Mech-DLK

Mech-Mind

Apr 11, 2022

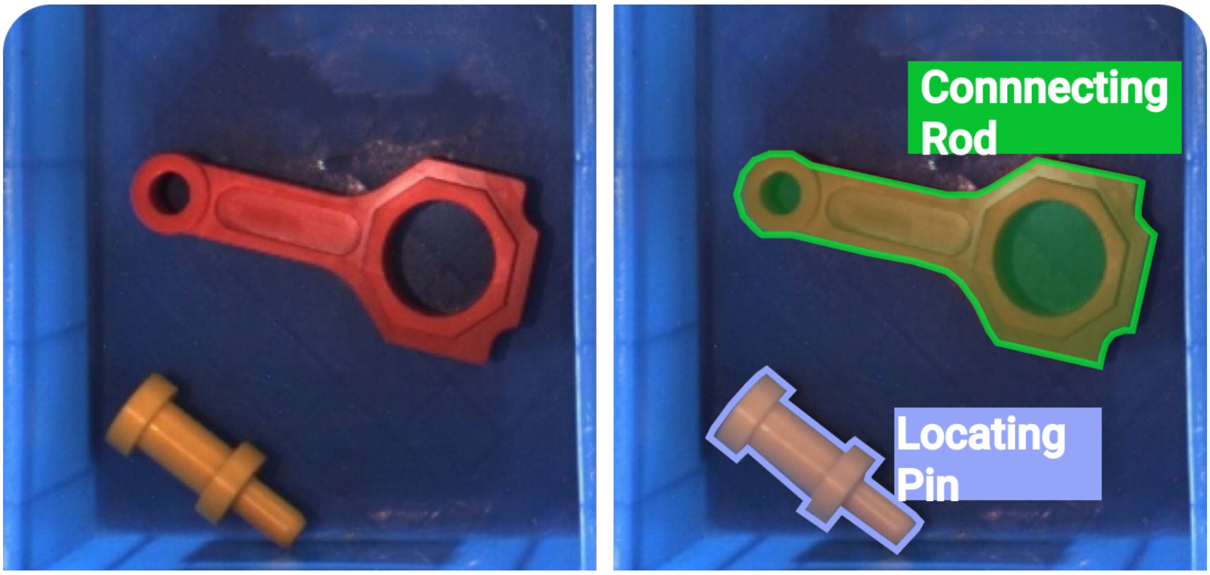
CONTENTS

Integrating deep learning algorithms into mature vision-based products, Mech-Mind Robotics has provided efficient and comprehensive solutions for clients in the automotive, 3C (computer, communication, consumer), manufacturing, and logistics industries all over the world, solving many real-world **intelligent recognition** problems including image classification, defect detection, feature recognition, etc.

Getting Started

Quick Guide to Deep Learning

Deep Learning Applications



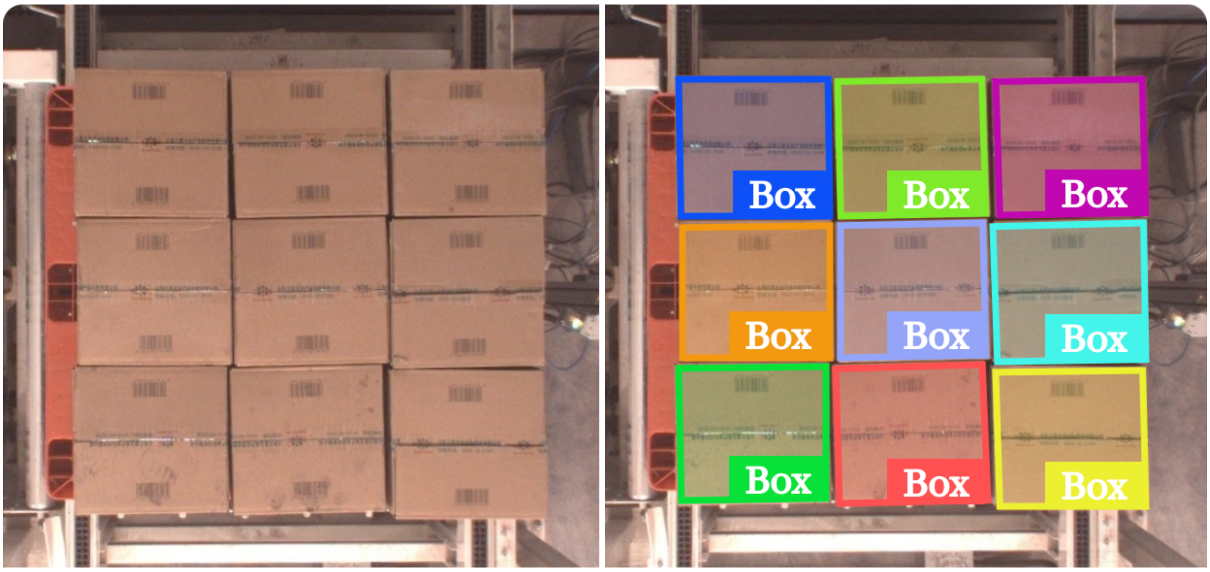
Instance Segmentation

Identify object contour, position, and class in images

Image Classification

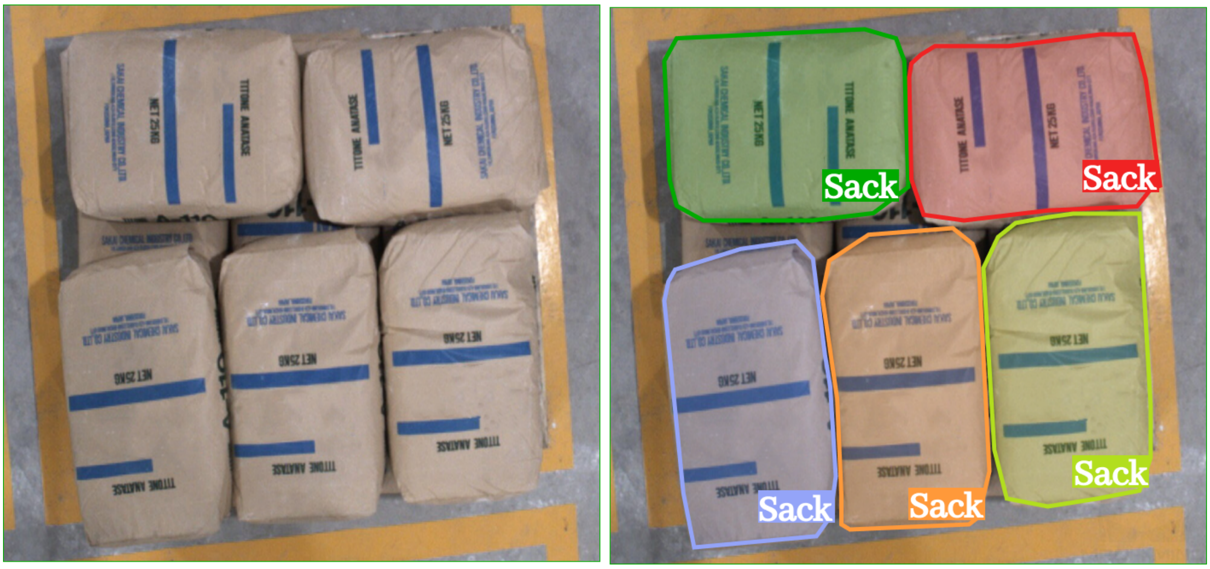
Assign a class to an image of a single object

Applying Deep Learning to Typical Projects



Box Palletizing/Depalletizing

Project workflow of box palletizing/depalletizing



Sack Palletizing/Depalletizing

Project workflow of sack palletizing/depalletizing

Mech-DLK User Guide & Deep Learning Environment Configuration

Mech-DLK Handbook

Deep learning training software for building deep learning models

Environment Configuration

Deep learning environment configuration and troubleshooting

FAQs

FAQ

FAQs about deep learning projects

CHAPTER

ONE

QUICK GUIDE TO DEEP LEARNING

1.1 Definitions

Deep learning means solving problems by learning the features of data.

Data collection and labeling: Take pictures of the objects that need to be recognized and then label the features that need to be recognized in the pictures.

Model training: Train a deep learning model on the object features from the images and labels and save the trained model as a file.

Application: Apply the model to recognition problems in actual project scenarios with input data similar to the training data.

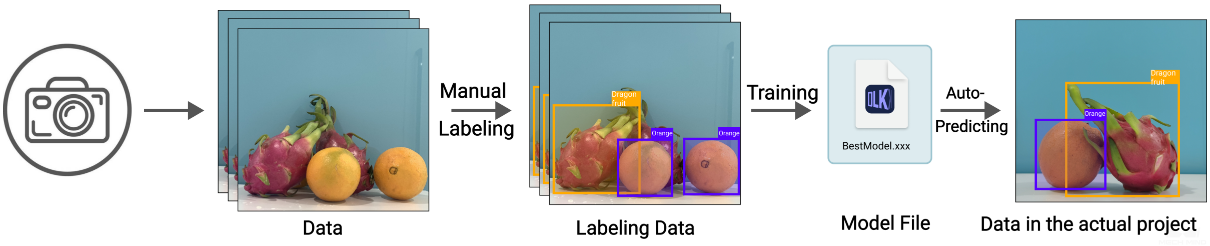


Figure 1. The process from data collection to application of a deep learning model

1.2 Usage

What problems can deep learning solve?

For different problems, the deep learning products developed by Mech-Mind Robotics provide different solutions:

Instance Segmentation	*Classification*
Typical Applications	
• Palletizing/depalletizing of cartons, sacks, turnover boxes • Machine tending • Order picking • Logistics parcel picking	• Classification, size/orientation recognition of workpieces for machine tending • Judging correctness of workpiece placing for assembly and picking • Distinguishing box/sack colors and types for palletizing/depalletizing
Functions	
• Locating • Classifying	• Classifying

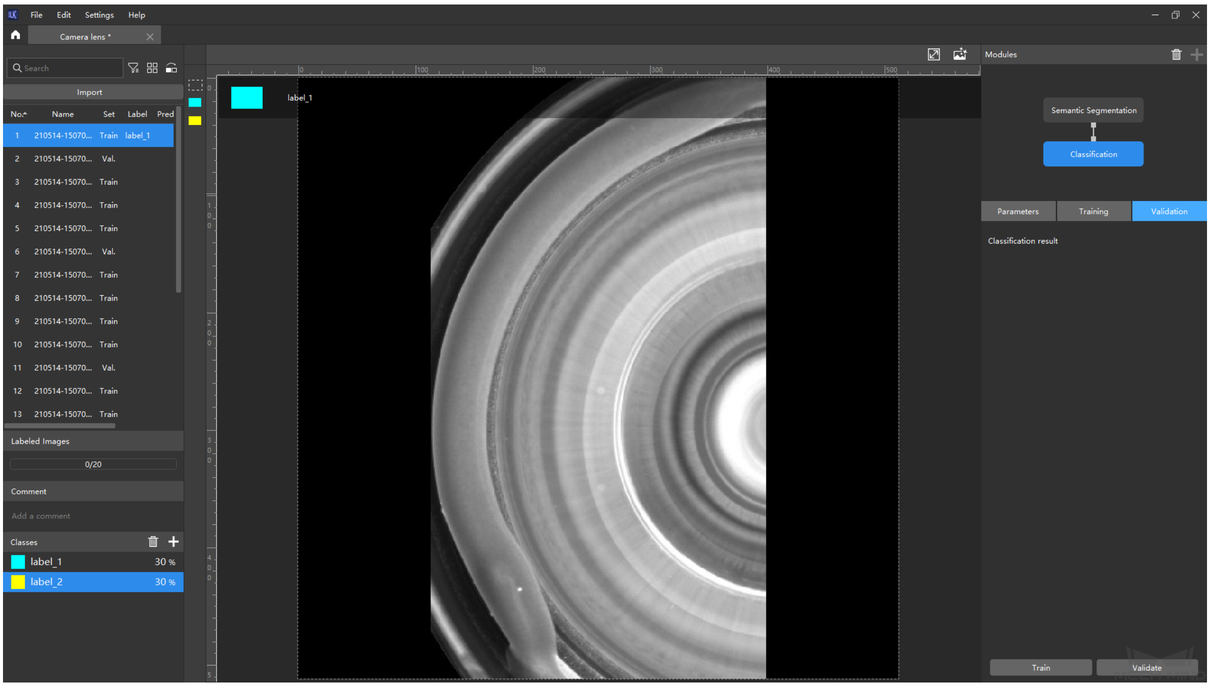
> **Attention:** Different deep learning algorithms have different specialties, which are not mutually exclusive. Whether to select one algorithm or a combination of algorithms depends on the actual requirements of a project.

1.3 Application Process

Step 1. Environment configuration: Please see *Environment Configuration* for details.

Step 2. Data preparation: Collect, label, and review the data.

1. Ensure that the image data required for deep learning is collected under conditions completely consistent with the actual application scenario.
2. Select the algorithm according to the project requirements and label the dataset by the rules.
3. Review whether there are incorrectly labeled data (Mech-DLK helps efficiently prepare the dataset required for training).

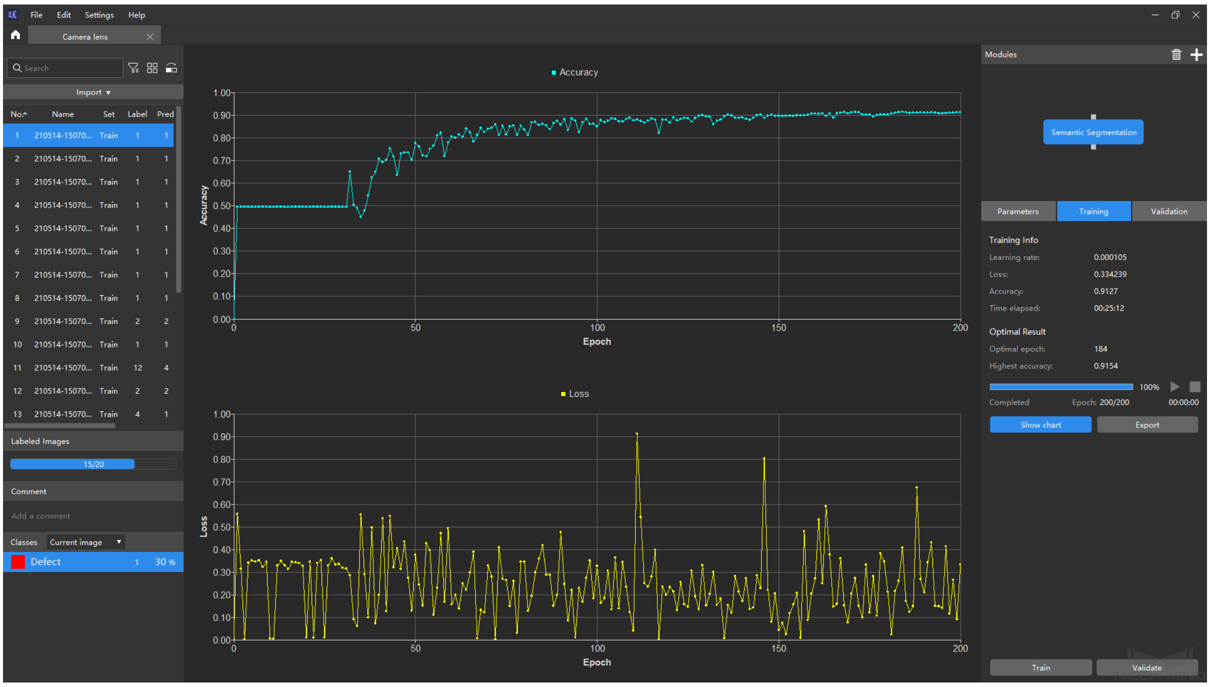


No.	Name	Set	Label	Pred
1	210514-15070...	Train	label_1	
2	210514-15070...	Val.		
3	210514-15070...	Train		
4	210514-15070...	Train		
5	210514-15070...	Train		
6	210514-15070...	Val.		
7	210514-15070...	Train		
8	210514-15070...	Train		
9	210514-15070...	Train		
10	210514-15070...	Train		
11	210514-15070...	Val.		
12	210514-15070...	Train		
13	210514-15070...	Train		

Classes	
label_1	30 %
label_2	30 %

Figure 2. Labeling data in Mech-DLK

Step 3. Training: Use the labeled data to train the required deep learning model in Mech-DLK. Please see *Mech-DLK Quick Start* for details.



No.	Name	Set	Label	Pred
1	210514-15070...	Train	1	1
2	210514-15070...	Train	1	1
3	210514-15070...	Train	1	1
4	210514-15070...	Train	1	1
5	210514-15070...	Train	1	1
6	210514-15070...	Train	1	1
7	210514-15070...	Train	1	1
8	210514-15070...	Train	1	1
9	210514-15070...	Train	2	2
10	210514-15070...	Train	1	1
11	210514-15070...	Train	12	4
12	210514-15070...	Train	2	2
13	210514-15070...	Train	4	1

Defect 1 30 %

Training Info

Learning rate: 0.000105

Loss: 0.334239

Accuracy: 0.9127

Time elapsed: 00:25:12

Optimal Result

Optimal epoch: 184

Highest accuracy: 0.9154

100%

Completed Epoch: 200/200 00:00:00

Figure 3. Training a deep learning model in Mech-DLK

Step 4. Evaluate the training effect: Use the reserved test set to verify whether the model’s performance meets the requirements.

