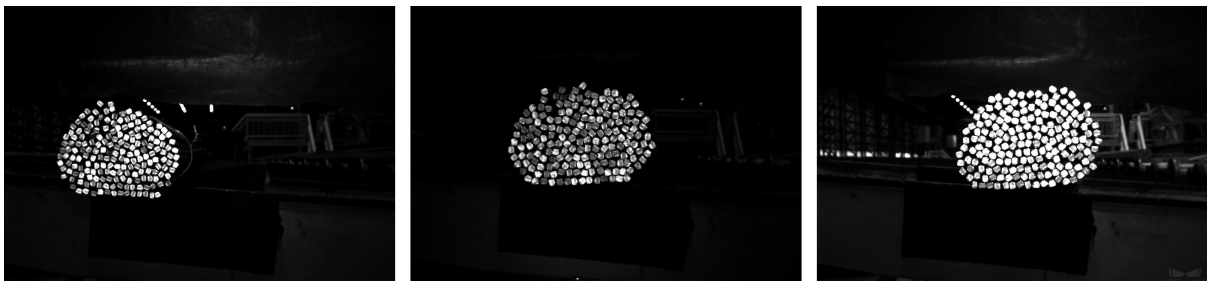
A workpiece inspection project

- The incoming objects are rotors scattered randomly.
- The project requires accurate detection of all rotor positions.
- 30 images were collected.
 - **Positions:** In the actual application, the rotors may be in any position in the bin, and the quantity will decrease after picking each time.
 - **Positional relationships:** The rotor may come scattered, neatly placed, or overlapped.



A steel bar counting project

- Steel bars have relatively simple features, so only the variations of **object positions** need to be considered. Images in which steel bars are in any position in the camera's field of view were captured.



Select the Right Dataset

1. Control dataset image quantities

For the first-time model building of the “Object Detection” module, capturing 20 images is recommended.

It is not true that the larger the number of images the better. Adding a large number of inadequate images in the early stage is not conducive to the later model improvement, and will make the training time longer.

2. Collect representative data

Dataset image capturing should consider all the conditions in terms of illumination, color, size, etc. of the objects to recognize.

- **Lighting:** Project sites usually have environmental lighting changes, and the datasets should contain images with different lighting conditions.
- **Color:** Objects may come in different colors, and the datasets should contain images of objects of all the colors.
- **Size:** Objects may come in different sizes, and the datasets should contain images of objects of all existing sizes.

Attention: If the actual on-site objects may be rotated, scaled in images, etc., and the corresponding image datasets cannot be collected, the datasets can be supplemented by adjusting the data augmentation training parameters to ensure that all on-site conditions are included in each dataset.

3. Balance data proportion

The number of images of different conditions/object classes in the datasets should be proportioned according to the actual project; otherwise, the training effect will be affected.

4. Dataset images should be consistent with those from the application site

The factors that need to be consistent include lighting conditions, object features, background, field of view, etc.

7.3.3 Ensure Labeling Quality

Labeling quality should be ensured in terms of completeness and accuracy.

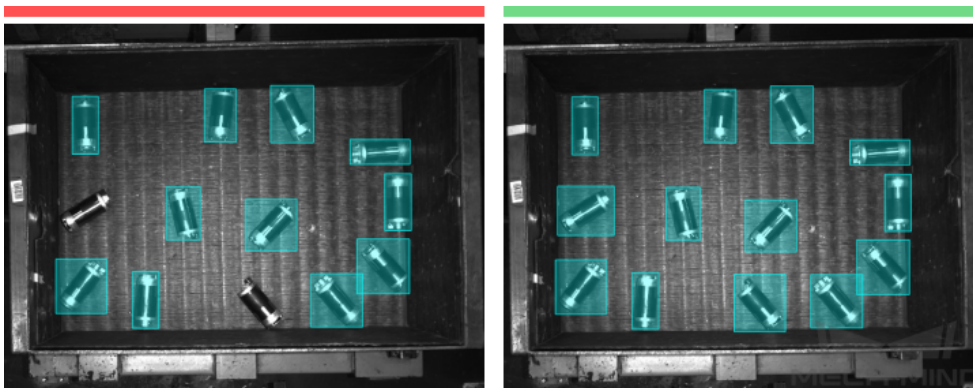
1. **Completeness:** Label all objects that meet the rules, and avoid missing any objects or object parts.
 - **Left image, bad example:** Omit objects that should be labeled.
 - **Right image, good example:** Label all objects.



Please make sure the camera height stays the same when capturing the training data and when deploying the project.

2. **Accuracy:** Each rectangular selection should contain the entire object. Please avoid missing any object parts, or including excess regions outside the object contours.

- **Left image, bad example:** Include multiple objects or incomplete objects in one selection.
- **Right image, good example:** Each selection corresponds to one object.

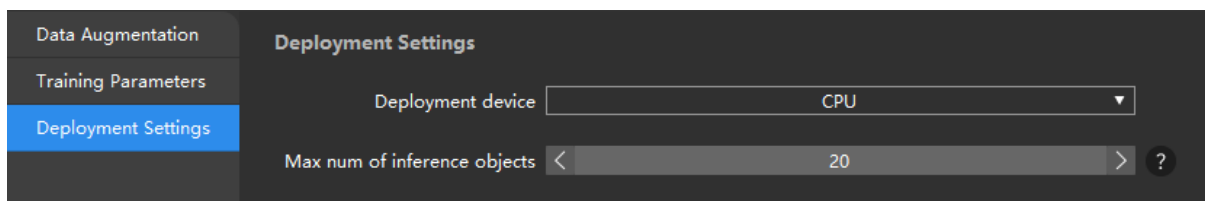


Please do not omit any object.

7.4 CPU and GPU Model Deployment

7.4.1 CPU Model Deployment

If the model is to be deployed on a CPU device, please select **CPU** for **Deployment device**.

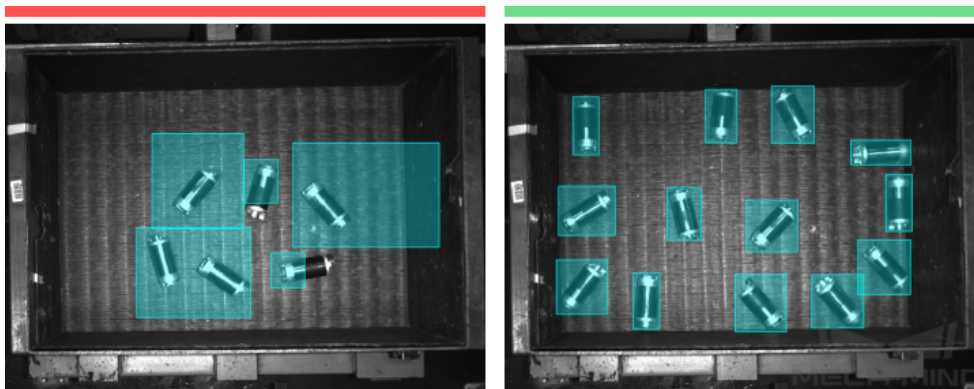
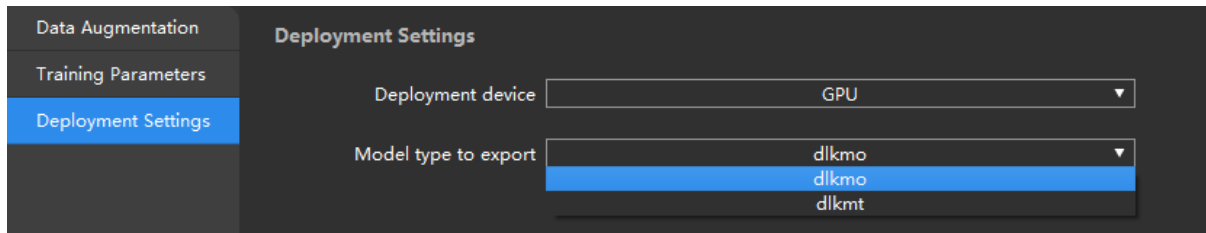


7.4.2 GPU Model Deployment

If the model is to be deployed on a GPU device, please select **GPU** for **Deployment device**.

For **Model type to export**, please select **dlkmt** only when the model training and deployment are done on GPU graphics cards of the same model.

Otherwise, please select **dlkmo** to avoid the problem that the model file cannot be used due to the difference in graphics card models for training and deployment.



Please do not include unnecessary regions or omit necessary object parts.

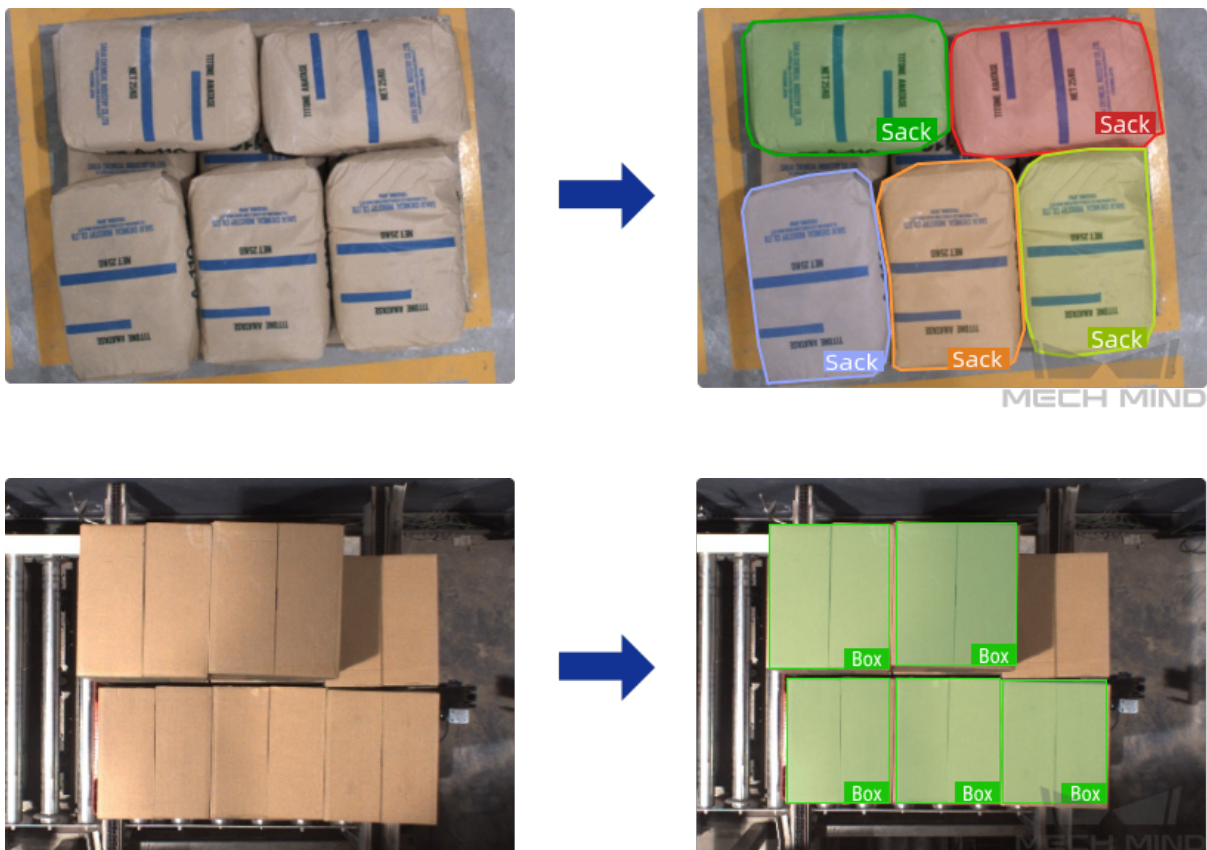
INSTANCE SEGMENTATION

8.1 Introduction

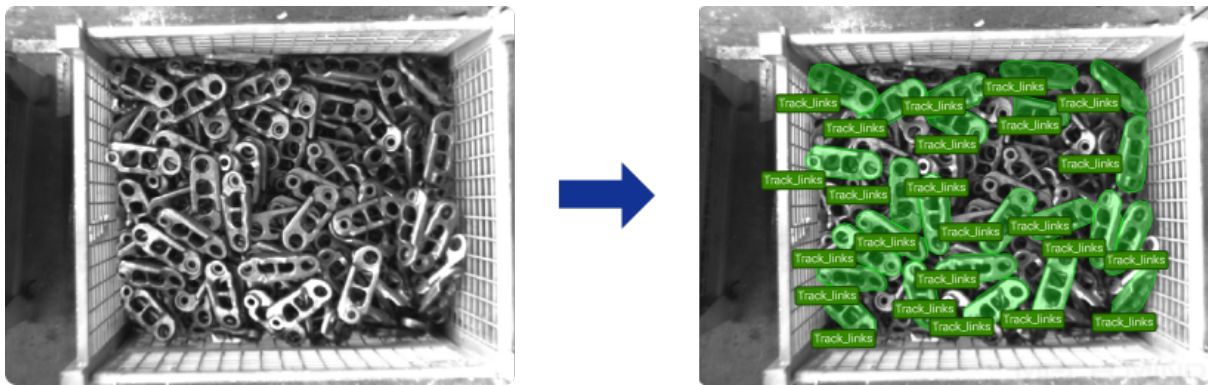
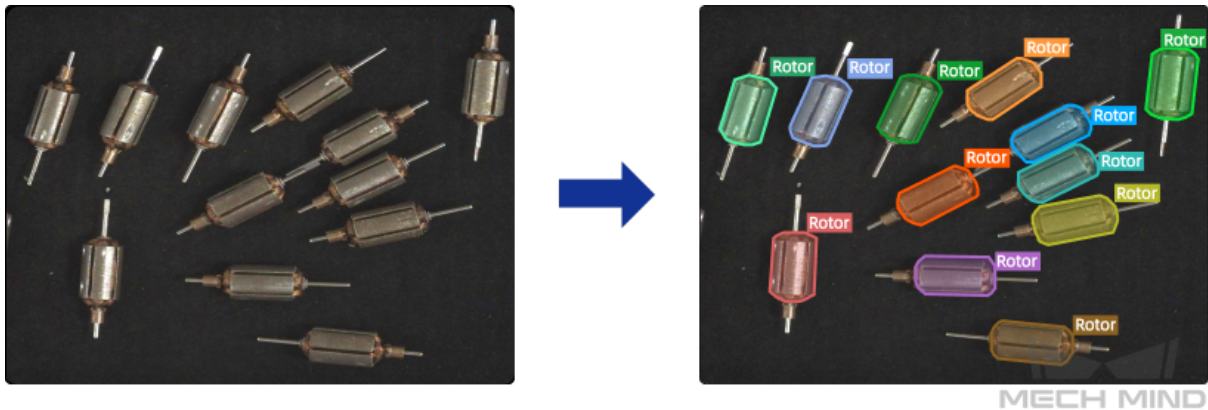
This module is used to segment the contour of target objects and output the corresponding labels of the classes.

8.1.1 Applicable Scenarios

Depalletizing: Applicable to scenarios where cartons, boxes, or sacks need to be depalletized from the pallets and then placed elsewhere (such as a bag break station or conveyor belt).

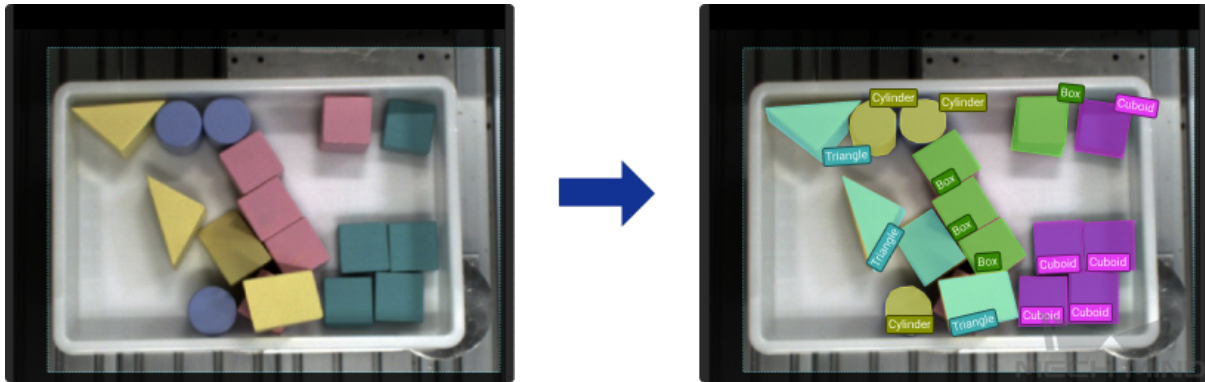


Machine tending: Applicable to machine tending of complex workpieces, structural parts, irregular parts in automobile, steel, machinery, and other industries.

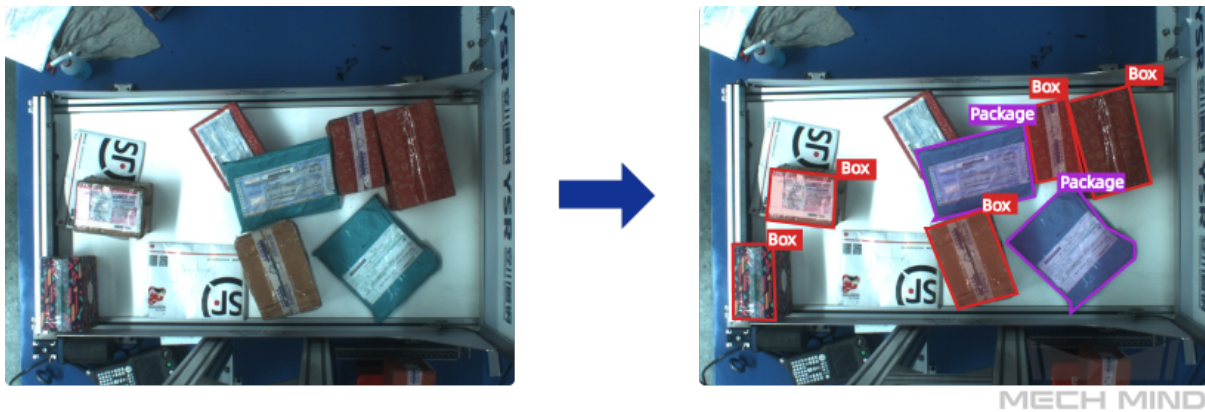


Product picking: Applicable to batch picking, discrete order picking, etc., in the e-commerce industry. This module is capable of processing various product packaging such as airbags, transparent packaging, aluminum cans, packaging of irregular shapes, etc.

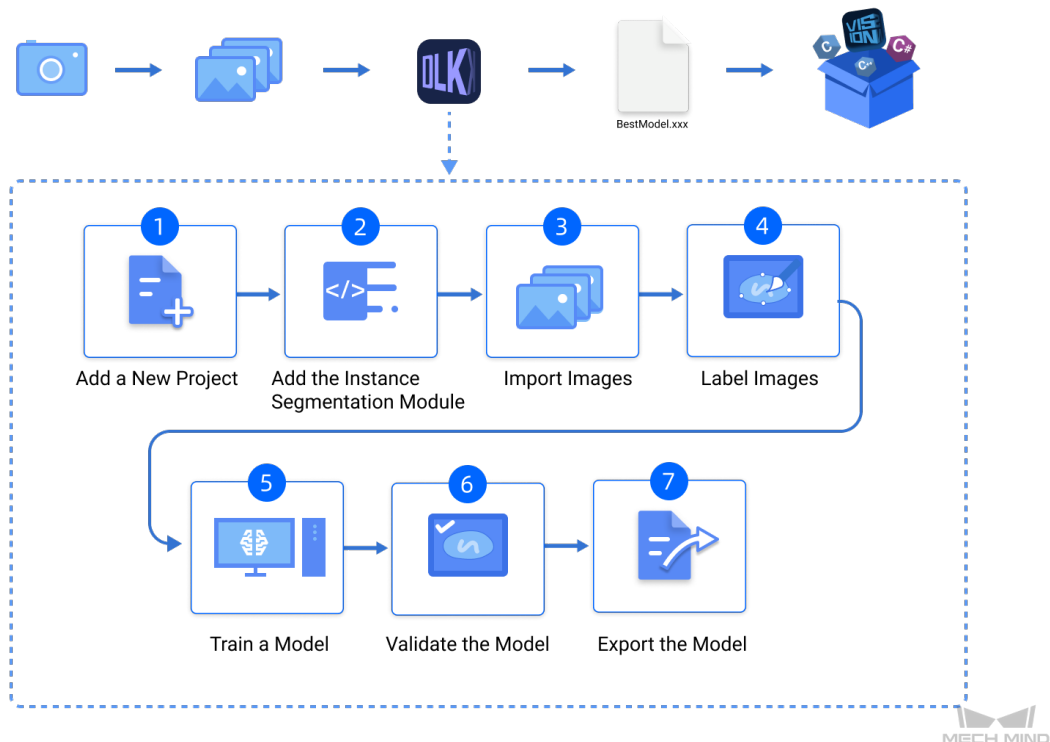




Express parcels: Suitable for detecting parcels of different shapes such as soft packages, postal envelopes, express cartons, padded envelopes, etc.



8.1.2 General Workflow



8.1.3 Tips


The performance of the trained model depends on the followings.

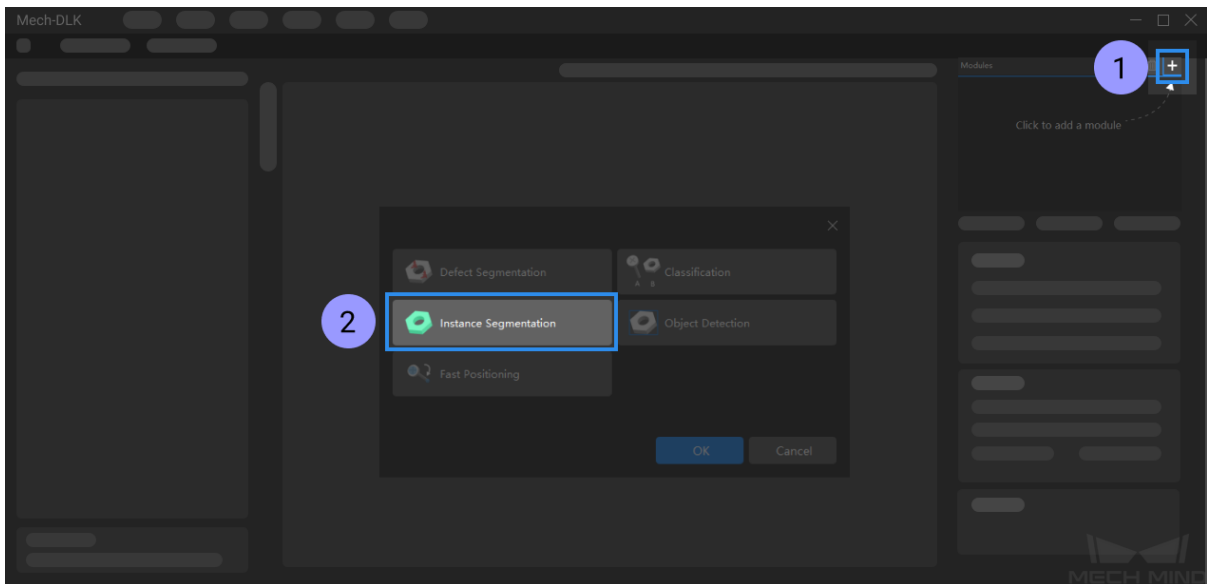
1. *The quality of the captured images.*
2. *The quality of the datasets.*
3. *The quality of labeling.*

8.2 Start Using the “Instance Segmentation” Module

Please click [here](#) to download an image dataset of wooden blocks. In this section, we will use an **Instance Segmentation** module and train a model to segment different types of wooden blocks and export the corresponding labels.

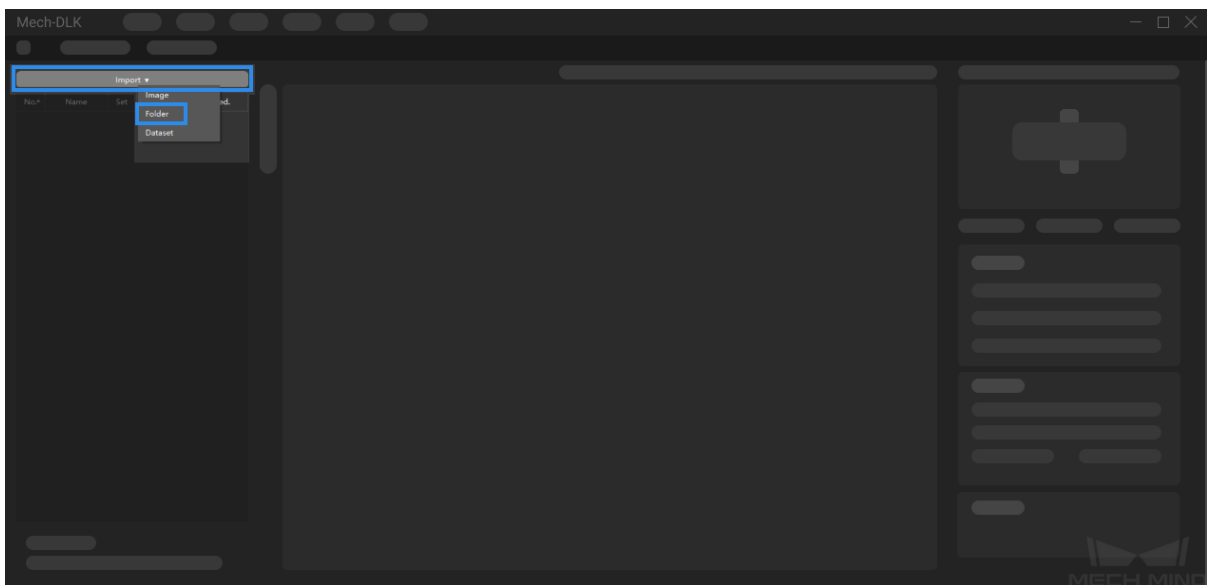
1. Create a new project and add the instance segmentation module

Click on *New Project* in the interface, name the project, and select a directory to save the project. Click on  in the upper right corner of the **Modules** panel and add the **Instance Segmentation** module.




2. Import the image dataset of wooden blocks

Decompress the downloaded dataset file. Click on the *Import* button in the upper left corner, select **Folder**, and import the image dataset. The wooden blocks in the images are of four different shapes and colors.




3. Select an ROI

Click on the ROI Tool button  and adjust the frame to select the bin containing wooden blocks in the image as an ROI, and click on *Apply* to save the settings. Setting the ROI can avoid interferences from the background and reduce processing time.




4. Create Labels

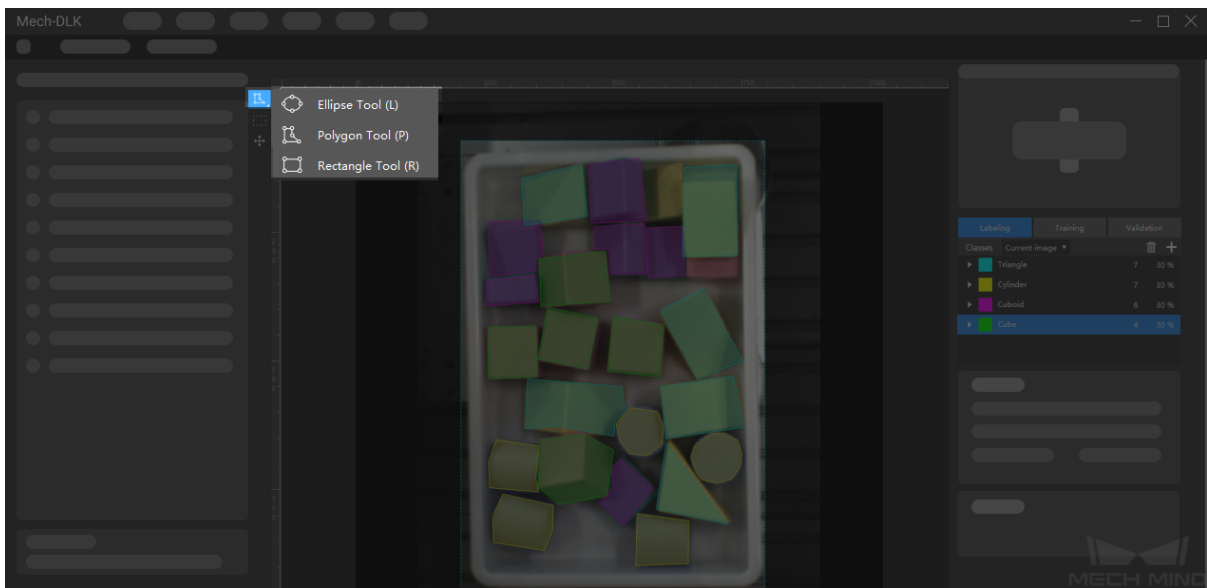
Select *Labeling* and click on the  button in the **Classes** panel to create labels based on the type or feature of different objects. In this example, the labels are named after the different shapes of the wooden blocks. You can also name the labels according to different colors.



5. Label images


Right-click on the  button and select a proper tool to label the image. In this example project, the contours of the wooden blocks need to be outlined for segmentation. In addition, please make sure that the different shapes of wooden blocks have been labeled

correctly.

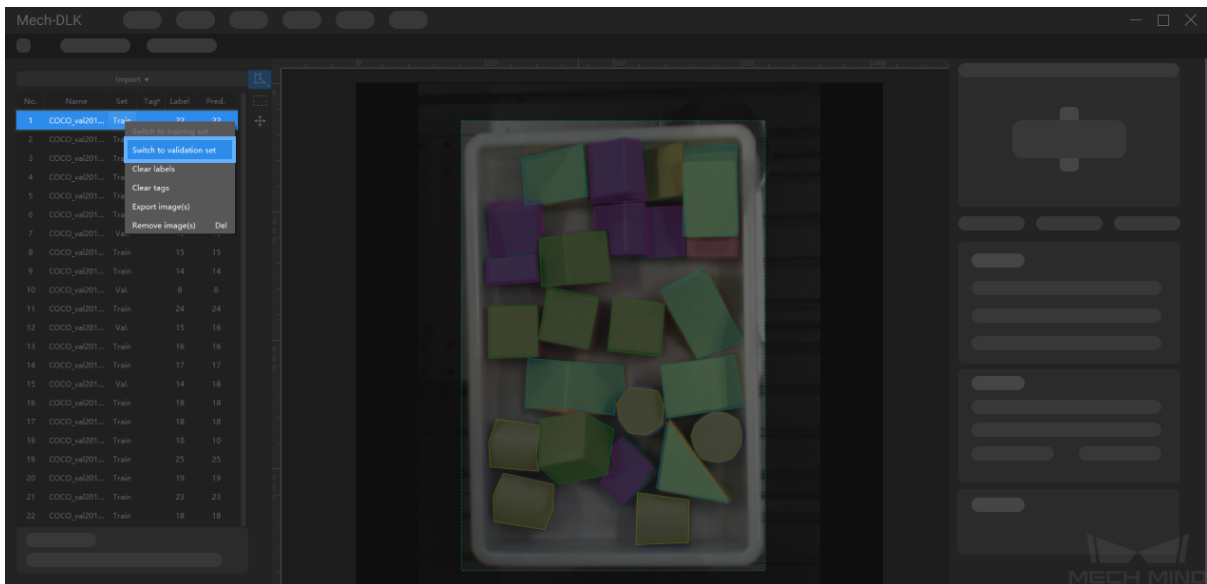


6. Split the dataset into the training set and validation set

By default, 80% of the images in the dataset will be split into the training set and the rest 20% will be split into the validation set. Please make sure that both the training set and validation set include **objects of all classes to be segmented**, which will guarantee that the algorithm can learn all different classes and validate the images properly.

If the default training set and validation set cannot meet this requirement, please click on  and drag the slider to adjust the proportion.

You can also right-click the individual image and switch it to the training/validation set manually.



7. Train the model

Keep the default training parameter settings and click on *Train* to start training the model.



8. Validate the model

After the training is completed, click on *Validate* to validate the model and check the results.



9. Export the model

Click on *Export* and select a directory to save the exported model (with file extension dlkpack). You can deploy the model according to actual needs.



8.3 Train a High-Quality Model

This section introduces several factors that most affect the model quality and how to train a high-quality image classification model.

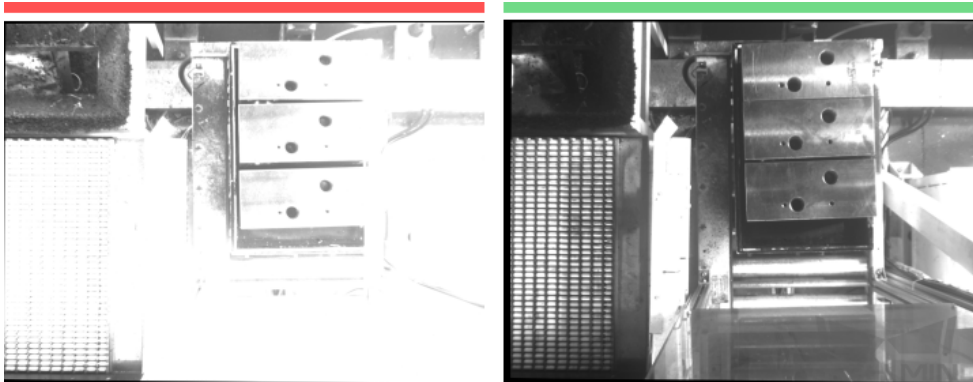
- *Ensure image quality*
- *Ensure dataset quality*
- *Ensure labeling quality*

8.3.1 Ensure Image Quality

1. Avoid **overexposure**, **dimming**, **color distortion**, **blur**, **occlusion**, etc. These conditions will lead to the loss of features that the deep learning model relies on, which will affect the model training effect.

- **Left image, bad example:** Overexposure.
- **Right image, good example:** Adequate exposure.

- **Left image, bad example:** Dim image.
- **Right image, good example:** Adequate exposure.



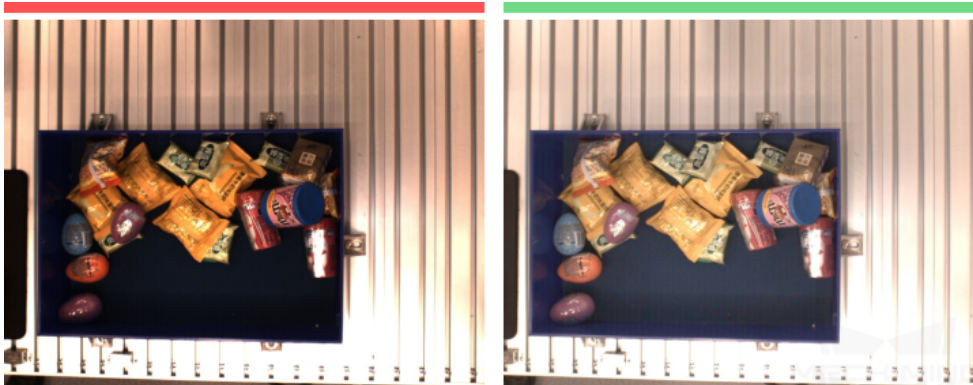
You can avoid overexposure by methods such as shading.

- **Left image, bad example:** Color distortion.
- **Right image, good example:** Normal color.



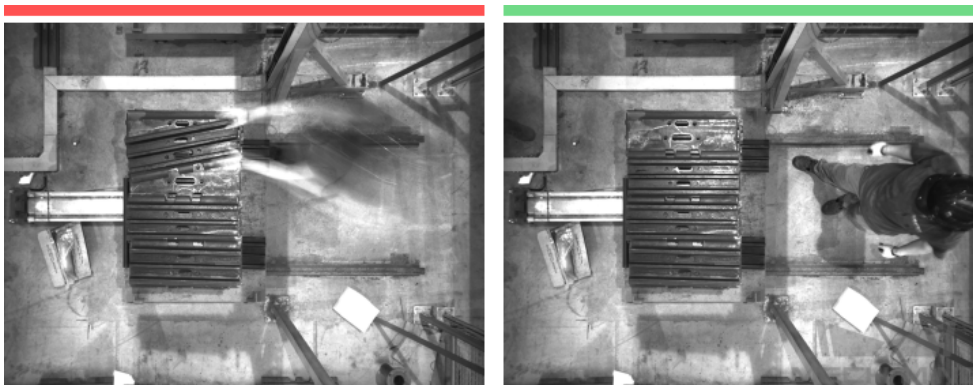
You can avoid dimming by methods such as supplementary light.

- **Left image, bad example:** Blur.
- **Right image, good example:** Clear.



Color distortion can be avoided by adjusting the white balance.

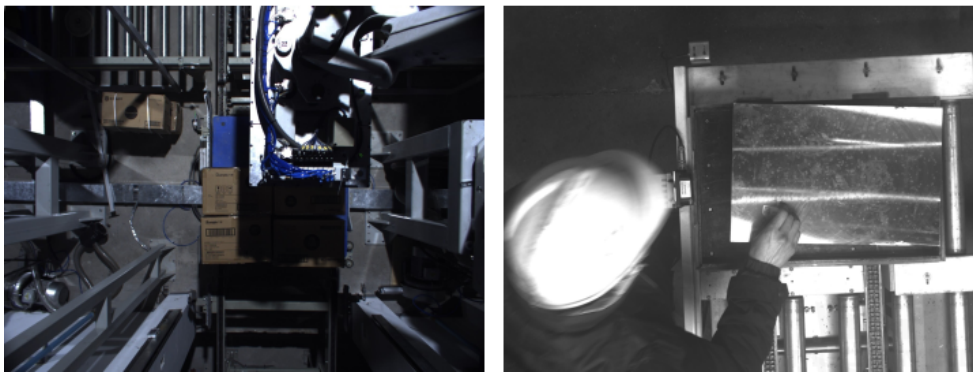
- **Left image, bad example:** Occluded by the robot arm.
- **Right image, good example:** Occluded by a human.



Please avoid capturing images when the camera or the objects are still moving.

2. Ensure that the **background, perspective, and height** of the image capturing process are consistent with the actual application. Any inconsistency will reduce the effect of deep learning in practical applications. In severe cases, data must be re-collected. Please confirm the conditions of the actual application in advance.

- **Bad example:** The background in the training data (left) is different from the background in the actual application (right).



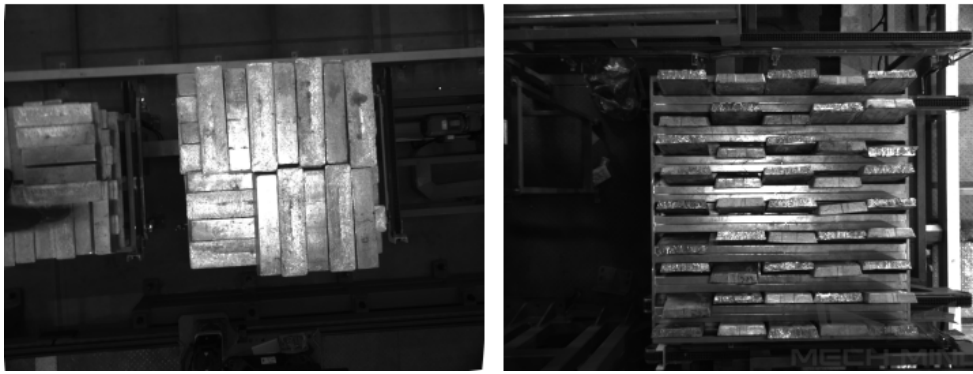
Please make sure there is no robot or human in the way from the camera to the objects.

- **Bad example:** The field of view and perspective in the training data (left) are different from that in the actual application (right).



Please make sure the background stays the same when capturing the training data and when deploying the project.

- **Bad example:** The camera height in the training data (left) is different from the background in the actual application (right).



Please make sure the field of view and perspective stay the same when capturing the training data and when deploying the project.

8.3.2 Ensure Dataset Quality

An instance segmentation model is trained by learning the features of the objects in the image. Then the model applies what is learned in the actual applications.

Therefore, to train a high-quality model, the conditions of the collected and selected dataset must be consistent with those of the actual applications.

1. *Collect Datasets*
2. *Select the Right Dataset*

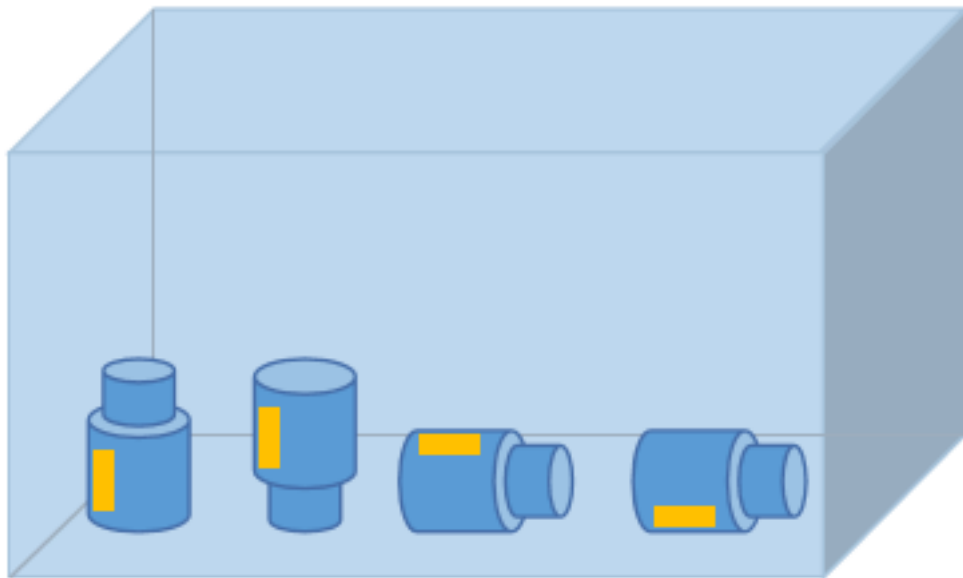
Collect Datasets

Various placement conditions need to be properly allocated. For example, if there are horizontal and vertical incoming materials in actual production, but only the data of horizontal incoming materials are collected for training, the classification effect of vertical incoming materials cannot be guaranteed.

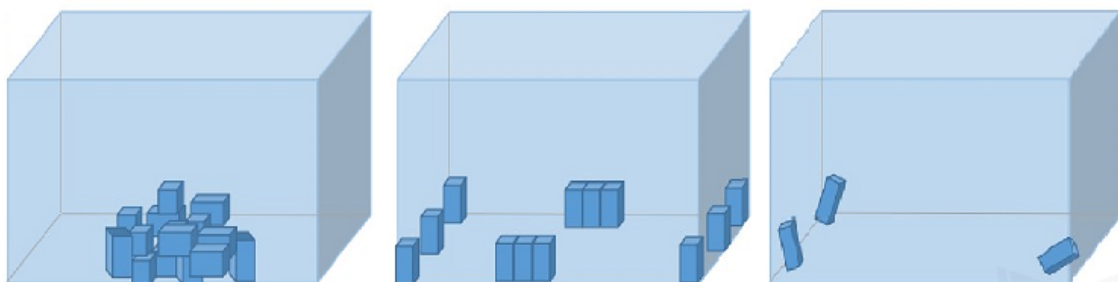
Therefore, when collecting data, it is necessary to **consider various conditions** of the actual application, including the features present given different object placement **orientations**, **positions**, and **positional relationships between objects**.

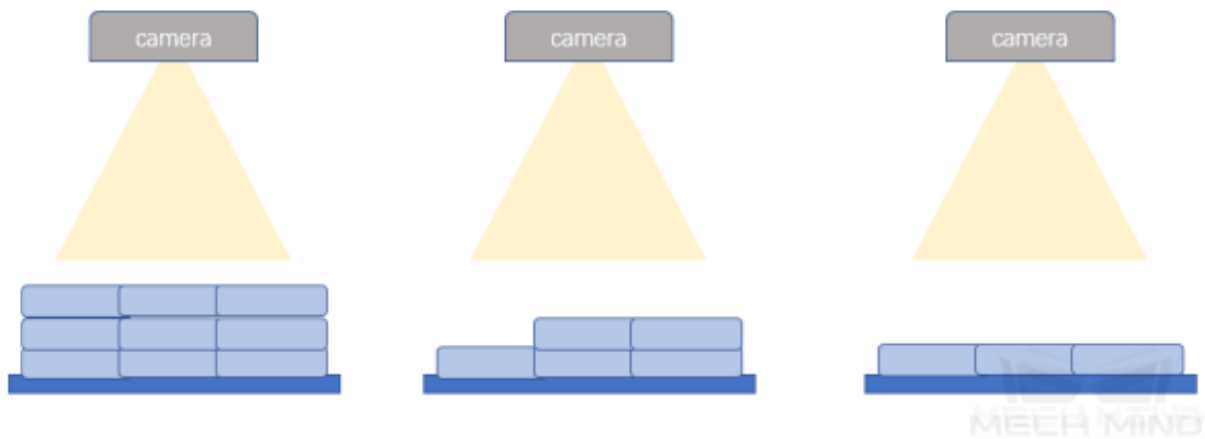
Attention: If some situations are not in the datasets, the deep learning model will not go through inadequate learning on the corresponding features, which will cause the model to be unable to effectively make recognitions given such conditions. In this case, data on such conditions must be collected and added to reduce the errors.

Orientations

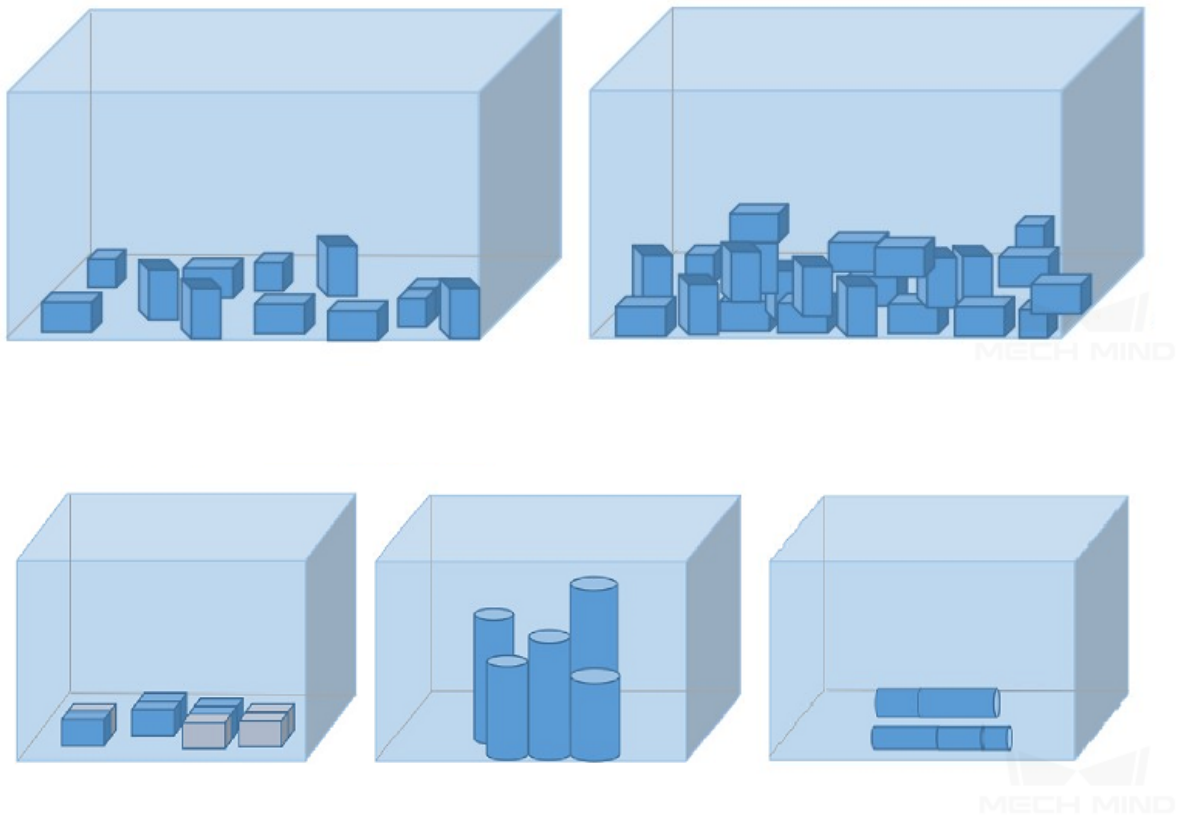


Positions





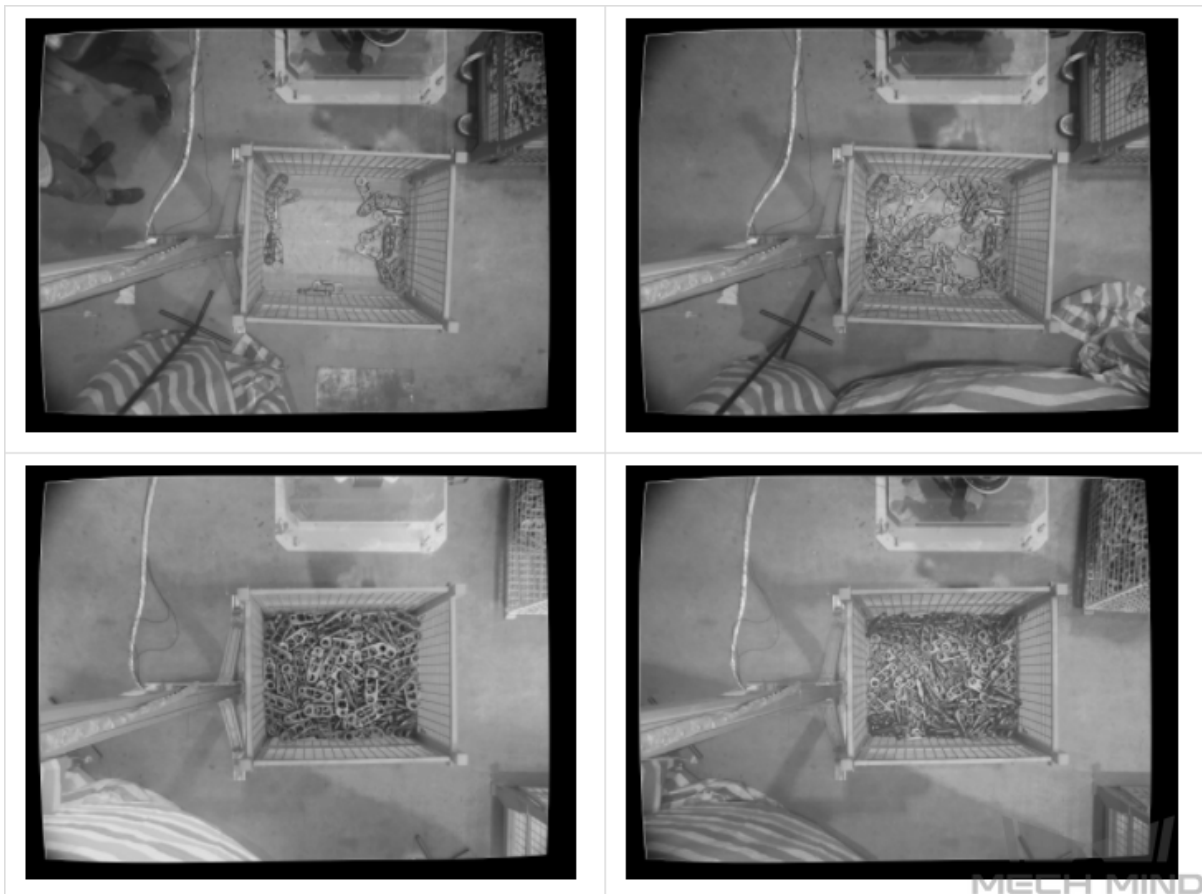
Positional relationships between objects



Data Collection Examples

A metal piece project

- Single object class.
- 50 images were collected.
- Object placement conditions of lying down and standing on the side need to be considered.
- Object positions at the bin center, edges, corners, and at different heights need to be considered.
- Object positional relationships of overlapping and parallel arrangement need to be considered.
- Samples of the collected images are as follows.





A grocery project

- 7 classes of objects are mixed and classification is required.
- The following situations need to be considered to fully capture object features.
 - Situation #1: objects of one class placed in different orientations
 - Situation #2: mixing objects of multiple classes

Number of images for situation #1: $5 * \text{number of object classes}$.

Number of images for situation #2: $20 * \text{number of object classes}$.

- The objects may come lying flat, standing on sides, or reclining, so images containing all faces of the objects need to be considered.
- The objects may be in the center, on the edges, and in the corners of the bins.
- The objects may be placed parallelly or fitted together.

Samples of the collected images are as follows:

Placed alone



Mixedly placed



A track shoe project

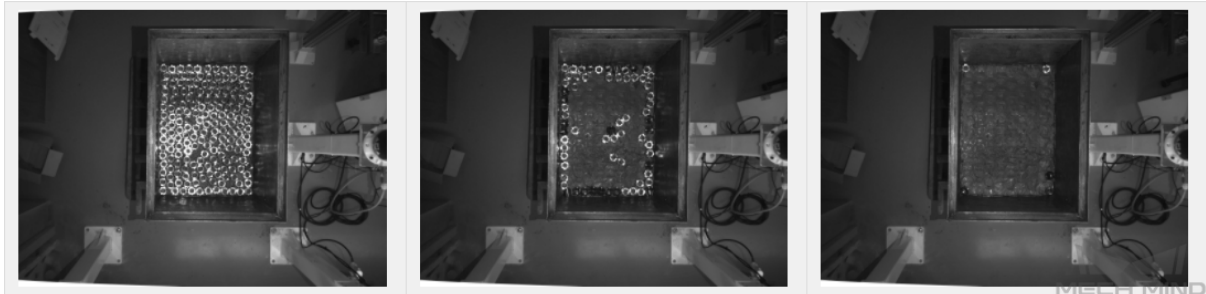
- The track shoes come in many models.
- The number of images captured was $30 * \text{number of models}$.

- The track shoes only face up, so only the facing-up condition needs to be considered.
- The track shoes may be on different heights under the camera.
- The track shoes are arranged regularly together, so the situation of closely fitting together needs to be considered.
- Samples of the collected images are as follows:



A metal piece project

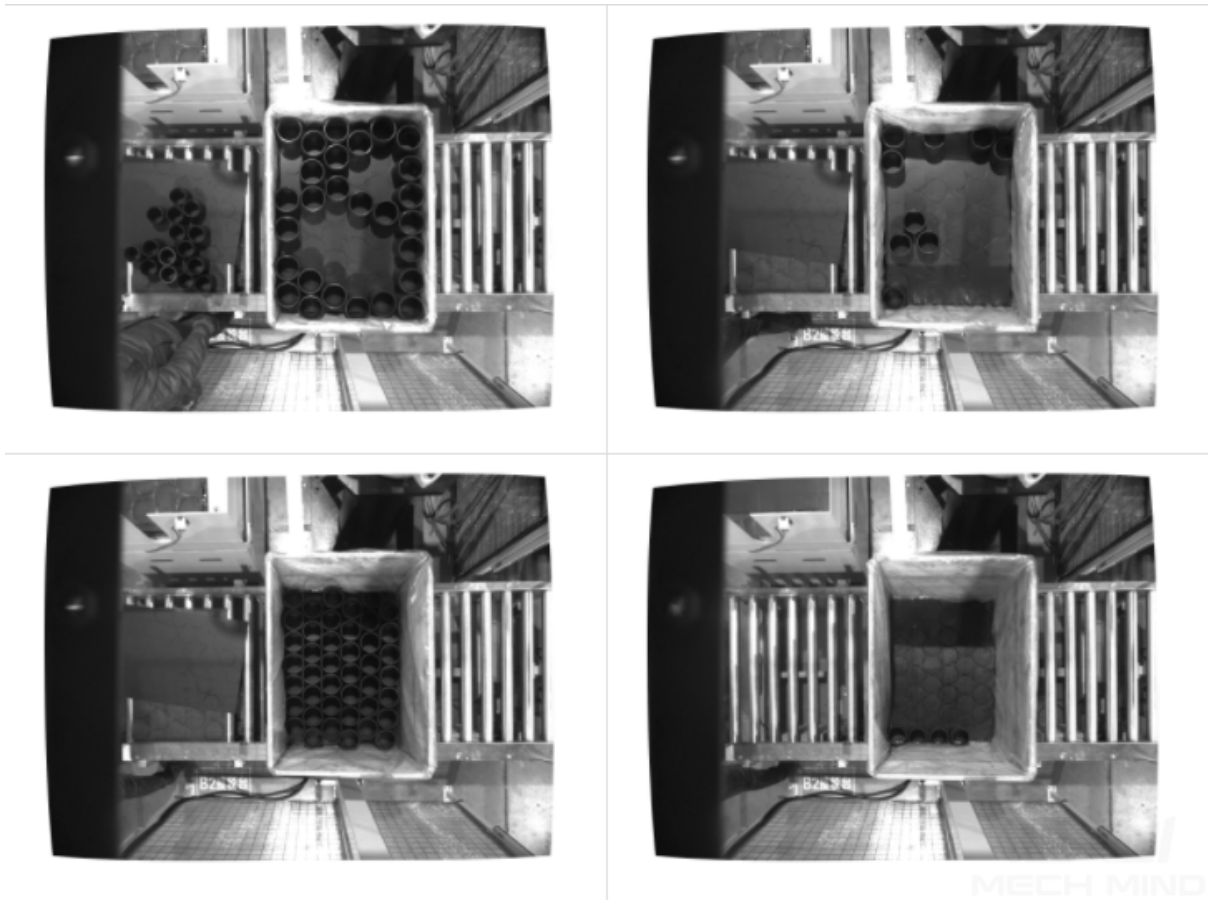
- Metal pieces are presented in one layer only. So only 50 images were captured.
- The metal pieces only face up.
- The metal pieces are in the center, edges, and corners of the bin.
- The metal pieces may be fitted closely together.
- Samples of the collected images are as follows:



A metal piece project

- Metal pieces are neatly placed in multiple layers.
- 30 images were collected.
- The metal pieces only face up.
- The metal pieces are in the center, edges, and corners of the bin and are on different heights under the camera.
- The metal pieces may be fitted closely together.

Samples of the collected images are as follows:



Select the Right Dataset

1. Control dataset image quantities

For the first-time model building of the “Instance Segmentation” module, capturing 30 images is recommended.

It is not true that the larger the number of images the better. Adding a large number of inadequate images in the early stage is not conducive to the later model improvement, and will make the training time longer.

2. Collect representative data

Dataset image capturing should consider all the conditions in terms of illumination, color, size, etc. of the objects to recognize.

- **Lighting:** Project sites usually have environmental lighting changes, and the datasets should contain images with different lighting conditions.
- **Color:** Objects may come in different colors, and the datasets should contain images of objects of all the colors.

- Size: Objects may come in different sizes, and the datasets should contain images of objects of all existing sizes.

Attention: If the actual on-site objects may be rotated, scaled in images, etc., and the corresponding image datasets cannot be collected, the datasets can be supplemented by adjusting the data augmentation training parameters to ensure that all on-site conditions are included in each dataset.

3. Balance data proportion

The number of images of different conditions/object classes in the datasets should be proportioned according to the actual project; otherwise, the training effect will be affected.

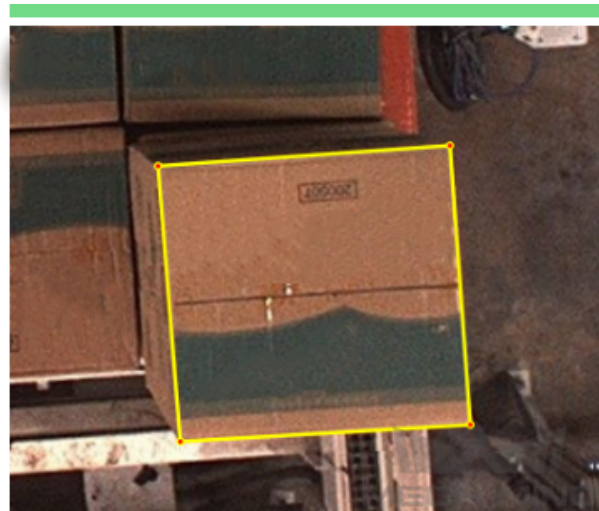
4. Dataset images should be consistent with those from the application site

The factors that need to be consistent include lighting conditions, object features, background, field of view, etc.

8.3.3 Ensure Labeling Quality

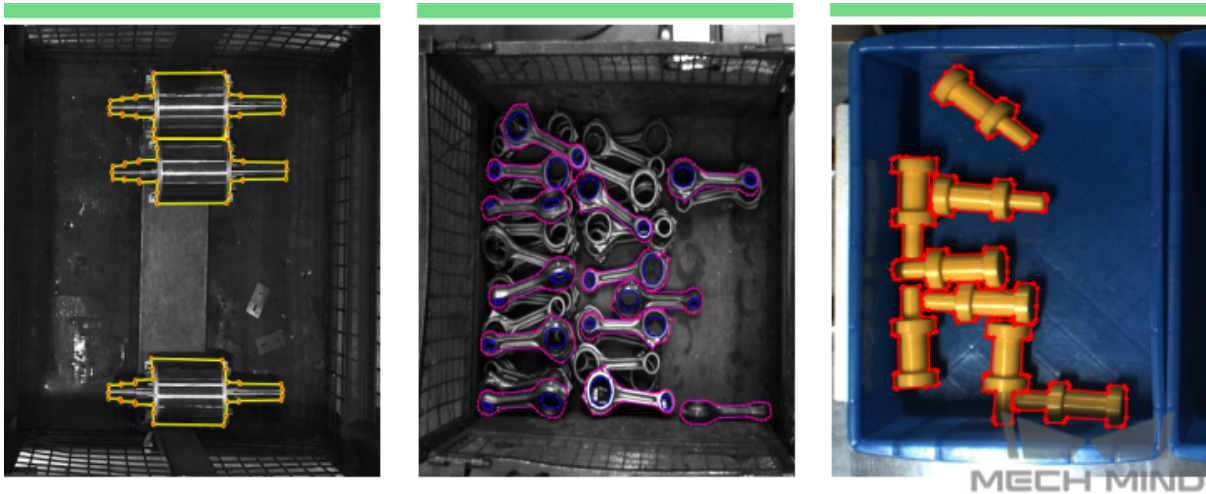
Determine the Labeling Method

1. **Label the upper surface' contour:** It is suitable for regular objects that are laid flat, such as cartons, medicine boxes, rectangular workpieces, etc., for which the pick points are calculated on the upper surface contour, and the user only needs to make rectangular selections on the images.
 - **Left image, bad example:** Select the entire box when only selecting the top surface is needed.
 - **Right image, good example:** Only select the top surface when necessary.



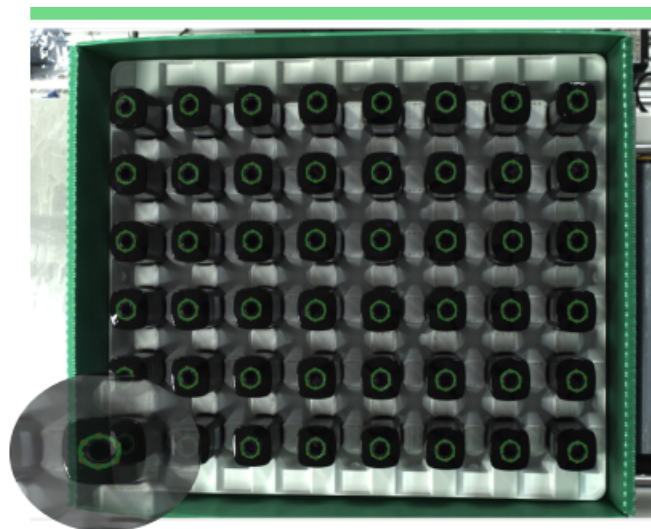
2. **Label the entire objects' contours:** It is suitable for sacks, various types of workpieces, etc., for which only labeling the object contours is the general method.

- **Good examples**



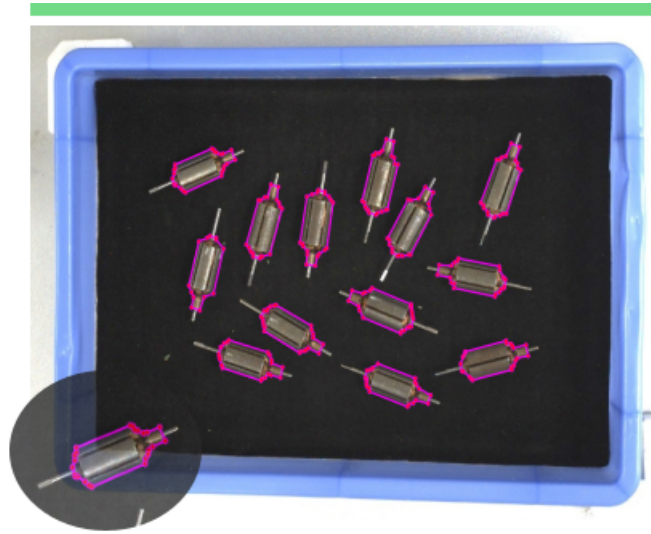
3. **Special cases:** For example, when the recognition result needs to conform to how the grippers work.

- It is necessary to ensure that the suction cup and the tip of the bottle to pick completely fit (high precision is required), and only the bottle tip contours need to be labeled.
 - **Good example:** Label bottle tips.



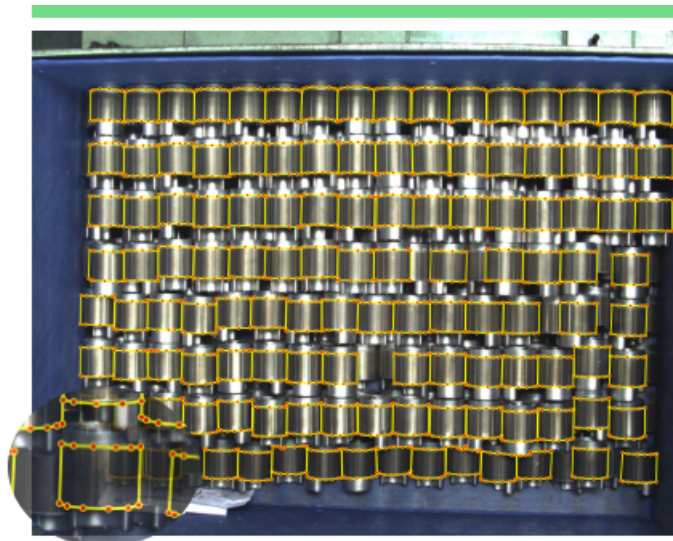
- The task of rotor picking involves recognizing rotor orientations. Only the middle parts whose orientations are clear can be labeled, and the thin rods at both ends cannot be labeled.

- **Good example:** Label middle parts of rotors.



- It is necessary to ensure that the suction parts are in the middle parts of the metal pieces, so only the middle parts of the metal pieces are labeled, and the ends do not need to be labeled.

- **Good example:** Label middle parts.

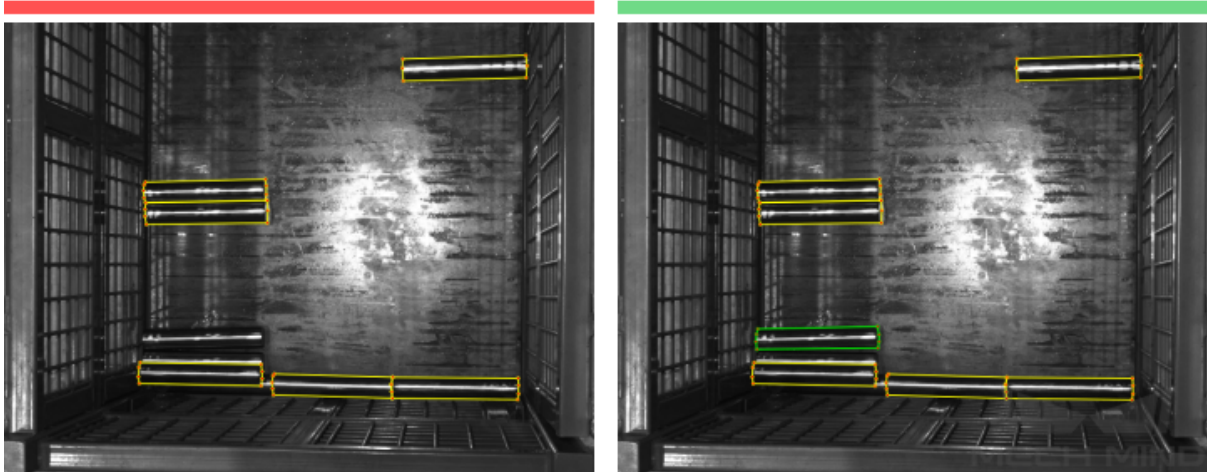


Check Labeling Quality

Labeling quality should be ensured in terms of completeness, correctness, consistency, and accuracy.

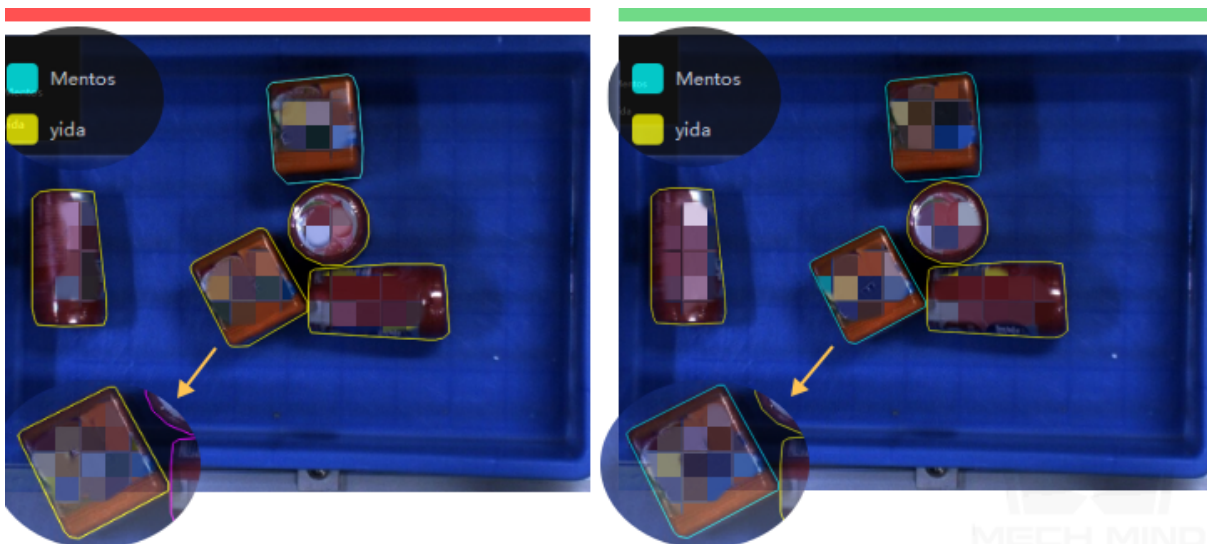
1. **Completeness:** Label all objects that meet the rules, and avoid missing any objects or object parts.

- **Left image, bad example:** Omit objects.
- **Right image, good example:** Label all objects.



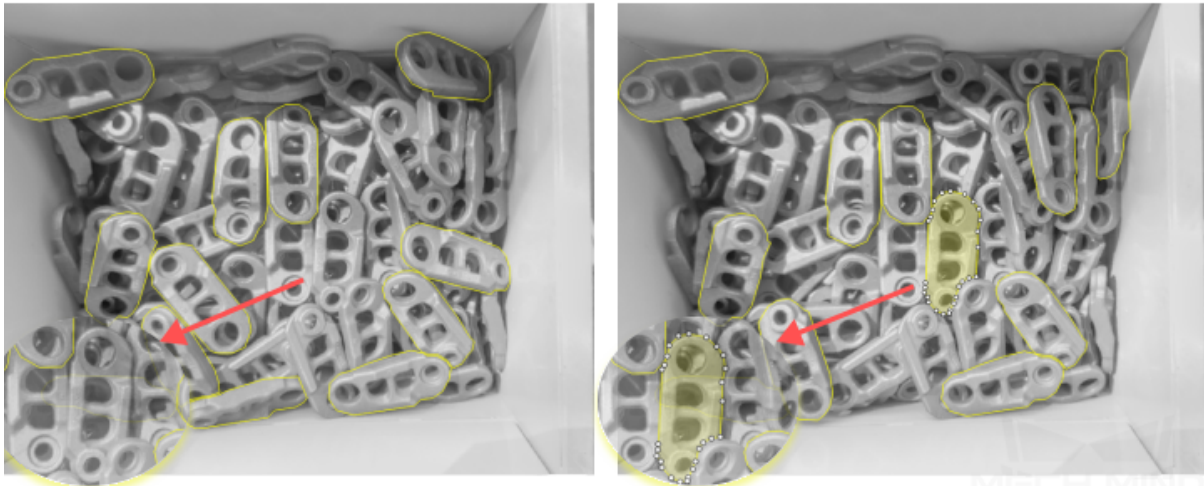
2. **Correctness:** Make sure that each object corresponds correctly to the label it belongs to, and avoid situations where the object does not match the label.

- **Left image, bad example:** Wrong label. A Mentos was labeled as a yida.
- **Right image, good example:** Correct labels.