Step 5. Application: The tested model that meets the project requirements can then be used in the actual project.

CHAPTER

TWO

DEEP LEARNING APPLICATIONS

2.1 Instance Segmentation

2.1.1 Introduction to Instance Segmentation

What Instance Segmentation Does

Instance segmentation solves the problems of **what is there, what it is, and where it is**, i.e., whether there is a target object in a picture, what kind of object it is, and where it is in the picture.

Examples:

- If the target objects are simply cartons, an instance segmentation model will judge whether there are cartons in an image (it will not recognize any other objects). If so, it will draw the contour of each carton and output the carton label indicating the object class (the label has been created during the data labeling process); otherwise, it will not output any results.

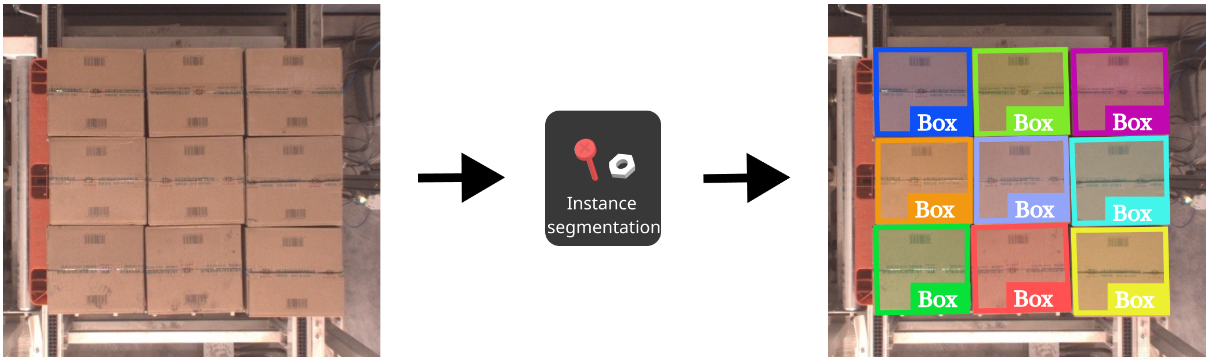


Figure 1. An instance segmentation model recognizing and labeling every carton

- If the target objects are of multiple classes, such as soap, toothbrush, shampoo, etc., an instance segmentation model will judge whether there are objects of these classes in an image. If so, it will draw the contour of each item and apply the corresponding label; otherwise, it will not output any results.

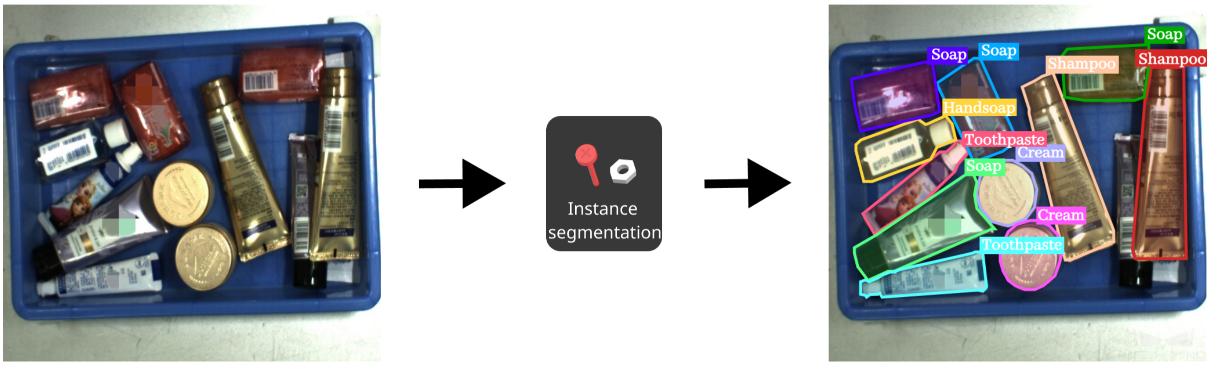


Figure 2. An instance segmentation model recognizing and labeling various objects

Typical Industrial Application Scenarios of Instance Segmentation

- **Palletizing/depalletizing**: Objects such as cartons, turnover boxes, sacks, etc. need to be removed from a pallet and placed on another pallet or equipment, such as bag break station, conveyor belt, etc.

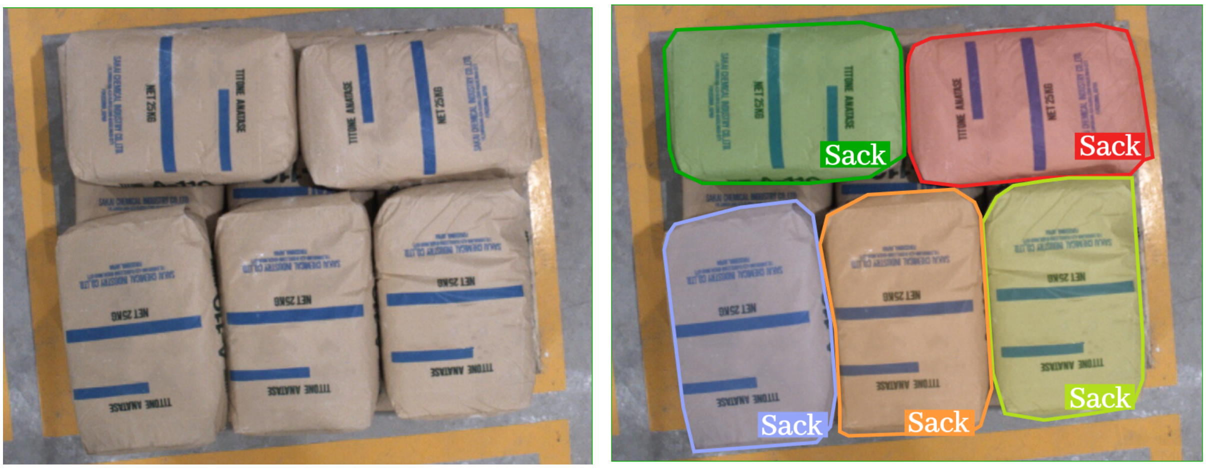


Figure 3. Segmenting sacks in an image in a palletizing/depalletizing project

- **Machine tending**: Handle and grasp complex workpieces, structural parts, irregular parts, etc., in the automotive, steel, machinery, and other industries.

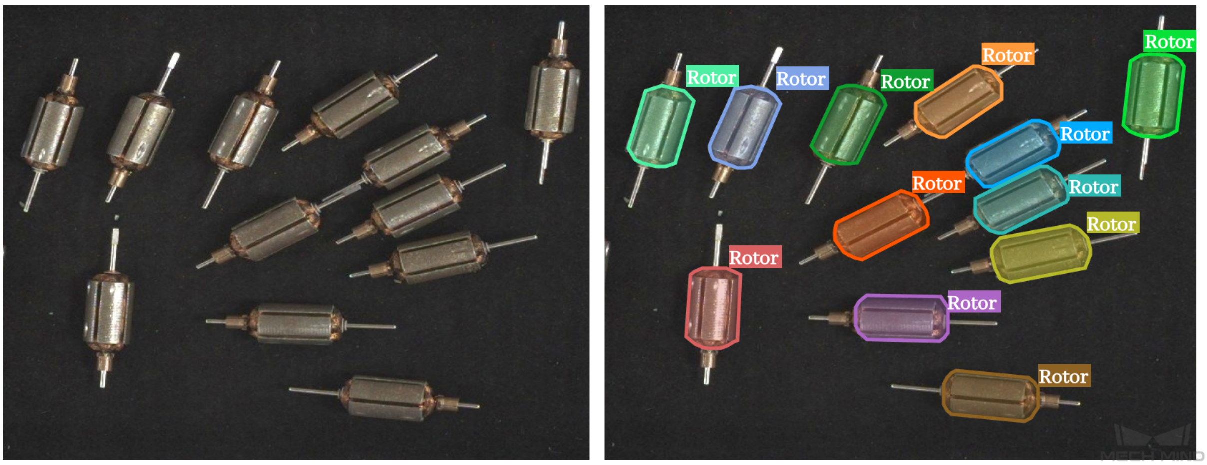


Figure 4. Segmenting workpieces in an image in a machine tending project

- **Order picking**: Frequently seen picking scenarios in various e-commerce warehouses include batch picking, discrete picking, sorting, etc. Supports various objects, including inflatable packaging, transparent packaging, bottles, aluminum cans, and irregularly-shaped goods like pots and pans.

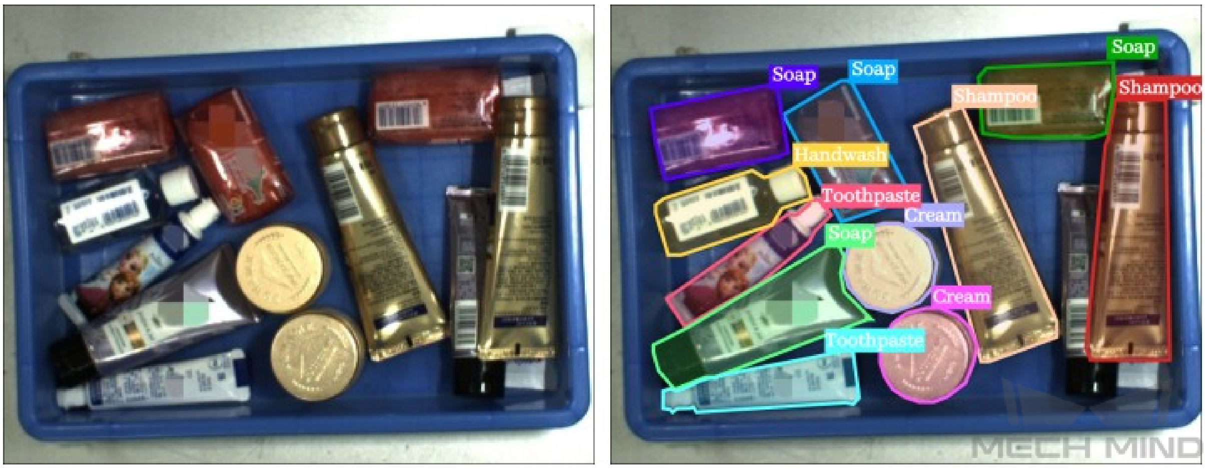


Figure 5. Segmenting goods in an image in an order picking project

- **Logistic parcel picking**: Supports picking various commonly seen packages such as shipping bags, postal envelopes, shipping boxes, padded envelopes, etc., and a variety of irregularly-shaped packages.

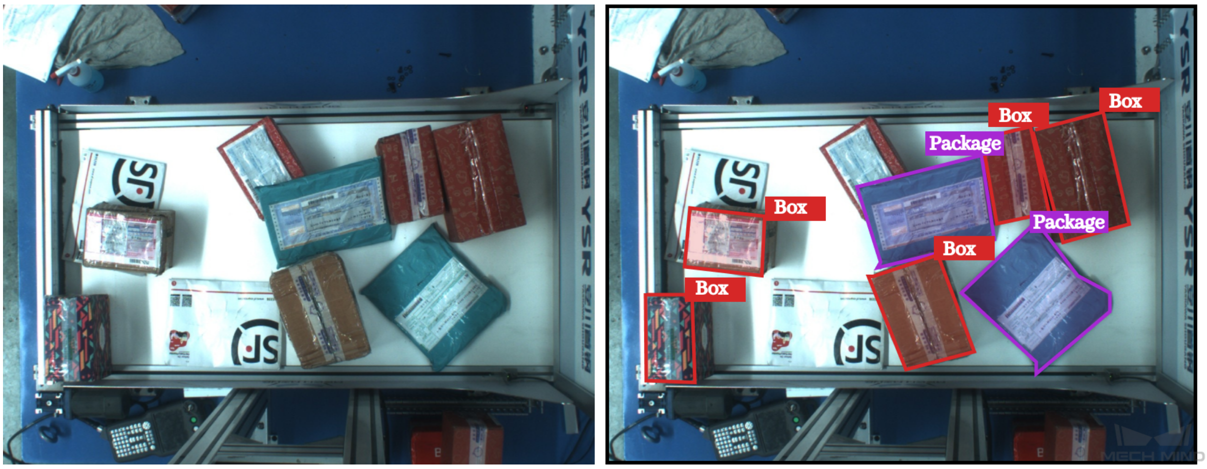


Figure 6. Segmenting shipping boxes and bags in an image in a logistic parcel picking project

Application Process of Instance Segmentation

Given enough image data from actual usage scenarios with object contours and classes correctly labeled, an instance segmentation model will learn how to segment the objects by itself. The application process of instance segmentation is as follows:

- *Collect the Training Data*: Take enough pictures of target objects with the camera.
- *Label the Training Data*: Label the contour and class of each object on each picture.
- *Train the Model*: Feed the labeled data to the instance segmentation model for training.
- *Use the Model*: Apply the trained model in an actual project.

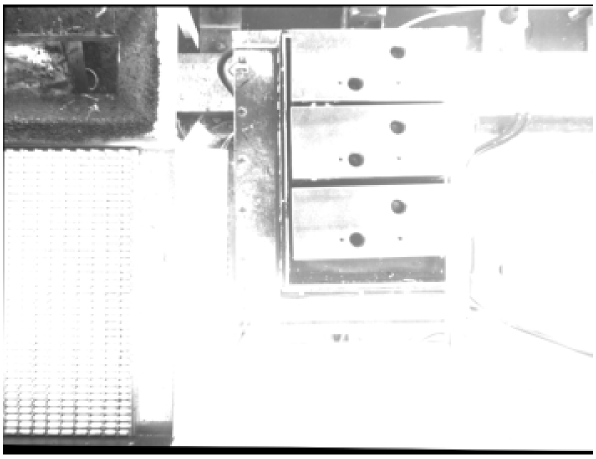
2.1.2 Collect the Training Data

> **Attention:** Collecting data is one of the most critical parts of a deep learning project. The final effect of the model largely depends on the quality of the training data. A high-quality dataset is a prerequisite for effective model training and accurate prediction.

Check the Data Collection Environment

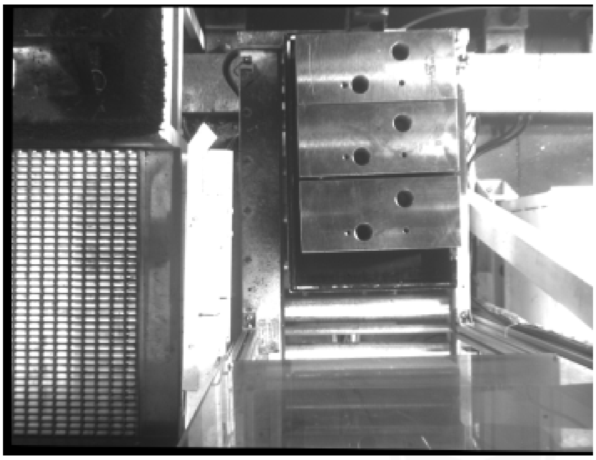
1. Please avoid conditions including **overexposure, underexposure, color distortion, blurriness, blockage**, etc., that will result in the loss of the features on which the deep learning model relies and thereby affect the model’s performance.

Bad example: Overexposure.

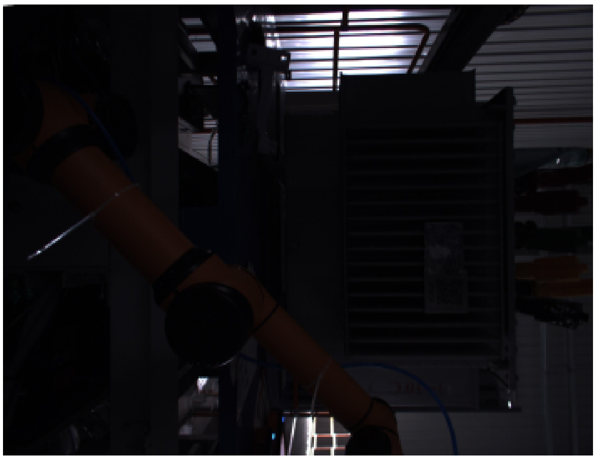


Optimization suggestion: Add shading.

Good example: Normal exposure.

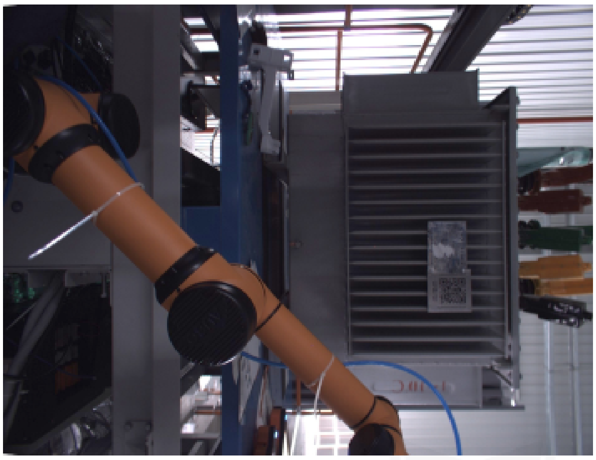


Bad example: Underexposure.