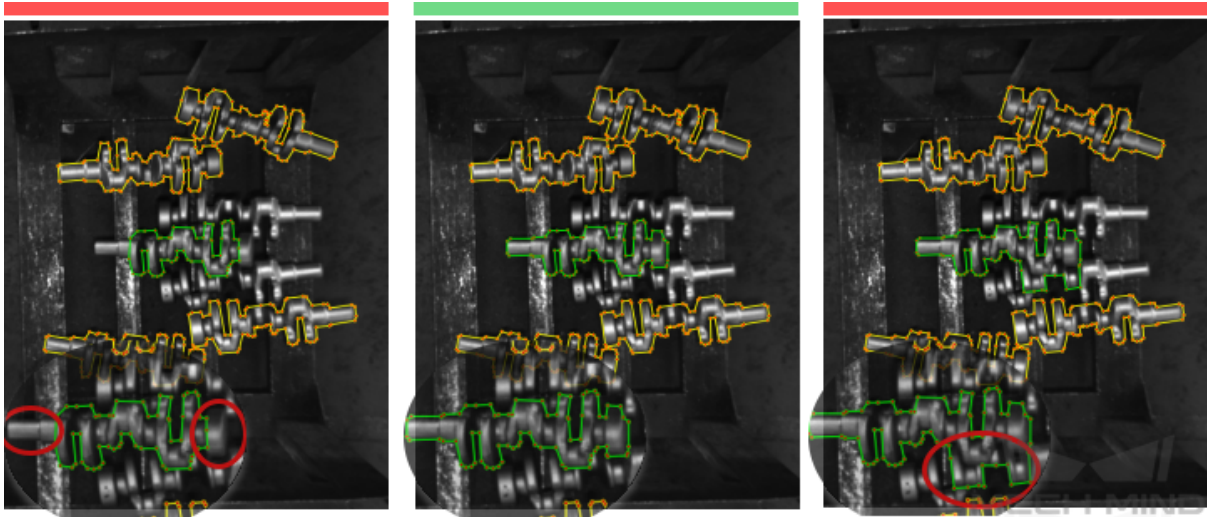
3. **Consistency:** All data should follow the same labeling rules. For example, if a labeling rule stipulates that only objects that are over 85% exposed in the images be labeled, then all objects that meet the rule should be labeled. Please avoid situations where one object is labeled but another similar object is not.

- **Bad example:** An object that should be labeled is labeled in one image but not labeled in another.



4. **Accuracy:** Make the region selection as fine as possible to ensure the selected regions' contours fit the actual object contours and avoid bluntly covering the defects with coarse large selections or omitting object parts.

- **Left image, bad example:** Omit object parts
- **Middle image:** Correct labeling selection.
- **Right image, good example:** Include parts of other objects in an object's selection.



8.4 CPU and GPU Model Training and Deployment

8.4.1 CPU Model Training and Deployment

If the model is to be deployed on a CPU device, it is recommended to select **Lite (better with CPU deployment)** .

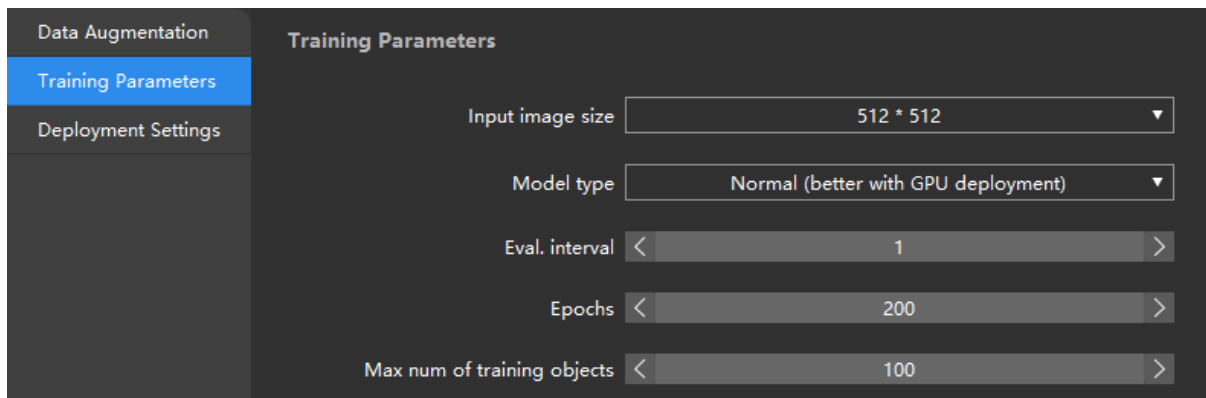
Data Augmentation	Training Parameters
Training Parameters	Input image size <input type="text" value="512 * 512"/>
Deployment Settings	Model type <input type="text" value="Lite (better with CPU deployment)"/>
	Eval. interval <input type="text" value="1"/>
	Epochs <input type="text" value="200"/>
	Max num of training objects <input type="text" value="100"/>

Select **CPU** for **Deployment device**.

Data Augmentation	Deployment Settings
Training Parameters	Deployment device <input type="text" value="CPU"/>
Deployment Settings	Max num of inference objects <input type="text" value="20"/>

8.4.2 GPU Model Training and Deployment

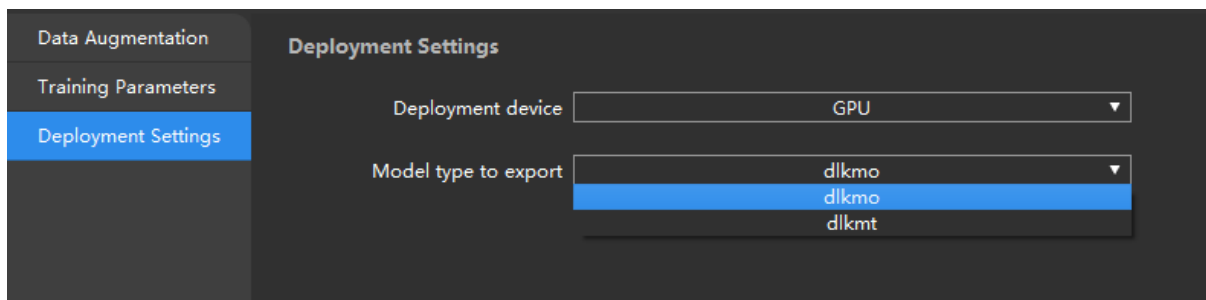
If the model is to be deployed on a GPU device, it is recommended to select **Normal (better with GPU deployment)**.



Please select **GPU** for **Deployment device**.

For **Model type to export**, please select **dlkmt** only when the model training and deployment are done on GPU graphics cards of the same model.

Otherwise, please select **dlkmo** to avoid the problem that the model file cannot be used due to the difference in graphics card models for training and deployment.



Please make sure the camera height stays the same when capturing the training data and when deploying the project.

MODULE CASCADING

The module cascading function is used to cascade two or more algorithm modules to train a model with multiple recognition functions.

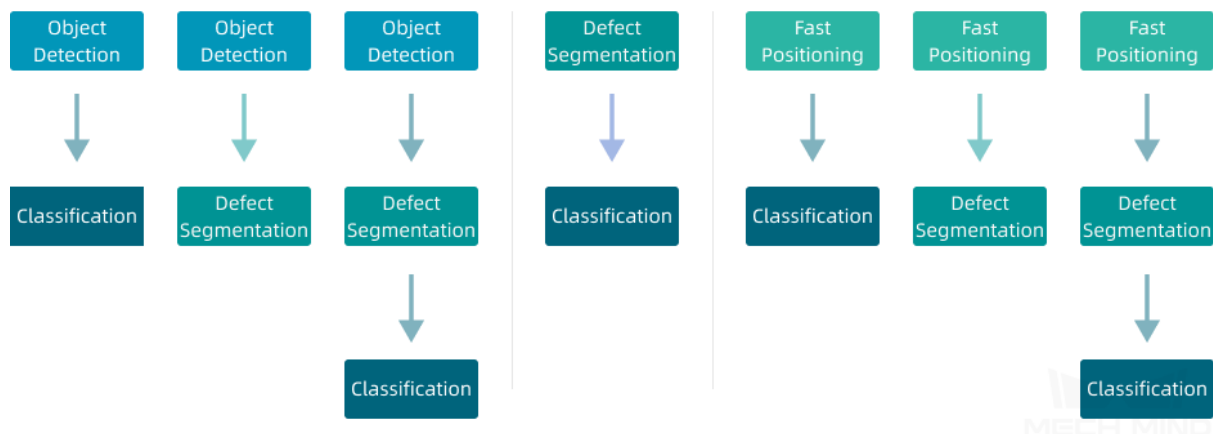
9.1 When to Use Module Cascading

When two or more identification requirements need to be met, module cascading can be used. For example, when both object defect segmentation and defect classification are required, the “Defect Segmentation” module and the “Classification” module can be cascaded.

With module cascading, there is no need to create multiple projects, which saves training and deployment time.

Please see *Train the First Model* for instructions on using module cascading.

9.2 Methods of Cascading



9.3 Create a Cascading Project

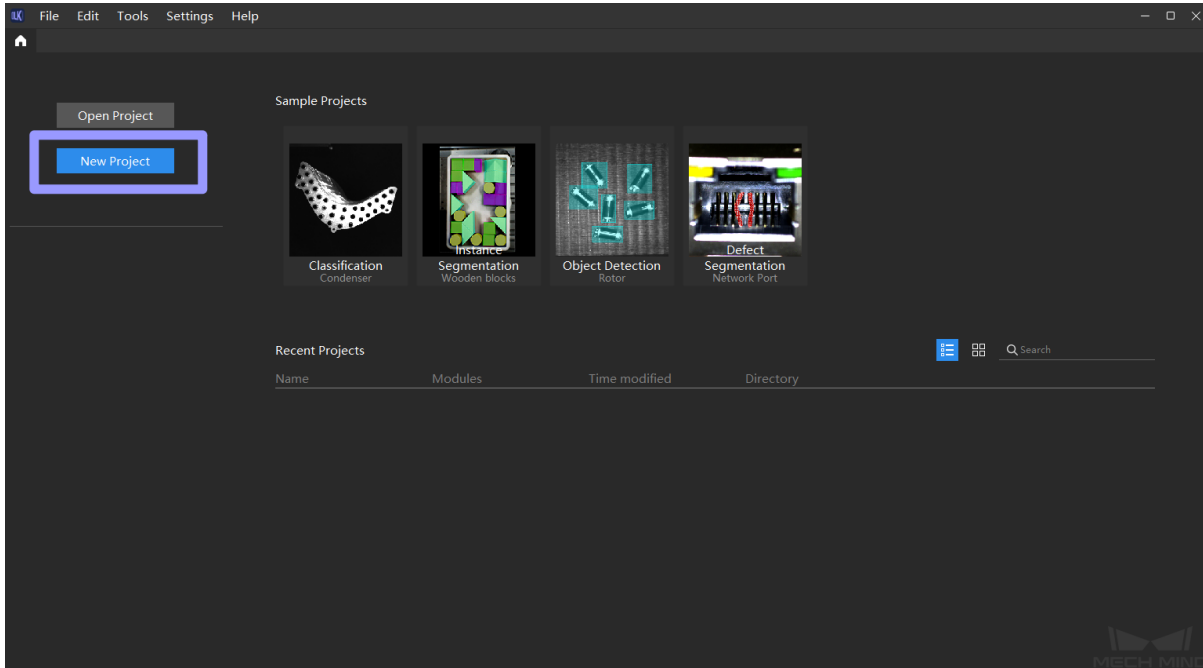
The following instructions show how to create a cascading project for **recognizing** object defects and **classifying** the defects.

The function is achieved by cascading two algorithm modules.

- *Defect Segmentation* recognizes defects in objects.
- *Classification* classifies the defects.


1. **Create the project**

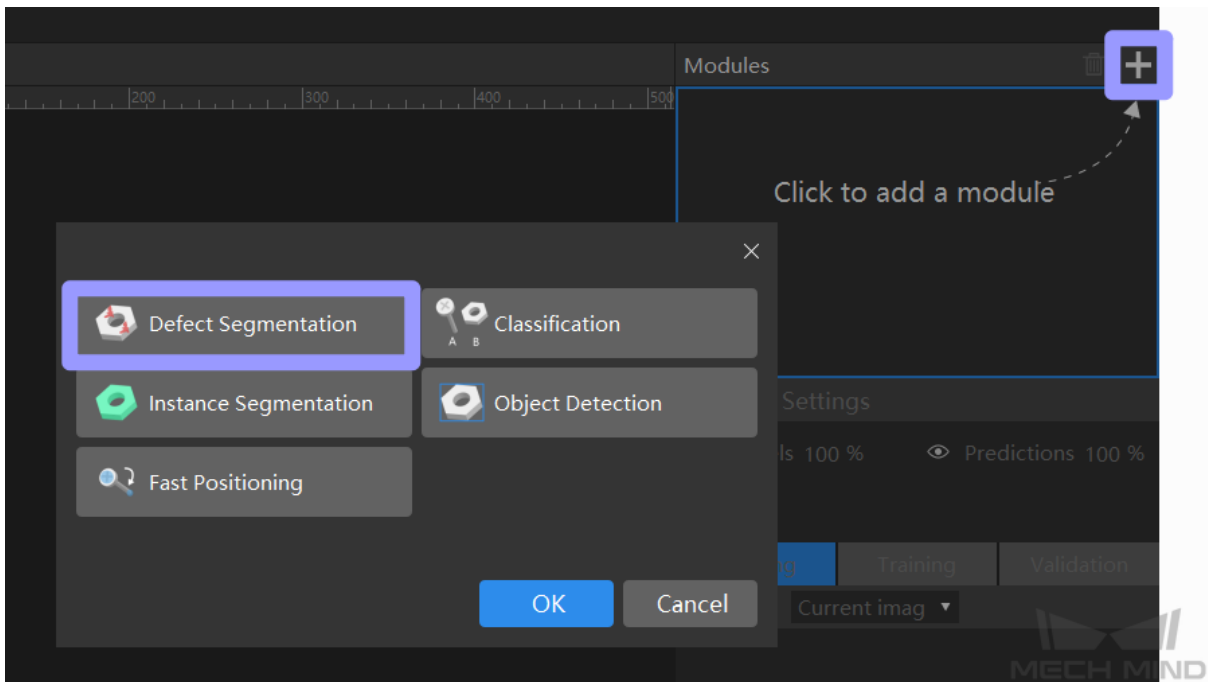
Click *New Project* on the main window, select the project path and enter the project name to create a new project.



2. **Add the “Defect Segmentation” module**

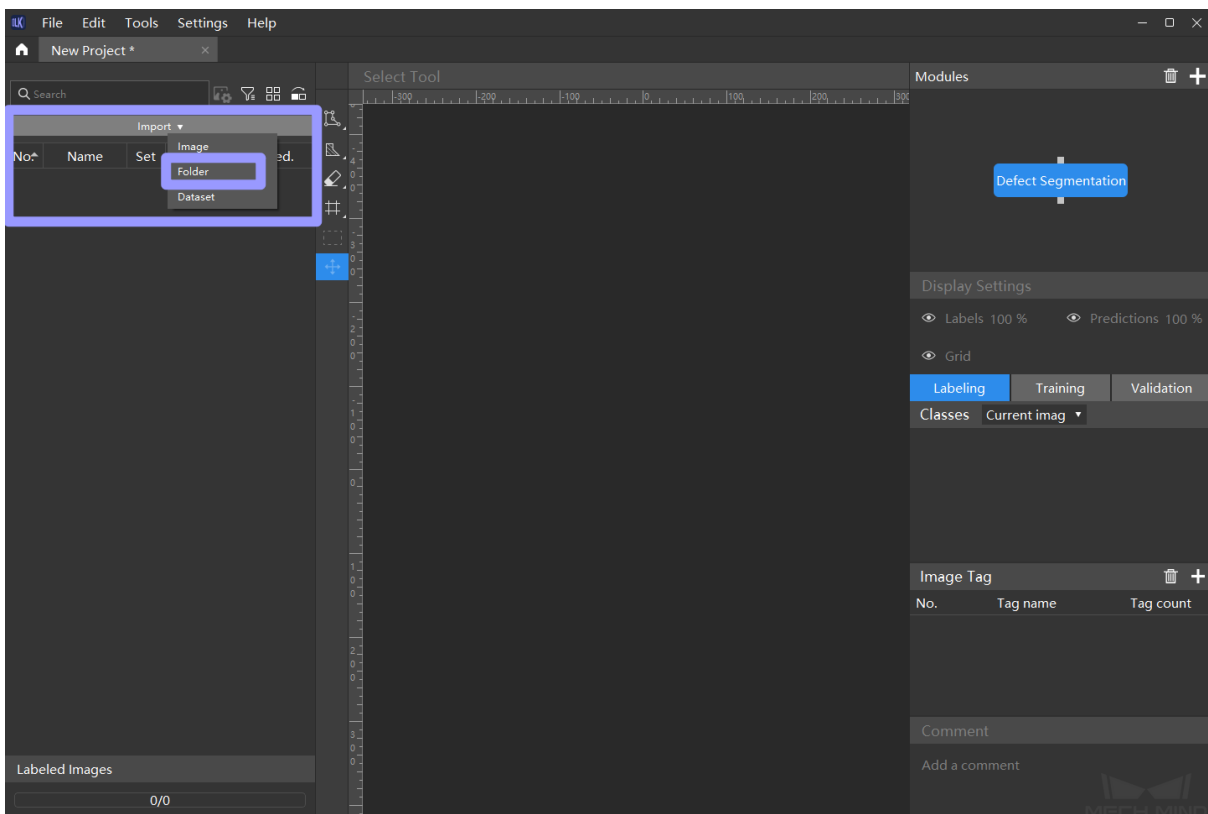


Click on  in the module section on the right part of the main window, select the *Defect Segmentation* module and click on *Confirm* .




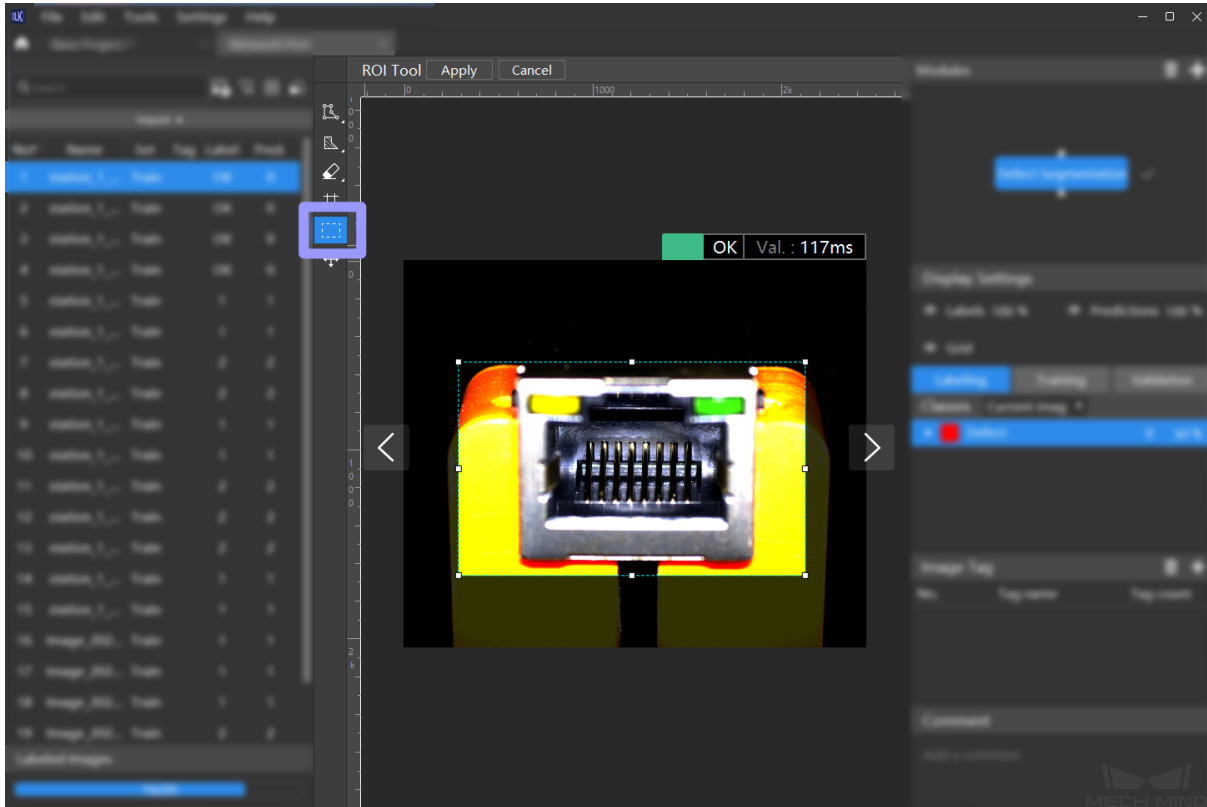
3. Import image data

Click *Import* in the upper left, and select how to import the image data.





4. Select ROI

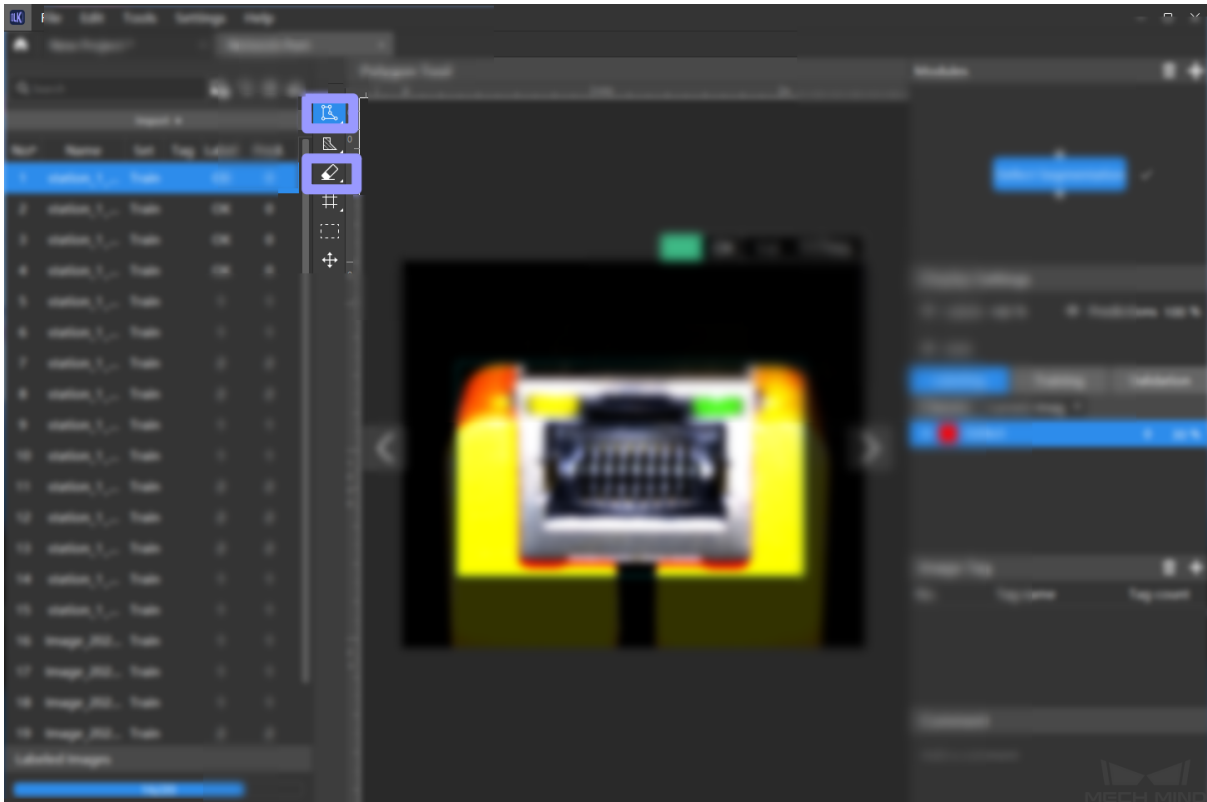
Click on  to select the region of interest from the image. The purpose is to reduce the interference of irrelevant background information and reduce the image processing time.



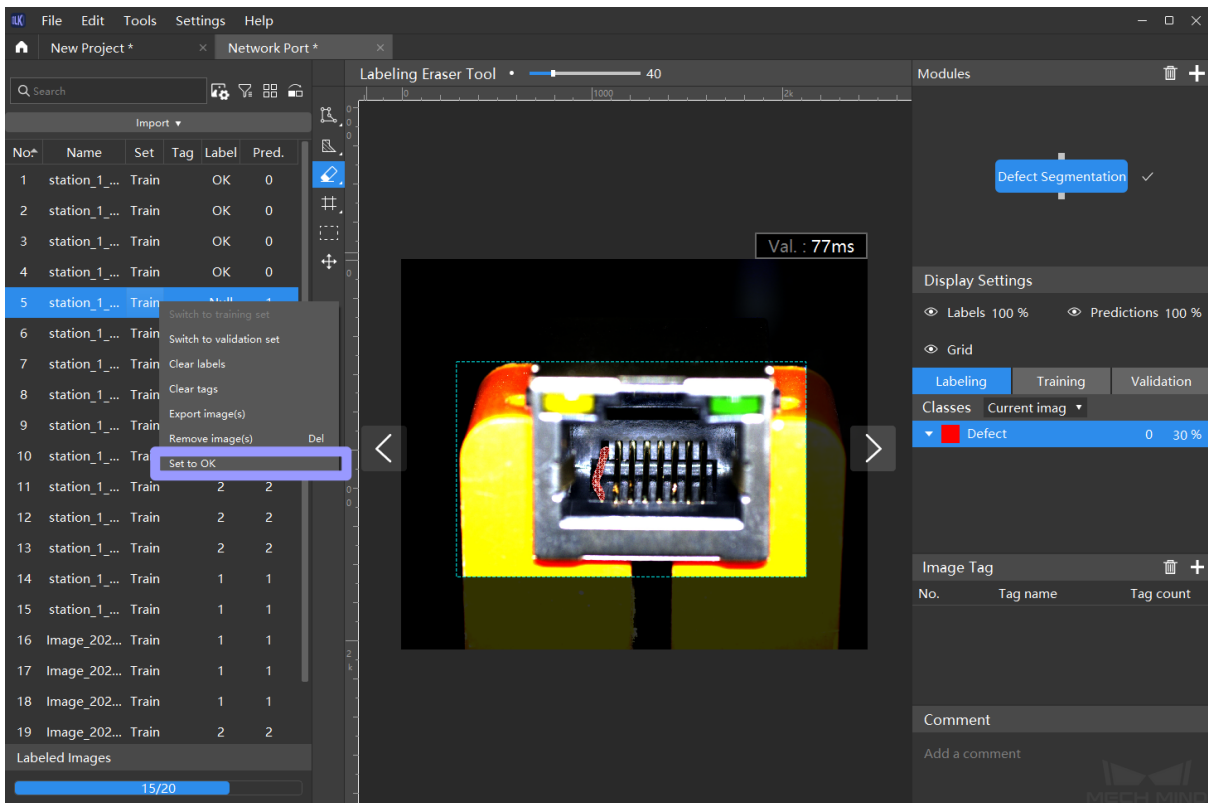
5. Label the images

Please label the OK images and the NG images that contain defects in each dataset. For NG images, please right-click on  on the left-side toolbar of the image, select the appropriate tools according to the shape of the defects, and select all defect regions in the images.

Click  to use the eraser tool for adjusting the selected regions.

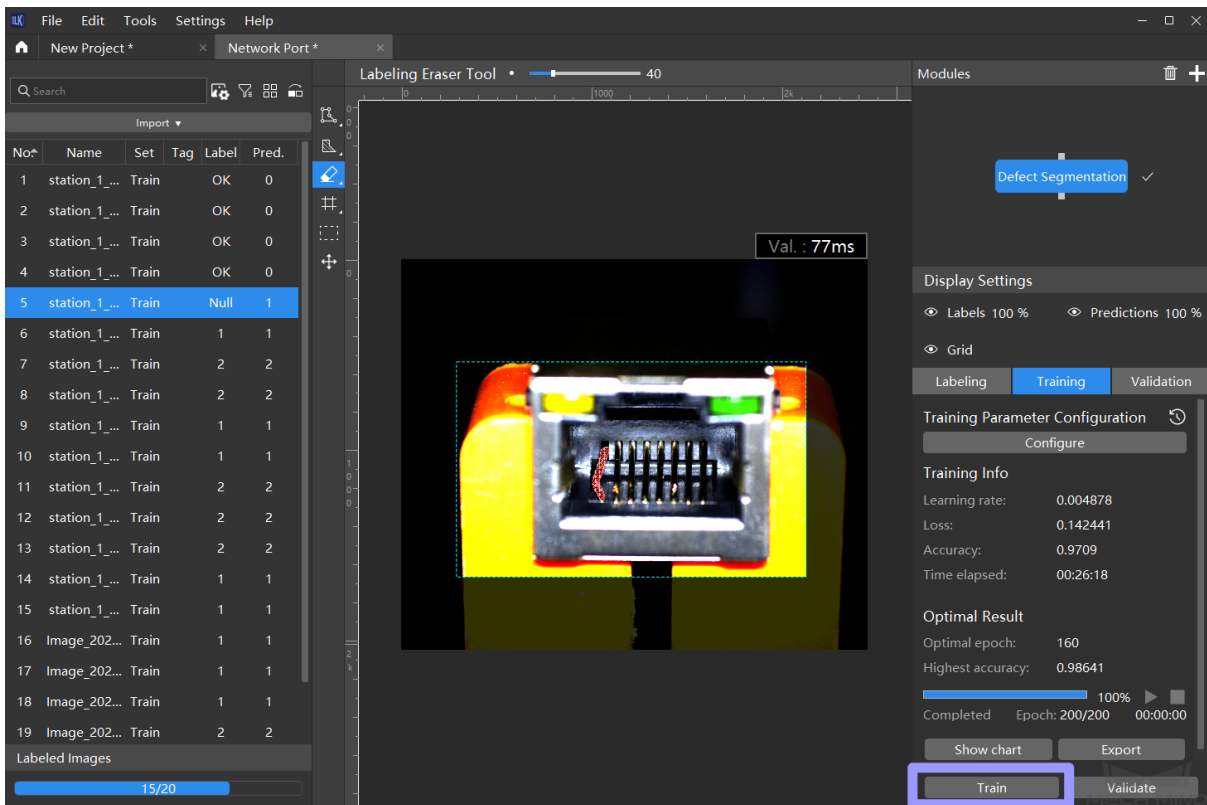


For an image that contains no defect, please select the image in the image list on the left, right-click, and select *Set to OK* .



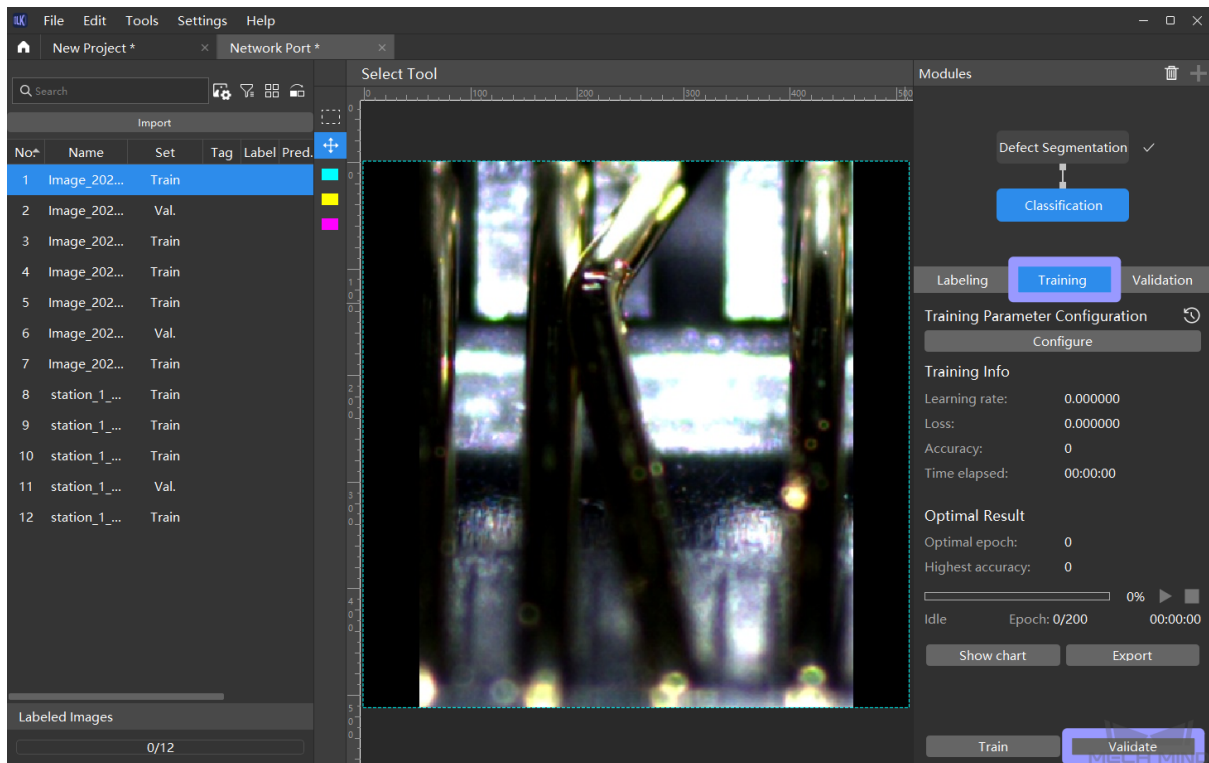
6. Train the model

Click *train* at the bottom right to start training.



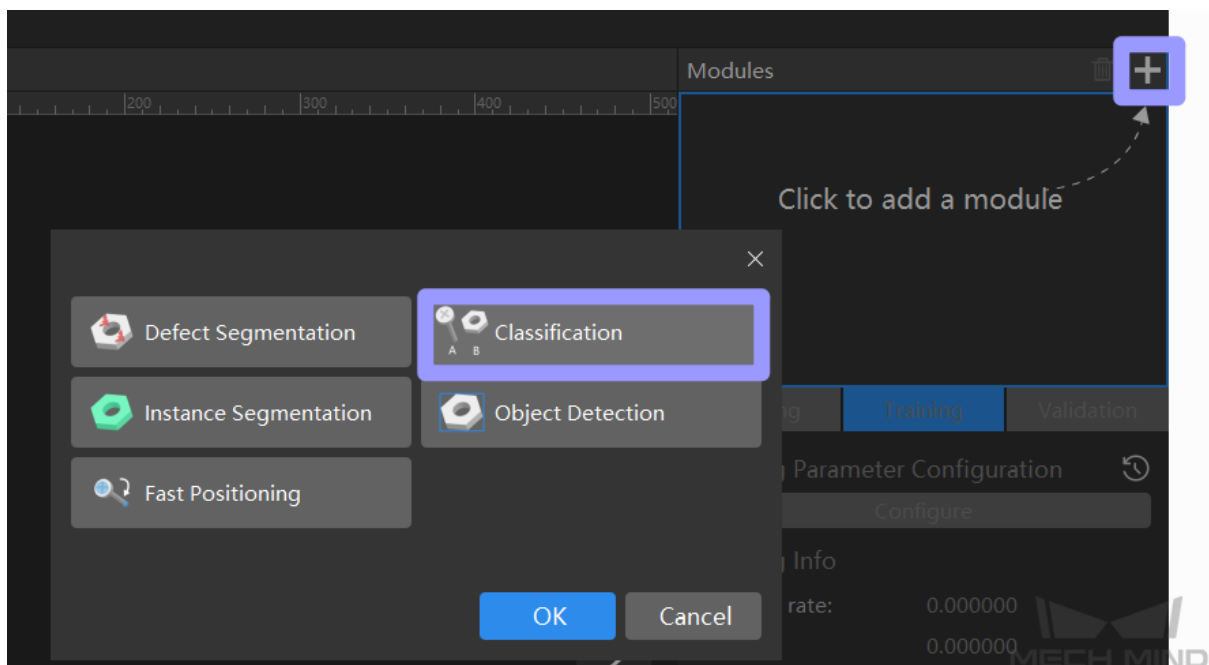
7. Validate the training effect

After the model training is completed, please click on *Validate Result* to see the model effect.



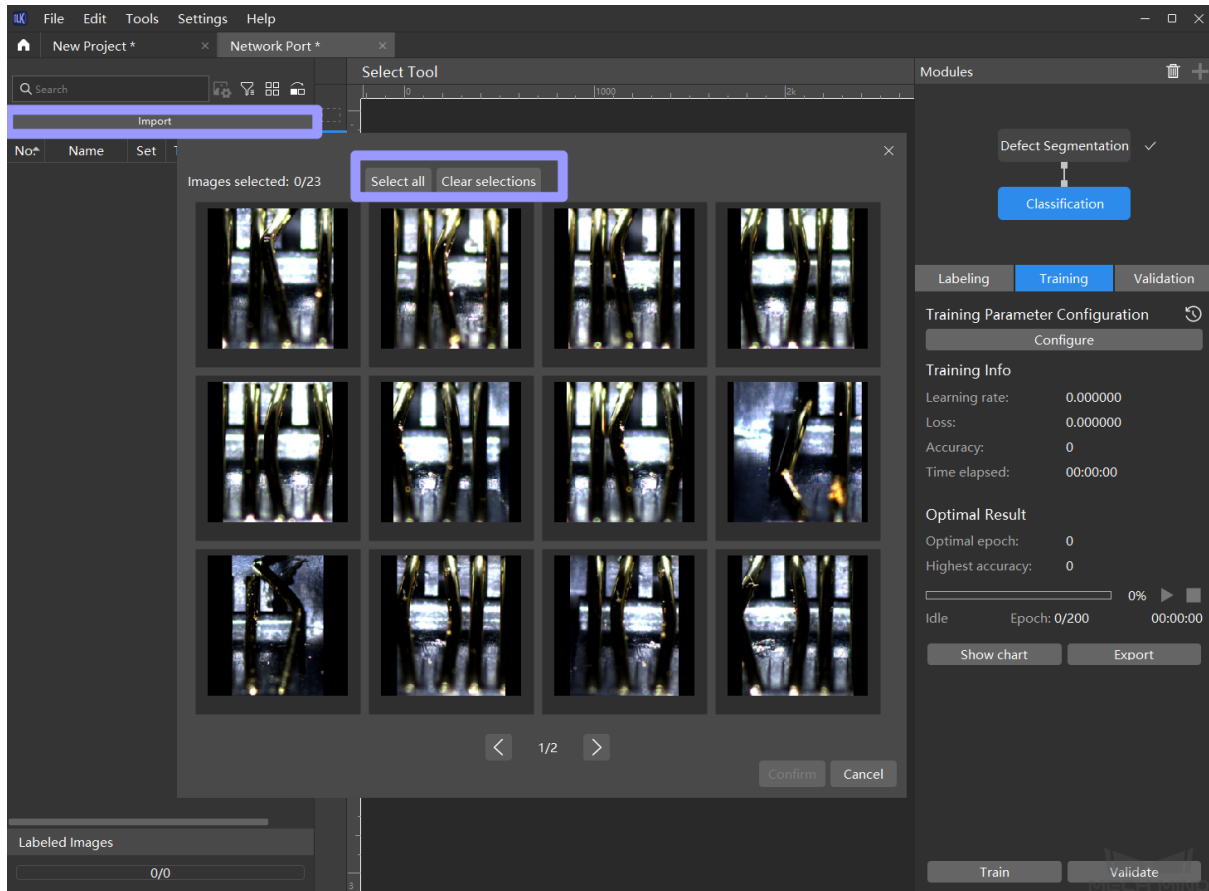
8. Add the “Classification” module

After validating the training effect of the trained defect segmentation model and confirming that the recognition performance meets the requirement, please click + in the “Module” section at the upper right of the window to add a “Classification” module.



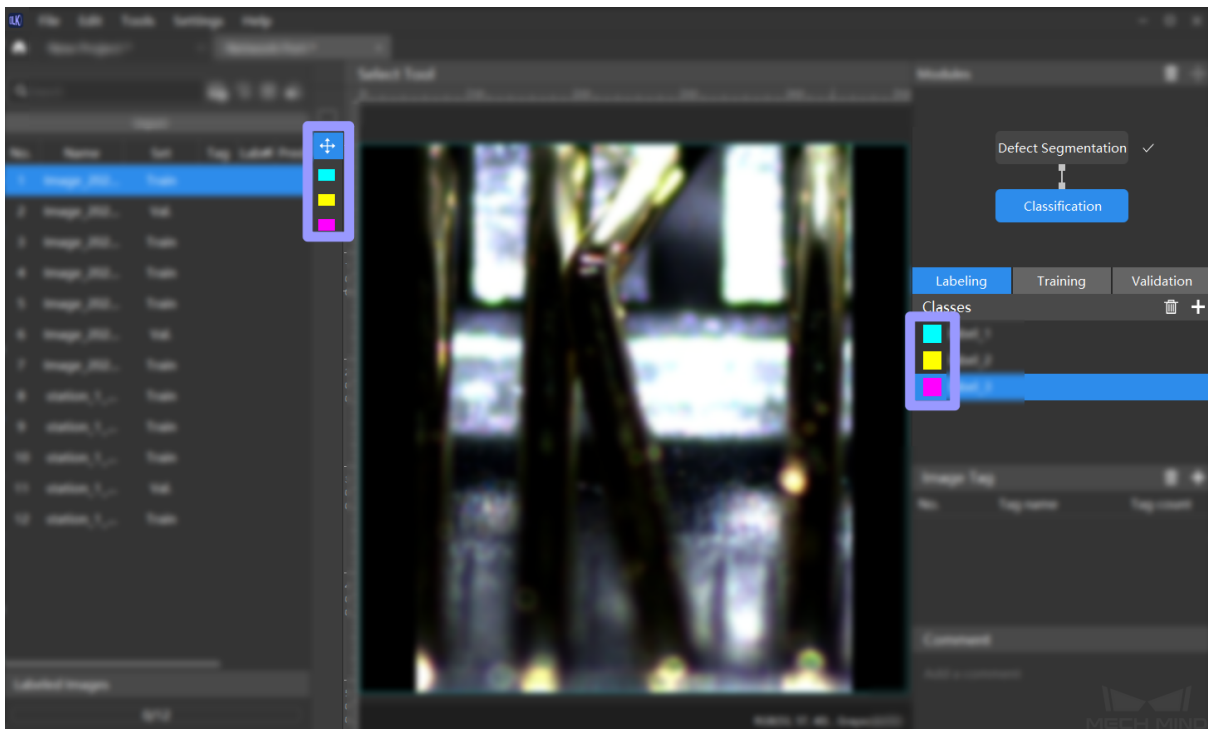
9. Import data into the “Classification” module

The above defect segmentation training results will be imported into the image classification module as a data source. Please click on *Import* , select the required data, and click on *Confirm* .



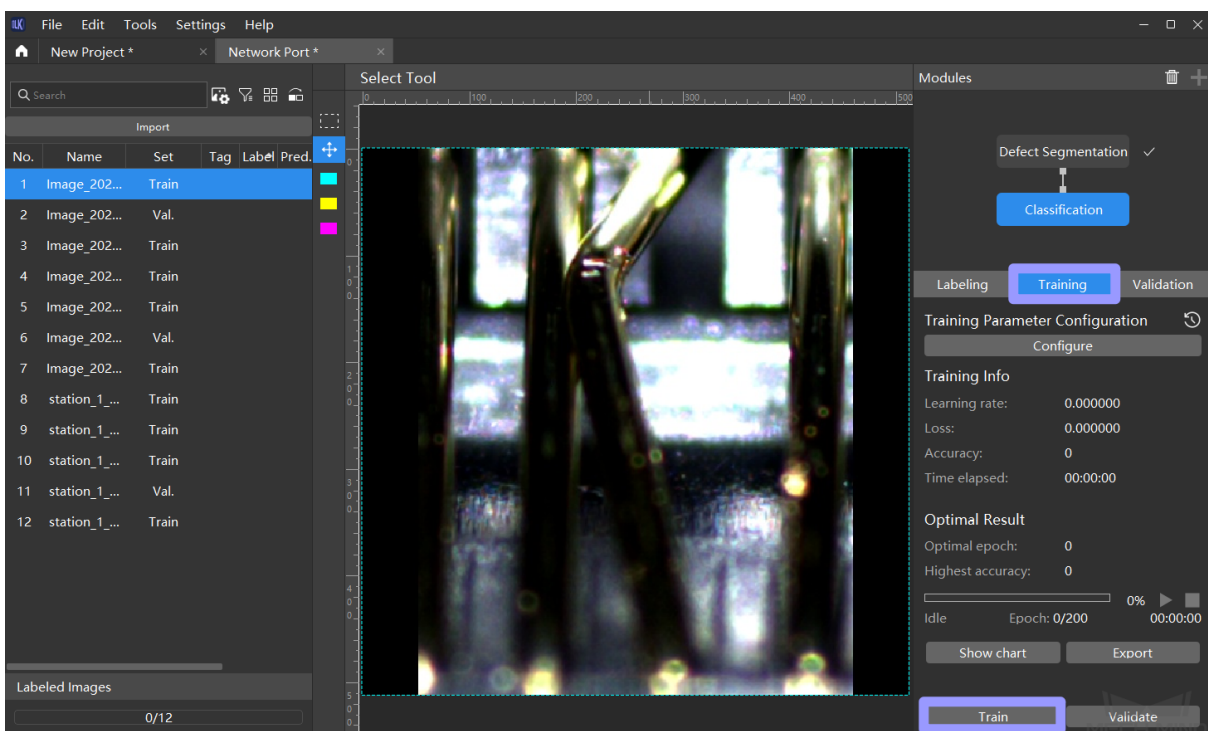
10. Create image classification labels and start labeling

Before labeling the *Classification* module, please click on + in the “Label” section on the right to create different labels according to the classes of the target objects. Then click a label on the left side of the “Label” section to label the images.



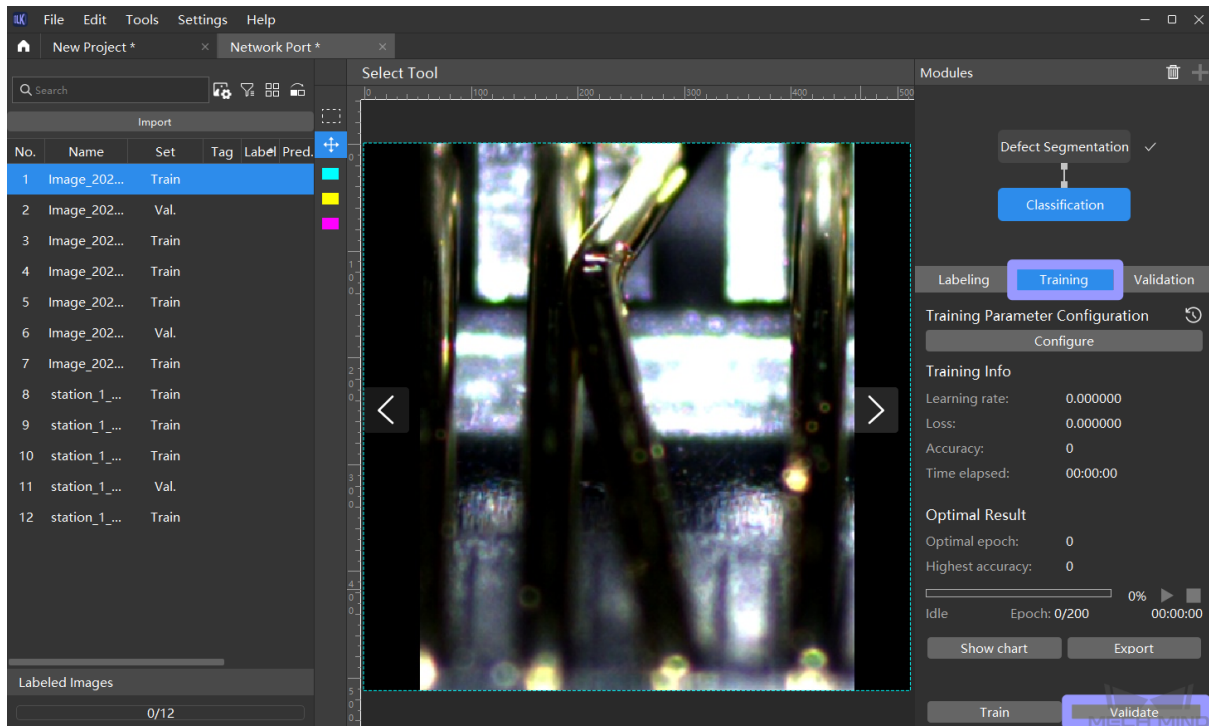
11. Train the model

Click *Train* to start training the model with default parameters.



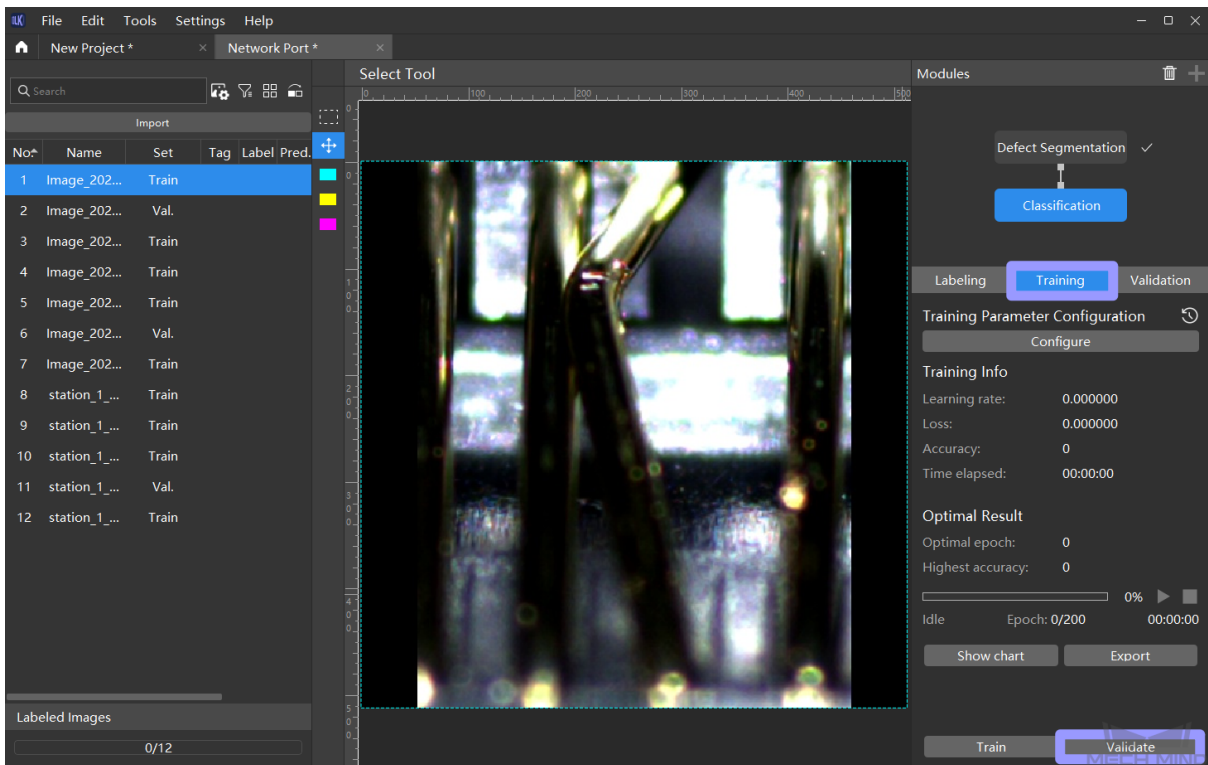
12. Validate the model

After training, click *Validate* to see the model's performance.



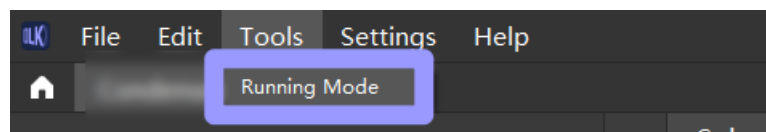
13. Export the model file

After the model training is completed, please click on *Export model* on the right to export the optimal model to the project folder.



RUNNING MODE

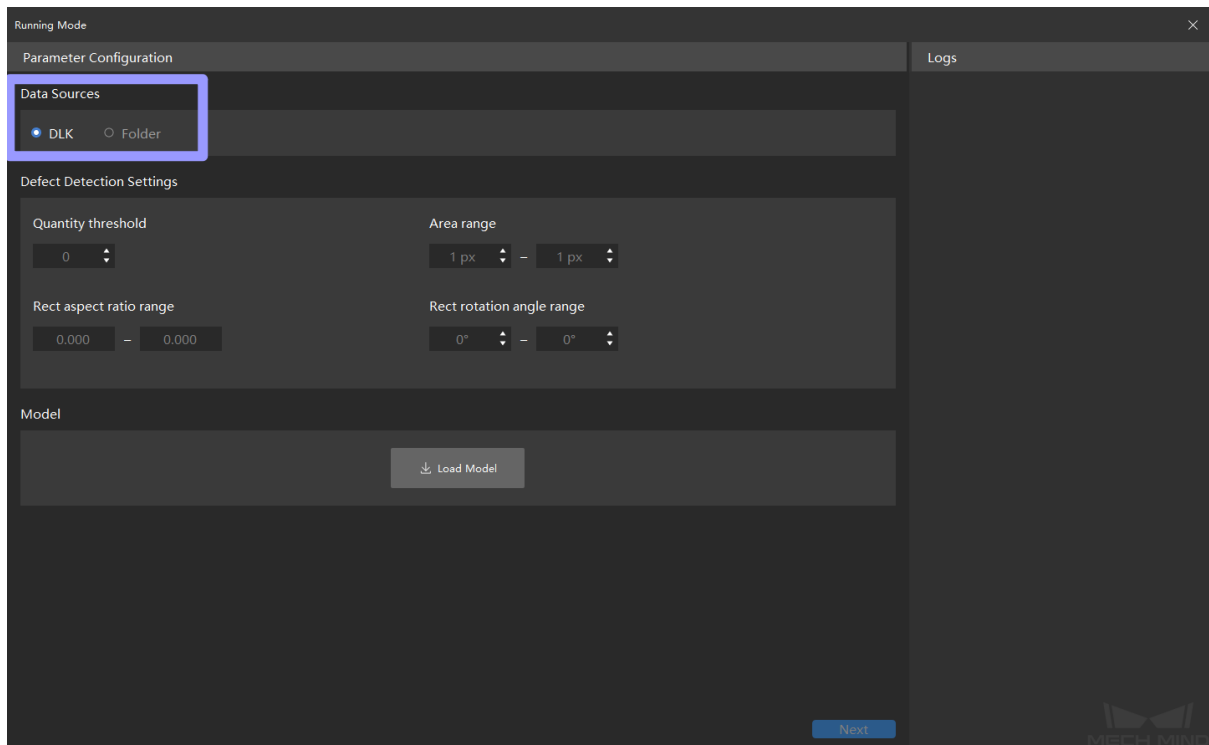
In the running mode, you can directly make inference and verify the model training effect, and export the inference report at the same time. The report records information such as accuracy, and false positive and false negative rates, etc.



1. Select data source

DLK : Directly import all data in the software (as shown in the “Data” section of the main window).

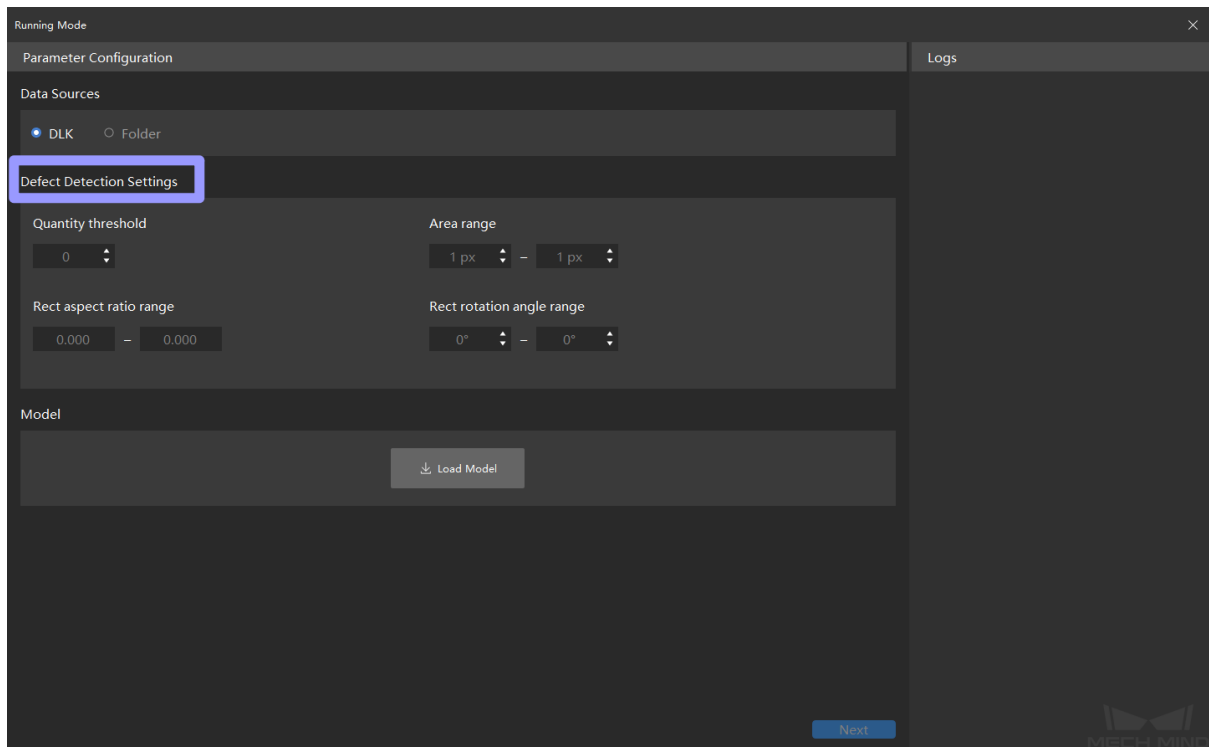
Folder : Select the image folder path outside the software. The new data loaded will be independent of the original dataset within the software.



Hint: Before using the running mode, if other images have been imported in addition to the training and validation set, selecting *DLK* will import the other images as well.

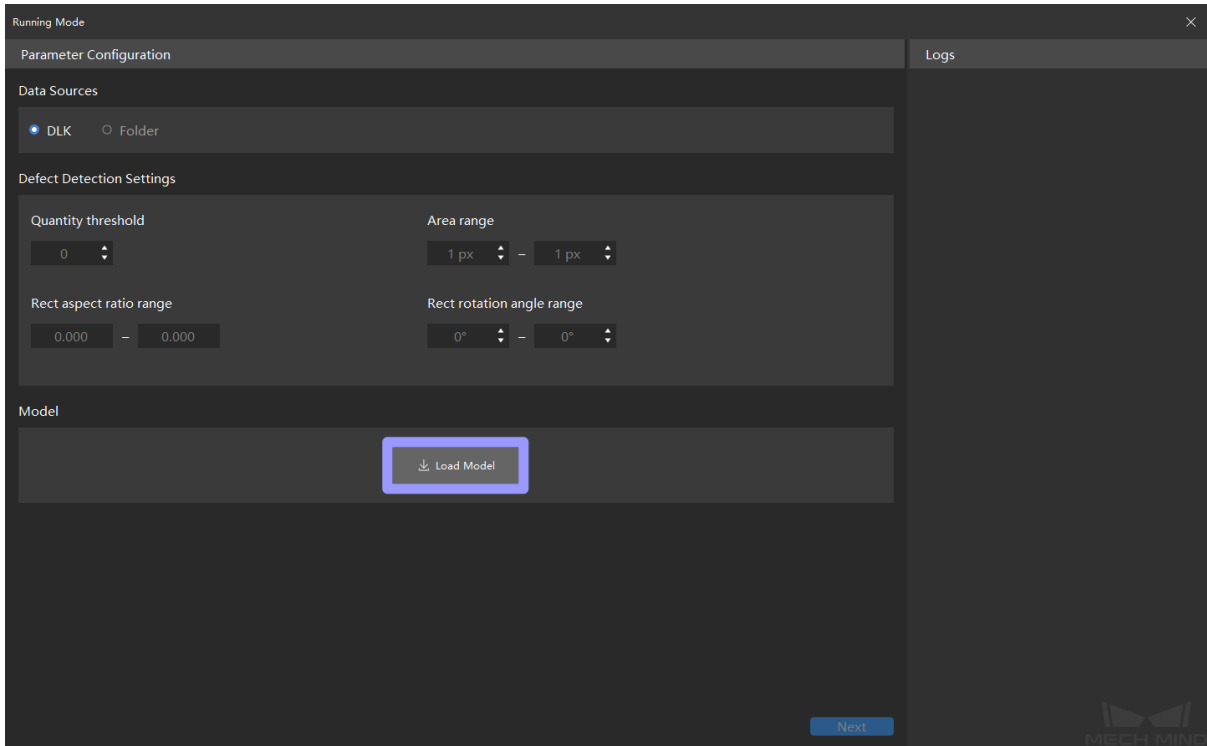
2. Defect detection settings

Set each parameter according to the defect judgment standard, and open the defect filtering related options in the software. The defect detection parameters are independent of the parameters in the Defect Judgement Rules.



3. Load the model

Click on *Load Model* , and click on *Next* after the loading is successful to enter the inference interface.

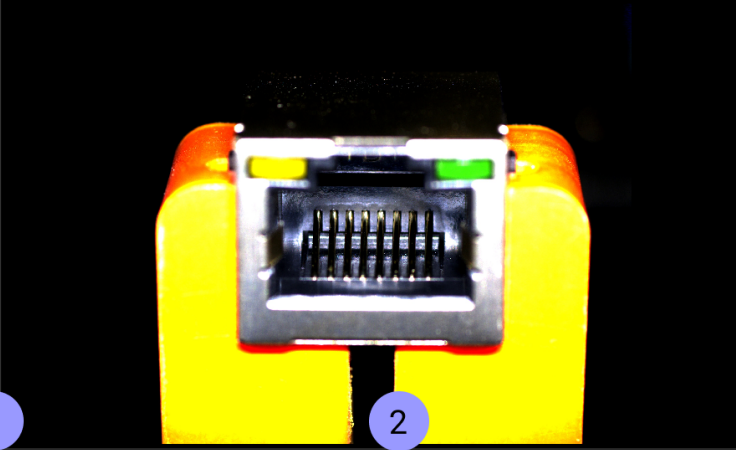


4. Make inference and export report

Click on the start button in the automatic inference section to start making the inference. After the inference is completed, check the validation results in the manual review section. After all the checks are completed, click *Export report* to view the accuracy rate, over-review rate, missed inspection rate, etc.

Running Mode

station_1_00082.jpg



Running Information

Infering result

Inference Time Cost

Single-image inference time:

Average inference time:

Defect Count

Over checked images:

Missed images:

GPU Utilization

Current GPU utilization: 4.76

1

2

Auto infer

Manual check

3

Back

Export report

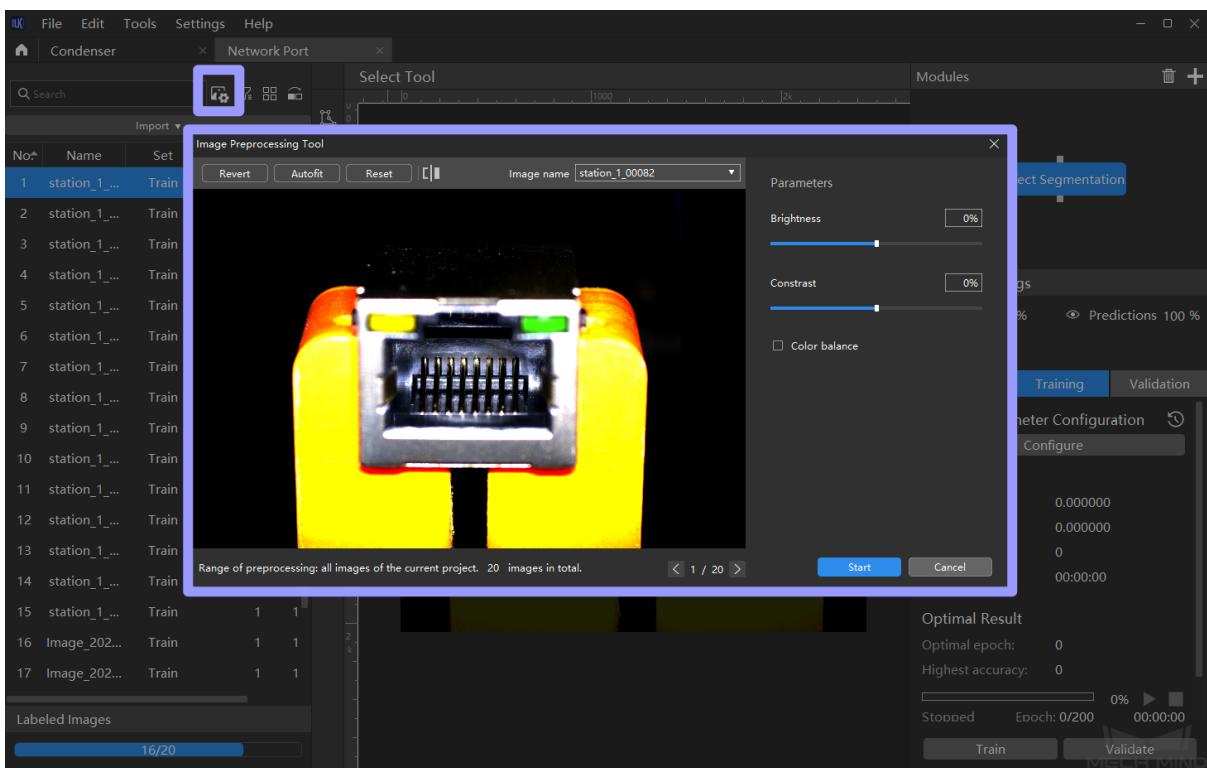
Complete

The interface shows a central image of a yellow and red mechanical component. To the left is a vertical list of image thumbnails, with the first one highlighted. Below the main image are two control panels: 'Auto infer' with a play button and 'Manual check' with green checkmark and red X buttons. On the right, a 'Running Information' panel displays inference results and GPU usage. At the bottom, there are buttons for 'Back', 'Export report', and 'Complete'. Three blue circles with numbers 1, 2, and 3 are overlaid on the interface to indicate specific areas of interest.

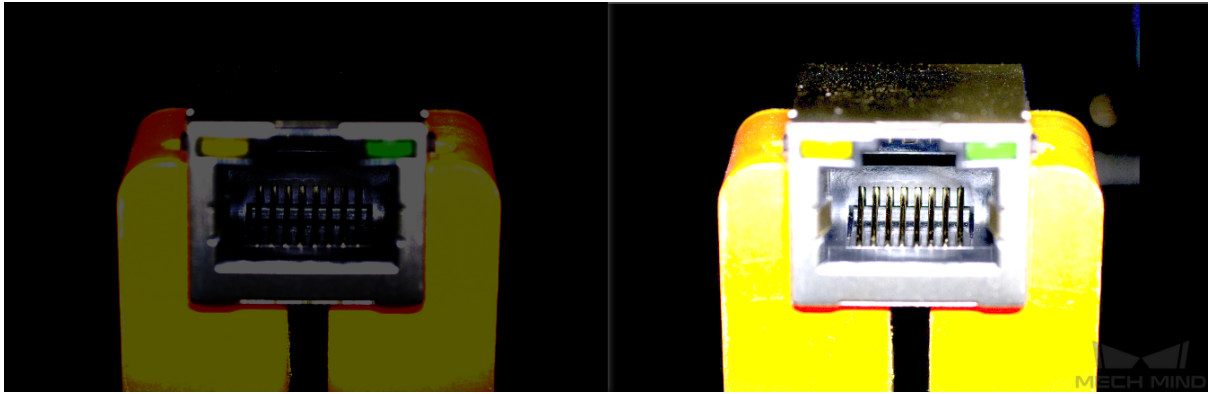
IMAGE PREPROCESSING TOOL

When the original image captured by the camera has problems such as darkness and indistinct object features due to lighting or other factors, image preprocessing tools can be used.

The tool preprocesses images by adjusting parameters for brightness, contrast, and color balance.



The images before and after processing are as follows:



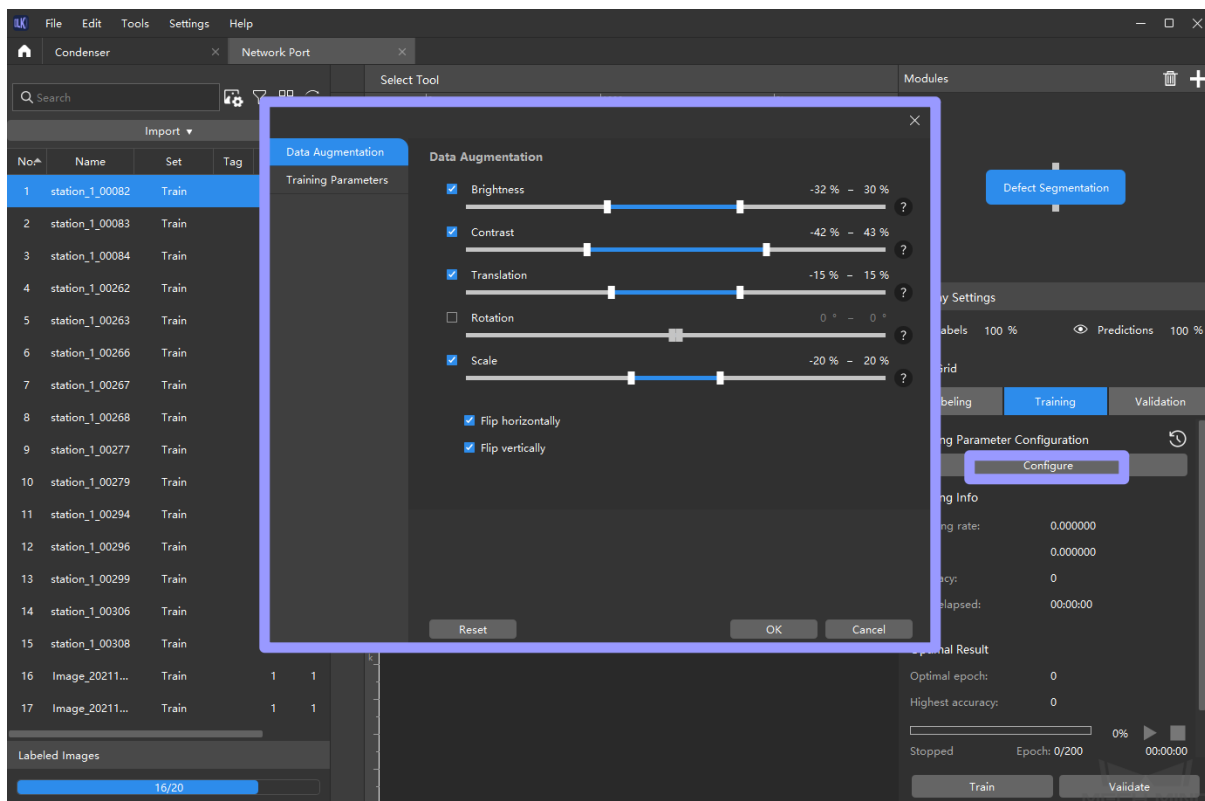
Hint: The image dataset will be changed directly after preprocessing, and then used as the training and validation set in model training.

DATA AUGMENTATION

In deep learning projects, the collected image datasets need to reflect all the situations on site, but in many application projects, the sites may not have the corresponding collection conditions.

For example, sometimes it is impossible to collect image data under different rotation angles, different moving ranges, etc.

For this issue, by adjusting the data augmentation parameters as shown in the figure below, more adequate datasets can be generated based on the original data.



Attention: Please make sure that the contents of the augmented image datasets are consistent with the actual situation on site. For example, when there is no rotation on site, if the rotation parameter is adjusted, the model training effect will be affected.

Brightness

When the actual lighting changes greatly, the dataset under different lighting conditions can be augmented by adjusting the brightness range.

Contrast

When the differences between the target objects and the background are not obvious, to better train the model on the object features, the contrast range can be adjusted appropriately.

Translation

When the moving ranges of the on-site objects (bins, trays, etc.) are large, the data can be augmented in terms of horizontal and vertical translation by adjusting the translation parameters.