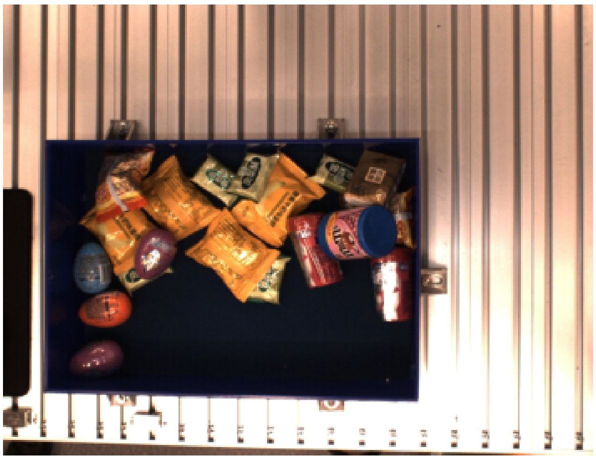


Optimization suggestion: Add supplementing light.

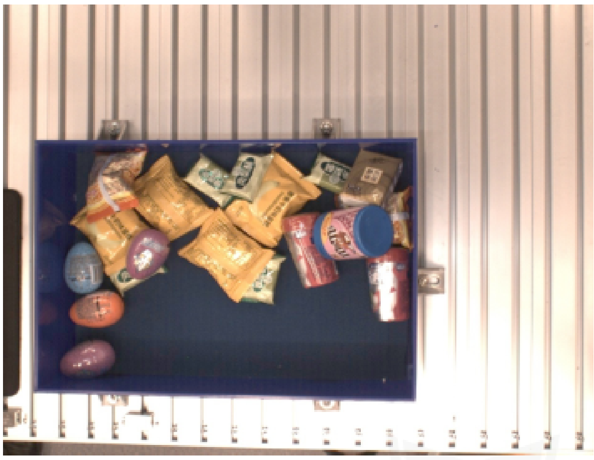
Good example: Normal exposure.



Bad example: Color distortion.

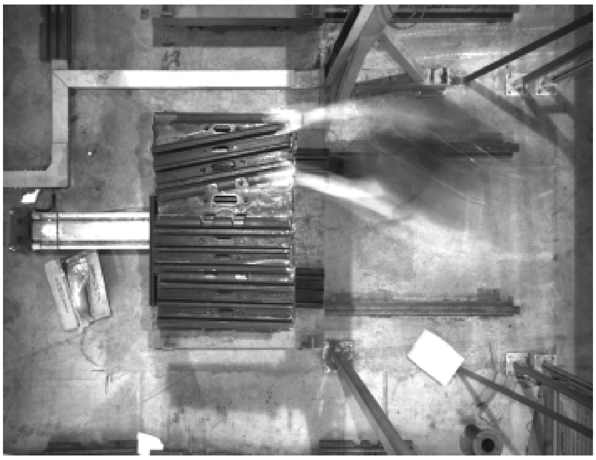


Good example: Normal color.

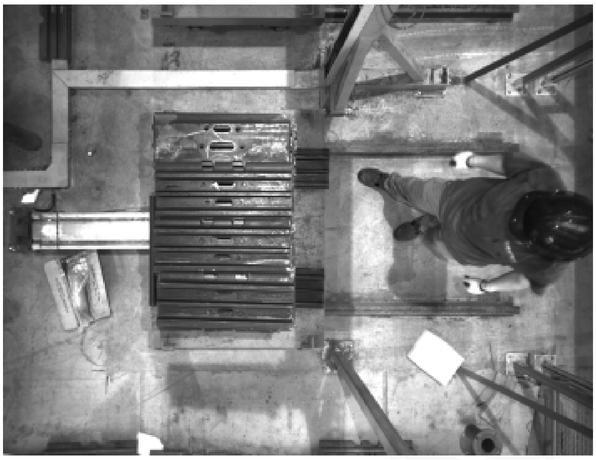


Optimization suggestion: Please adjust the camera's white balance to avoid color distortion.

Bad example: Blurry imaging.



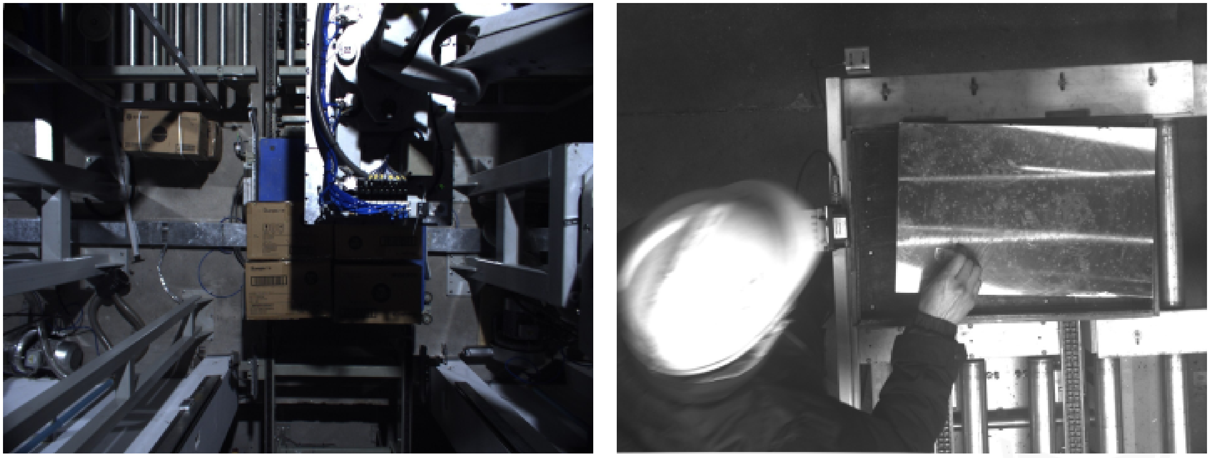
Good example: Clear imaging.



Optimization suggestion: Only take pictures when the camera and the objects are still.

Bad example: Blocked by the robot arm.

Bad example: Blocked by human staff.

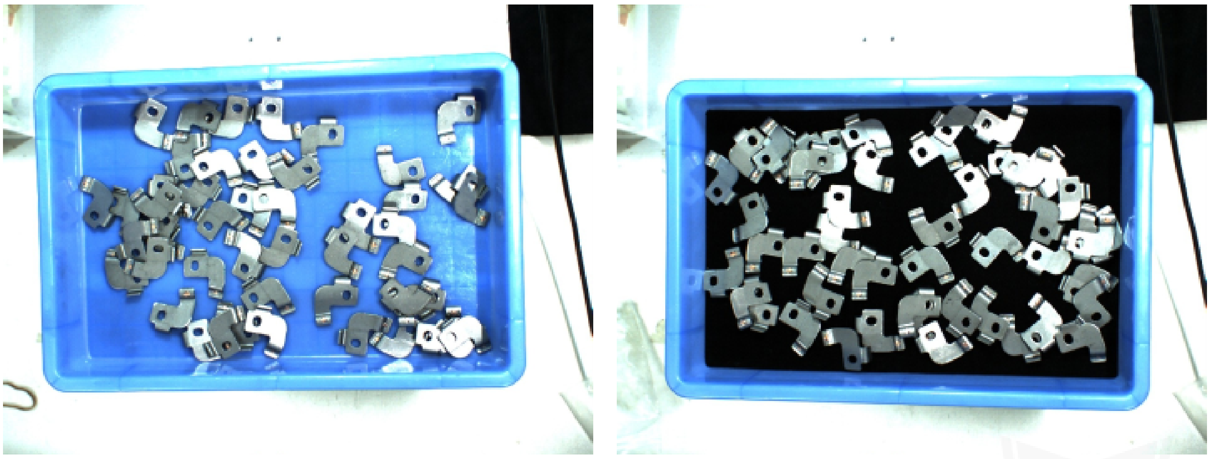


Optimization suggestion: Ensure there is no robotic arm or human staff in the camera's field of view.

Figure 1. Examples of data collection environment conditions

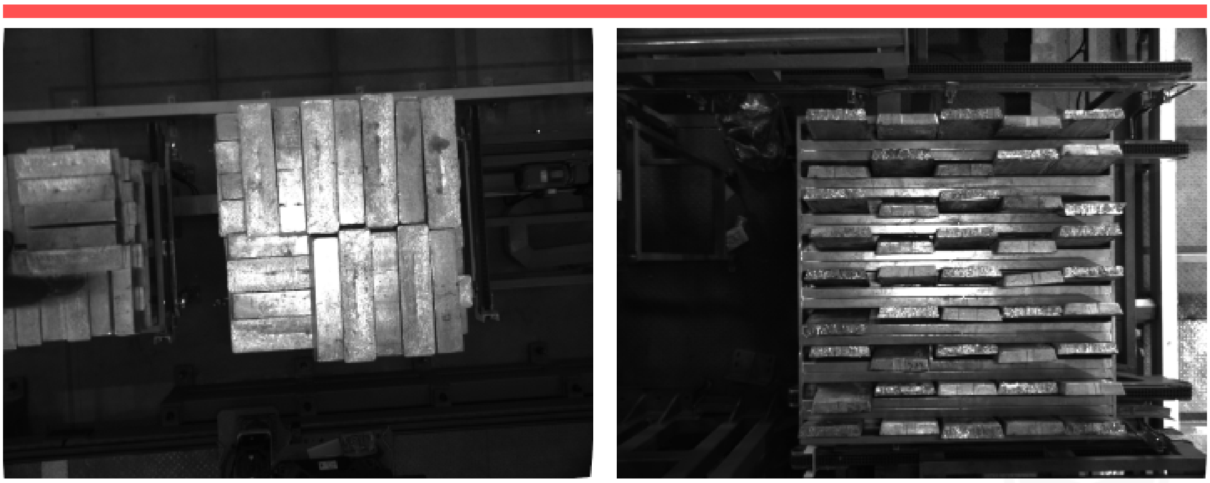
2. Please ensure that the **backgrounds, perspectives, and camera distances from the objects** for data collection are consistent with those of the actual application scenarios. Any inconsistencies will reduce the performance of the model in the actual application. In severe cases, data need to be recollected and the model needs to be re-trained. Therefore, please confirm the detailed conditions of the actual application scenario before data collection.

Bad example: The training data's background (left) is inconsistent with that of the scenario (right).



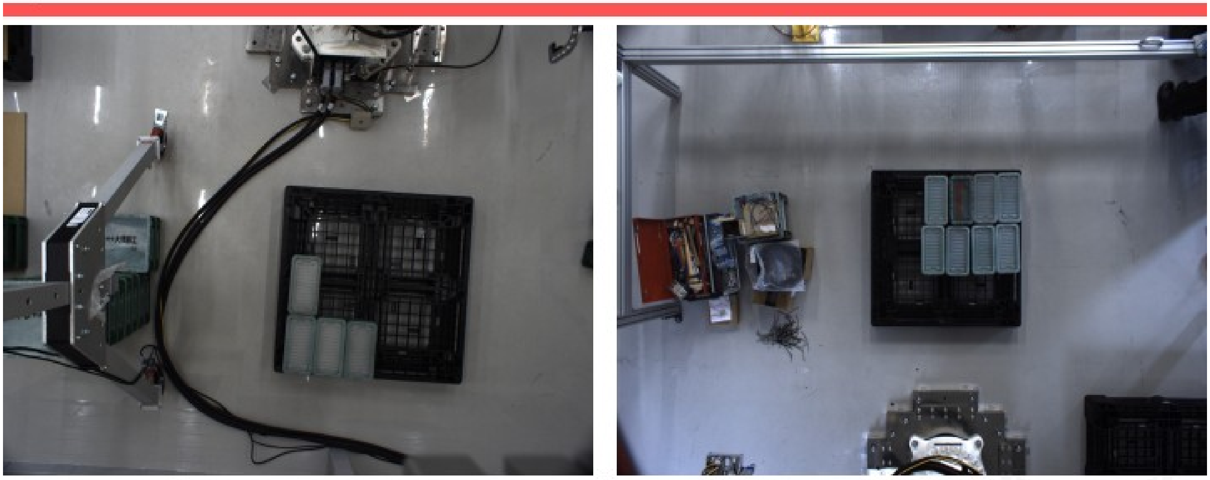
Optimization suggestion: Ensure that the background of the training data is consistent with that of the scenario.

Bad example: The training data's field of view (left) is inconsistent with that of the scenario (right).



Optimization suggestion: Ensure that the training data's field of view is consistent with that of the scenario.

Bad example: The training data's camera height (left) is inconsistent with that of the scenario (right).



Optimization suggestion: Ensure that the training data's camera height is consistent with that of the scenario (right).

Figure 2. Inconsistencies between data collection environment and application scenario

Quantity of Data to Collect

- If there is only one object class, please collect around 50 images.
- If there are multiple object classes, please collect around 30 images for each class. **Total number of images to collect = 30 * number of classes**.
- The above is a general guideline for the quantity of data to collect, and typical industrial applications have more specific requirements. Please see *Data Collection Examples from Past Projects* for an example.

Attention: If the training dataset is too small, the model will not have enough samples and can not be trained effectively; the test error rate will also be high. If the training dataset is too large, the training time will be significantly increased. Please make sure the size of the dataset is appropriate for actual needs.

Object Placing for Data Collection

All different placing conditions should be included in the dataset, and the number of images for each placing condition should be reasonably allocated based on the actual project conditions.

For example, if the objects come in horizontal and vertical poses in the actual application, but only images of horizontal incoming objects are collected and used for training, then the resulting model' s performance on the vertical objects cannot be guaranteed.

Another example is that, if the objects come overlapping in the actual application, but only images of separately placed objects are collected and used for training, then the resulting model' s performance on the overlapping objects cannot be guaranteed.

Therefore, when collecting data, please **take all circumstances in the actual application into consideration** as much as possible. Factors include the following:

- All **object orientations** that may appear in the actual application;
- All **object positions** that may appear in the actual application;
- All **spatial relationships** between objects that may appear in the actual application.

Attention: If some circumstances are omitted from data collection, the deep learning model will not be trained properly for and will fail to output satisfactory results in such circumstances. In this case, please include data on omitted circumstances to avoid errors.

1. **Object orientation**

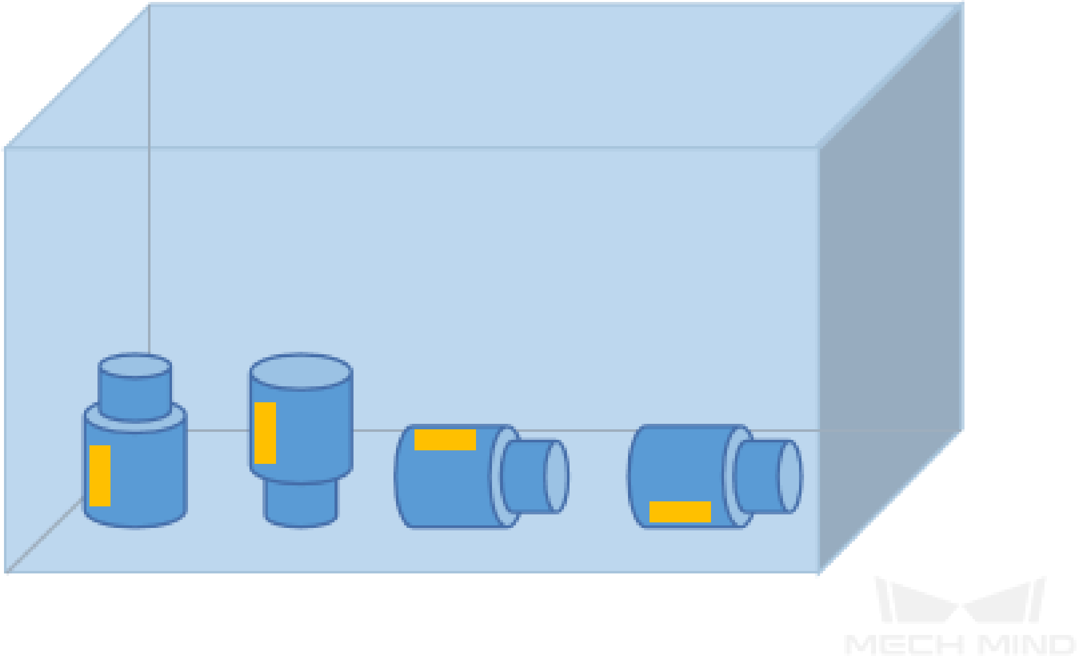


Figure 3. Objects’ different sides face up

2. **Object position**

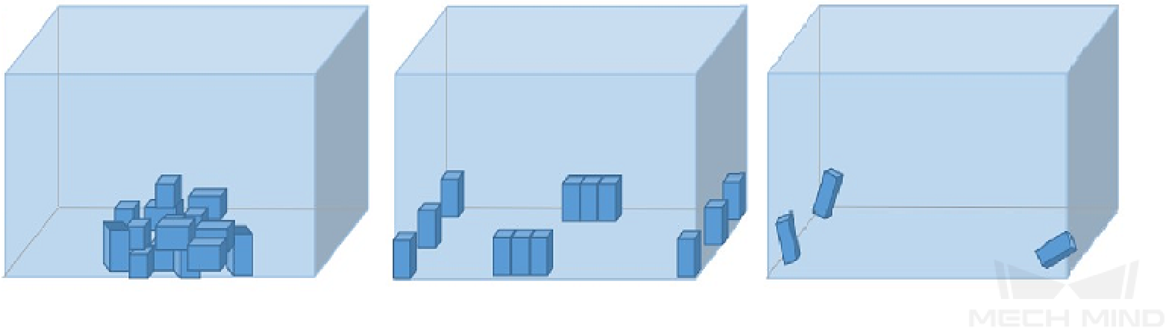


Figure 4. Objects are in the center, along the edges, or in the corners of the bin