Rotation

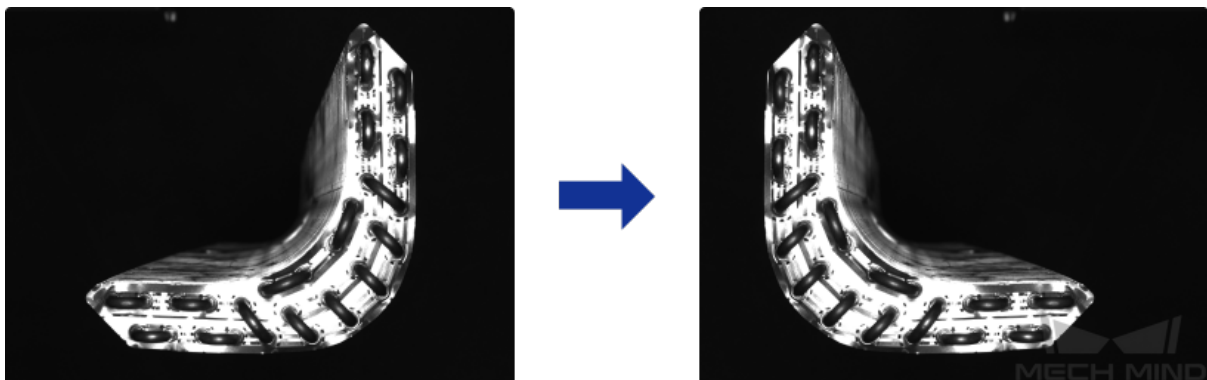
When the placement orientations of the objects change greatly, the data can be augmented by adjusting the rotation parameters according to the actual situation. Usually, you can keep the default parameter settings.

Scale

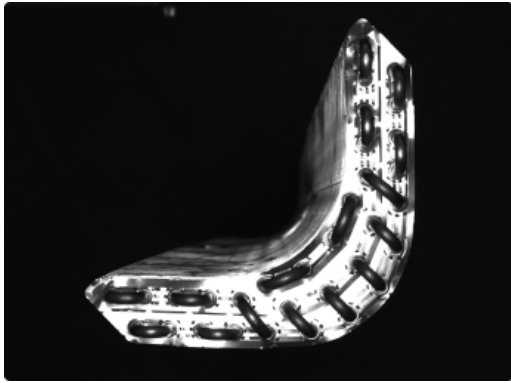
When the height differences between the objects are large, the data can be augmented in terms of different target object heights by adjusting the scale.

Flip horizontally

If the object to be recognized has left-right symmetry, this feature can be enabled.

**Flip vertically**

If the object to be recognized has up-down symmetry, this feature can be enabled.



KEYBOARD SHORTCUTS

No.	Feature	Shortcut	Note
1	New Project	Ctrl + n/N	
2	Save Project	Ctrl + s/S	
3	Open Project	Ctrl + o/O	
4	Undo labeling	Ctrl + y/Y	
5	Redo labeling	Ctrl + z/Z	
6	Copy labeling	Ctrl + c/C	
7	Paste labeling	Ctrl + v/V	
8	Select all labeling	Ctrl + a/A	
9	Delete labeling	Delete	Please move the cursor to the labeling
10	Ellipse Tool	l / L	
11	Polygon Tool	p / P	
12	Rectangle Tool	r / R	
13	Brush Tool	b / B	
14	Autofill Lasso Tool	a / A	
15	Labeling Eraser Tool	e / E	
16	Auto Labeling Tool	t / T	
17	Mask Polygon Tool	Shift + p/P	
18	Mask Brush Tool	Shift + b/B	
19	Mask Lasso Tool	Shift + a/A	
20	Mask Eraser Tool	Shift + e/E	
21	Grid Cutting Tool	u / U	
22	Grid Selection Tool	i / I	
23	Feature Group Labeling Tool	f / F	
24	ROI Tool	o / O	
25	Selection Tool	s / S	
26	Clear labeling of the current image	Ctrl + d/D	
27	Switch labeling display	Ctrl + l/L	
28	Switch prediction display	Ctrl + p/P	
29	Delete dataset image	Delete	Please move the cursor to the dataset
30	Switching between datasets or in tables/dragging sliders	↑↓→←	Please click on the corresponding com
31	Image page turning: previous page	Alt + ←	
32	Image page turning: next page	Alt + →	

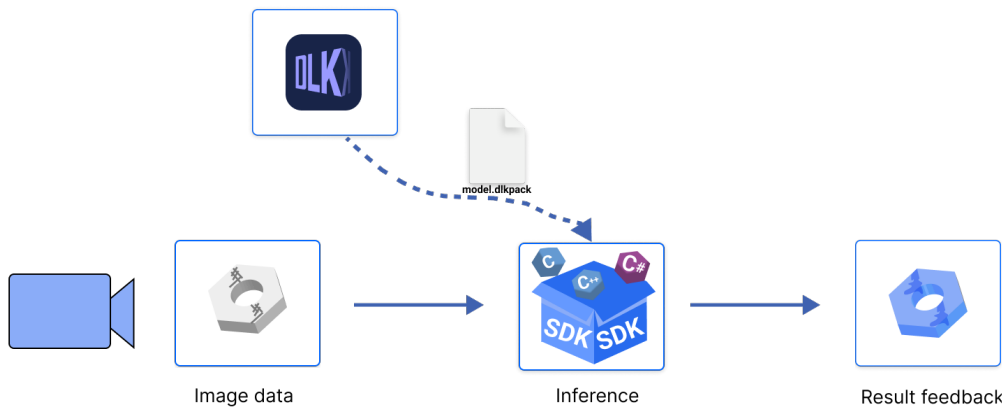
Tip: Hover over a tool icon in the interface to quickly view the keyboard shortcut corresponding to

that tool.

ABOUT MECH-DLK SDK

14.1 What is SDK

The SDK of Mech-DLK supports C, C#, and other multi-language interfaces, and provides DLL libraries, examples, etc. to meet the needs of users in model deployments and integrations.



14.2 System Requirements and Licensing

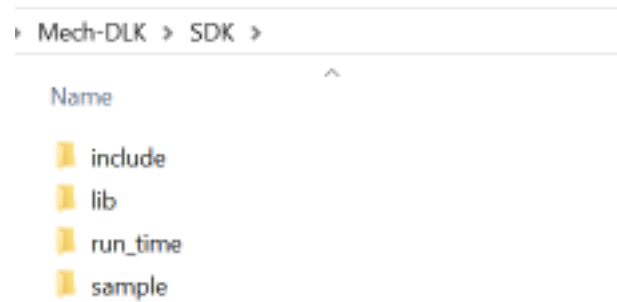
- Please make sure the dongle is authorized;
- Please ensure that the computer system, software and hardware meet the SDK deployment requirements:

Operating system	Windows 10 (recommended)/Windows 11
CPU	Intel(R) Core(TM)i5 or later
Memory	8 GB or larger
Graphics card	GeForce GTX 1650(4 GB) or later
Graphics driver	471.68 or later
.NET environment	.net 4.61 or later
C/C++	C99 and above compilers

Attention: The configurations of the computer system and software and hardware have a great impact on the performance of the SDK after deployment, so it is necessary to ensure that the above requirements are met.

14.3 Location and Structure

Location



Structure

include: C API header files for C projects.

lib: Static library for the C API, used in C projects.

run_time: Dynamic runtime library for C/C# projects.

sample: SDK sample project, with C# samples at present.

GETTING STARTED WITH SDK

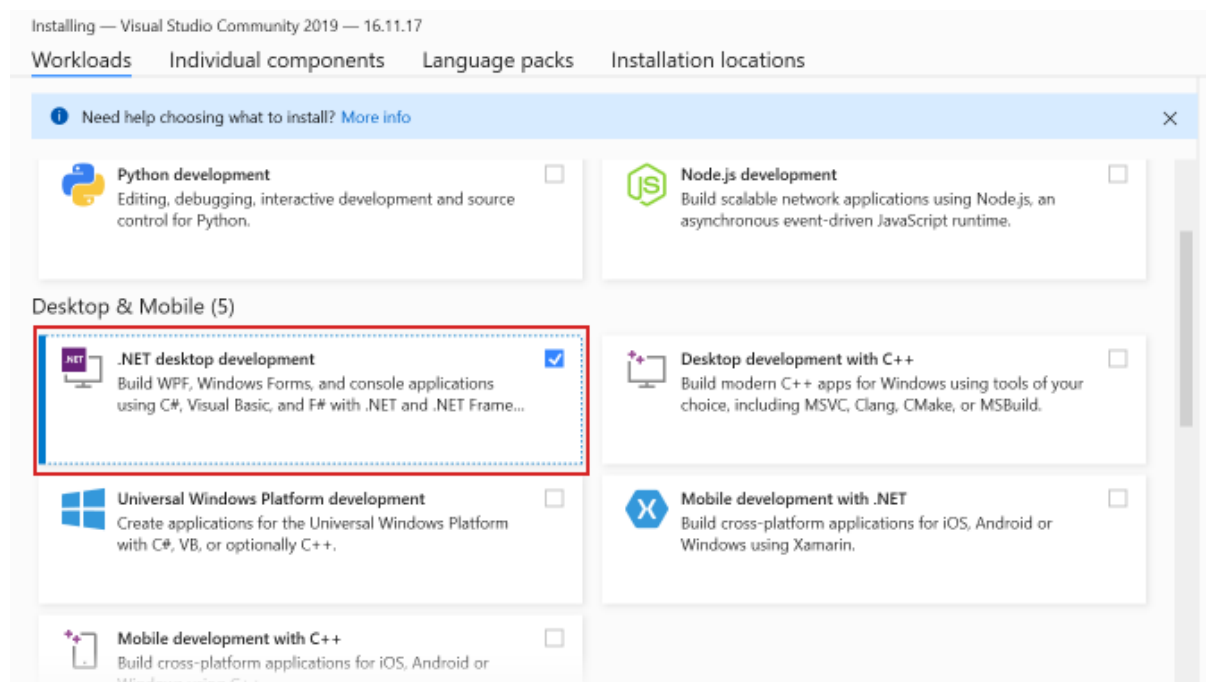
This section guides users who use the SDK for the first time to learn the relevant operation procedures, and quickly create and complete a new project of the Mech-DLK SDK.

15.1 Requirements

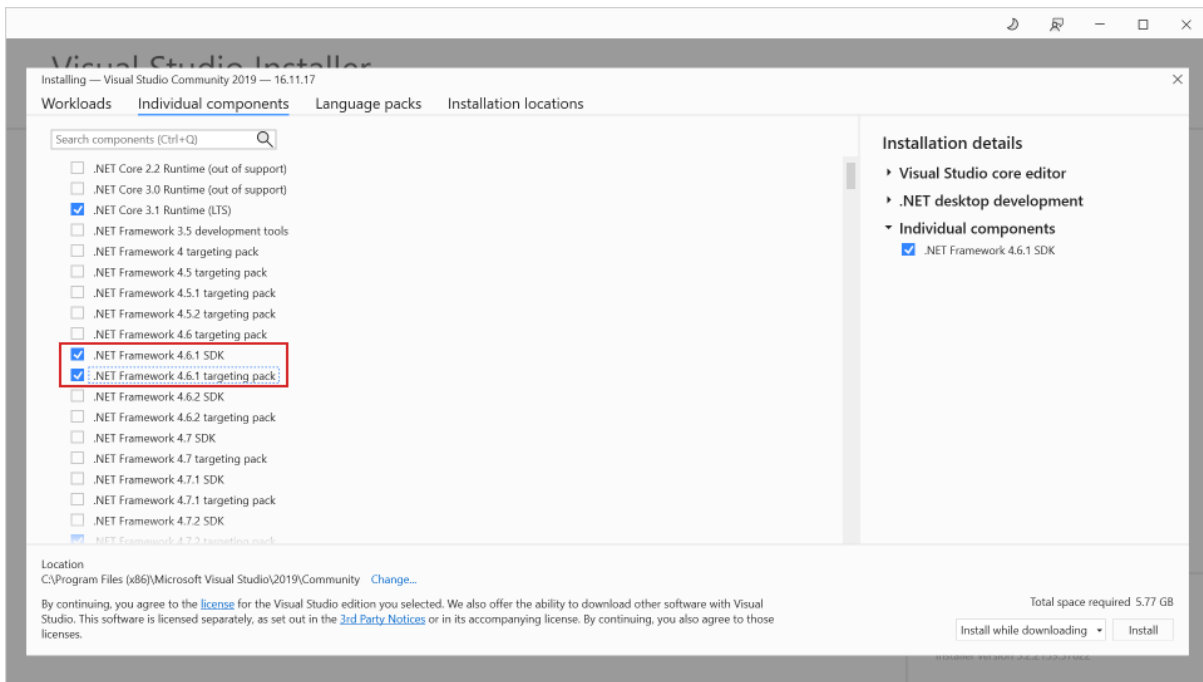
Download Visual Studio 2019.

Install components for Visual Studio

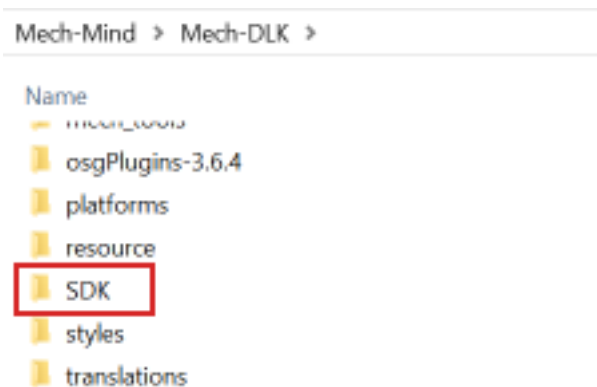
Workloads → *.NET desktop development* .



Individual components → *.NET Framework 4.6.1 target pack* .



Make sure there are SDK files in the Mech-DLK folder.



15.2 Run the SDK C# Example Project

The SDK_Example project provides examples showing how to use the Mech-DLK SDK to implement the required functions in the application.

Taking defect segmentation as an example, this section provides the required model and image dataset to lead the user to run the SDK project:

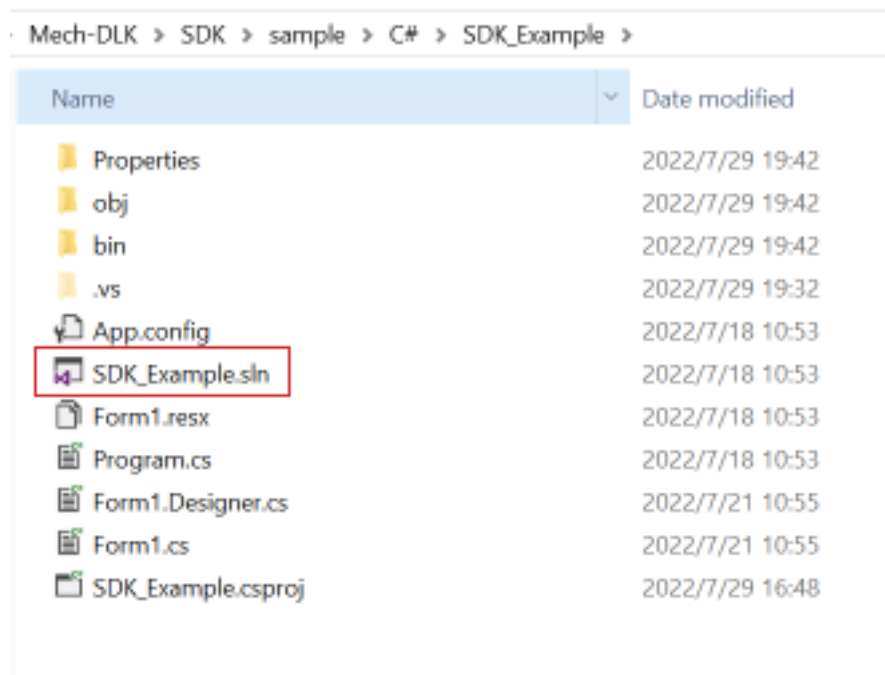
- Click to download the image dataset

- Click to download the model



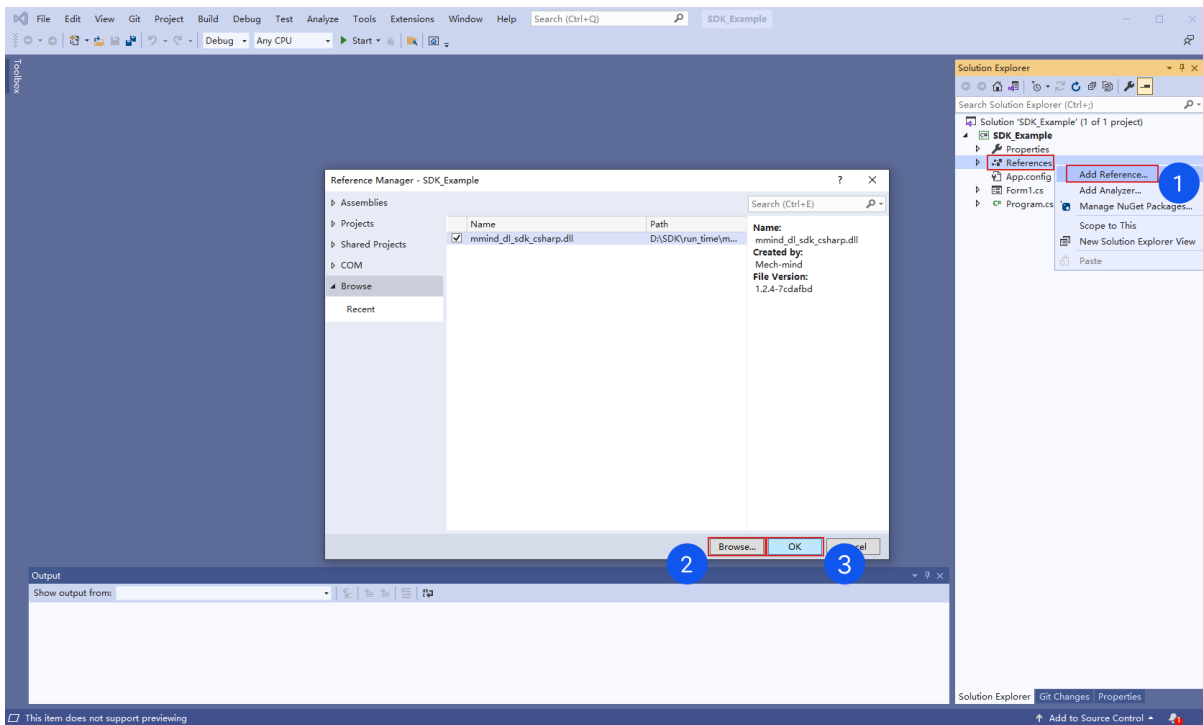
1. Open the example project

Double-click the file *SDK_Example.sln* in the path shown below to open it with Visual Studio 2019.



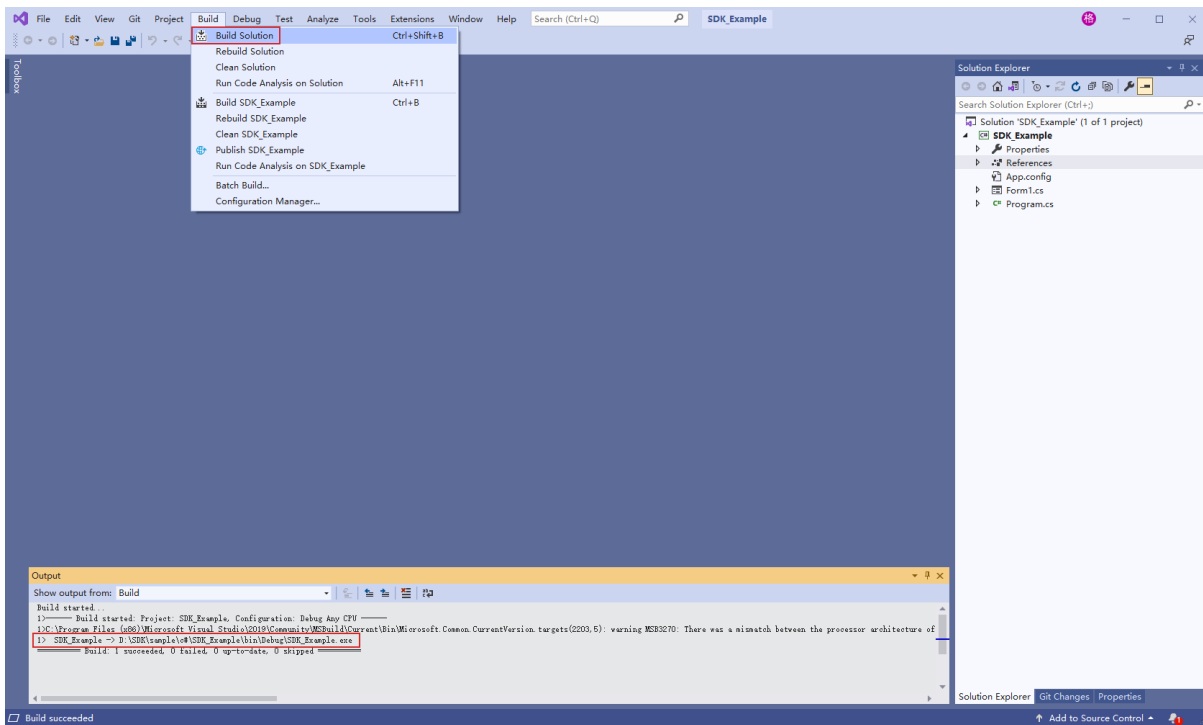
2. Referencing SDK

Right-click on *Reference* -> *Add Reference*. The reference manager window will pop up. Click on *Browse...*, check the file *mmind_dl_sdk_csharp.dll* in the folder *run_time* and click on *OK*.



3. Build solution

Click on *Build* -> *Build Solution* to generate the executable file. Check the path generated by the executable in the output section at the bottom of the interface.




4. Add the “run_time” library

Copy all the dll format files in the folder below to the release directory of the executable file generated in the previous step. If there are duplicate files, just overwrite them.

Original file location:

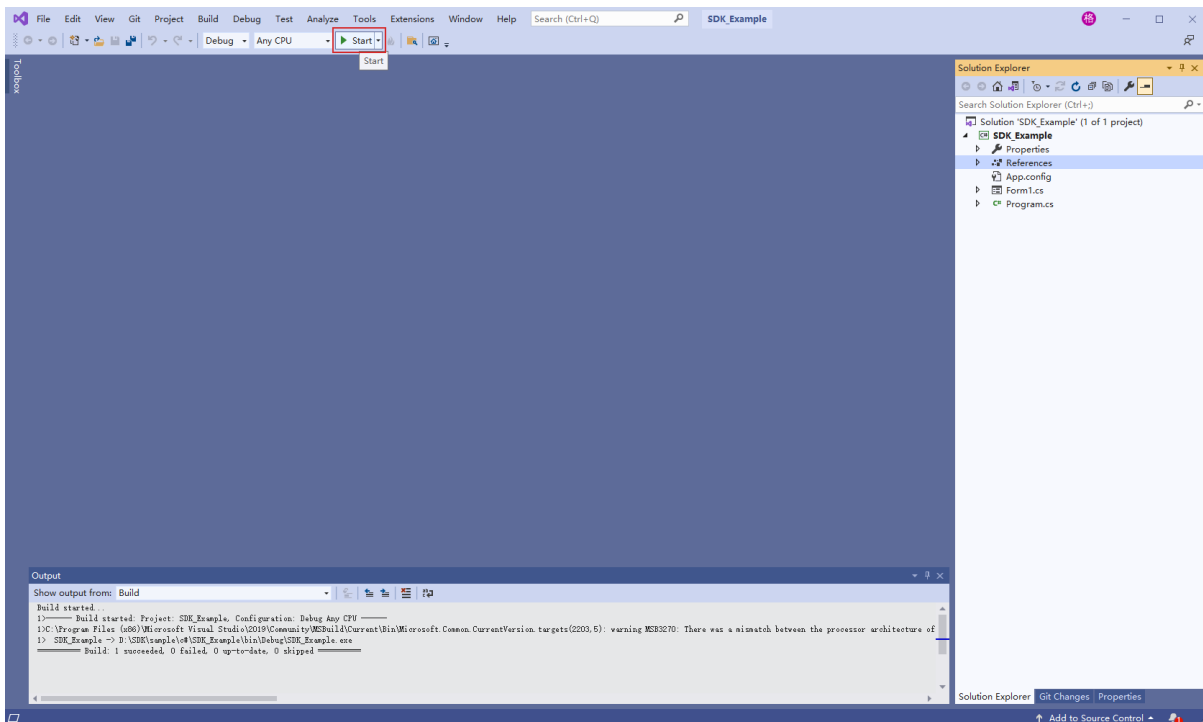
 << Mech_DLK > SDK > run_time

Target path:

 << Mech_DLK > SDK > sample > C# > SDK_Example > bin > x64 > Debug

5. Start project

Click on *Start* to run the program.



Program interface after startup:

Load Image: Load the image to be inferred.

Load Model: Load a model in dlkpack format exported by Mech-DLK.

Infer: perform inference.

Form1



Load Image

Load Model

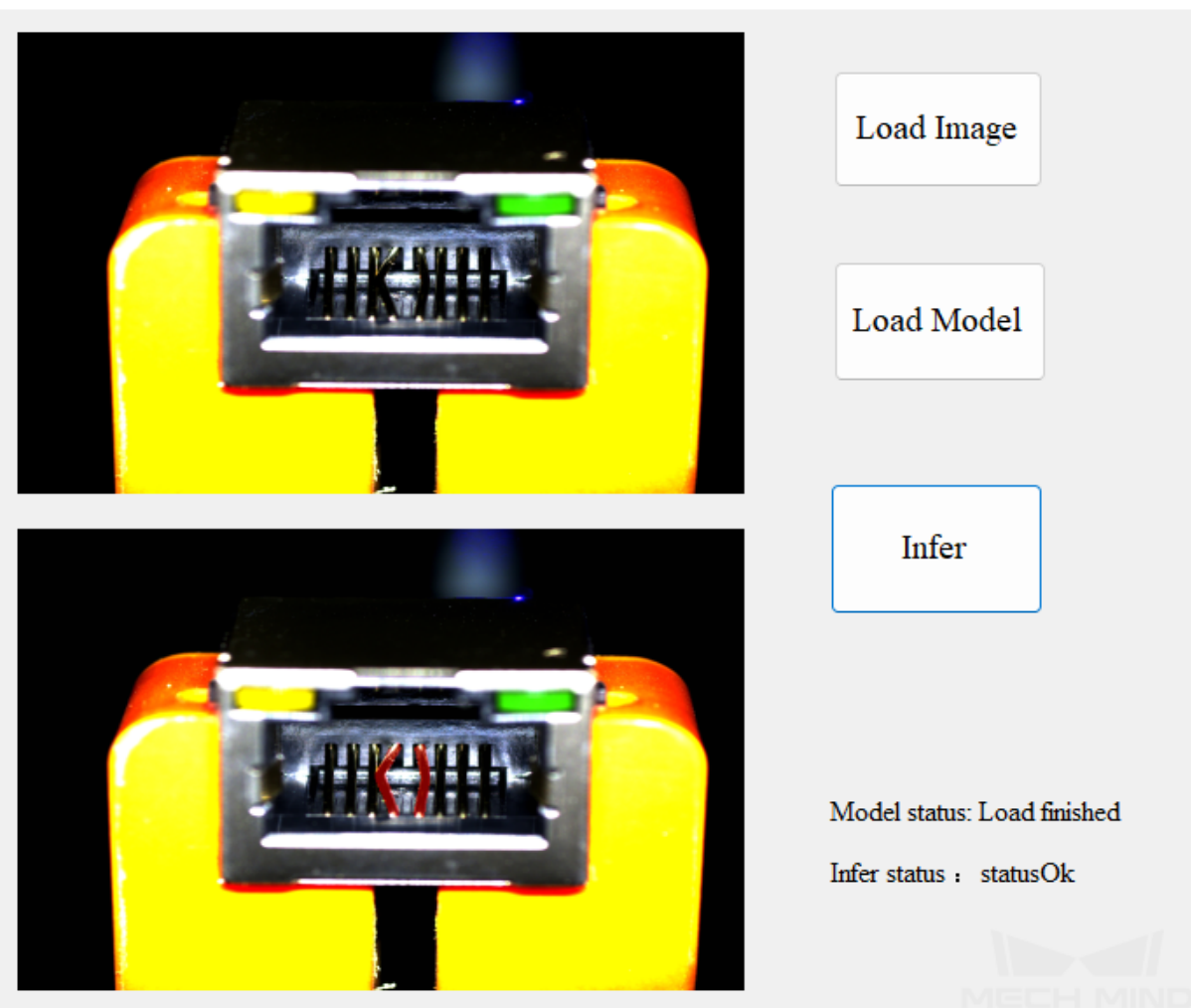
Infer

Model status:NA

Infer status:NA



Execution result:



Attention: It takes a little longer to load the model for the first time. About 5 minutes. It is recommended to use multiple threads to call the corresponding interface.

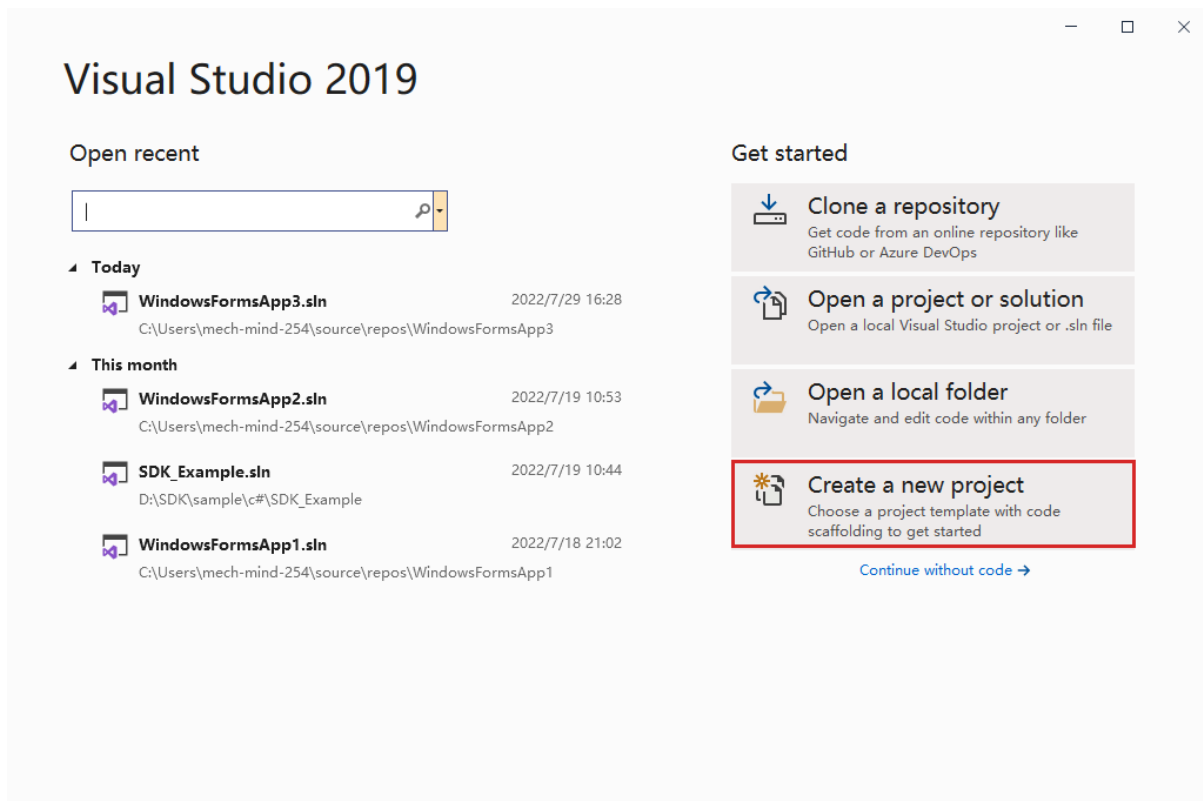
15.3 Create a New Project

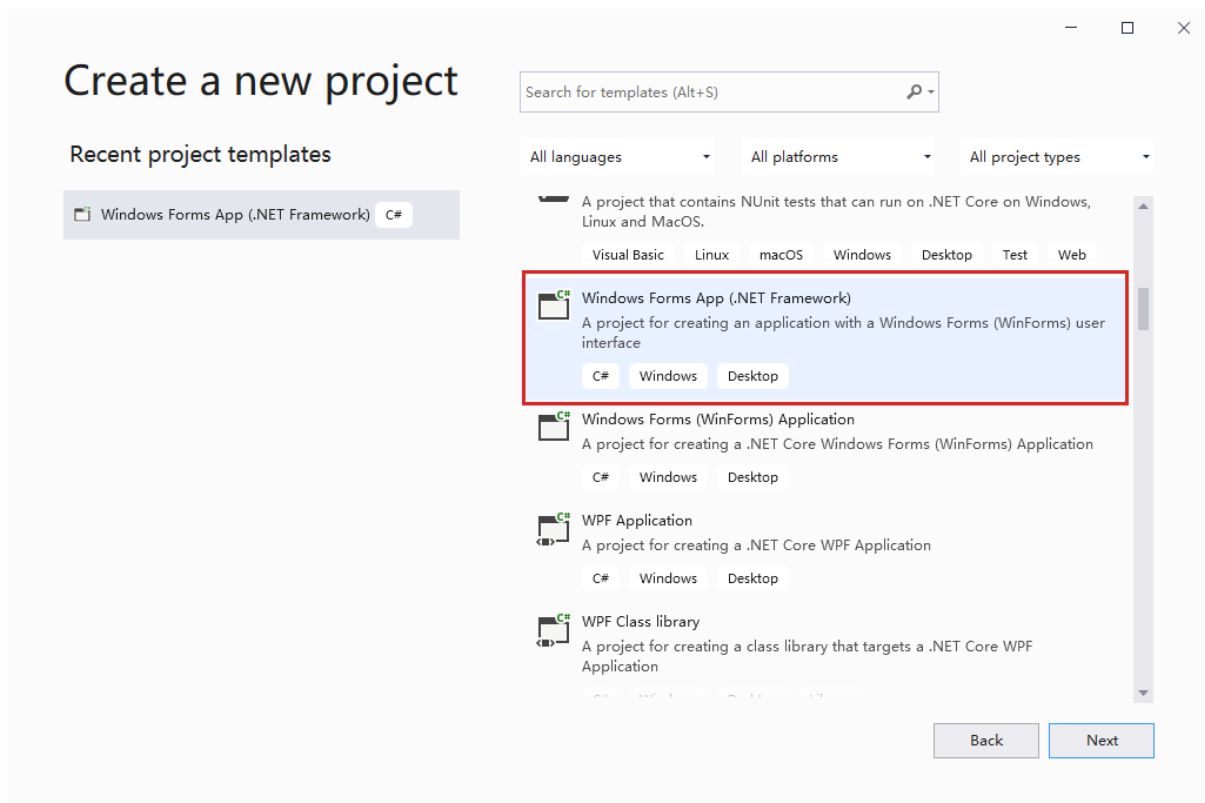
After ensuring that the system environment meets the requirements for integration, you can refer to the following instructions to create a new project using the SDK. The instructions take .NET (Visual Studio 2019) as an example:

1. **Create a new project**

Open Visual Studio 2019, if you are to create a new project for the first time, please click on *File* → *New Project* from the upper left corner, and on the Create New Project page,

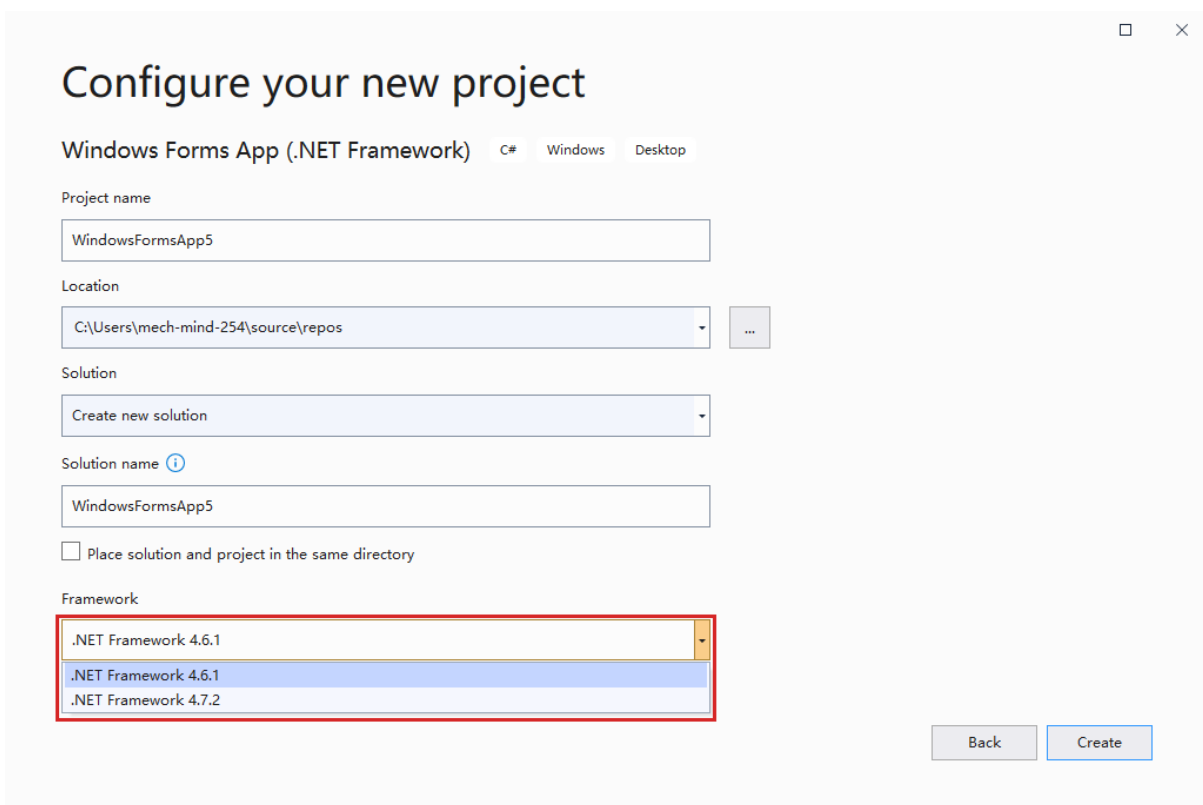
click Windows Desktop Wizard to create a Windows application.





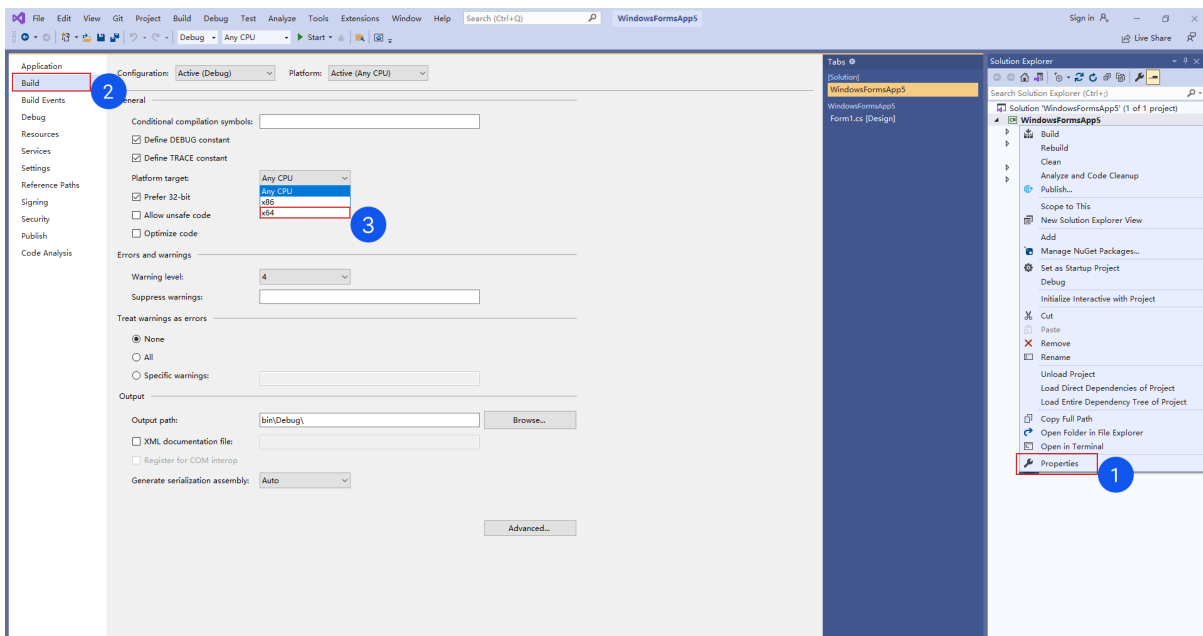
2. Configure the new project

Select .NET Framework 4.6.1 (4.6.1 and above are supported).



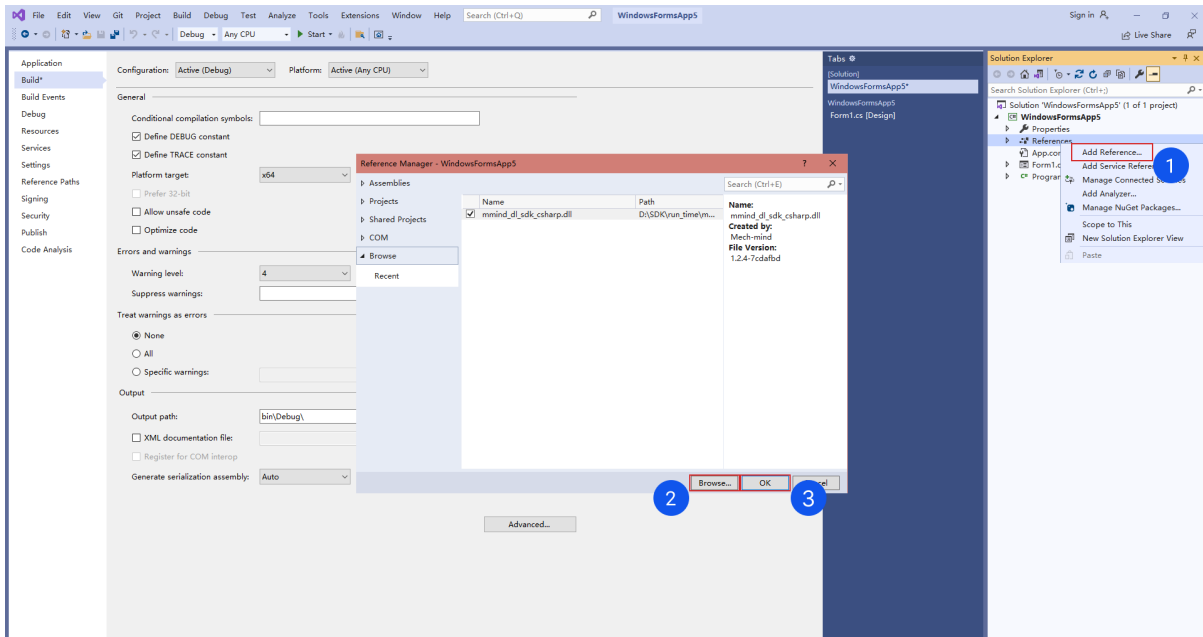
3. Configure project properties

Right-click the new project name in the right project bar, and select *Properties* → *Build* → *x64* for the target platform. You can also add the x64 active solutions platform via the configuration manager.



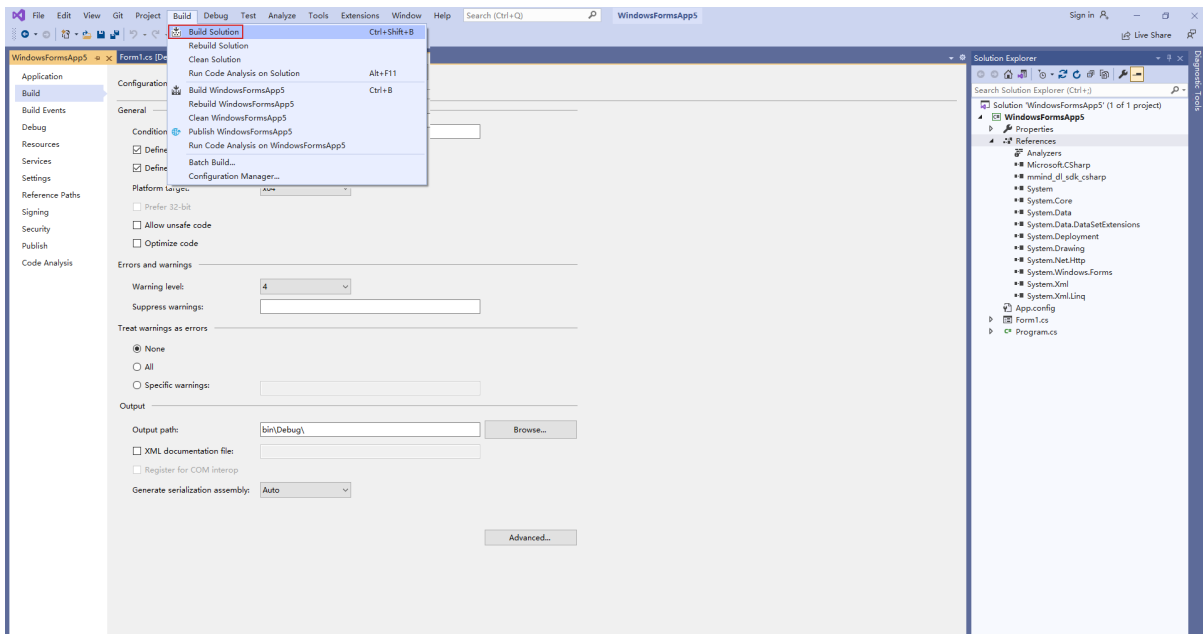
4. Add referencing

Right-click on *Reference* in the project section and select *Add reference*, click on *Browse* in the pop-up window, check the file *mmind_dl_sdk_csharp.dll* under the run time folder, and click on *OK*. If the file already exists in the window, select it directly.

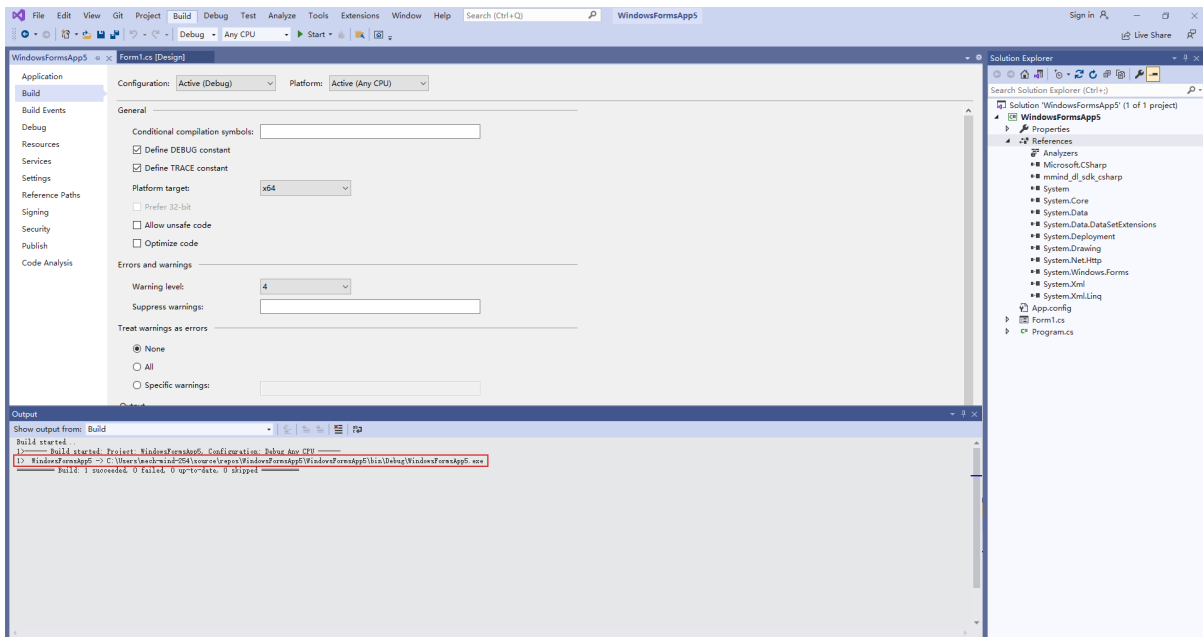


5. Build solution

Click on *Build* → *Build Solution* on the menu at the top to generate an executable file.



Copy all the DLL files in the run_time under the SDK folder to the exe directory. Then, you can find the exe directory path in the red box in the figure below.



6. Start integration

You can refer to the code of the example project or *API Reference Guide* to complete subsequent integrations.

API REFERENCE GUIDE

16.1 C# API

16.1.1 Interface Functions

1. setVisible

```
void setVisible(Boolean flag);
```

Set up the result visualization. If this function sets visualization to true, the SDK will automatically fuse the predicted result with the input data, and the result will be in *Returned Results of Interface Function Inference*. If you have high requirements on performance, please do not set it to true.

Input parameter:

- flag: The boolean flag for visualization. The value defaults to false.

Returned value:

None

2. setConvertInseg

```
void setConvertInseg(Boolean Convertflag);
```

Set the processing of instance segmentation results. If true is input, the inference result of instance segmentation models will have information of the inscribed rectangles and the circumscribed rectangles, and the result will be in *Returned Results of Interface Function Inference*. If you have high requirements on performance, please do not set it to true.

Input parameter:

Convertflag: The Boolean flag for drawing the inscribed and circumscribed rectangles of the instance masks. The value defaults to false.

Returned value:

None

3. LoadModel

```
SDKStatus LoadModel(string packPath);
```

Load the model exported by Mech-DLK. This interface is a synchronous interface and may be blocked. It is recommended to use thread calls.

Input parameter:

- packPath: dlkpack file path.

Returned value:

- SDKStatus: 1000 means normal, other values mean errors. Please see *Returned Values of Interface Function Call* for details.

4. Predict

```
InferPackResult Predict(Bitmap img);
```

Perform inference.

Input parameter:

- img: bitmap image.

Returned value:

InferPackResult: Inference result. See *Returned Results of Interface Function Inference* for details.

16.1.2 Returned Values of Interface Function Call

Status code returned	Enumeration value	Note
STATUS_OK	1000	No interface call exception.
FILE_NOT_FOUND	DI001	File not found.
VALUE_OUTOF_LEFTRANGE	1002	The parameter value exceeds the left limit. Example: If the parameter value range is [0.0, 1.0], it will be generated when the parameter value is less than 0.0.
VALUE_OUTOF_RIGHTRANGE	1003	The parameter value exceeds the right limit. Example: If the parameter value range is [0.0, 1.0], it will be generated when the parameter value is greater than 1.0.
MODEL-TYPE_ERROR	1004	Model type error.
MODEL_REGISTEREDLIMIT_ERROR	1005	Registered models beyond the limit.
MODEL_CREATE_ERROR	1006	Model creation error.
MODEL_DESTROY_ERROR	1007	Model unregistration error.
MODEL_LOAD_ERROR	1008	Model loading error.
MODEL_INIT_ERROR	1009	Model initialization error.
MODEL_INFER_ERROR	1010	Model inference error.
GET_RESULT_ERROR	1011	Error getting model result .
IMG-PATH_ERROR	1012	Invalid image file path.
IMGNULL_ERROR	1013	Empty image.
IMGCHANNEL_ERROR	1014	Image channel error.
IMGDEPTH_ERROR	1015	Image bit depth error.
IMG-SIZE_ERROR	1016	Image size error.
INCONSISTENT_IMAGEDEPTH_ERROR	1017	Inconsistent bit depths of two images.
INCONSISTENT_IMAGESIZE_ERROR	1018	Inconsistent sizes of two images.
INCONSISTENT_IMAGETYPE_ERROR	1019	Inconsistent types of two images.
IMGROI_ERROR	1020	Incorrect image ROI parameters.
IMGROI_NULL_ERROR	1021	Empty image ROI parameters.
IMG_CONVERTTYPE_ERROR	1022	Incorrect conversion parameters for image color space transformation.
CONFIG_PARAM_ERROR	1023	Configuration parameter error.
UNKNOWN_ERROR	1024	Unknown error.

16.1.3 Returned Results of Interface Function Inference

Status code returned	Enumeration value	Note
status	SDKStatus	1000 means normal, other values mean errors. Please see <i>Returned Values of Interface Function Call</i> for details.
type	Type	Model type. 0: unknown model; 1: defect segmentation; 2: instance segmentation; 3: object detection; 4: image classification.
imageShow	Bitmap	Visualization image.
masks	Bitmap[]	Result masks.
labels	int[]	Inference result labels.
bboxes	float[]	Bounding box coordinates. Every 4 values (upper left X, upper left Y, lower right X, lower right Y) are for one box.
external-Rect	float[]	Instance min circumscribed rectangle coordinates. Every 4 values (upper left X, upper left Y, lower right X, lower right Y) are for one rectangle.
internal-Rect	float[]	Instance max inscribed rectangle coordinates. Every 4 values (upper left X, upper left Y, lower right X, lower right Y) are for one rectangle.
confidence	float[]	Result confidences.

16.2 C API

16.2.1 Interface Functions

1. mmind_packInfer_create

```
int mmind_packInfer_create(Engine* engine, const char* packPath);
```

Parse the dlkpack file and create the corresponding inference engine.

Input parameters:

- engine: Inference engine information (pointer).
- packPath: dlkpack file path.

Returned value:

- int: 1000 means no error occurred; other values mean errors. Please see *Returned Values of Interface Function Call* for details.

2. mmind_packInfer_destroy

```
int mmind_packInfer_destroy(Engine engine);
```

Unregister the inference engine.

Input parameter:

- engine: Inference engine information.

Returned value:

- int: 1000 means no error occurred; other values mean errors. Please see *Returned Values of Interface Function Call* for details.

3. mmind_packInfer_infer

```
int mmind_packInfer_infer(Engine engine, Image* img, Image* segMask, Image*
↳ insegMask, int& maskNumber, int* h, int* w, int& labelNumber, int* labels, int&
↳ bboxNumber, float* bboxes, int& confidenceNumber, float* confidences);
```

Let the inference engine perform inference.

Input parameters:

- engine: Inference engine information.
- img: Input image.
- segMask: Mask result for defect segmentation.
- insegMask: Mask result for instance segmentation.
- maskNumber: Number of instance segmentation masks.
- h: Mask image height for instance segmentation.
- w: Mask image width for instance segmentation.
- labelNumber: Number of labels.
- labels: Inference result labels.
- bboxNumber: Number of bounding boxes.
- bboxes: Bounding box coordinates. Each 4 values (upper left X, upper left Y, lower right X, lower right Y) are for one box.
- confidenceNumber: Number of result confidences.
- confidences: Result confidences.

Returned value:

- int: 1000 means no error occurred; other values mean errors. Please see *Returned Values of Interface Function Call* for details.

16.2.2 Returned Values of Interface Function Call

Status code returned	Enumeration value	Note
STATUS_OK	1000	No interface call exception.
FILE_NOT_FOUND	DI001	File not found.
VALUE_OUTOF_LEFTRANGE	1002	The parameter value exceeds the left limit. Example: If the parameter value range is [0.0, 1.0], it will be generated when the parameter value is less than 0.0.
VALUE_OUTOF_RIGHTRANGE	1003	The parameter value exceeds the right limit. Example: If the parameter value range is [0.0, 1.0], it will be generated when the parameter value is greater than 1.0.
MODEL-TYPE_ERROR	1004	Model type error.
MODEL_REGISTEREDLIMIT_ERROR	1005	Registered models beyond the limit.
MODEL_CREATE_ERROR	1006	Model creation error.
MODEL_DESTROY_ERROR	1007	Model unregistration error.
MODEL_LOAD_ERROR	1008	Model loading error.
MODEL_INIT_ERROR	1009	Model initialization error.
MODEL_INFER_ERROR	1010	Model inference error.
GET_RESULT_ERROR	1011	Error getting model result .
IMG-PATH_ERROR	1012	Invalid image file path.
IMGNULL_ERROR	1013	Empty image.
IMGCHANNEL_ERROR	1014	Image channel error.
IMGDEPTH_ERROR	1015	Image bit depth error.
IMG-SIZE_ERROR	1016	Image size error.
INCONSISTENT_IMAGEDEPTH_ERROR	1017	Inconsistent bit depths of two images.
INCONSISTENT_IMAGESIZE_ERROR	1018	Inconsistent sizes of two images.
INCONSISTENT_IMAGE_TYPE_ERROR	1019	Inconsistent types of two images.
IMGROI_ERROR	1020	Incorrect image ROI parameters.
IMGROI_NULL_ERROR	1021	Empty image ROI parameters.
IMG_CONVERTTYPE_ERROR	1022	Incorrect conversion parameters for image color space transformation.
CONFIG_PARAM_ERROR	1023	Configuration parameter error.
INVALID_ENGINE_ID	2000	Invalid inference engine object.
UNKNOWN_ERROR	9000	Unknown error.

PREREQUISITES FOR USING MECH-MIND SOFTWARE

To make sure Mech-Mind software can function properly, please complete the following actions:

- *Check Interfaces and Drivers*
- *Set up Software License*
- *Check Python Version (Mech-DLK 2.1.0 and below)*
- *Turn off Windows Defender Firewall*
- *Prevent Windows from Updating (Recommended)*

17.1 Check Interfaces and Drivers

Devices that communicate with Mech-Mind software are connected to the IPC through network interfaces. Mech-Mind software also require GPU to process data.


Therefore, it is recommended to check whether your network interfaces and GPU are functioning properly first.

Check the following in Windows **Control Panel**:

- **Network and Internet**: make sure that the network interfaces used to connect to other devices are functioning properly.
- **Device Manager**: under **Display adapters** and **Network adapters**, make sure the required drivers for your network interfaces and GPU are installed.

17.2 Set up Software License

Mech-Mind uses CodeMeter from Wibu-Systems as the license system for its software.

- Plug the license dongle you received into the IPC.
- Run the CodeMeter installer received from Mech-Mind to install CodeMeter.
- Make sure that CodeMeter is running: in the system tray, check if the CodeMeter icon  is displayed.

Note:

-
- If you need to obtain a trial license that does not require a license dongle, please click [here](#).
 - If you need to update an existing license, please click [here](#).
-

17.3 Check Python Version (Mech-DLK 2.1.0 and below)

Note: Starting from version 2.2.0, the Python 3.6.5 environment is integrated into Mech-DLK. This section only applies to users of version 2.1.0 or below.

Mech-DLK requires Python 3.6.5 to function. This version of Python is automatically installed by running Mech-Mind Software Environment installation package.

If you have multiple versions of Python installed, Mech-DLK may not be able to call the correct version and thus may not function properly.

If not needed, please uninstall the other versions of Python.

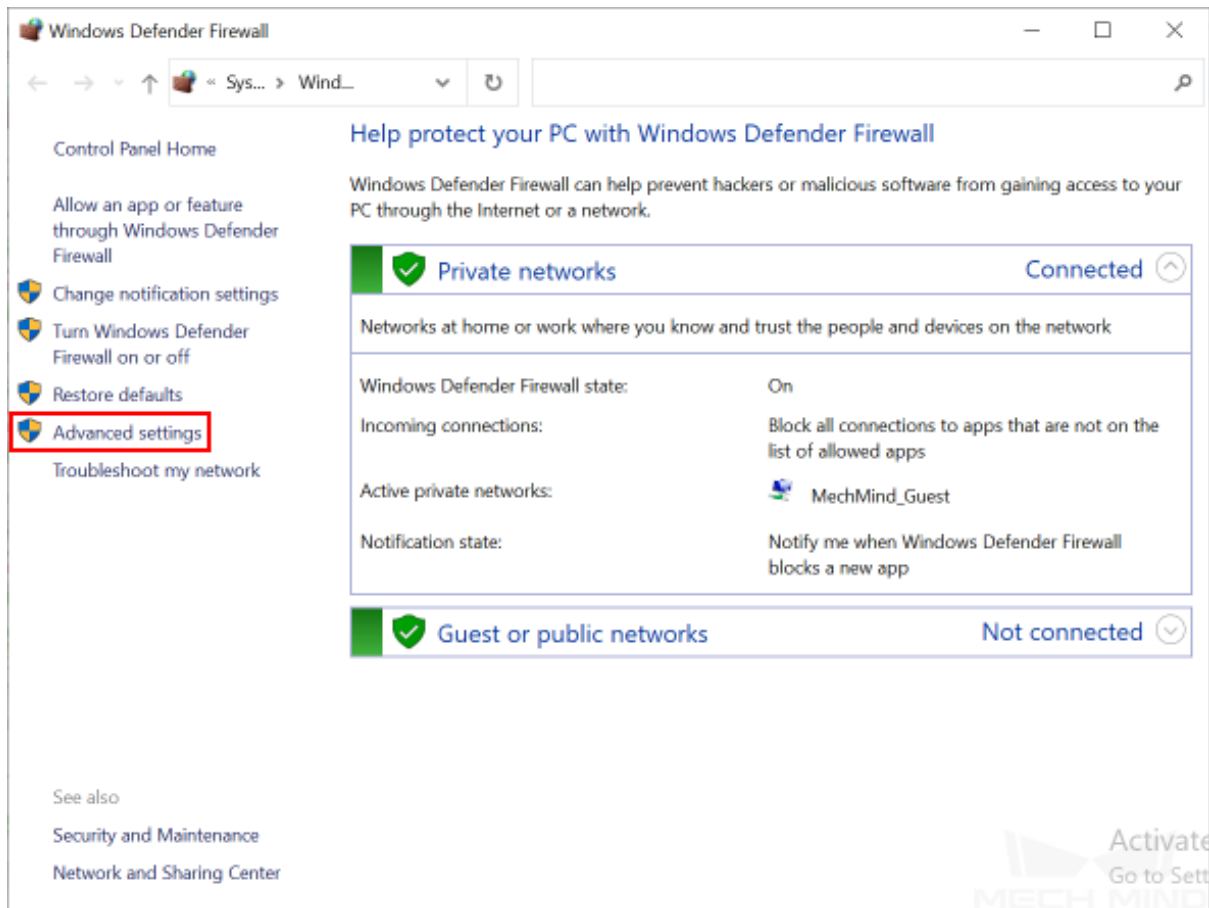
Note: Using the corresponding Python executable installer of each version to uninstall ensures that all related files are removed.

If you must keep the other versions of Python to satisfy other needs, please refer to *Designate Python Version* for detailed instructions on resolving the problem.

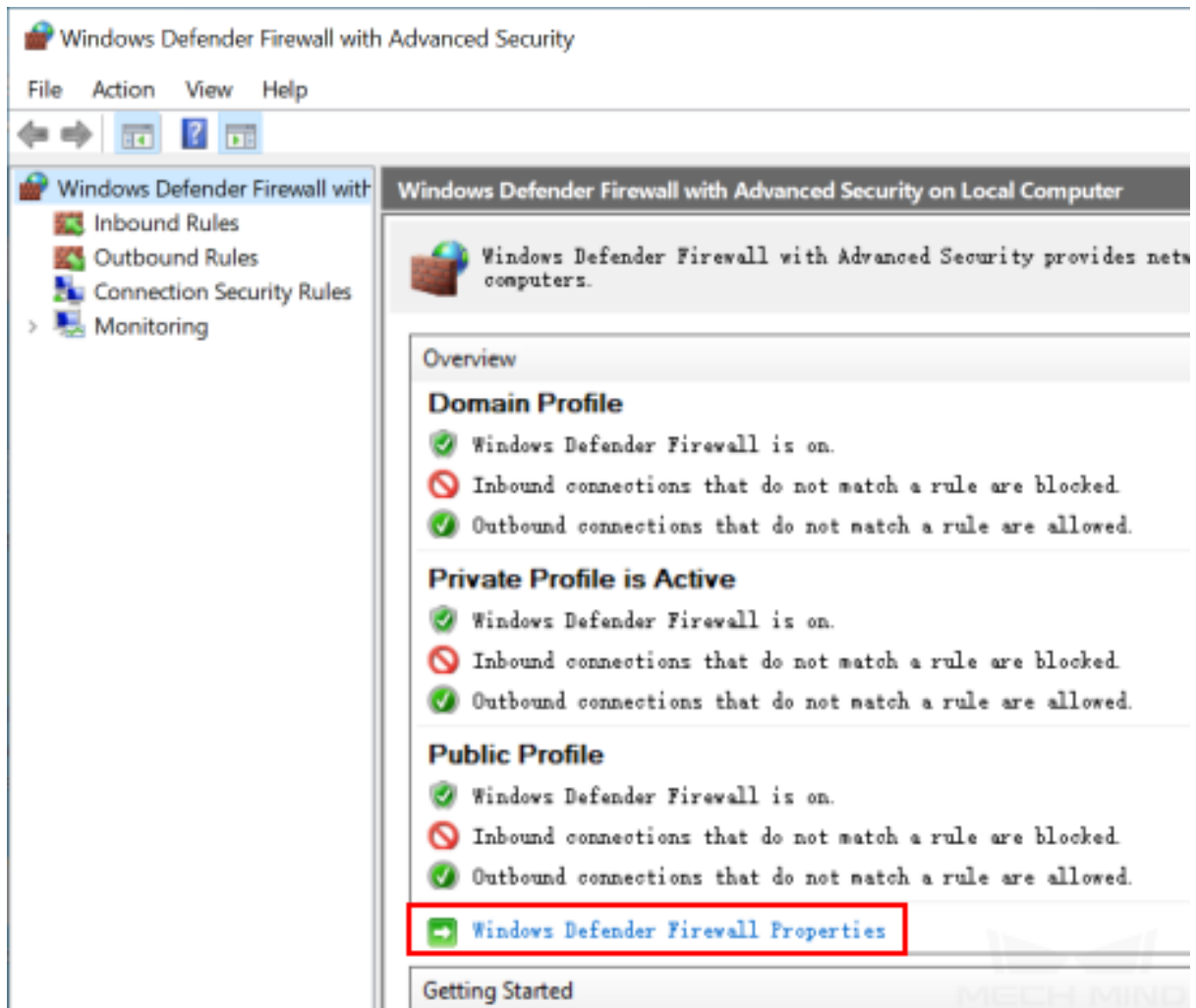
17.4 Turn off Windows Defender Firewall

Windows Defender Firewall might block the normal communication between Mech-Mind software and devices connected to the IPC. Therefore, it is necessary to turn off Windows Defender Firewall's protection for the network interfaces connected to devices that communicate with Mech-Mind software, such as Mech-Mind Eye Industrial 3D Camera, robot controller, and PLC.

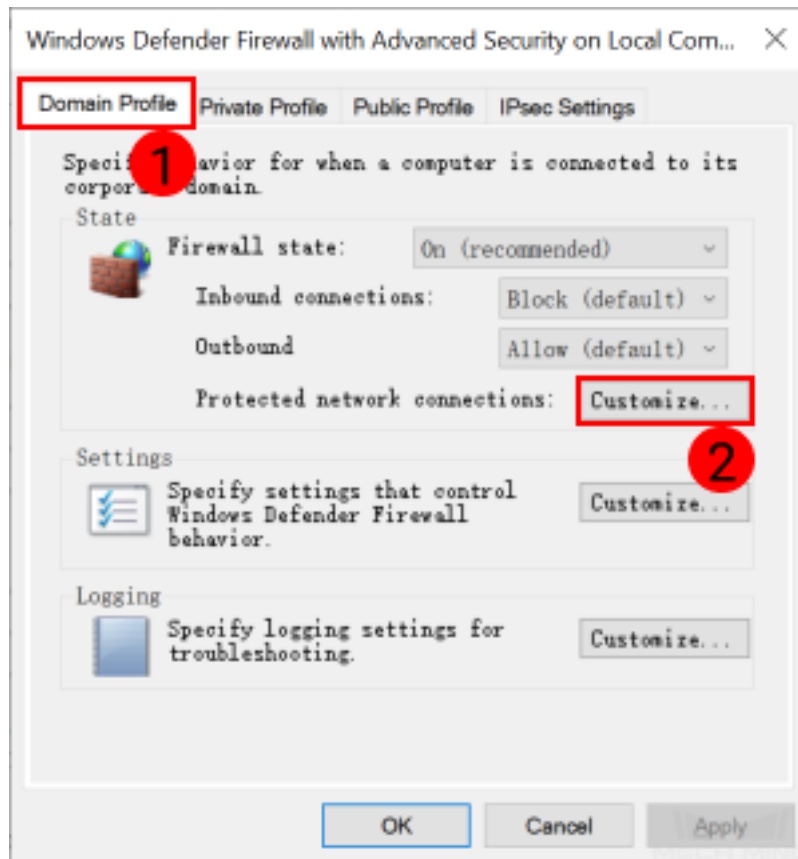
- Click on the magnifying glass icon in the taskbar, and search for **Windows Defender Firewall**.
- Click on **Windows Defender Firewall** in the search results to open it, and click on **Advanced Settings**.



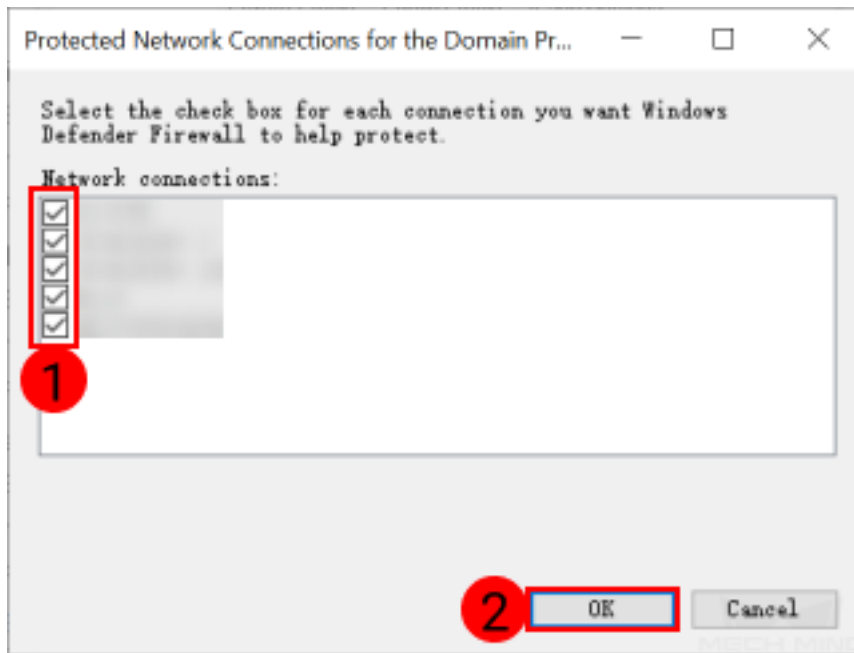
- In the pop-up window, click on **Windows Defender Firewall Properties**.



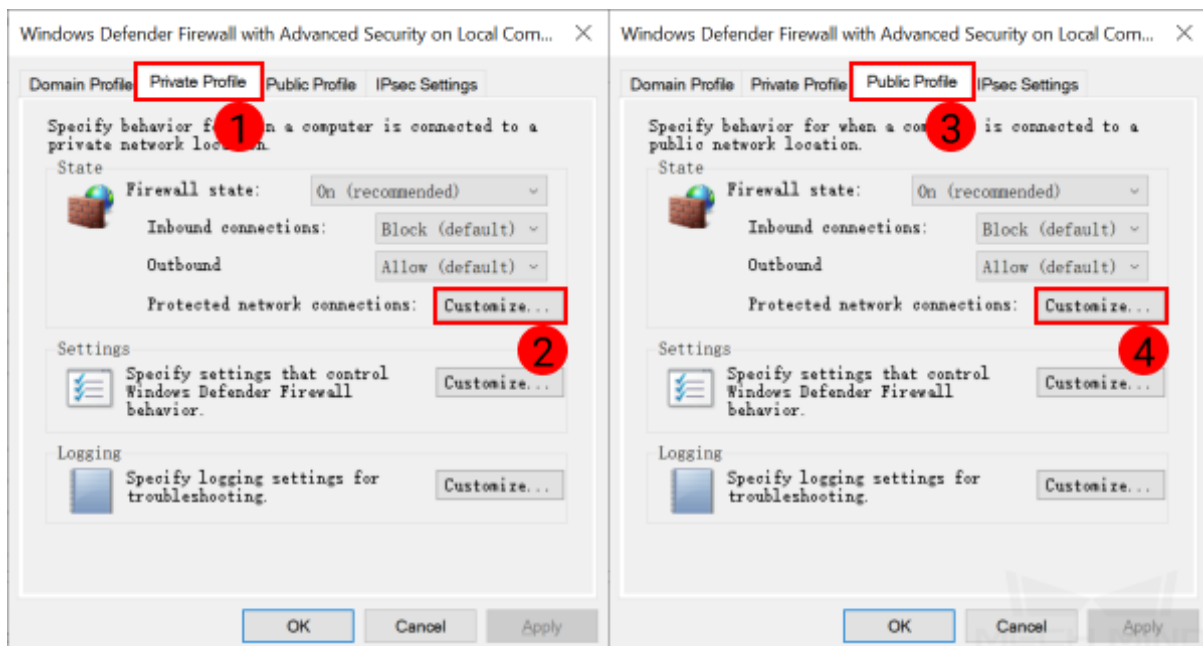
- Under the **Domain Profile** tab, click on *Customize...* next to **Protected network connections**.



- In the pop-up window, uncheck all the network interfaces connected to devices that communicate with Mech-Mind software. Then, click on *OK*.



- Repeat steps 4 and 5 for the **Private Profile** and **Public Profile** tabs.