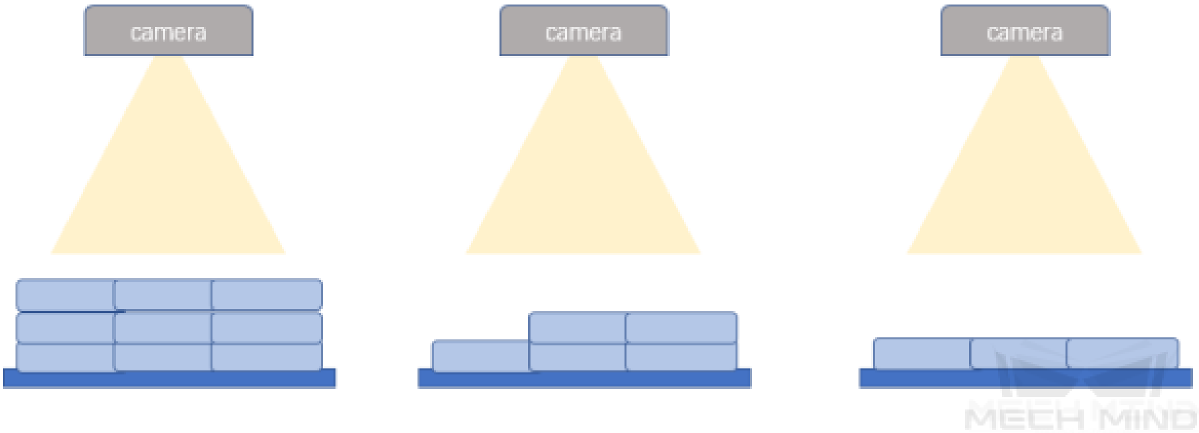


Figure 5. Objects are on different layers

3. **Spatial relationship between objects**

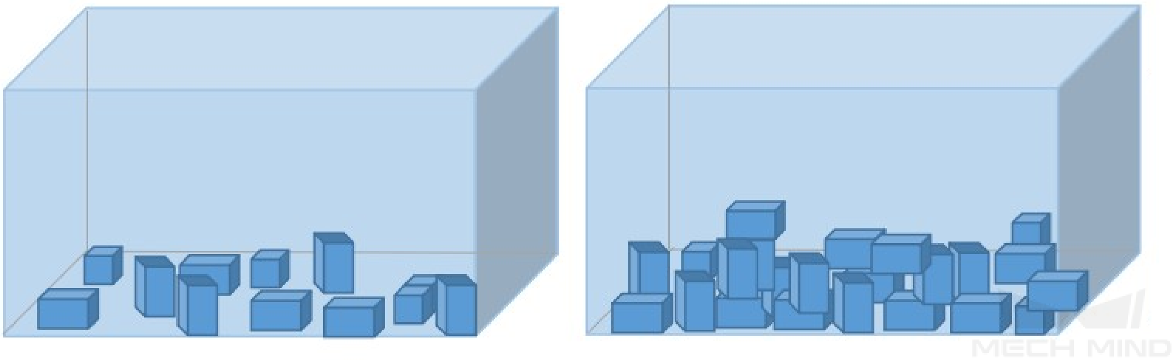


Figure 6. Objects are separately placed or overlapping

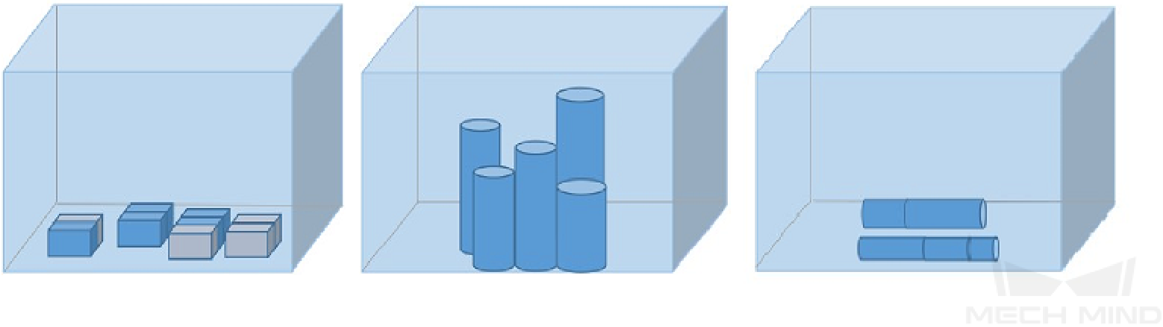


Figure 7. Objects are closely fitted

Use Mech-Vision to Collect Data

After checking the data collection environment, determining the data quantity to collect, and listing all the possible ways of object placing, please use the following Steps in Mech-Vision to collect the image data. See Capture Images From Camera for detailed instructions.

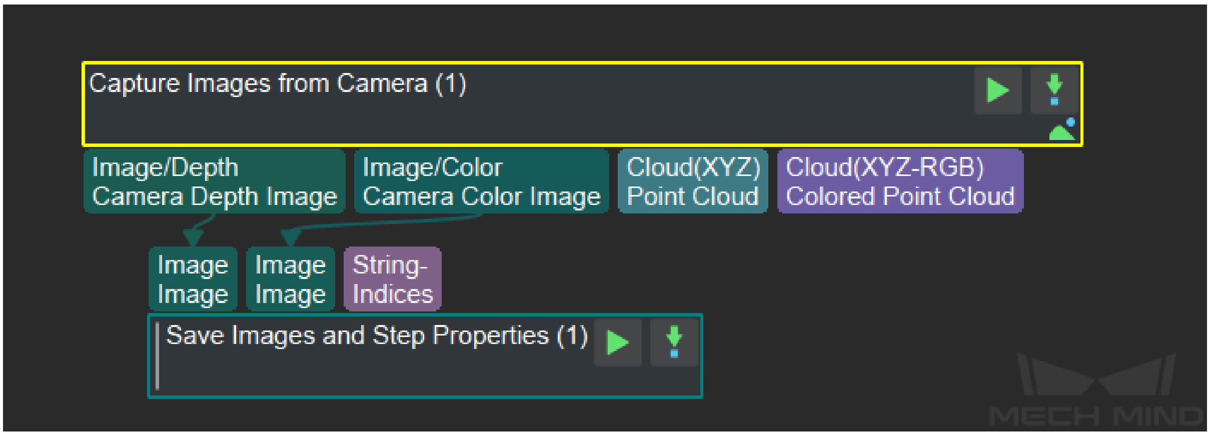


Figure 8. Data collection Steps in Mech-Vision

Data Collection Examples from Past Projects

Metal workpiece, single class

- Data quantity: 50 pictures for single-class objects.
- Orientation: The objects may lie flat or stand on sides, both of which need to be considered.
- Position: The objects may be in the center, along the edges, or in the corners of the bin, or placed on different layers.
- Spatial relationship: The objects may be overlapping, and occasionally parallelly placed.

The following are some examples of the images collected:

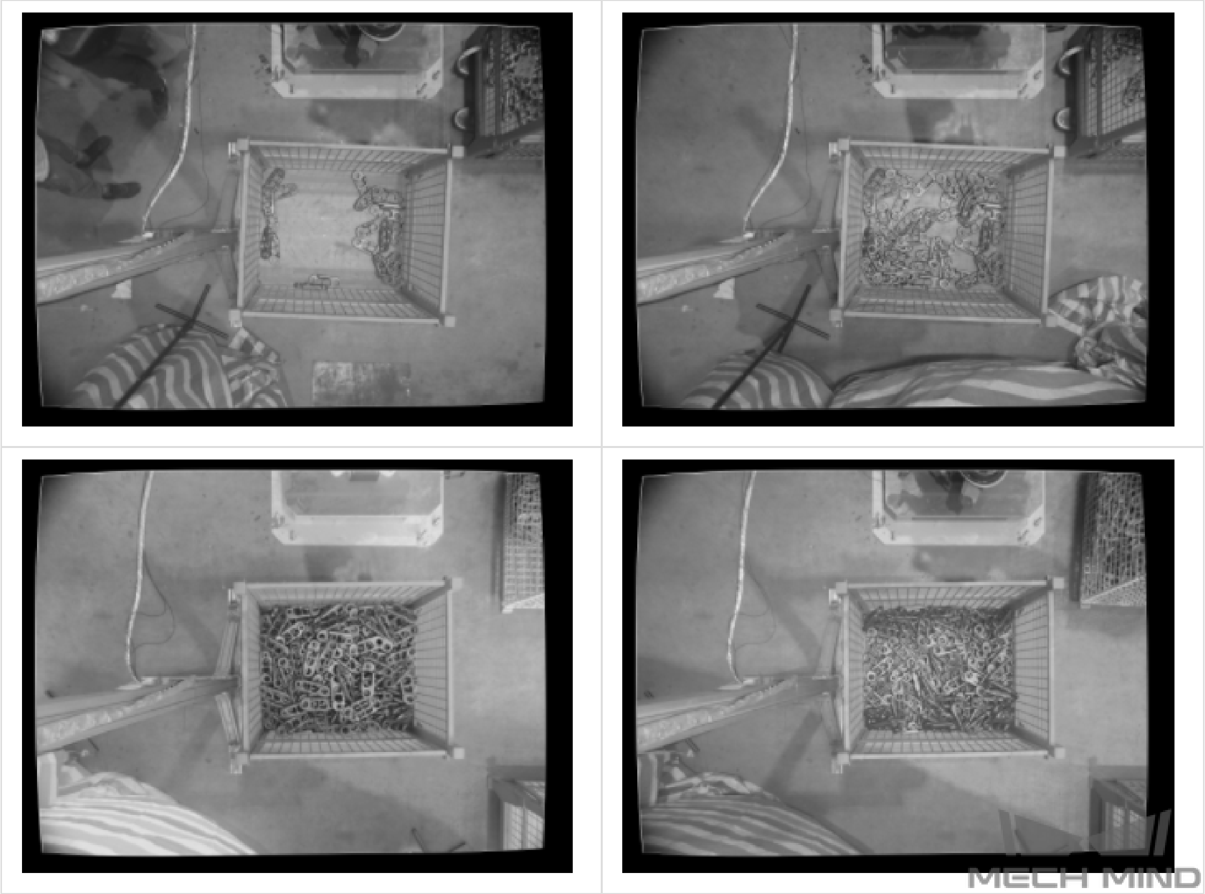


Figure 9. Sparsely scattered (top left), densely scattered (top right), overlapping (bottom left), and very densely scattered (bottom right)

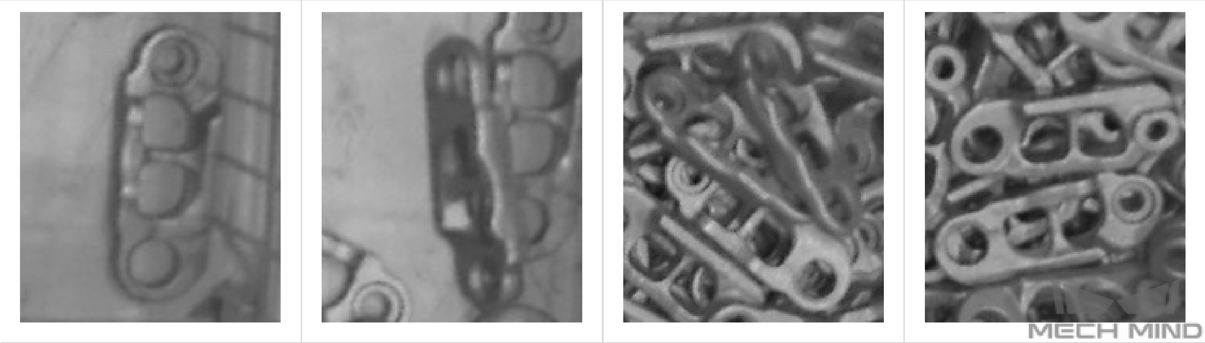


Figure 10. Lying flat, standing on sides, overlapping, and parallelly placed

Beauty and personal care products, seven classes

- Classification is required as there is more than one class of objects.
- Cases, where products of the same class are placed in many orientations and products of multiple classes are placed together, need to be considered to fully capture the object features.
- For cases where only a single class of objects are placed in the bin, five images for each class should be collected. For cases where objects of multiple classes are mixed in the bin, the total number of images to collect should be (20 * number of classes).
- Orientation: The objects may lie flat, stand on sides, or lean at an angle. All sides of the objects need to be captured.
- Position: The objects may be in the center, along the edges, or in the corners of the bin.
- Spatial relationship: The objects may be overlapping, occasionally parallelly placed, and tightly fitted.

The following are some examples of the images collected:

One class

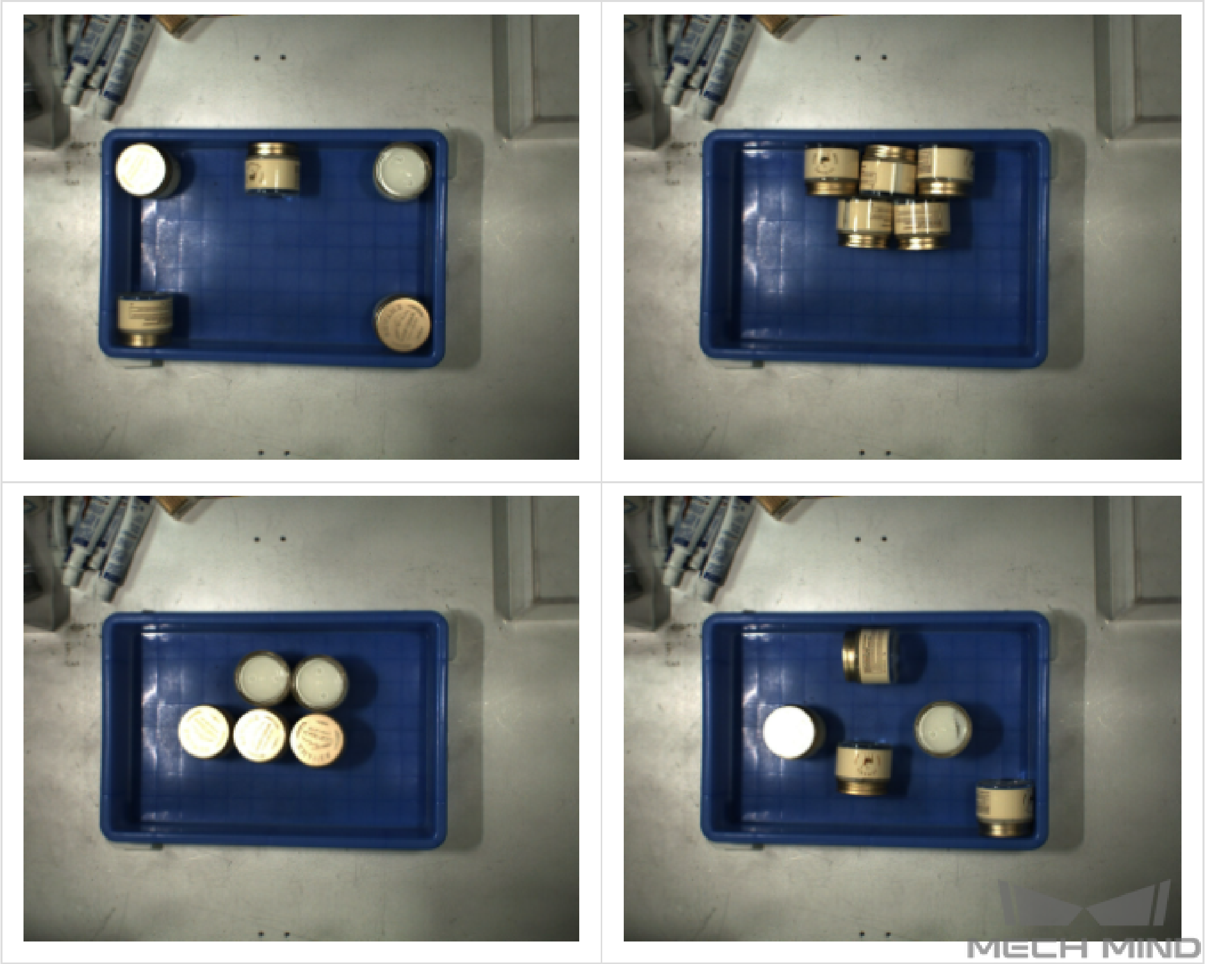


Figure 11. In the corners (top left), tightly fitted (top right), closely placed (bottom left), and sparsely scattered (bottom right)

Multiple classes

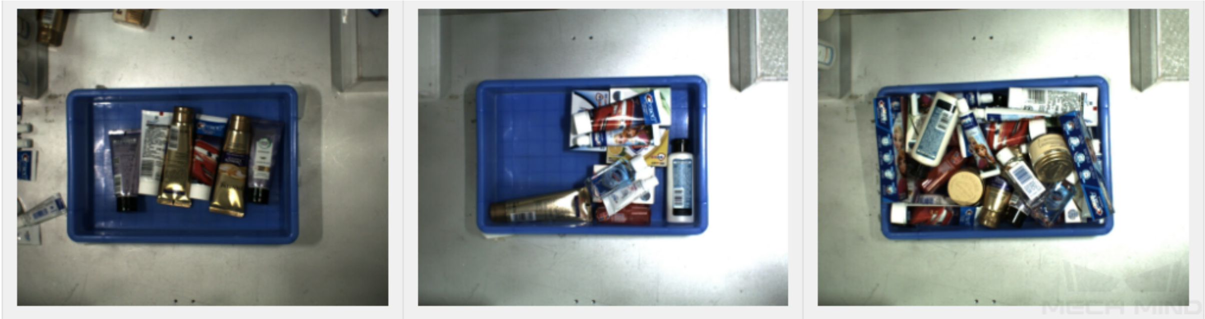


Figure 12. Closely placed, jammed in bin corners, and scattered and overlapping

Track shoe unit, multiple classes (models)

- Number of images to collect: 30 * number of models.
- Orientation: Only the case where the front face is up needs to be considered.
- Position: Relatively simple. Only objects in the top, middle, and bottom layers need to be considered.
- Spatial relationship: The objects are orderly placed and tightly fitted.