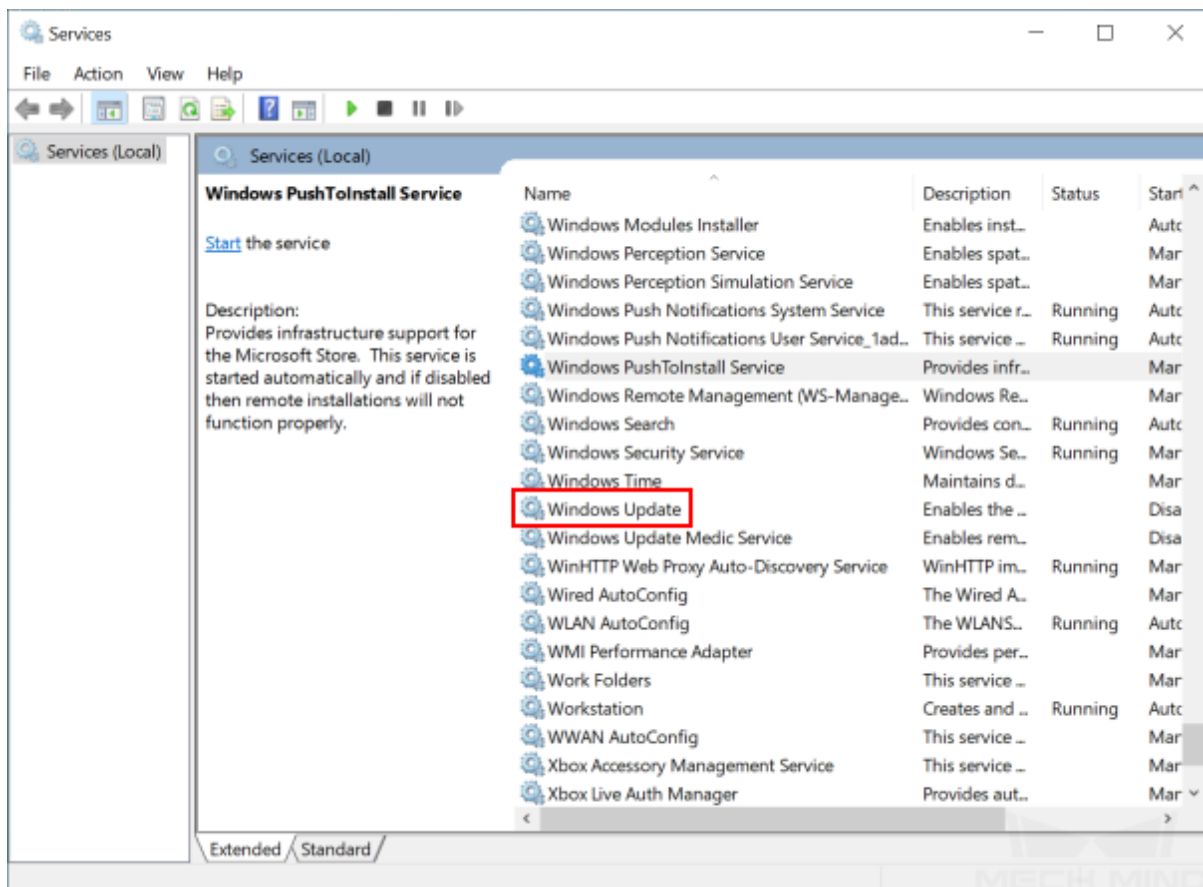
17.5 Prevent Windows from Updating (Recommended)

Windows updates might force the IPC to shut down/restart during production in order to complete the updates, which would end Mech-Mind software and thus affect normal production. Therefore, we highly recommend preventing Windows from updating to avoid unexpected downtime.

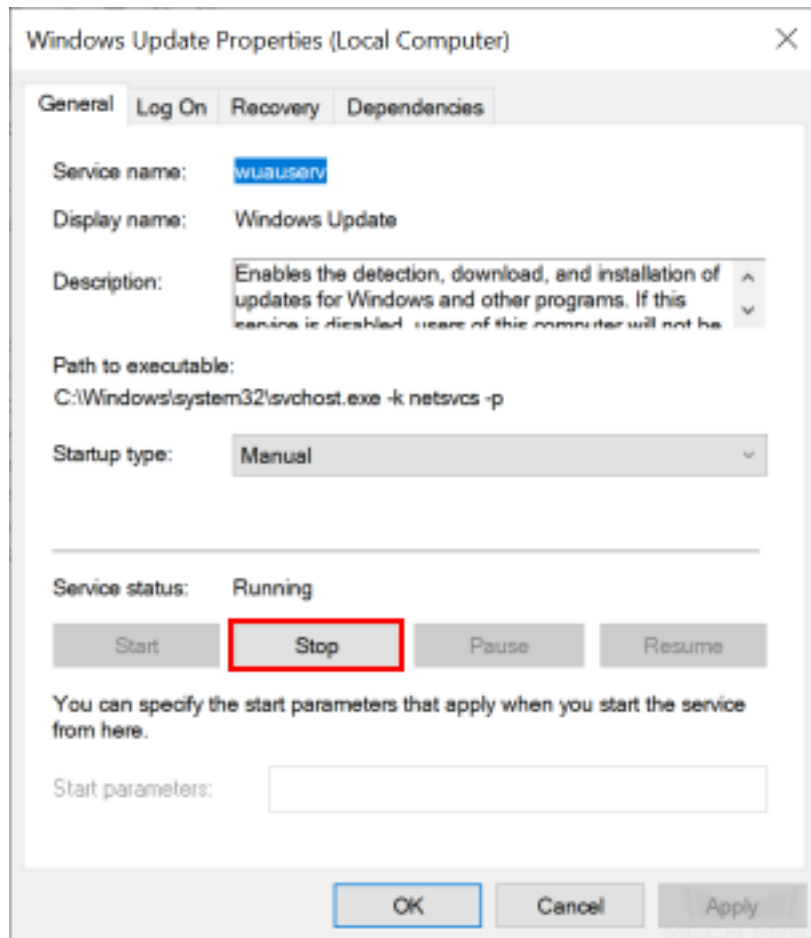
Note: If you choose to keep Windows Update enabled, please take measures to ensure that shut-down/restart occurs during planned downtime, such as setting active hours for Windows Update.

17.5.1 Disable Windows Update

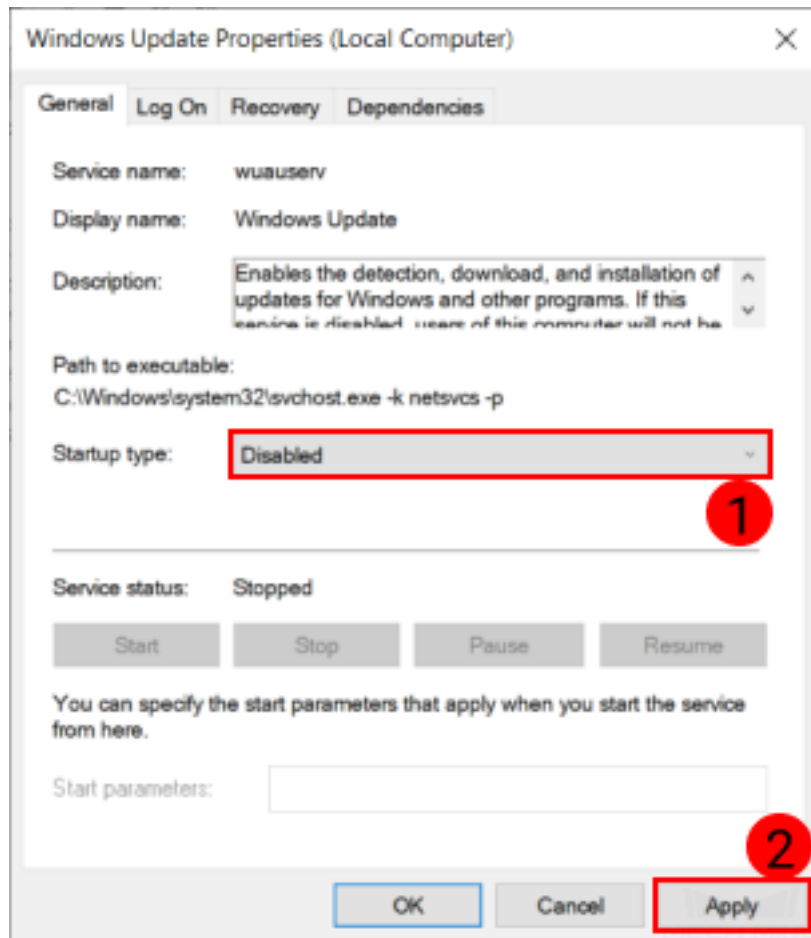
- Click on the magnifying glass icon in the taskbar, and search for **Services**.
- Click on **Services** in the search results to open it, and scroll down to find **Windows Update**. Double-click on it to open **Windows Update Properties**.



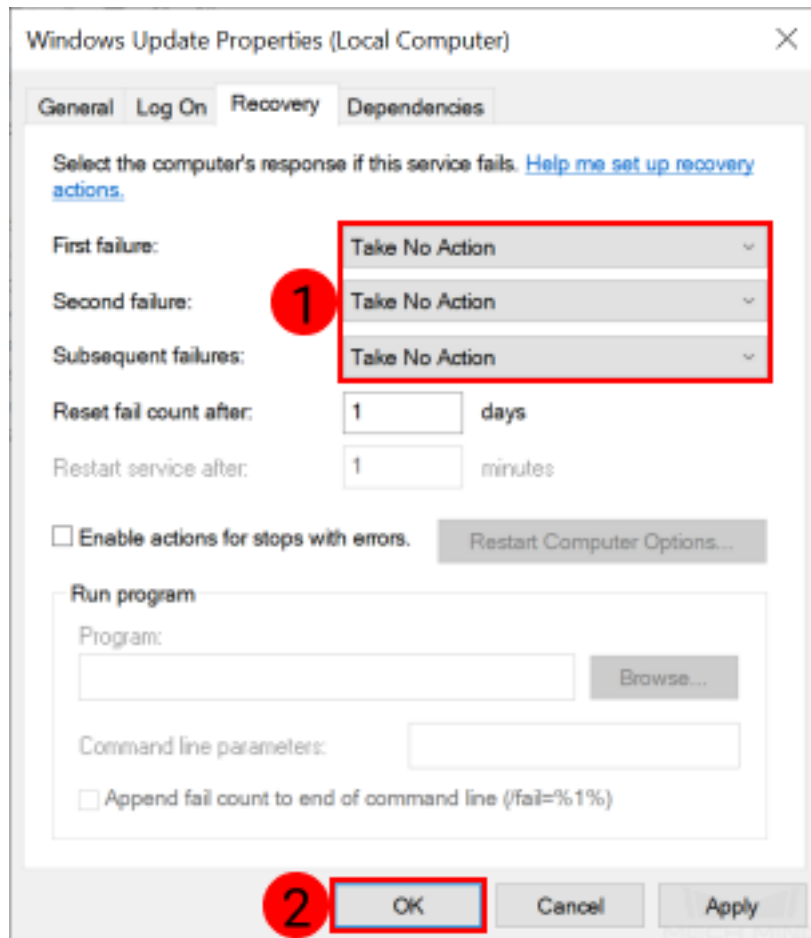
- Click on *Stop* to stop the service first.



- In the drop-down menu of **Startup type**, select **Disabled**, and then click on **Apply**.



- Click on the **Recovery** tab, and change the response for all failures to **Take No Action**. Then, click on *OK*.

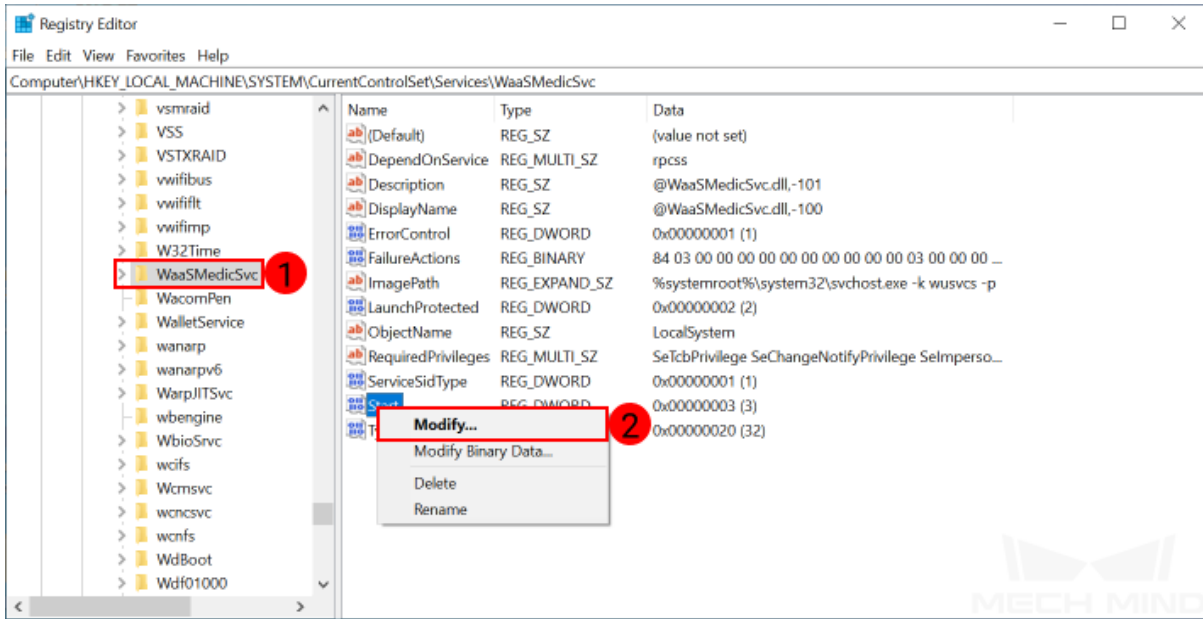


17.5.2 Disable Windows Update Medic Service

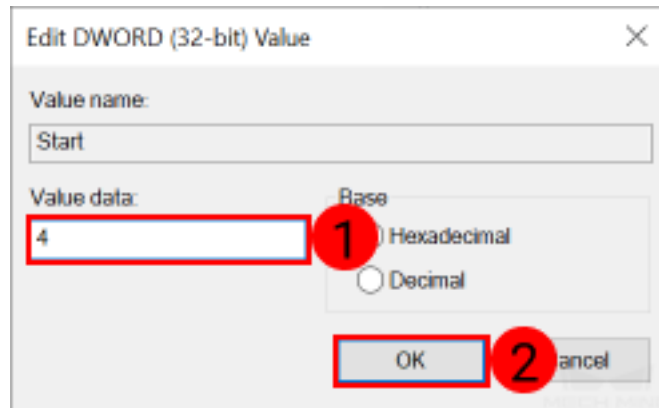
Windows Update Medic Service fixes problems in Windows Update and ensures that your computer continues to receive updates. That is, even if you have disabled Windows Update, Windows Update Medic Service will eventually re-enable it. Therefore, it is necessary to disable Windows Update Medic Service as well.

Windows does not allow you to disable Windows Update Medic Service through simple button-clicking. To disable it, you'll need to go to the Registry Editor as instructed below.

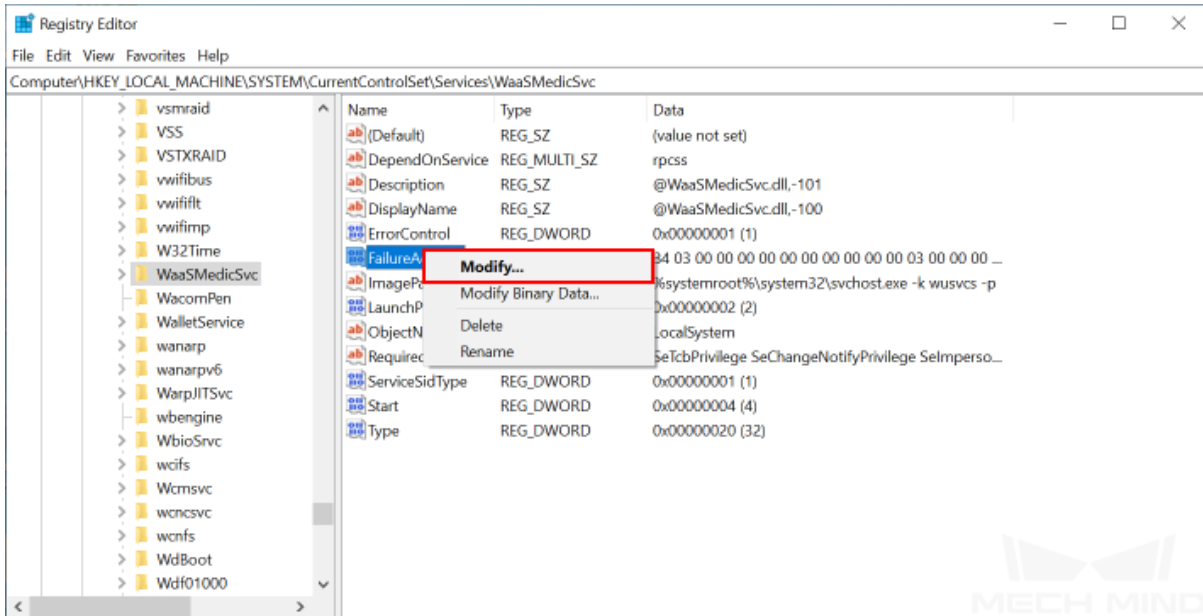
- Click on the magnifying glass icon in the taskbar, and search for **regedit**.
- Click on **Registry Editor** in the search result to open it.
- In the left pane, navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WaaSMedicSvc`. Then in the right pane, right-click on **Start** and select **Modify**.



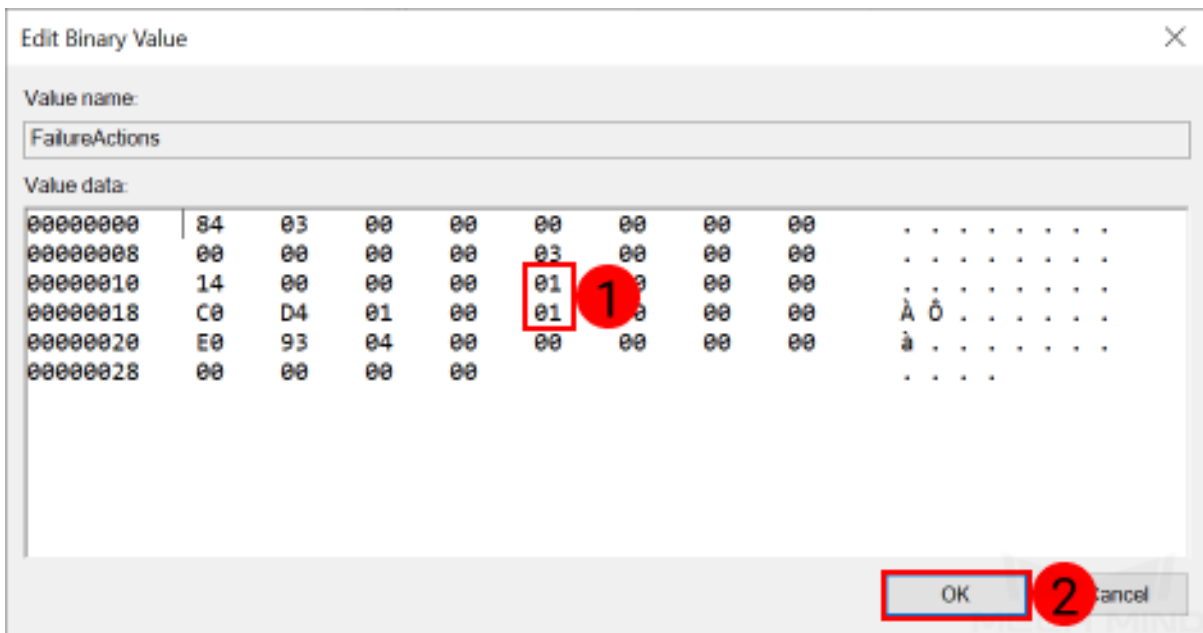
- In the pop-up window, change **Value data** to 4, and then click on *OK*.



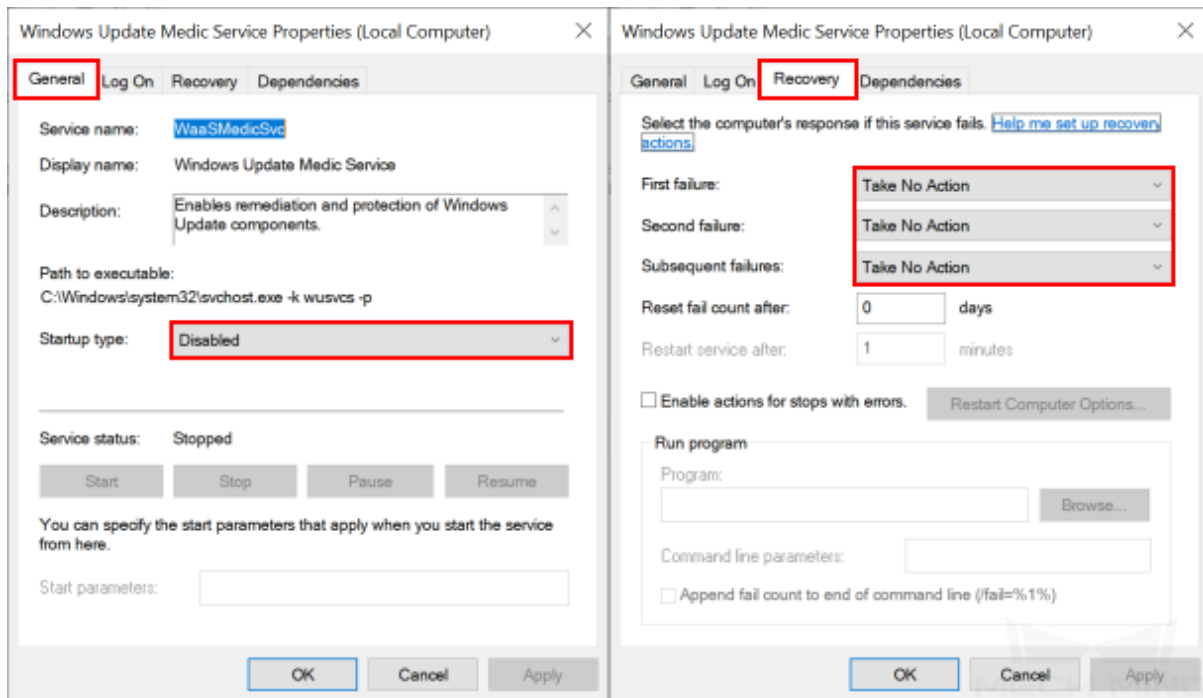
- Right-click on **FailureActions** and select **Modify**.



- In the pop-up window, change the values in the fifth column from the left for **00000010** and **00000018** to **01**. Then, click on **OK**.



- Return to **Services**, find **Windows Update Medic Service** and double-click to open it. Now the **Startup** type in the **General** tab should be **Disabled**, and all responses for failures in the **Recovery** tab should be **Take No Action**.



TERMINOLOGY

ROI A region of interest is a region selected from an image. The region is the focus of the analyses of the images. Selecting regions of interest helps reduce processing time and improve accuracy

Labeling Labeling refers to the process of selecting object features or contours from images and adding labels indicating features or defects to the selections, thus telling the model what contents it should learn.

Dataset A dataset contains the original data and labelings. In Mech-DLK, datasets are saved in dlkdb files.

Unlabeled data Original data without and labelings.

Training set The part of the dataset allocated for model training.

Validation set The part of the dataset allocated for model validation.

OK image In defect detection, an OK image is an image that contains no defects.

NG image in defect detection, an NG image is an image that contains any defects.

Training Training refers to the process of letting the model learn on the training set.

Validation Validation refers to the process of verifying the trained model on the validation set.

Accuracy Accuracy refers to the ratio of the number of correctly predicted samples to the total number of samples in the validation set when validating the trained model.

Loss Loss is a measure of the inconsistency between the model's predictions on the validation set and the ground truth of the validation set.

Epochs The number of times the model goes through the training set when training.

COMPATIBILITY

19.1 Instance Segmentation

Mech-Vision version	Deep learning environment version	Mech-Vision Step	Mech-DLK version for model	Model and configuration file extensions	
1.4.0	1.4.0	Instance Segmentation *	1.4.0	.pth/.py	
1.5.x	2.0.0/2.1.0		1.4.0	.pth/.py	
1.6.0	2.0.0/2.1.0		Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	.dllkmp/.dllkcfg
				1.4.0	
	No deep learning environment required		2.0.0/2.1.0		
1.6.1	No deep learning environment required	Deep Learning Model Package CPU Inference/ Deep Learning Inference (DLK 2.2.0+)	2.2.1	.dllk-packC/.dllkpack	

* Please start the deep learning server for the Step.

19.2 Image Classification

Mech-Vision version	Deep learning environment version	Mech-Vision Step	Mech-DLK version for model	Model and configuration file extensions
1.4.0	1.4.0	Image Classification *	1.4.0	.pth/.json
1.5.x	2.0.0/2.1.0	Image Classification *	1.4.0	.pth/.json
		Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	.dlkpack
1.6.0	2.0.0/2.1.0	Image Classification *	1.4.0	
	No deep learning environment required	Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	
		Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.0	
1.6.1	No deep learning environment required	Deep Learning Model Package CPU Inference/ Deep Learning Inference (DLK 2.2.0+)	2.2.1	.dlk-packC/.dlkpack

* Please start the deep learning server for the Step.

19.3 Object Detection

Mech-Vision version	Deep learning environment version	Mech-Vision Step	Mech-DLK version for model	Model and configuration file extensions
1.4.0	1.4.0	Object Detection *	1.4.0	.pth/.py
1.5.x	2.0.0/2.1.0	Object Detection *	1.4.0	.pth/.py
		Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	.dlkpack
1.6.0	2.0.0/2.1.0	Object Detection *	1.4.0	
	No deep learning environment required	Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	
		Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.0	
1.6.1	No deep learning environment required	Deep Learning Model Package CPU Inference/ Deep Learning Inference (DLK 2.2.0+)	2.2.1	.dlk-packC/.dlkpack

* Please start the deep learning server for the Step.

19.4 Defect Segmentation

Mech-Vision version	Deep learning environment version	Mech-Vision Step	Mech-DLK version for model	Model and configuration file extensions
1.4.0	1.4.0	Defect Detection *	1.4.0	.pth/.py
1.5.x	2.0.0/2.1.0	Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	.dlkpack
1.6.0	No deep learning environment required	Deep Learning Inference (DLK 2.1.0/2.0.0)	2.0.0/2.1.0	.dlkpack
		Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.0	
1.6.1	No deep learning environment required	Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.1	.dlkpack

* Please start the deep learning server for the Step.

19.5 Fast Positioning

Mech-Vision version	Deep learning environment version	Mech-Vision Step	Mech-DLK version for model	Model and configuration file extension
1.6.0	No deep learning environment required	Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.0	.dlkpack
1.6.1	No deep learning environment required	Deep Learning Model Package Inference (DLK 2.2.0+)	2.2.1	.dlkpack

This section provides information that help you solve problems encountered in using Mech-Mind Vision System.

20.1 Update Software License


CodeMeter will notify you if your license(s) is about to expire. In such case, please contact Mech-Mind Technical Support to request an update.

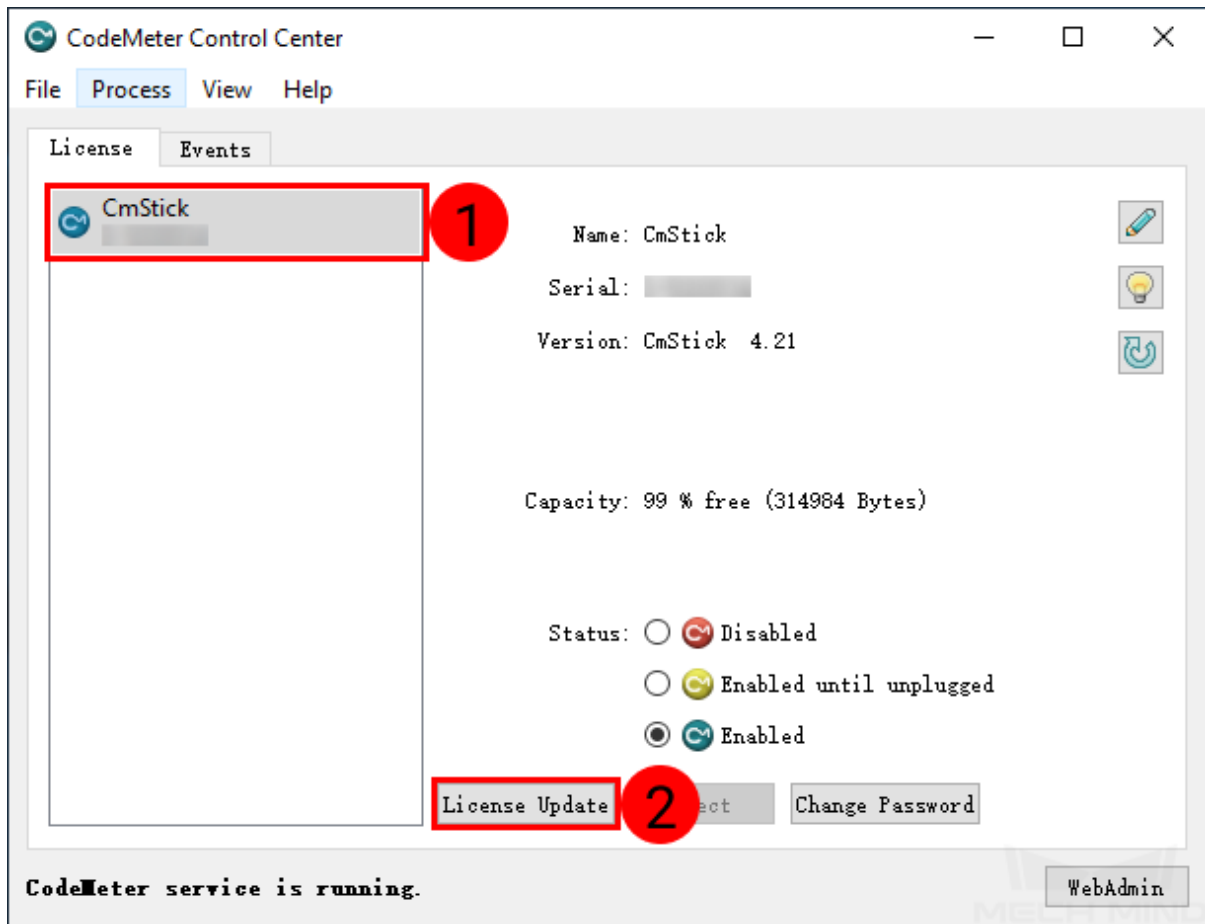
Once the update has been arranged, you' ll need to export a license request file to send to Mech-Mind first. Then, Mech-Mind will send back a license update file that extends the duration of your license.

20.1.1 Export License Request File

Note: Before proceeding, please make sure that:

- CodeMeter is installed;
 - License dongle(s) is inserted.
-

1. Click on  in the system tray to open CodeMeter Control Center.
2. Select the license dongle whose license needs to be updated, and click on *License Update*.



3. Click on *Next* in the pop-up window.

← CmFAS Assistant

Welcome to the CmFAS Assistant!

The CodeMeter Field Activation Service (CmFAS) assistant helps you adding, changing and deleting licenses from the license management system CodeMeter.

With the CmFAS assistant you can create license request files, which you can send to the vendor of the software by email. You can also import the received license update files with the CmFAS assistant into the license management and create a receipt of the import for the vendor.

Next >

Help

4. Select **Create license request**, and then click on *Next*.

← CmFAS Assistant

Please select the desired action

Create license request

1

Choose this option if you want to create a license request file in order to send it to the vendor of the software.

Import license update

Choose this option, if you received a license update file from the software vendor and want to import this file.

Create receipt

Choose this option if you want to confirm the successful import of a license update file for the software vendor.

2

Next >

Help

5. Select **Extend existing license**, and then click on *Next*.

← CmFAS Assistant

Please choose an option for the license request

Extend existing license

1

Choose this option if you want to change an existing license or to add new licenses to an existing license of the same vendor.

Add license of a new vendor

Choose this option if you want to add a new license and there are no licenses from this vendor in the selected license container.

2

Next >

Help

6. Choose **Mech-Mind** as the vendor, and then click on *Next*.

← CmFAS Assistant

Please choose the vendor


Mech-Mind. (103130) **1**

Select the software vendor to which you want to send the license request file. The vendor will only see the data which you select here. So you can ensure that the vendor doesn't see which other licenses from other vendors you have.

2

Note: The company code may differ.


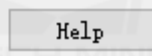
7. On the next page, click on .. to select a location for saving the license request file, and then click on *Commit*.

←  CmFAS Assistant

Please select the file name

Select a file name for storing the license request file. Then click on 'commit' to create the file. You can then send this file to the vendor by email.

  Commit 

Hint: If licenses on multiple license dongles need to be updated, please repeat steps 2 to 7 to export a license request file for each dongle.


8. Send the license request file to Mech-Mind Technical Support.

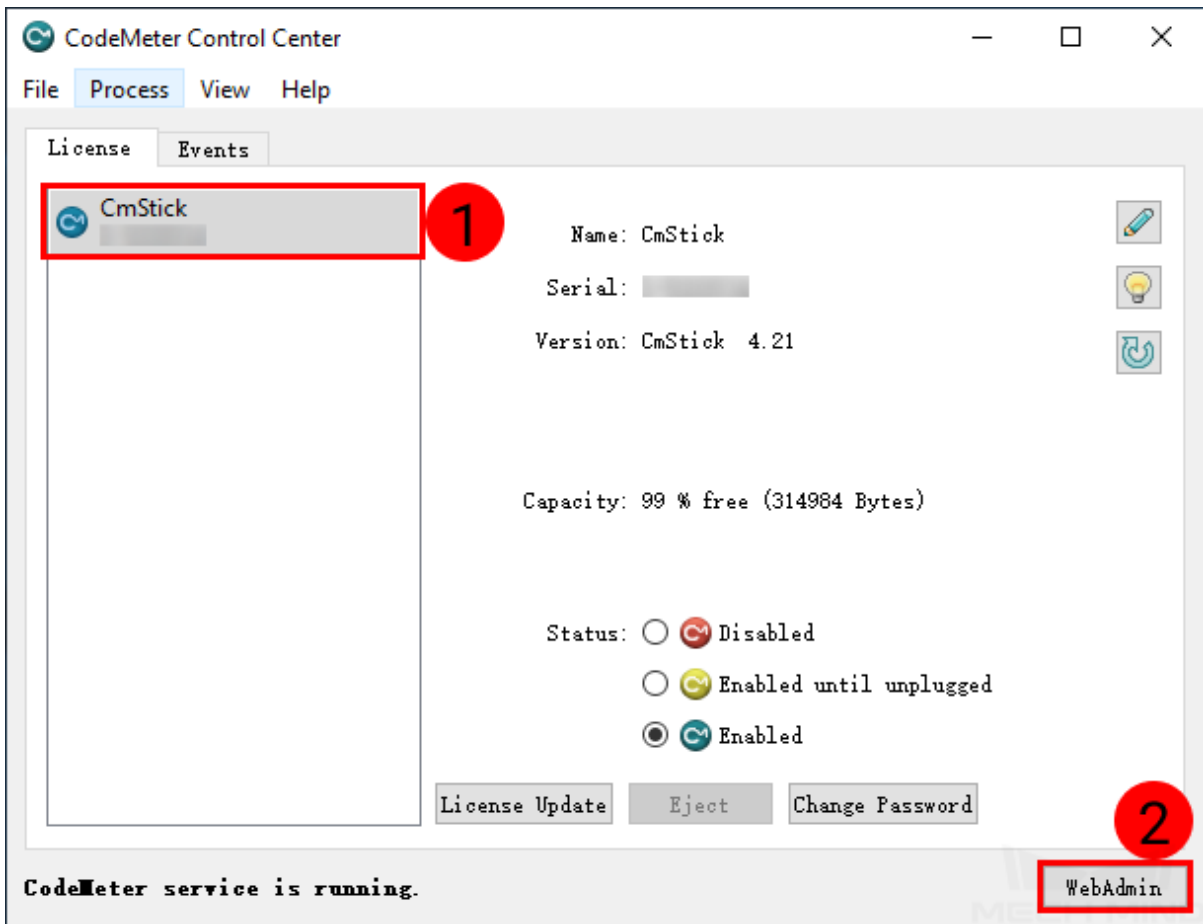
20.1.2 Update the License

After you send the license request file, Mech-Mind will send back a license update file in WIBUCMRAU format.

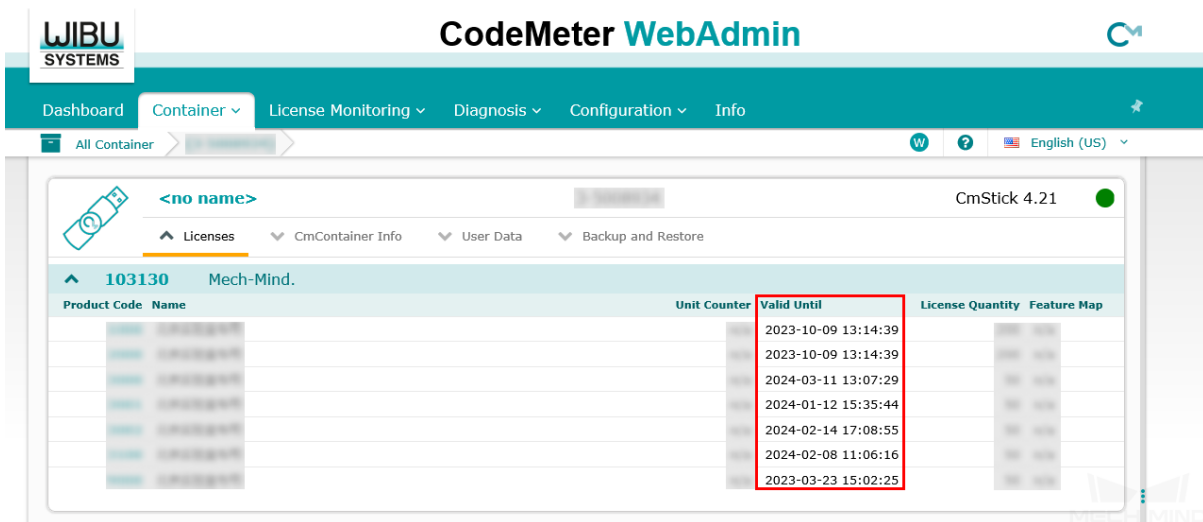
Double-click on this file to update the corresponding license.

Follow the steps below to check if the license has been successfully updated.

1. Click on  in the system tray to open CodeMeter Control Center.
2. Select the license dongle you'd like to check, and then click on *WebAdmin* in the lower right.




3. The CodeMeter WebAdmin webpage will be opened, and you can check the date under Valid Until to see if the license has been updated.

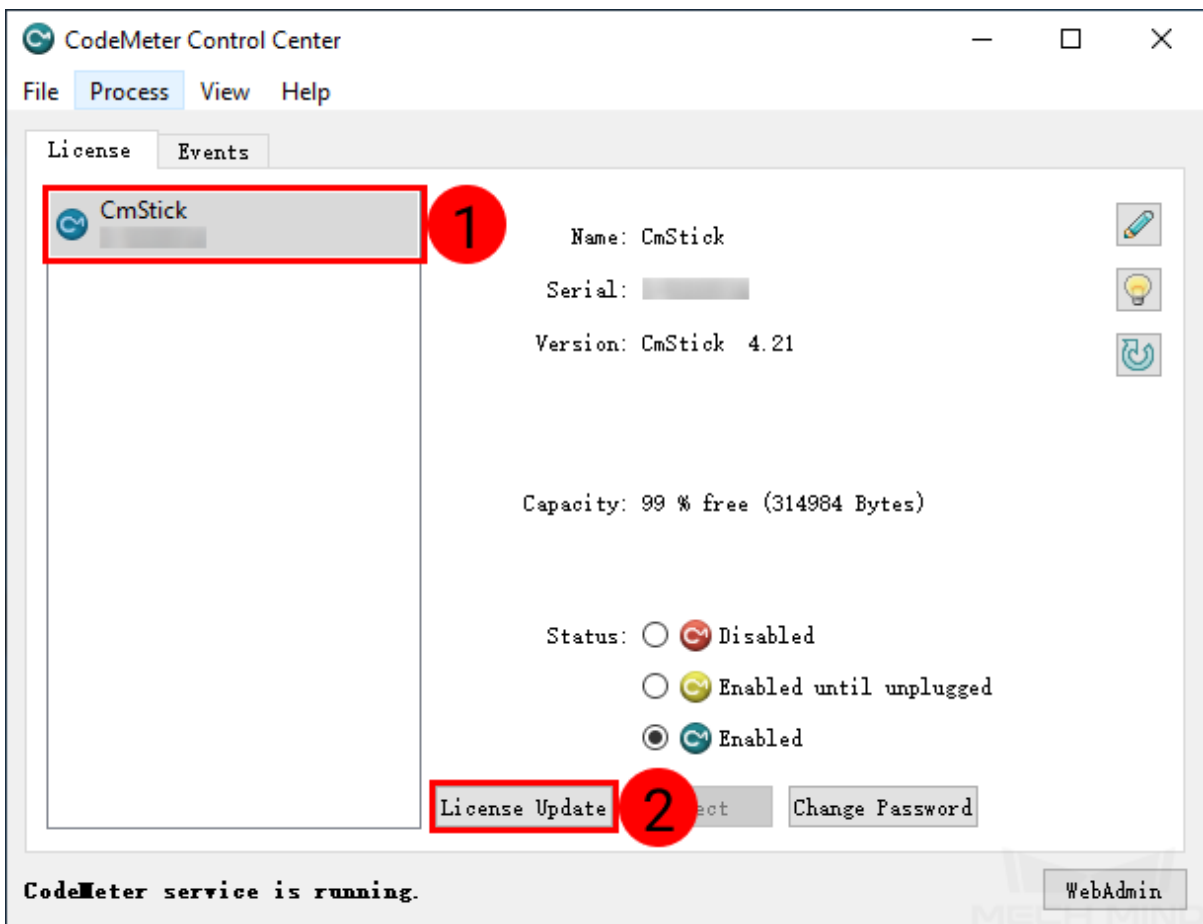


Note:

- The **Valid Until** date for a perpetual license is **n/a**.
- You can select a different license dongle to view by clicking on the **Container** tab at the top.

If for some reason, double-clicking on the WIBUCMRAU file does not update your license, please follow the steps below to manually import the license update file.

1. Click on  in the system tray to open CodeMeter Control Center.
2. Select the license dongle whose license needs to be updated, and click on *License Update*.



3. Click on *Next* in the pop-up window.

← CmFAS Assistant

Welcome to the CmFAS Assistant!

The CodeMeter Field Activation Service (CmFAS) assistant helps you adding, changing and deleting licenses from the license management system CodeMeter.

With the CmFAS assistant you can create license request files, which you can send to the vendor of the software by email. You can also import the received license update files with the CmFAS assistant into the license management and create a receipt of the import for the vendor.

Next >

Help

4. Select **Import license update**, and then click on *Next*.

← CmFAS Assistant

Please select the desired action

Create license request

Choose this option if you want to create a license request file in order to send it to the vendor of the software.

Import license update

Choose this option, if you received a license update file from the software vendor and want to import this file.

Create receipt

Choose this option if you want to confirm the successful import of a license update file for the software vendor.

Next >

Help

5. Click on .. to select the license update file, and then click on *Commit*.

← CmFAS Assistant

Please select the file name



Select a file under which the license update file is stored on your computer. Then click on 'commit' to import the new licenses.

2
Commit
Help

20.2 Obtain a Trial Software License

Mech-Mind uses CodeMeter from Wibu-Systems as the license system for its software.

We provide trial software licenses that do not require a license dongle. Please contact Mech-Mind Sales to obtain a **ticket** code before proceeding.

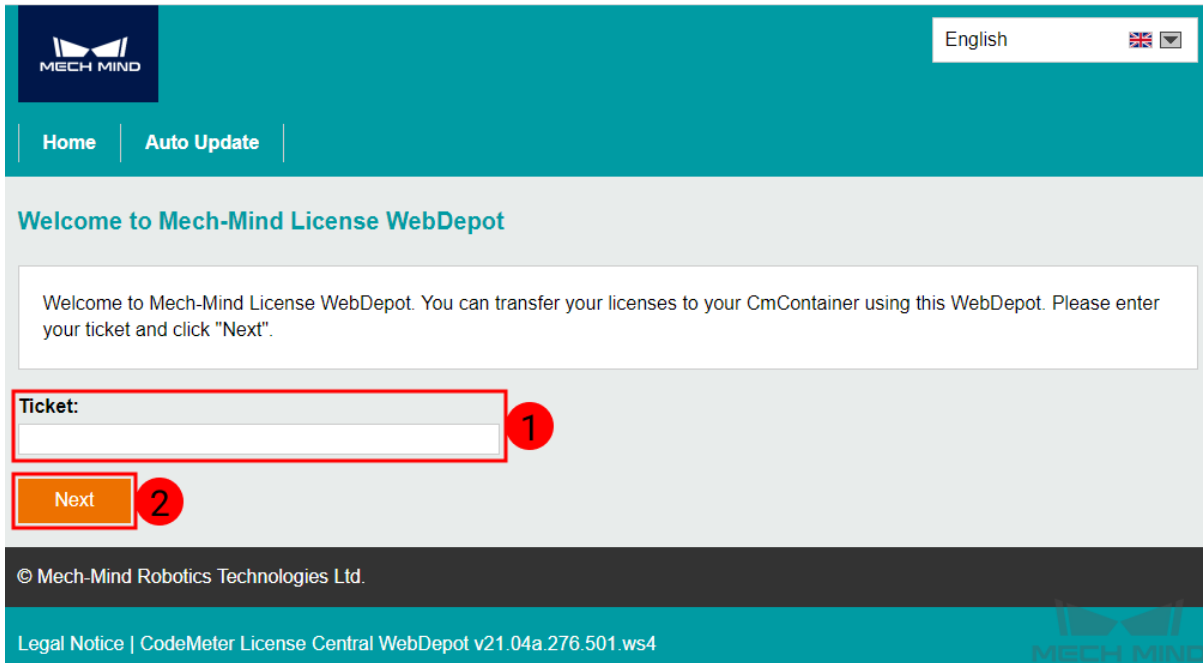
20.2.1 Install CodeMeter


Please run the CodeMeter installer received from Mech-Mind to install CodeMeter.

20.2.2 Activate the License

To activate your license, the ticket code sent to you via email is required. It is a 25-character string made up of numbers, alphabets, and hyphens.

1. Go to [Mech-Mind License WebDepot](#).
2. Copy and paste the ticket code into the **Ticket** field, and click on *Next*.



English 

Home | Auto Update

Welcome to Mech-Mind License WebDepot

Welcome to Mech-Mind License WebDepot. You can transfer your licenses to your CmContainer using this WebDepot. Please enter your ticket and click "Next".

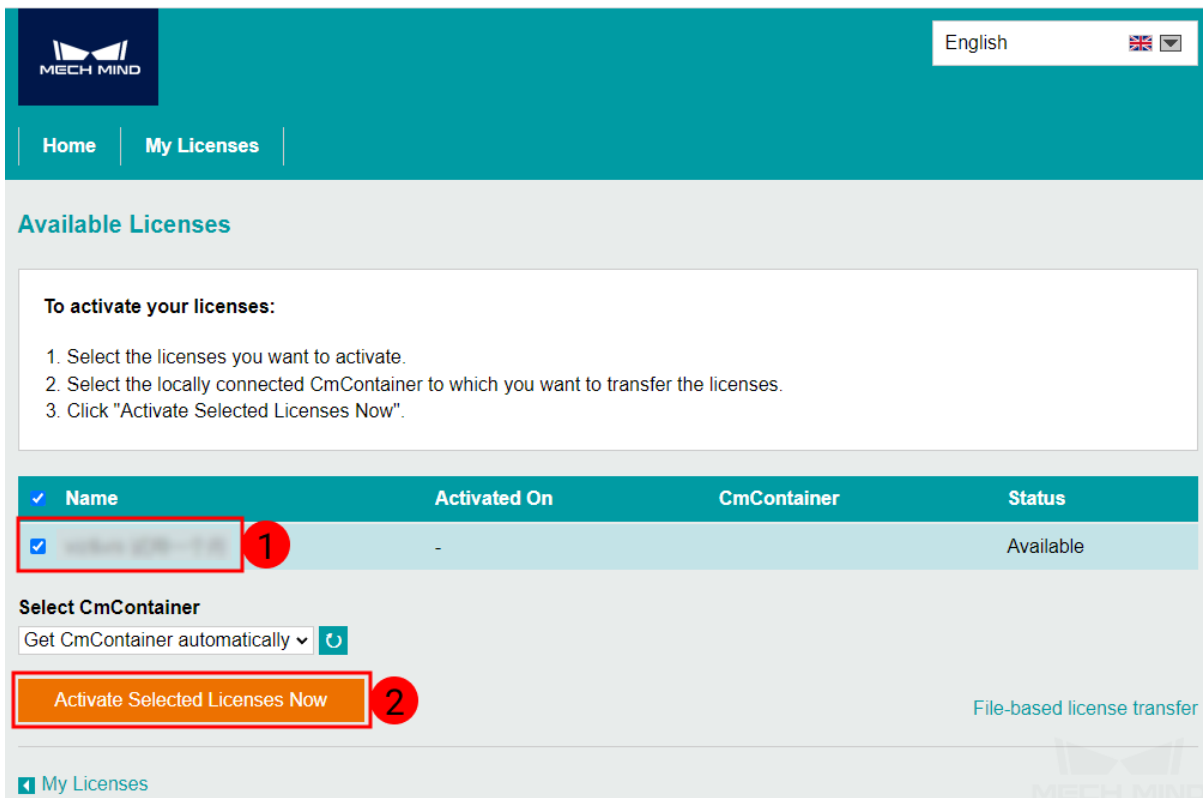
Ticket:


Next

© Mech-Mind Robotics Technologies Ltd.

Legal Notice | CodeMeter License Central WebDepot v21.04a.276.501.ws4

3. Select the license, and click on *Activate Selected Licenses Now*.



English 

Home | My Licenses


Available Licenses

To activate your licenses:

1. Select the licenses you want to activate.
2. Select the locally connected CmContainer to which you want to transfer the licenses.
3. Click "Activate Selected Licenses Now".

<input checked="" type="checkbox"/>	Name	Activated On	CmContainer	Status
<input checked="" type="checkbox"/>	MECH MIND	-		Available

Select CmContainer

Get CmContainer automatically 

Activate Selected Licenses Now

File-based license transfer

[My Licenses](#)

4. When the pop-up window displays the following message, click on *OK* to complete the license

transfer:

License transfer completed successfully!

Online License Transfer

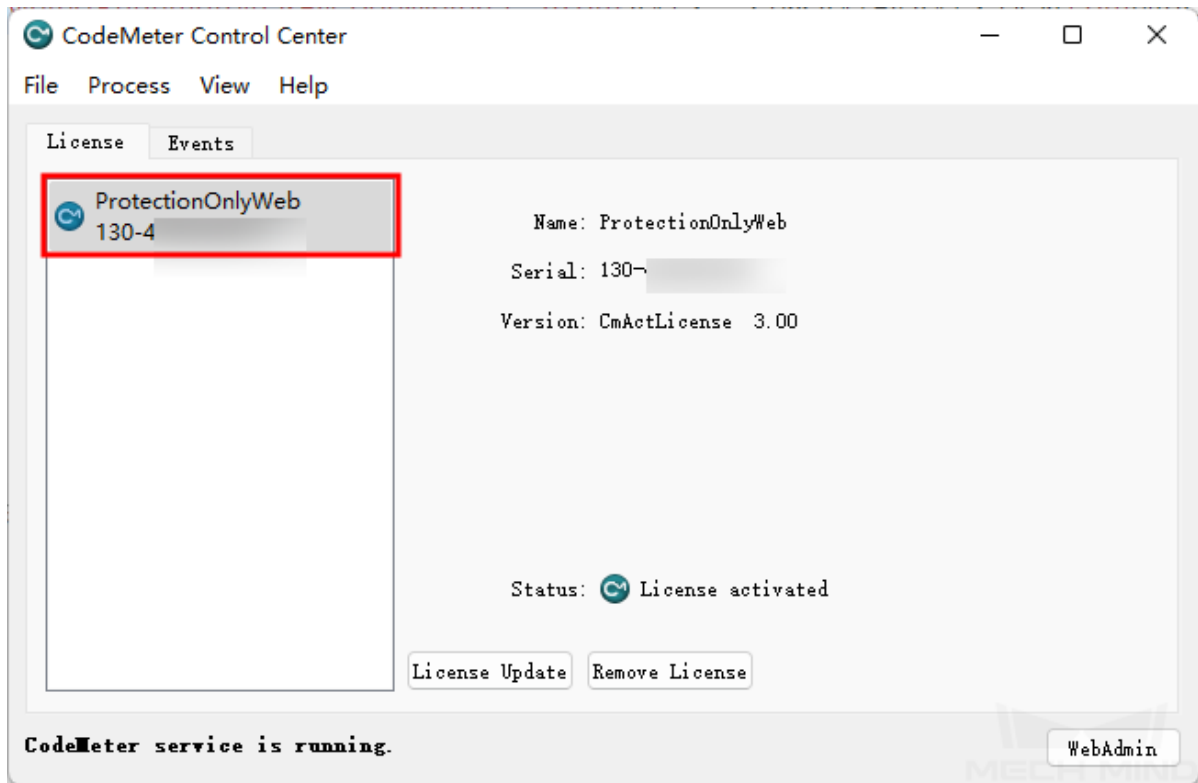
Starting license transfer.
Downloading license template.
Registering license template.
Creating license request.
Downloading license update.
Importing license update to CmContainer.
Creating receipt.
Uploading receipt.



License transfer completed successfully!

OK

5. Open CodeMeter Control Center, and you should see the license displayed.



20.3 Designate Python Version

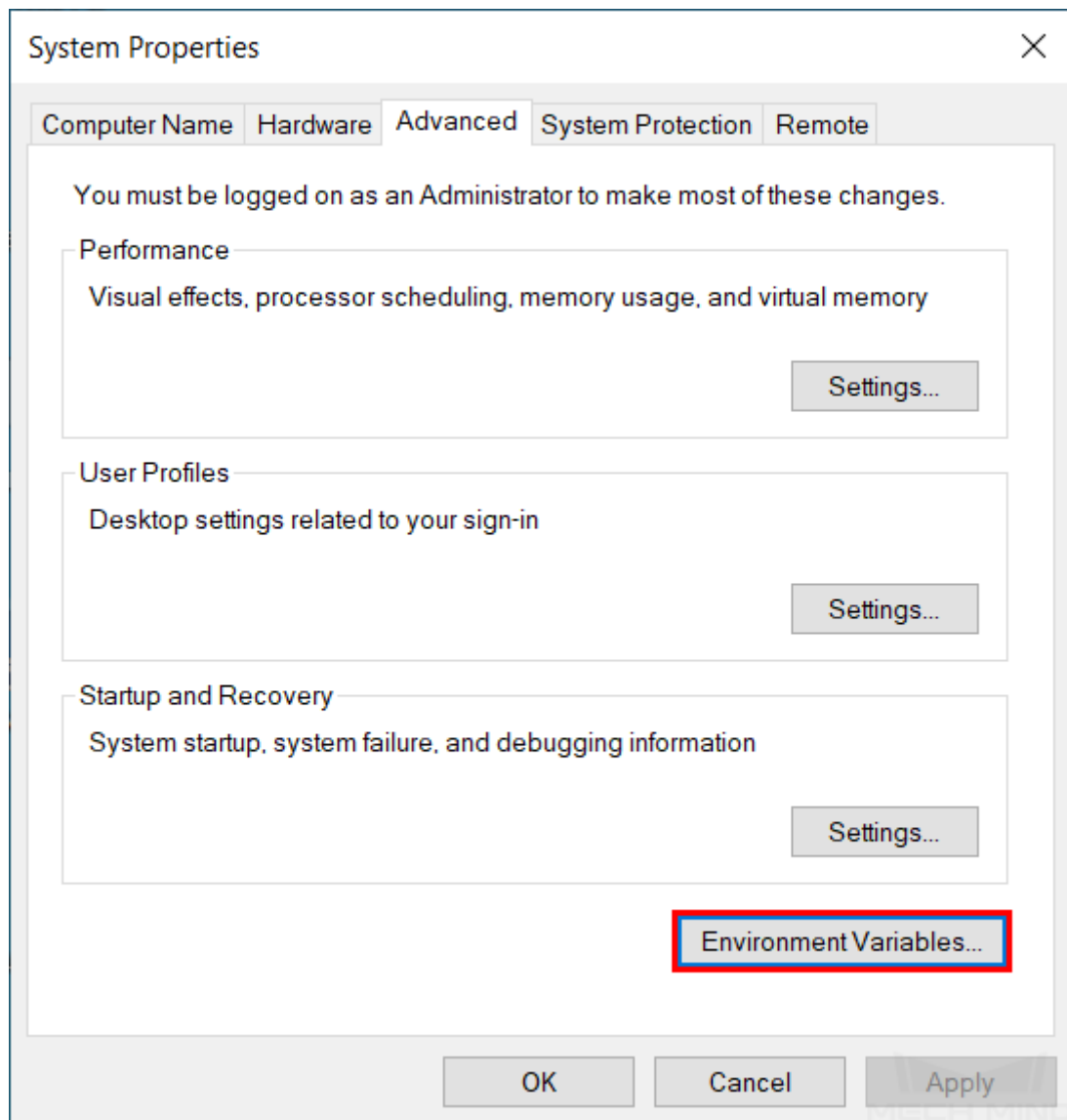
Mech-DLK requires Python 3.6.5 to function. If you have multiple versions of Python installed, Mech-DLK may not be able to call the correct version and thus may not function properly.

If not needed, please uninstall the other versions of Python.

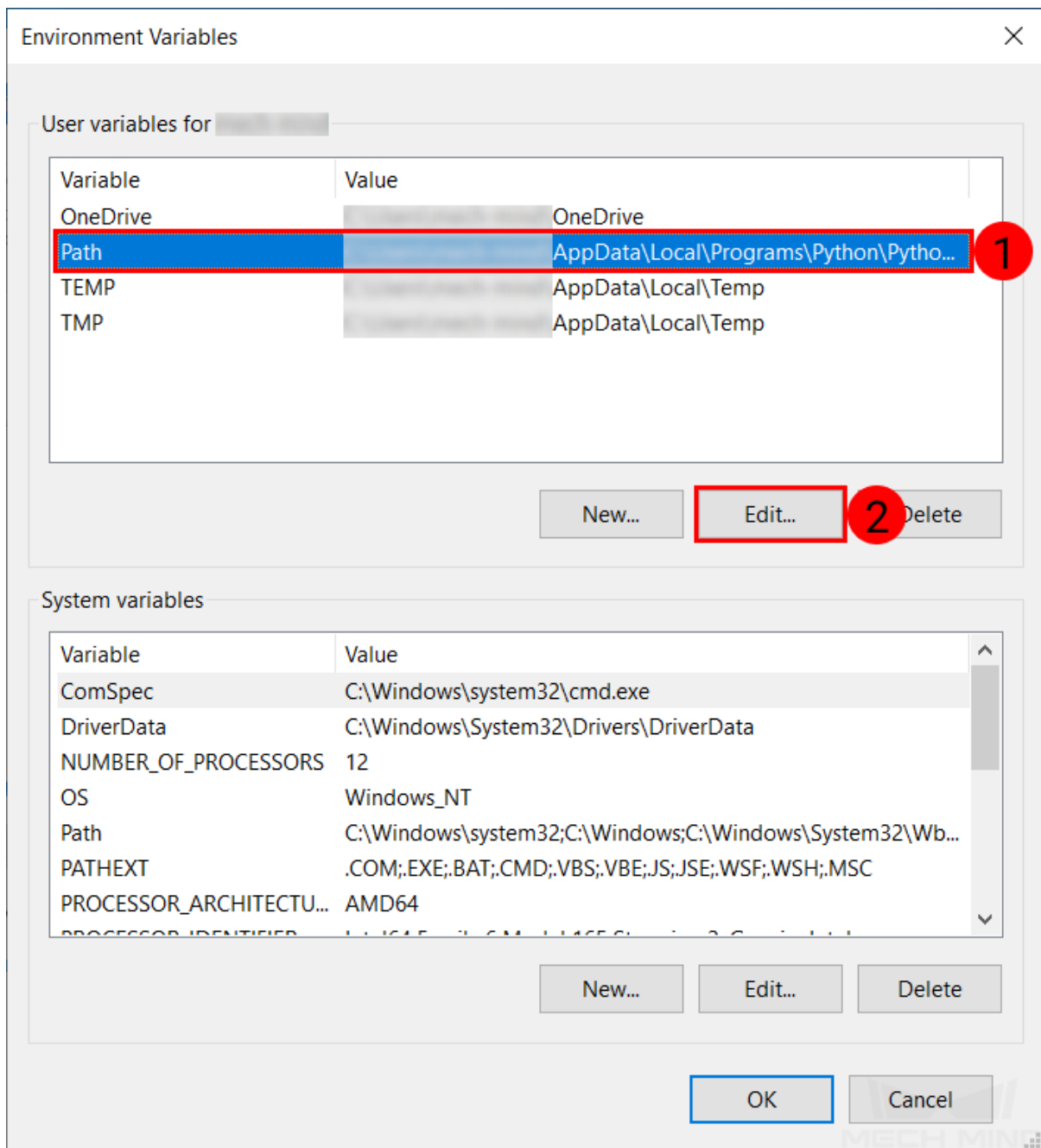
Note: Using the corresponding Python executable installer of each version to uninstall ensures that all related files are removed.

If you must keep the other versions of Python, please follow the steps below to make sure Mech-DLK can call the correct version.

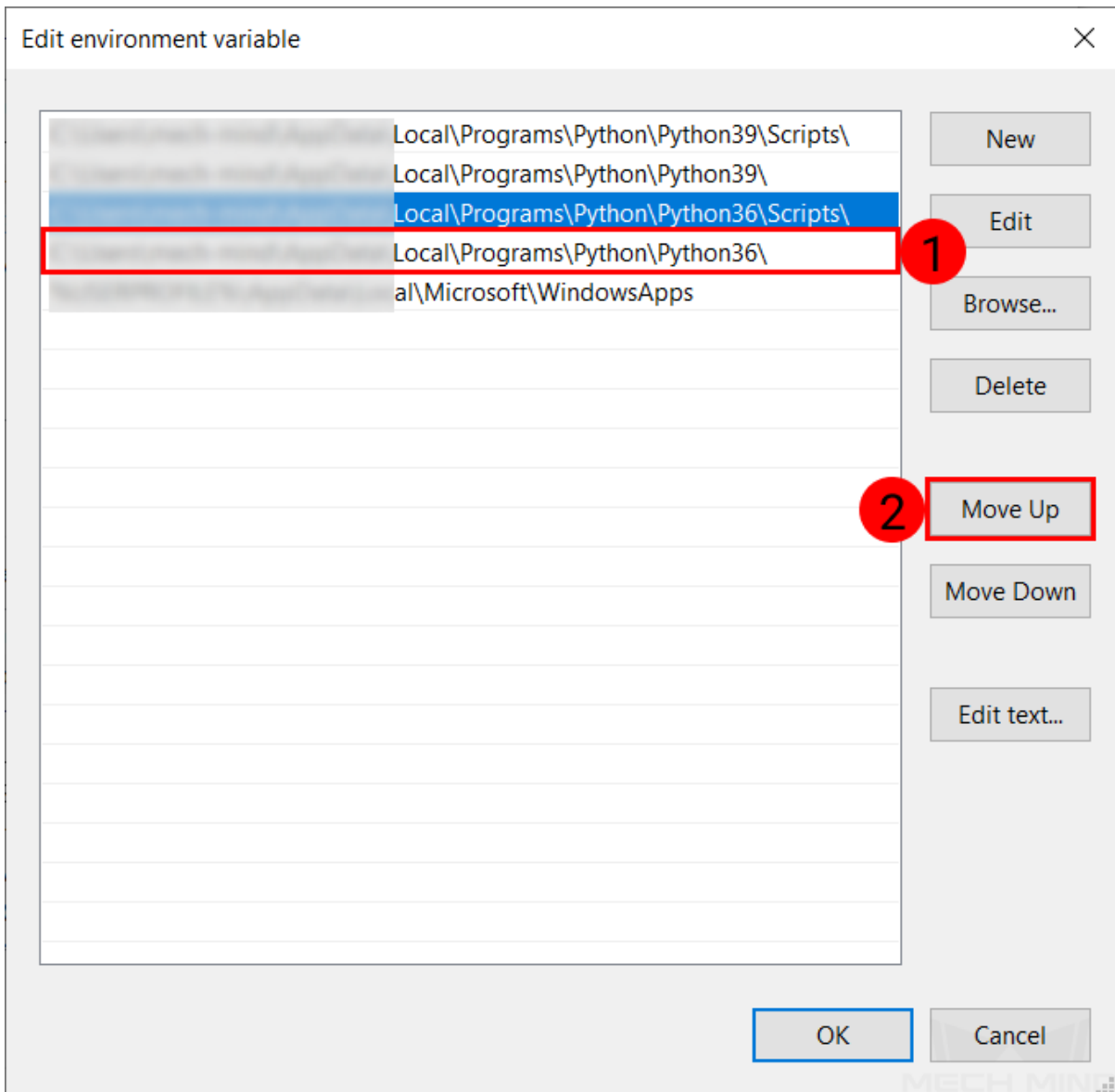
1. Right-click on **My PC** and select **Properties**.
2. Click on **Advanced system settings**, which may be on the left or right of the pop-up window, depending on the edition of Windows you are using.
3. Click on *Environment Variables*.



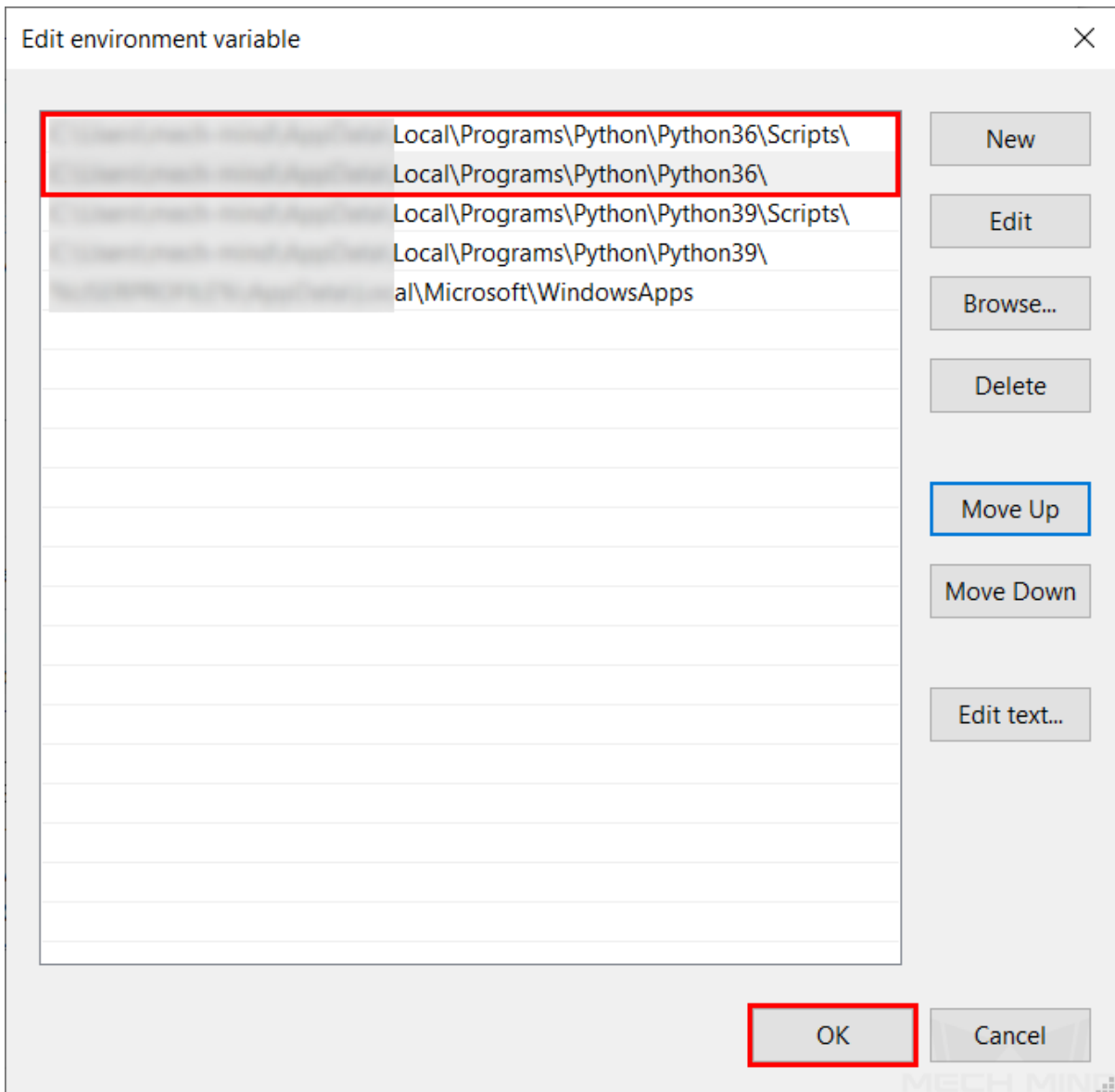
4. Click on **Path**, and then click on *Edit*.



5. Select the path that ends with `\Python36\`, and click on *Move Up* until this path is at the top of the list.



6. Similarly, move the path that ends with `\Python36\Scripts\` to the top of the list. Then, click *OK* to save the changes.



1. **Does it work to simulate changes in lighting conditions during data collection by manually adjusting the camera exposure or adding supplemental light?**

No. Simulated lighting conditions may not reflect the actual conditions accurately, and thus image data collected under such conditions cannot provide accurate object features to train the model. Therefore, if the lighting conditions on site change over the day, please collect image data respectively under different conditions.

2. **In the actual application, the camera is fixed, and the incoming objects' positions vary slightly. Does it work to simulate the position changes of the objects by moving the camera during data collection?**

No. The camera should be fixed in position before any data collection. Moving the camera during data collection will affect the extrinsic parameters of the camera and the training effect.

For the case in question, setting a larger ROI can capture the changes in object position.

3. **If the previously used camera has unsatisfactory imaging quality and is replaced by a new camera, is it necessary to add the images taken by the old camera to the dataset?**

No. After camera replacement, all data used for model training should come from the new camera. Please conduct data collection again using the new camera and use the data for training.

4. **Will changing the background affect model performance?**

Yes. Changing the background will lead to recognition errors, such as false recognition or failure to recognize a target object. Therefore, once the background is set in the early stage of data collection, it is best not to change the background afterward.

5. **Is it possible to use the image data collected with different camera models at different heights together to train one model?**

Yes, but please work on the ROI settings. Select different ROIs for images taken at different heights to reduce the differences among images.

6. **For highly reflective metal parts, what factors should be taken into consideration during data collection?**

Please avoid overexposure and underexposure. If overexposure in parts of the image is inevitable, make sure the contour of the object is clear.

7. **If the model performs poorly, how to identify the possible reasons?**

Factors to consider: quantity and quality of the training data, data diversity, on-site ROI parameters, and on-site lighting conditions.

- Quantity: whether the quantity of training data is enough to make the model achieve good performance.
- Quality: whether the data quality is up to standard, whether images are clear enough and are not over-/underexposed.
- Data diversity: whether the data cover all the situations that may occur on-site.
- ROI parameters: whether the ROI parameters for data collection are consistent with those for the actual application.
- Lighting conditions: Whether the lighting conditions during the actual application change, and whether the conditions are consistent with those during data collection.

8. How to improve unstable model performance due to complicated on-site lighting conditions, e.g., objects are covered by shadows?

Please add shading or supplemental light as needed.

9. Why does the inconsistency between the ROI settings of on-site data and training data affect the confidence of instance segmentation?

The inconsistency will result in objects being out of the optimal recognition range of the model, thus affecting the confidence. Therefore, please keep the ROI settings of the on-site data and training data consistent.

10. What scenarios is the Super Model for boxes suitable for?

It is suitable for palletizing/depalletizing boxes of single or multiple colors and surface patterns. However, please note that this Super Model is only applicable to boxes placed in horizontal layers and are not at an angle to the ground.

11. How to collect data for the Super Model for boxes?

Please test the Super Model first. If it cannot segment correctly sometimes, collect about 20 images of situations where the model does not perform well.

12. Does the image classification model work without a GPU?

No.

13. How to upgrade the earlier-version deep learning environment?

Please see *Install the Mech-Mind Software Environment* for instructions on upgrading the deep learning environment.

14. ROI position deviations may occur when opening old projects with the newer-version Mech-DLK.

The ROI will be corrected after clicking on *Validate*.