The following are some examples of the images collected:

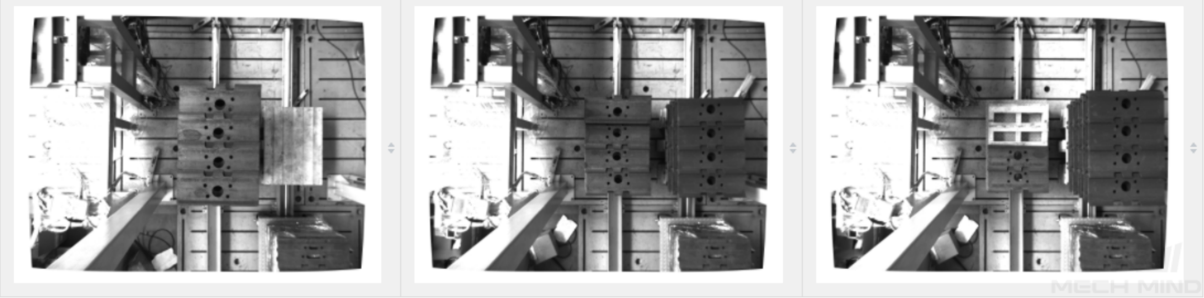


Figure 13. Objects in the top, middle, and bottom layers

Metal workpiece, single class

- Data quantity: The objects are of a single class and are placed in a single layer, so 50 images need to be collected.
- Orientation: Only the case where the front face is up needs to be considered.
- Position: The objects may be in the center, along the edges, or in the corners of the bin.
- Spatial relationship: The case where the objects are tightly fitted needs to be considered.

The following are some examples of the images collected:

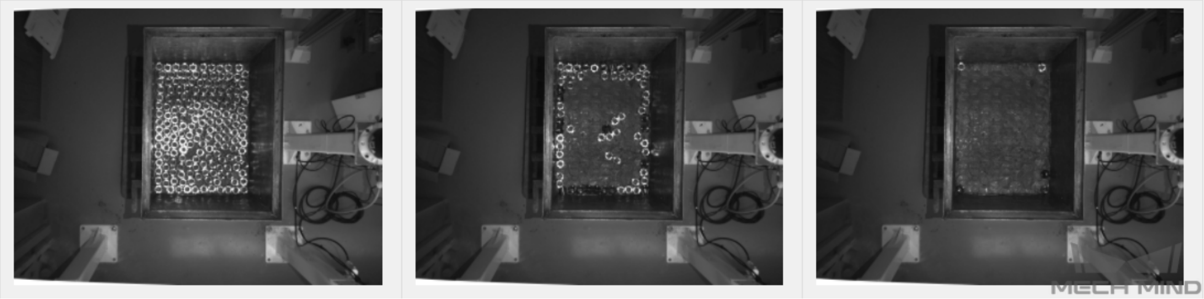


Figure 14. One full layer, along the edges, and in the corners of the bin

Metal workpiece, single class

- The objects are stacked in multiple layers, and 30 images need to be collected.
- Orientation: Only the case where the front face is up needs to be considered.
- Position: The objects may be in the center, along the edges, or in the corners of the bin, as well as in the top, middle, and bottom layers.
- Spatial relationship: The case where the objects are tightly fitted needs to be considered.

The following are some examples of the images collected:

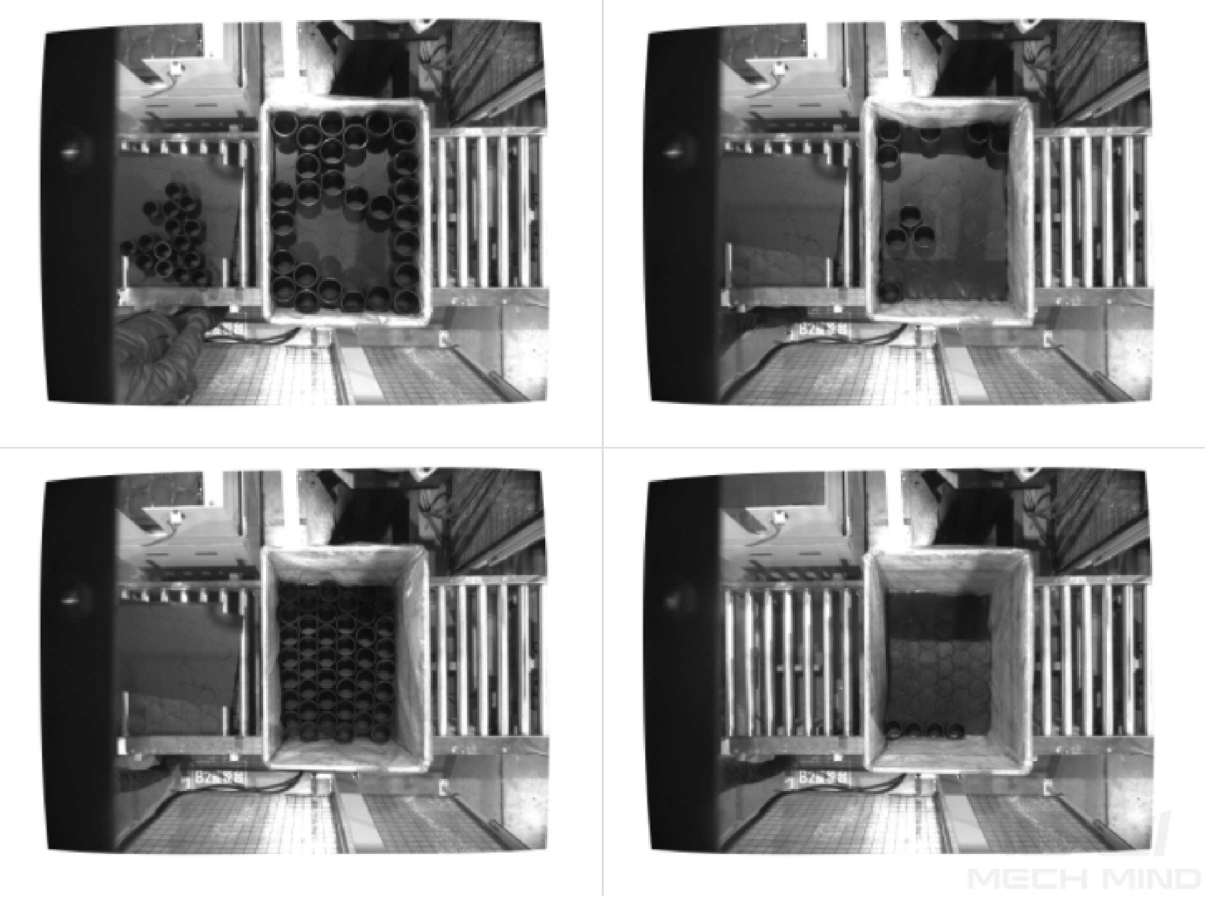


Figure 15. In the top layer (top left), a small amount in the top layer (top right), a full bottom layer (bottom left), and in the bottom layer along the bin edges (bottom right)

2.1.3 Label the Training Data

Create Label(s)

Please confirm whether the project requires the classification of objects. If so, please create multiple labels, each corresponding to an object class; otherwise, please create one label.

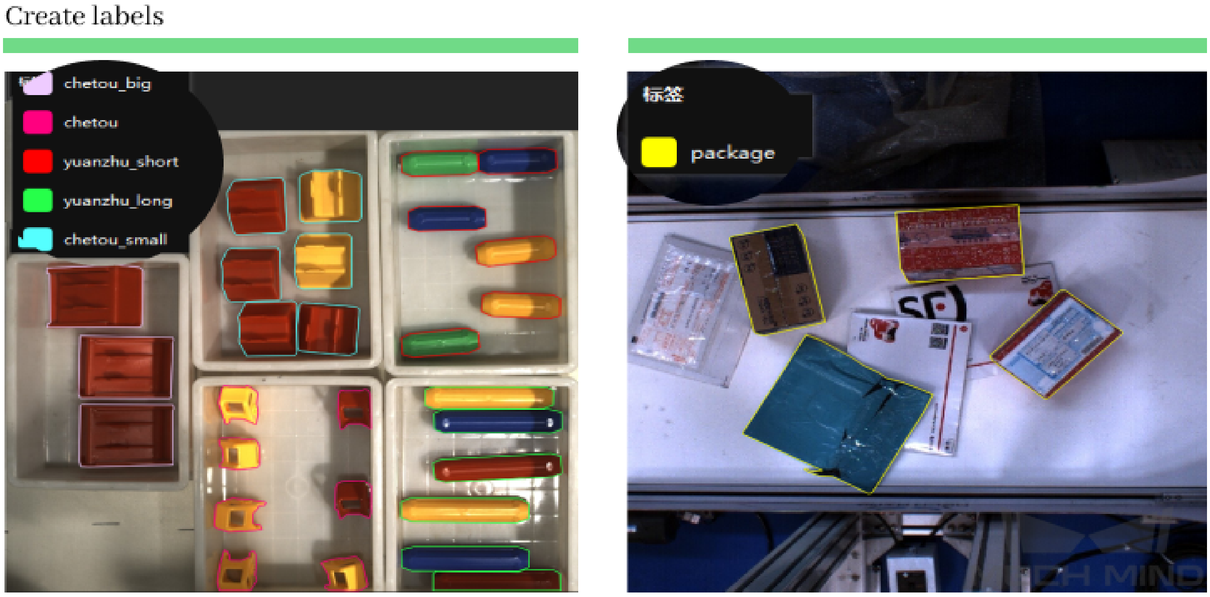


Figure 1. Cases of classification with multiple labels and no classification with a single label

> **Attention:** Label names should be relevant to the objects and easily recognizable. Please do not use meaningless names like a, b, tmp, etc. Label names should only include letters or numbers.

Determine Method of Labeling

1. **Label the contour of the upper surface**: Suitable for orderly-placed and regularly-shaped objects, such as cartons, medicine boxes, rectangular workpieces, etc. Pick points are calculated based on the contour of the upper surface.

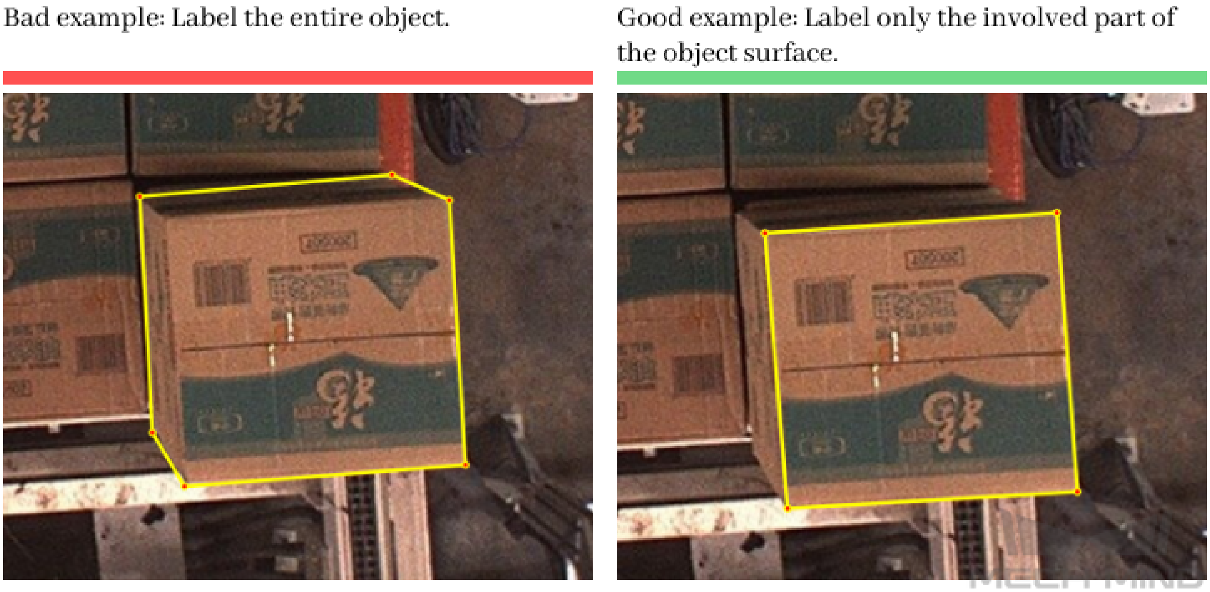

Bad example: Label the entire object.
Good example: Label only the involved part of the object surface.

Figure 2. Label the contour of the upper surface

2. **Label the contour of the entire object**: Suitable for sacks, various workpieces, etc. Labeling the contour of the entire object is a general labeling method.

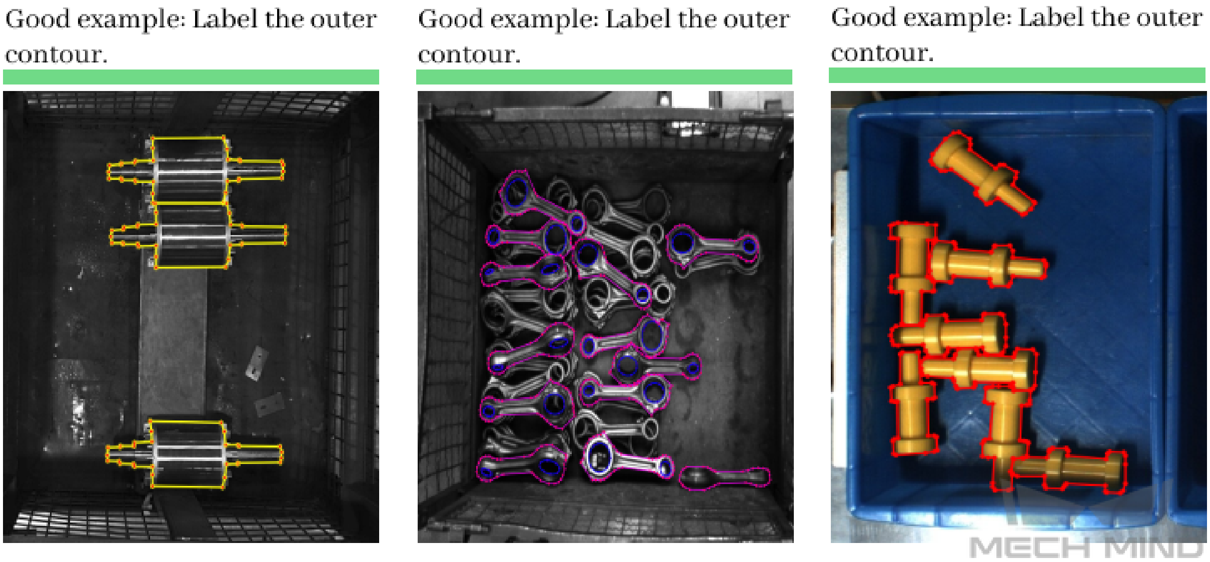

Good example: Label the outer contour.
Good example: Label the outer contour.
Good example: Label the outer contour.
MECH MIND

Figure 3. Label the contour of the entire object

3. **Special cases**: Some picking tasks involve special end effectors and/or picking methods.

   Case #1: The suction cup needs to fit perfectly with the bottle caps (requires high accuracy), thus only the contour of the bottle caps needs labeling.

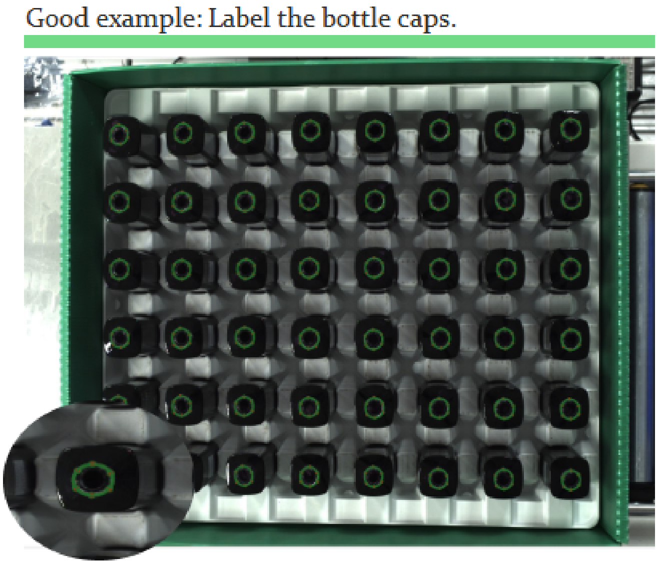


Figure 4. Only label the bottle caps

Case #2: The rotor picking task involves distinguishing the orientation of rotors. Only the middle parts where the orientation is easily recognizable need labeling and the thin rods at both ends should be excluded.

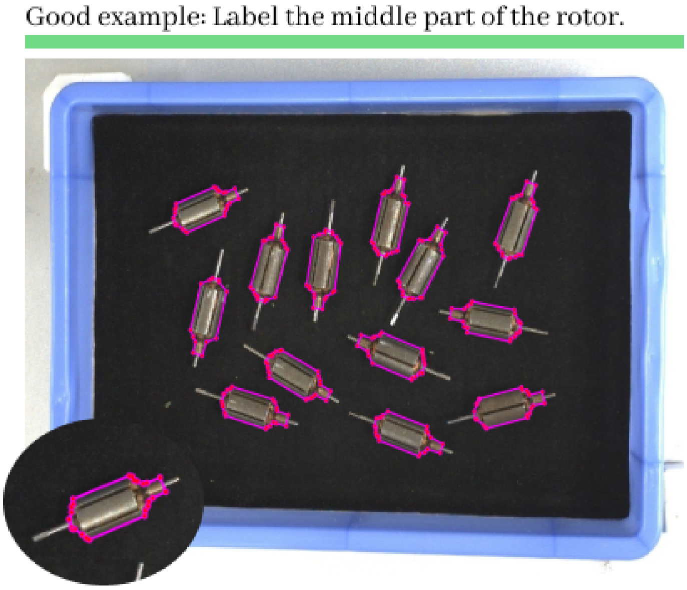
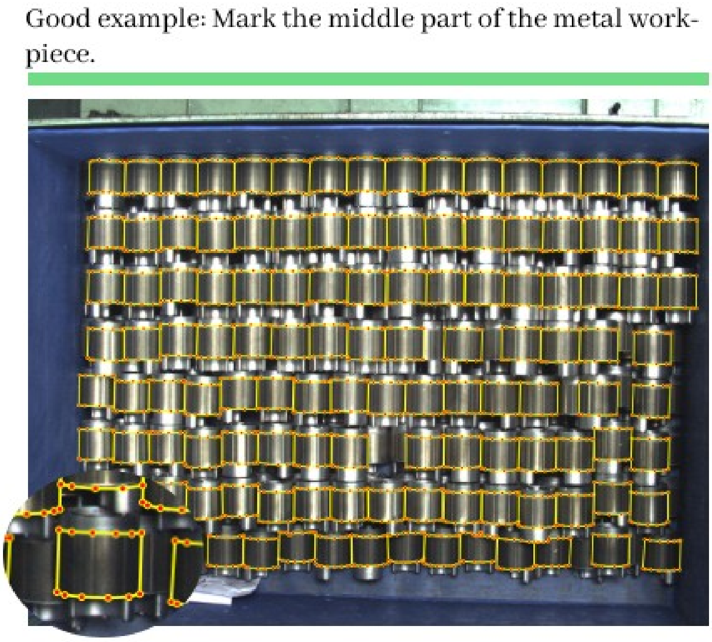


Figure 5. Only label the middle parts of rotors

Case #3: The pick point is required to be in the middle part of the metal workpiece, so only the middle parts need labeling, and the two ends should be excluded.



Good example: Mark the middle part of the metal workpiece.

Figure 6. Only label the middle parts of metal workpieces

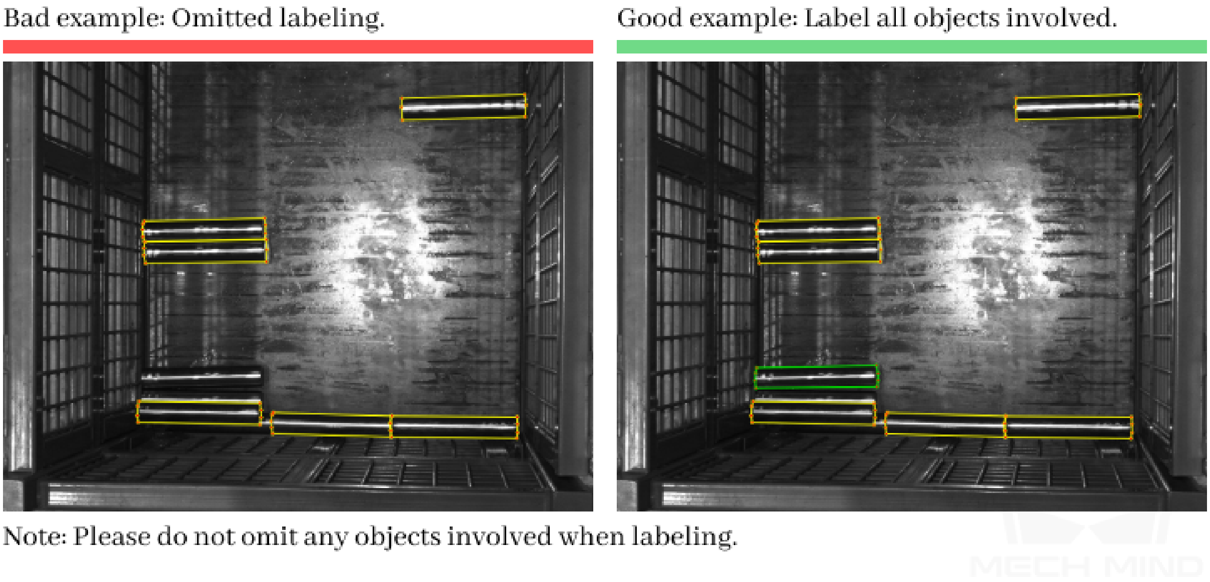
Use Mech-DLK to Label Data

Different algorithms require different methods of data labeling. Please see *Data Labeling in Mech-DLK* for detailed instructions.

> **Attention:** Please make sure to check the labeling quality: After labeling, be sure to check whether all images have been labeled and verify whether every label corresponds with the labeled object. Labeling errors act as counterexamples in the dataset and will adversely affect the training process and the model performance.

The quality of labeling should be evaluated in terms of completeness, correctness, consistency, and accuracy:

1. **Completeness:** Label all objects that meet the labeling rules without omission.



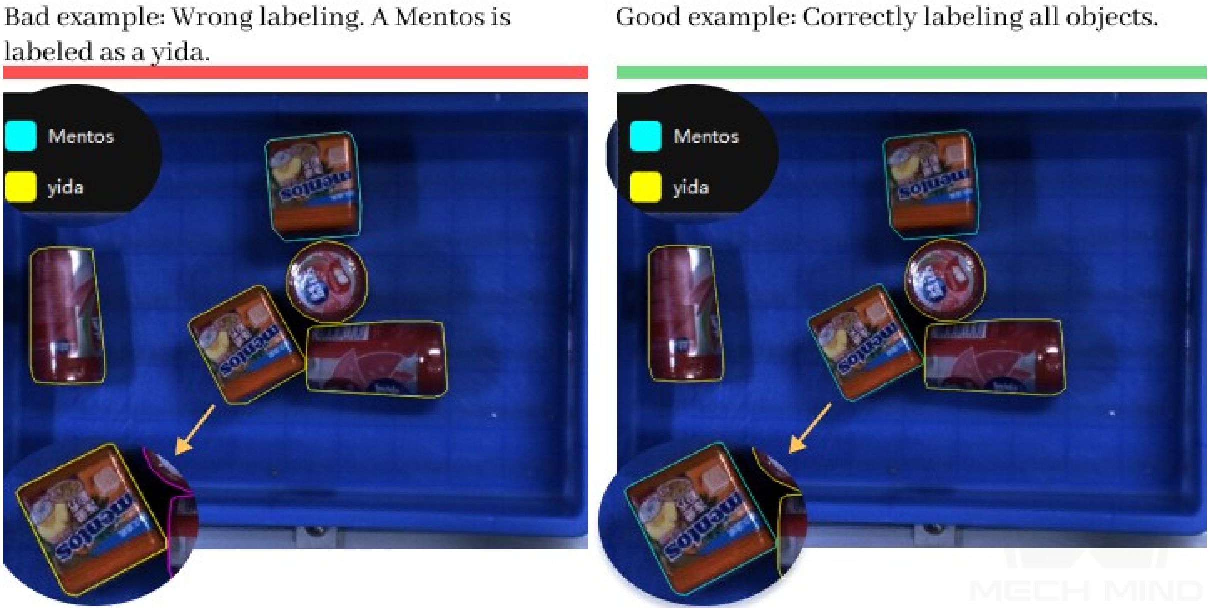
Bad example: Omitted labeling.

Good example: Label all objects involved.

Note: Please do not omit any objects involved when labeling.

Figure 7. Completeness of labeling

2. **Correctness**: Ensure that each object correctly corresponds to its label, without any mismatches between objects and their labels.



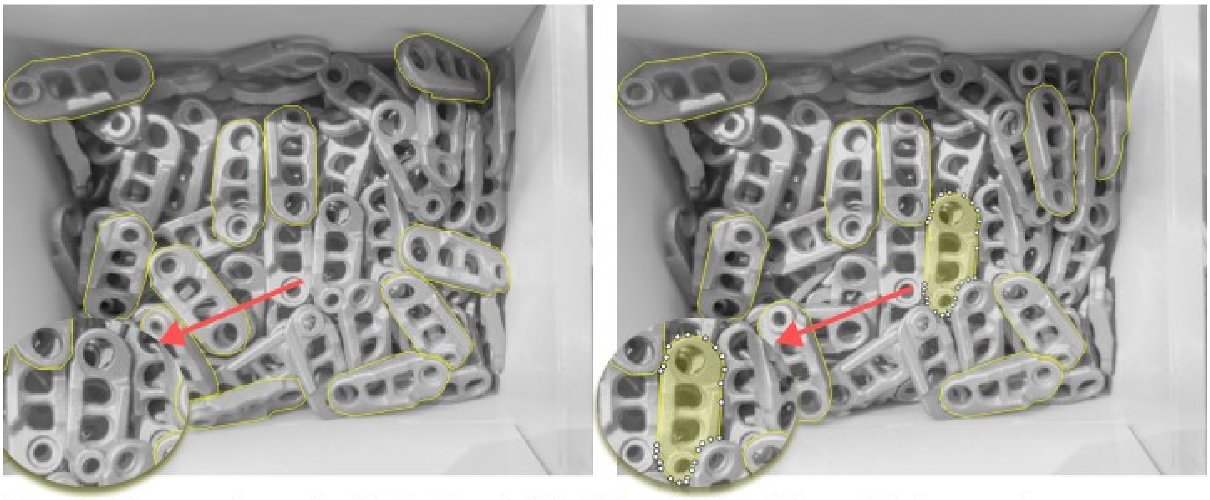
Bad example: Wrong labeling. A Mentos is labeled as a yida.

Good example: Correctly labeling all objects.

Figure 8. Correctness of labeling

3. **Consistency**: All data should be labeled under the same rules. For example, if the labeling rules stipulate that only objects with 85% of the entire surface exposed should be labeled, then all objects that meet this rule should be labeled without exception.

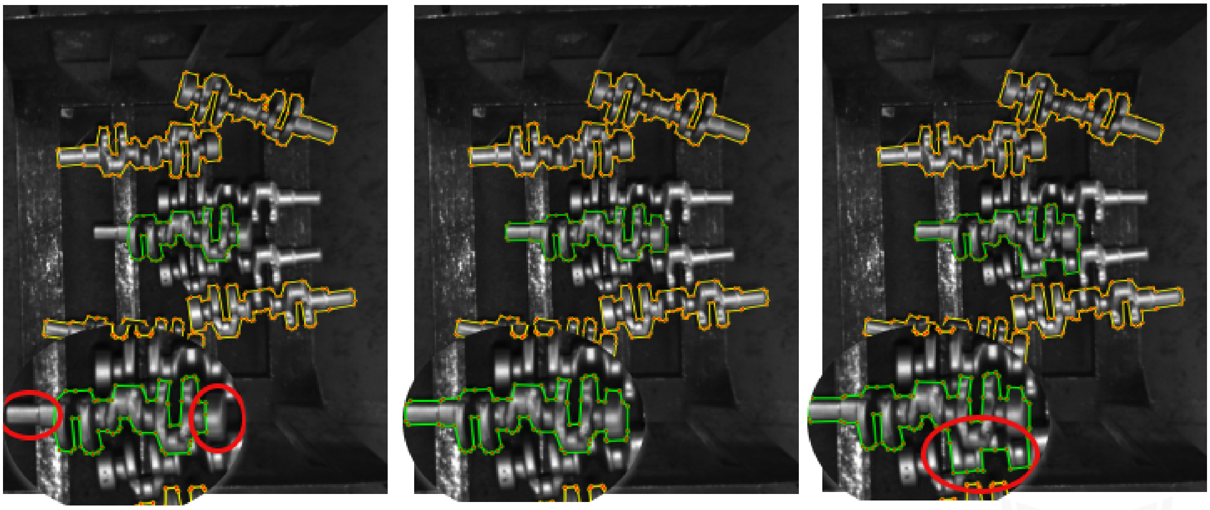
Bad example: The same track link's labeling status differs across training data images.



Note: In this case, please check whether the label(s) applied to this track link meets the requirements. If yes, please label the track link in the images in which it is not yet labeled; otherwise, please delete all the label(s) applied to this track link.

Figure 9. Consistency of labeling

4. **Accuracy**: The drawn contour of the labeled object should closely fit the actual outline of the object, not excluding parts of the object or including parts of another object.



Bad example: Label only part of an object.

Good example: Correct labeling.

Bad example: Include other objects in one labeling.

Note: Please label each crankshaft completely and exclusively and do not omit necessary pixels or include additional pixels.

Figure 10. Comparison of different labeling accuracies

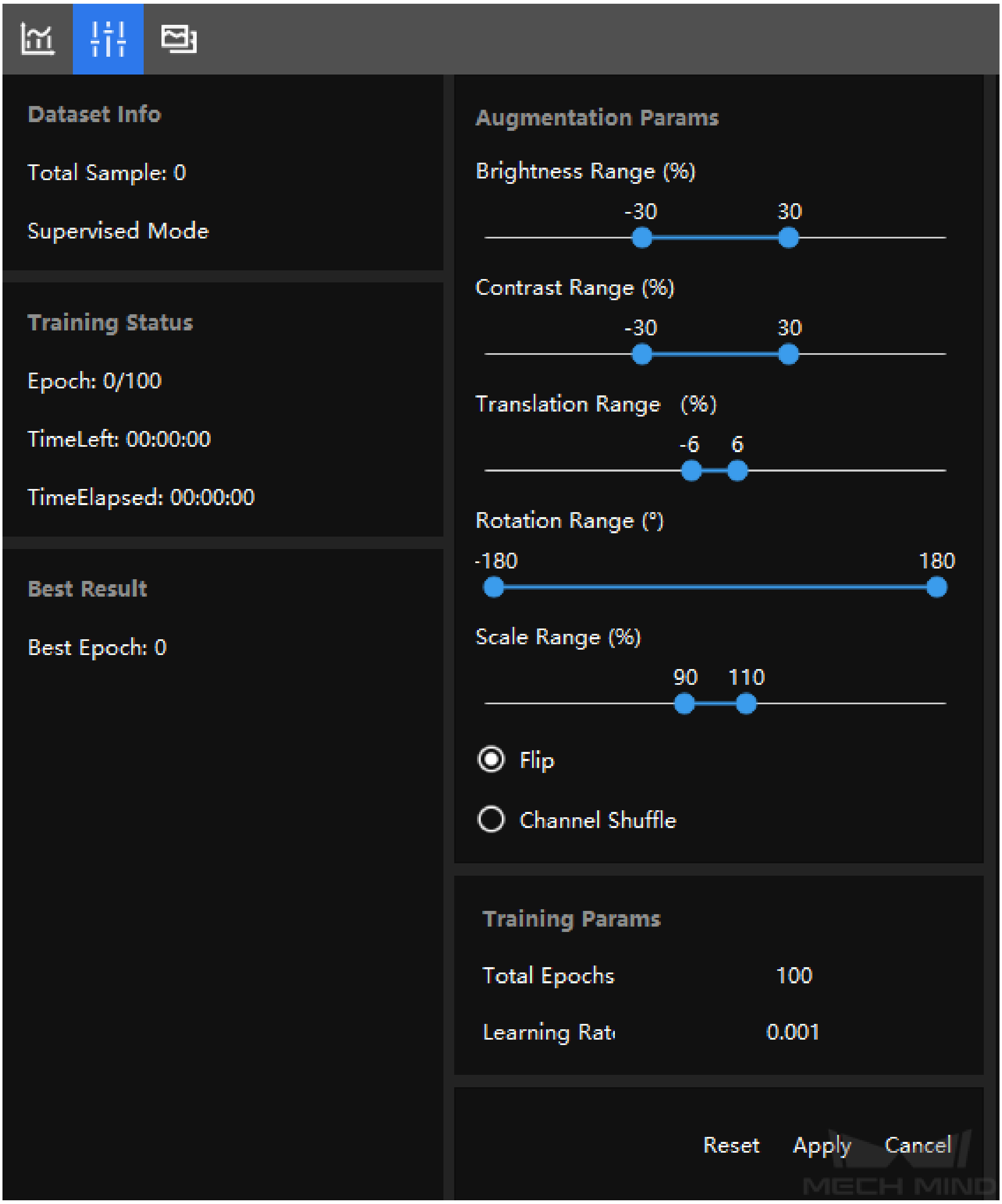
2.1.4 Train the Model

Using Mech-DLK to Train the Model

Please see *Mech-DLK Quick Start* for detailed instructions.

Training Parameters

In most cases, training with the default parameters is sufficient; the following parameters should only be adjusted under special requirements.



Dataset Info

Total Sample: 0

Supervised Mode

Training Status

Epoch: 0/100

TimeLeft: 00:00:00

TimeElapsed: 00:00:00

Best Result

Best Epoch: 0

Augmentation Params

Brightness Range (%): -30 30

Contrast Range (%): -30 30

Translation Range (%): -6 6

Rotation Range (°): -180 180

Scale Range (%): 90 110

● Flip

○ Channel Shuffle

Training Params

Total Epochs	100
Learning Rat	0.001

Figure 1. Training Parameters in Mech-DLK

Brightness Range

When the on-site lighting conditions vary greatly and cannot be stabilized or compensated by shading or supplemental light, it is necessary to increase the brightness range accordingly. Otherwise, please keep the default values.

Contrast Range

When **the difference between the object and the background is not obvious**, the contrast range can be adjusted accordingly to facilitate the model' s learning on object features. The contrast range is usually adjusted in conjunction with the brightness range. This case seldom occurs, so normally the contrast range does not need to be adjusted.

Translation Range

If the object container, such as bin, pallet, etc., moves over a relatively wide range, the translation range needs to be increased. Otherwise, please keep the default value.

Rotation Range

When the object is at a fixed position and its orientation needs to be distinguished, the range needs to be adjusted to 0–0 to avoid affecting the model' s learning on orientation features due to rotation. Otherwise, please keep the default value.

Scale Range

When **the heights of objects**, i.e., distances to the camera, vary greatly, or **the volumes of objects** at the same height vary greatly, please increase the scale range. Otherwise, please keep the default value.

Flip

When the object is at a fixed position and its orientation needs to be distinguished, please deselect the Flip option to avoid affecting the model' s learning on orientation features. Otherwise, please keep the default setting. This parameter is usually adjusted in conjunction with the Rotation Range parameter.

Channel Shuffle

This option enables the color channel shuffle function during image data processing to enhance the model' s generalization ability. When **objects have similar shapes and the classification relies on object colors**, please deselect this option to avoid affecting the model' s learning on color features. Otherwise, please keep the default setting.

Total Epochs

When no classification is needed and object features are simple, please set the total number of epochs to be within 600. Otherwise, please set it to be within 1,000.

Learning Rate

This parameter normally does not need to be adjusted. When the accuracy is low (< 0.8) or drops steeply, please set it to one-tenth of the current value.

Attention: Larger ranges of parameters do not necessarily lead to a better training effect. If any of the parameters' ranges unnecessarily cover values nonexistent in reality, the training effect will be adversely affected. For instance, when the light conditions are stable but the brightness range is too large, the training effect will not be as good as when the brightness range is of the proper value.

2.1.5 Use the Model

Using Instance Segmentation Model in Mech-Vision

Below is a typical piece of programming involving instance segmentation in Mech-Vision.

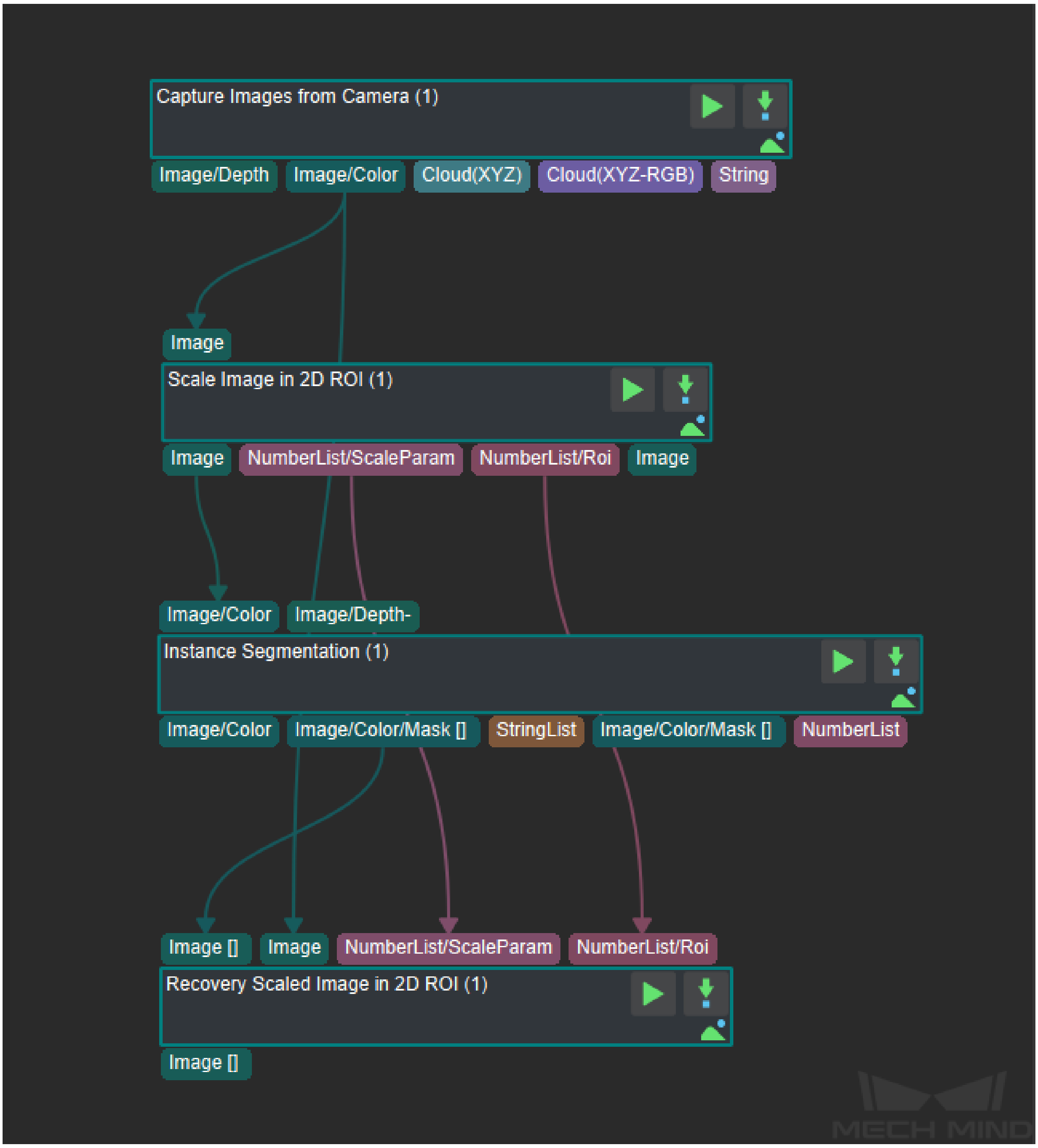
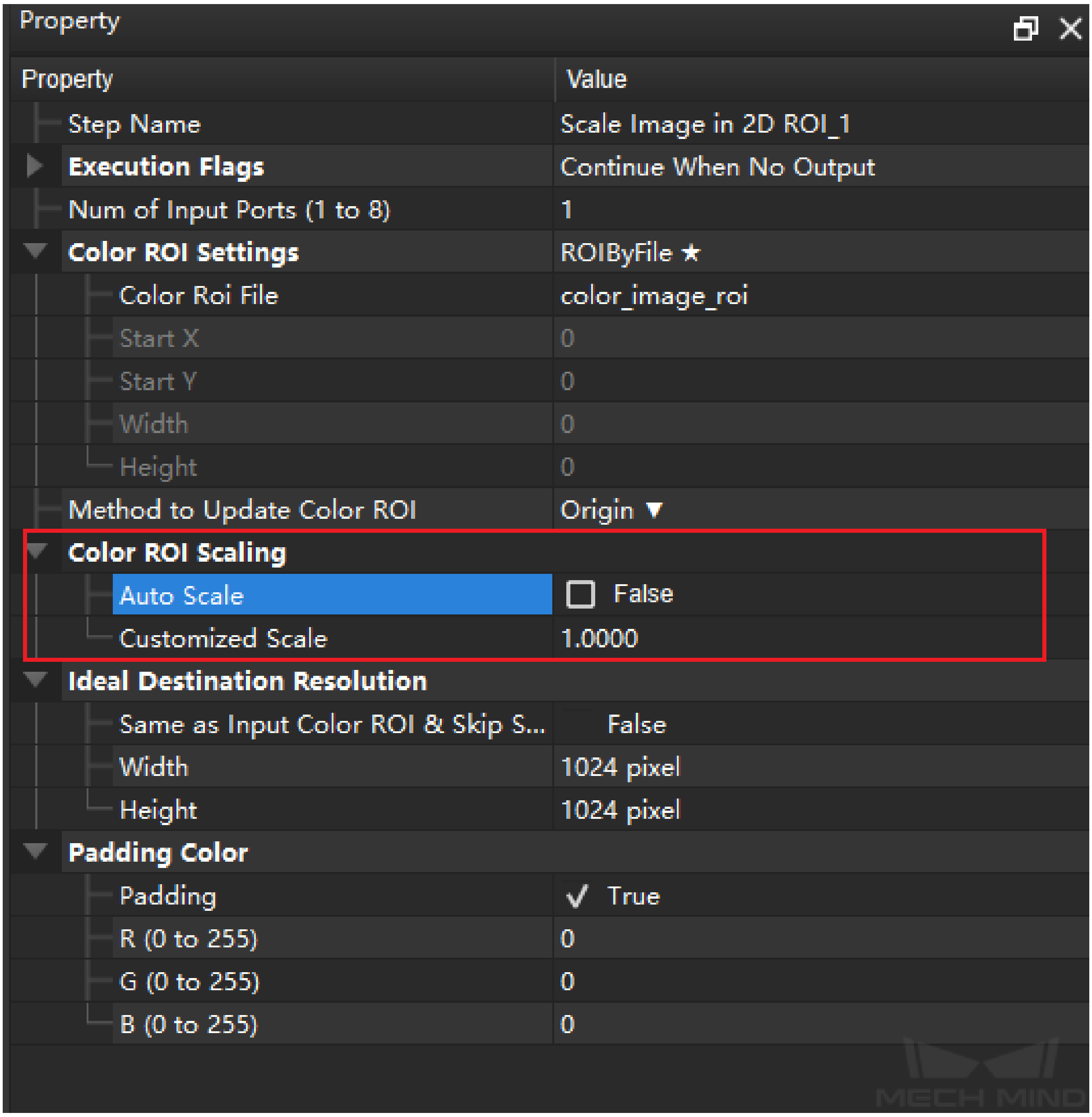


Figure 1. Programming involving instance segmentation in Mech-Vision

Step 1: **Capture images from the camera.**

Step 2: **Scale the images within the 2D ROI** to keep the ROI of the actual data consistent with that of the training data. Normally, please set **Auto Scale** under **Color ROI Scaling** in the parameter panel to `True`, as shown in *Figure 2*.



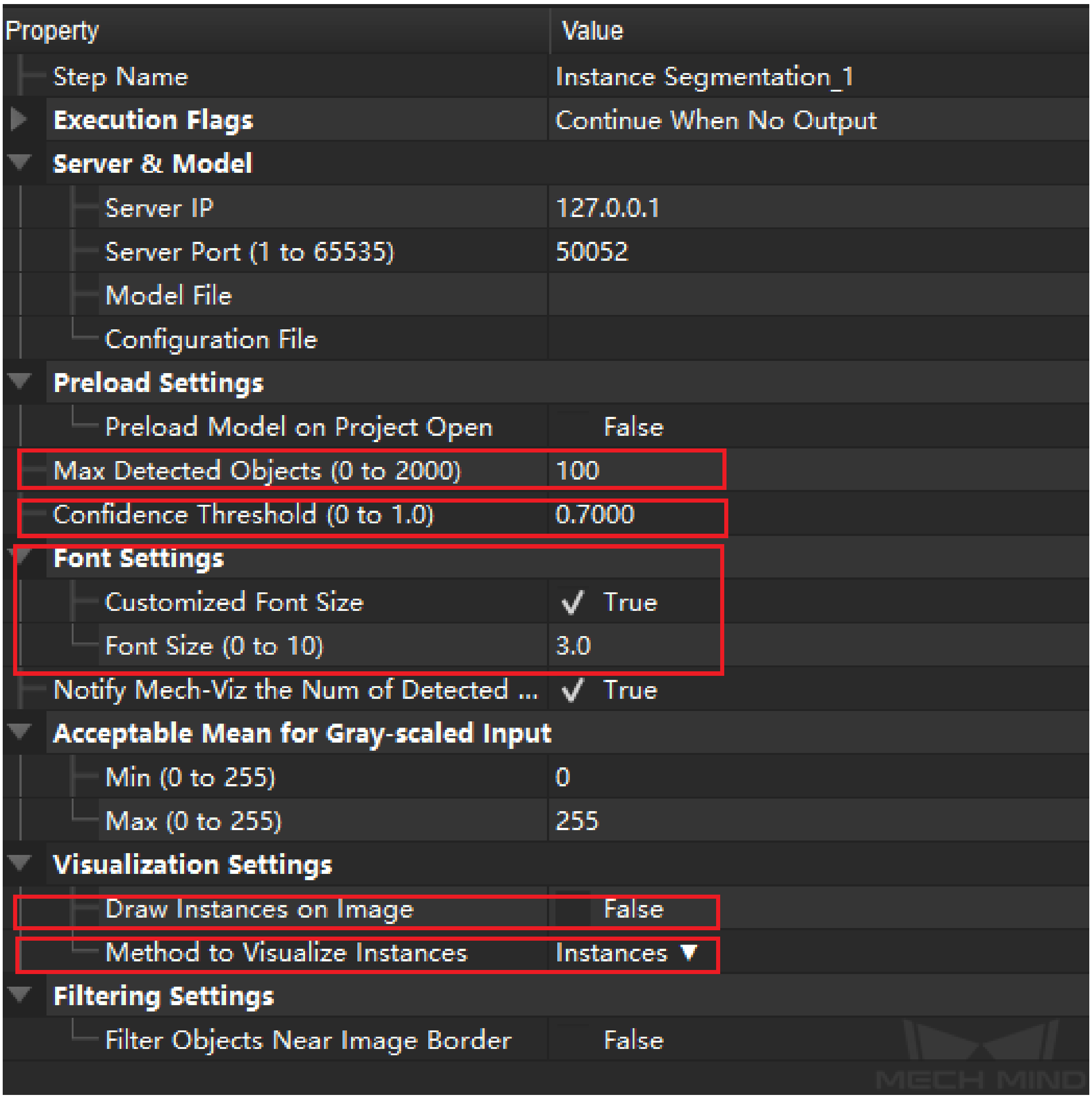
Property

Property	Value
Step Name	Scale Image in 2D ROI_1
Execution Flags	Continue When No Output
Num of Input Ports (1 to 8)	1
Color ROI Settings	ROIByFile ★
Color Roi File	color_image_roi
Start X	0
Start Y	0
Width	0
Height	0
Method to Update Color ROI	Origin ▼
Color ROI Scaling	
Auto Scale	□ False
Customized Scale	1.0000
Ideal Destination Resolution	
Same as Input Color ROI & Skip S...	False
Width	1024 pixel
Height	1024 pixel
Padding Color	
Padding	✓ True
R (0 to 255)	0
G (0 to 255)	0
B (0 to 255)	0

Figure 2. Auto scale

Step 3: **Instance Segmentation** outputs the instance segmentation results. Please determine whether the results meet the project requirements, whether the masks are complete, and whether there are omissions or mistakes.

After ensuring the paths of the **Model File** and the **Configuration File** are correct, please check the settings of the parameters marked with red boxes in *Figure 3* based on the following rules.



Property	Value
Step Name	Instance Segmentation_1
Execution Flags	Continue When No Output
Server & Model	
Server IP	127.0.0.1
Server Port (1 to 65535)	50052
Model File	
Configuration File	
Preload Settings	
Preload Model on Project Open	False
Max Detected Objects (0 to 2000)	100
Confidence Threshold (0 to 1.0)	0.7000
Font Settings	
Customized Font Size	✓ True
Font Size (0 to 10)	3.0
Notify Mech-Viz the Num of Detected ...	✓ True
Acceptable Mean for Gray-scaled Input	
Min (0 to 255)	0
Max (0 to 255)	255
Visualization Settings	
Draw Instances on Image	False
Method to Visualize Instances	Instances ▼
Filtering Settings	
Filter Objects Near Image Border	False

Figure 3. Important parameters of the **Instance Segmentation** Step

Preload Settings -> Max Detected Objects (0 to 2000) Please increase the value when the model needs to detect many objects at a time. If the number of objects that the model needs to detect at a time is small, and the project requires a short cycle time, please decrease the value of this parameter to optimize the cycle time.

Preload Settings -> Confidence Threshold (0 to 1.0) Normally, the confidence threshold can be kept at the default value `0.7`. When a detected object’ s confidence is lower than this value, this object is considered not meeting the requirements for picking and its confidence will be displayed in red. Please tune this parameter based on actual project needs. For instance, when objects with lower confidences still have comparatively complete masks and meet the requirements for picking, the confidence threshold can be lowered to include more objects as candidates for picking.

Font Settings -> Customized Font Size Normally the default value works well. When the objects to recognize are small, please decrease the font size to better display the results.

Visualization Settings -> Draw Instances on Image When set to `True`, the recognized objects' masks are displayed to show the effect of instance segmentation. Setting it to `False` during the actual project run can reduce the cycle time of the project.

Visualization Settings -> Method to Visualize Instances

- When set to `Classes`: objects' masks are displayed in different colors based on object labels.
- When set to `Instances`: the mask of each object is displayed in a different color.
- When set to `Threshold`: the masks of objects whose confidences are above the threshold are displayed in green, otherwise, red.

Tip: If a project requires two instance segmentation models, please add two **Instance Segmentation** Steps, and set the model files and configuration files separately. In addition, please set the **Server IP** of the two Steps to `127.0.0.1:50052` and `127.0.0.1:50053`, respectively; otherwise, port conflicts will occur.

Step 4: **Recovery Scaled Image in 2D ROI** restores the image to its original size.

2.2 Image Classification

2.2.1 Introduction to Image Classification

What Image Classification Does

Image classification solves the problem of **what an object is**.

In industrial scenarios, image classification means recognizing workpieces' types, models, front and back faces, and correctness of placing, etc.

In the example below, the target objects are almonds, walnuts, and cashews. Given an image, the image classification model determines whether it is an almond, walnut, or cashew in the image and assigns the corresponding label to the image.

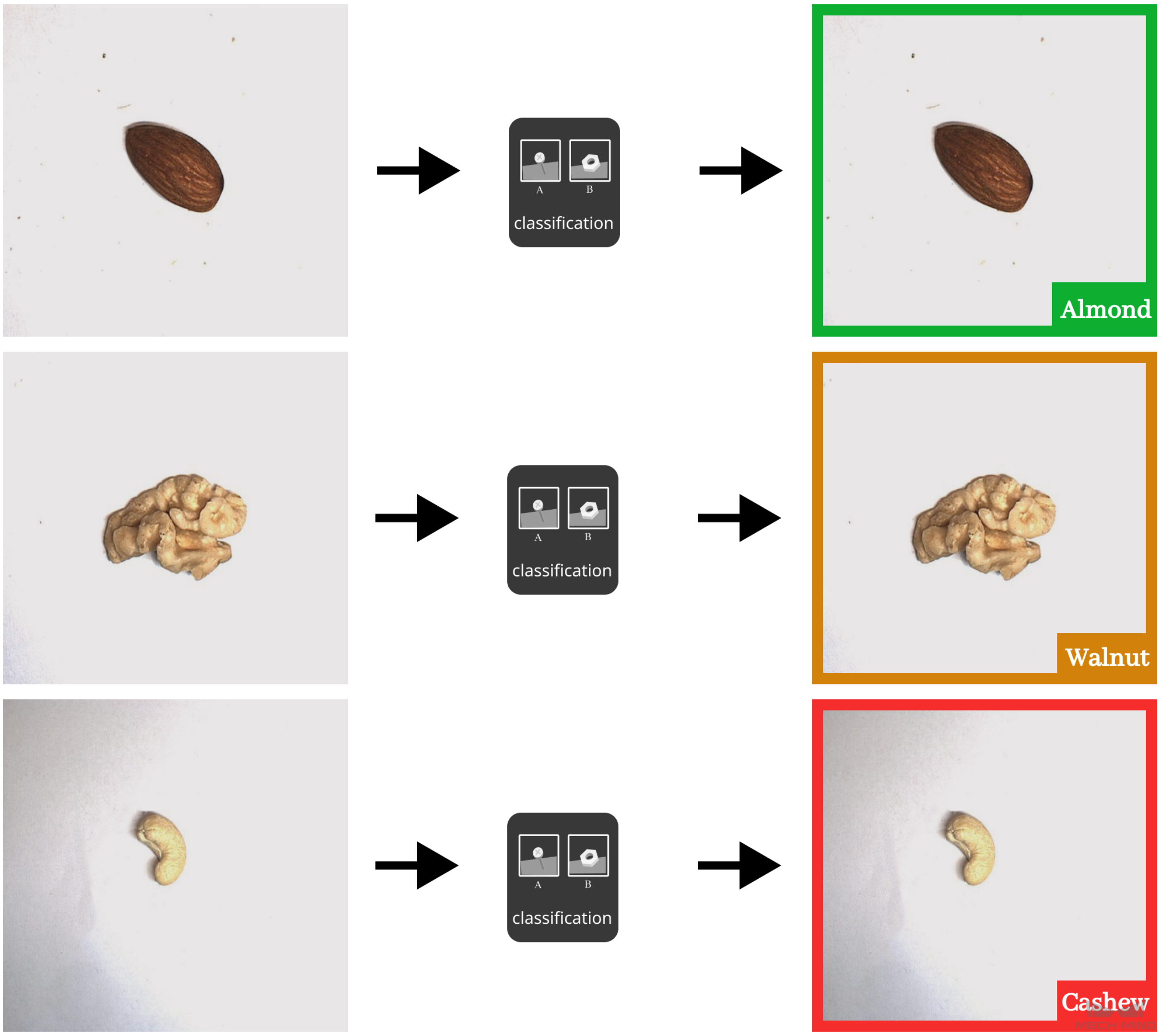


Figure 1. Classifying and assigning labels to images

Attention: A label assigned by an image classification model is for the entire image. If an image contains multiple objects of different classes, classification of such an image requires the image to be segmented first so that each segment contains only one object. Alternatively, *instance segmentation* or object detection can be used based on actual needs.

Typical Industrial Application Scenarios of Image Classification

The image classification model can be used for projects that involve classifying different images. The following are some typical application scenarios:

- Distinguishing the type, orientation, front and back faces of workpieces for machine tending projects.

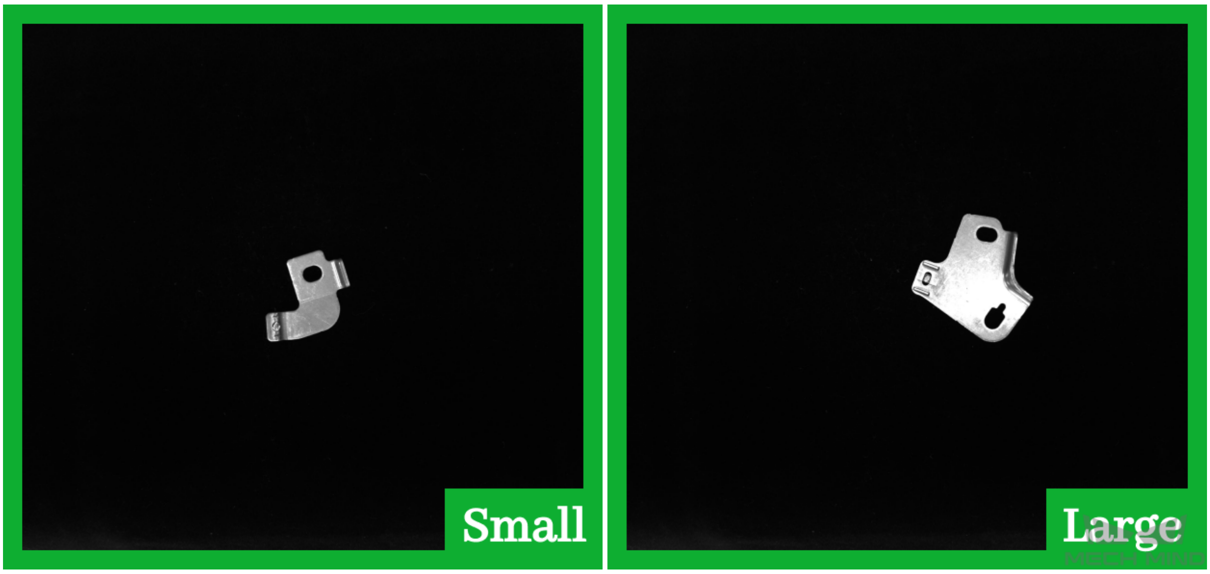


Figure 2. Recognizing different types of workpieces for machine tending

- Judging whether the objects are placed correctly for assembly or picking.

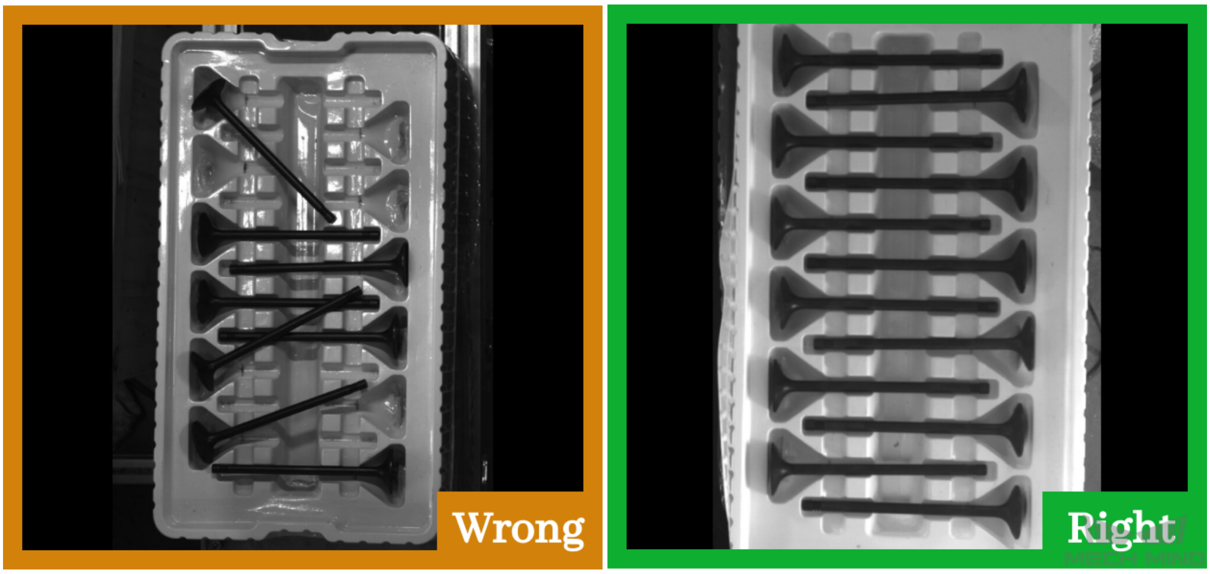


Figure 3. Determining whether objects are placed correctly

Application Process of Image Classification

Given a sufficient quantity of correctly labeled training image data obtained from actual application scenarios, an image classification model can be properly trained to correctly classify target objects. The application process of image classification is as follows:

- *Collect the Training Data*: take enough pictures of target objects in each class with the camera.
- *Label the Training Data*: assign a label corresponding to the object to each picture.
- *Train the Model*: feed the labeled data to the image classification model for training.
- *Use the Model*: apply the trained model in an actual project.

2.2.2 Collect the Training Data

Check the Data Collection Environment

Please see *Collect the Training Data* for details about setting up the image data collection environment.

Attention: Image classification model is susceptible to changes in **lighting conditions**. When collecting data, please keep the lighting conditions consistent throughout the process. If the on-site lighting conditions change over the day, image data should be collected respectively under different conditions.

Quantity of Data to Collect

- The recommended quantity is 20 images for each class.

Object Placing for Data Collection

All different placing conditions should be included in the dataset, and the number of images for each placing condition should be reasonably allocated based on the actual project conditions.

For example, if the objects come in horizontal and vertical poses in the actual application, but only images of horizontal incoming objects are collected and used for training, then the resulting model’s performance on the vertical objects cannot be guaranteed.

Therefore, when collecting data, please **take all circumstances in the actual application into consideration** as much as possible. Factors include the following:

- All **object orientations** that may appear in the actual application;
- All **object positions** that may appear in the actual application.

1. **Orientations**

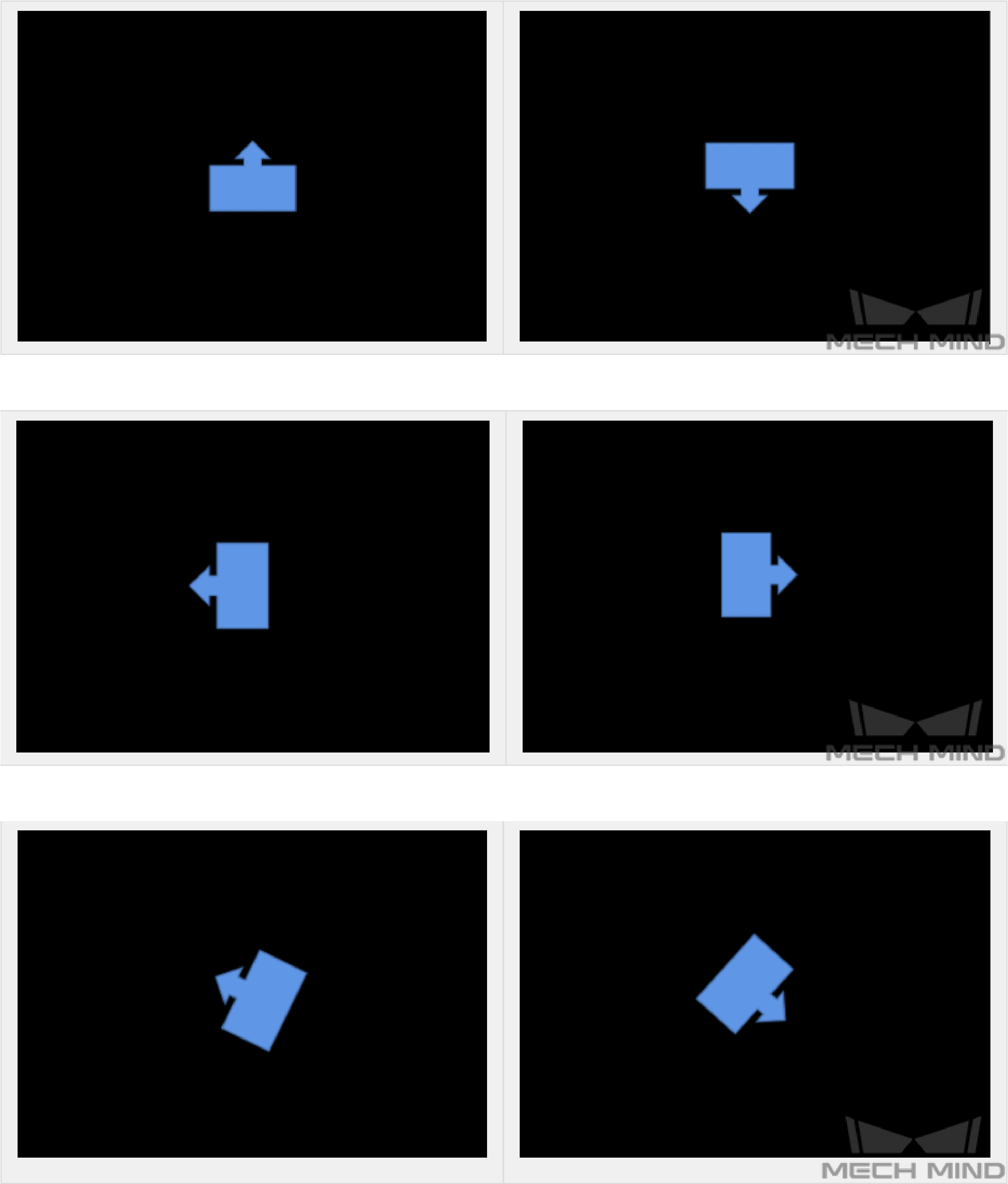


Figure 1. Different orientations

2. **Positions**

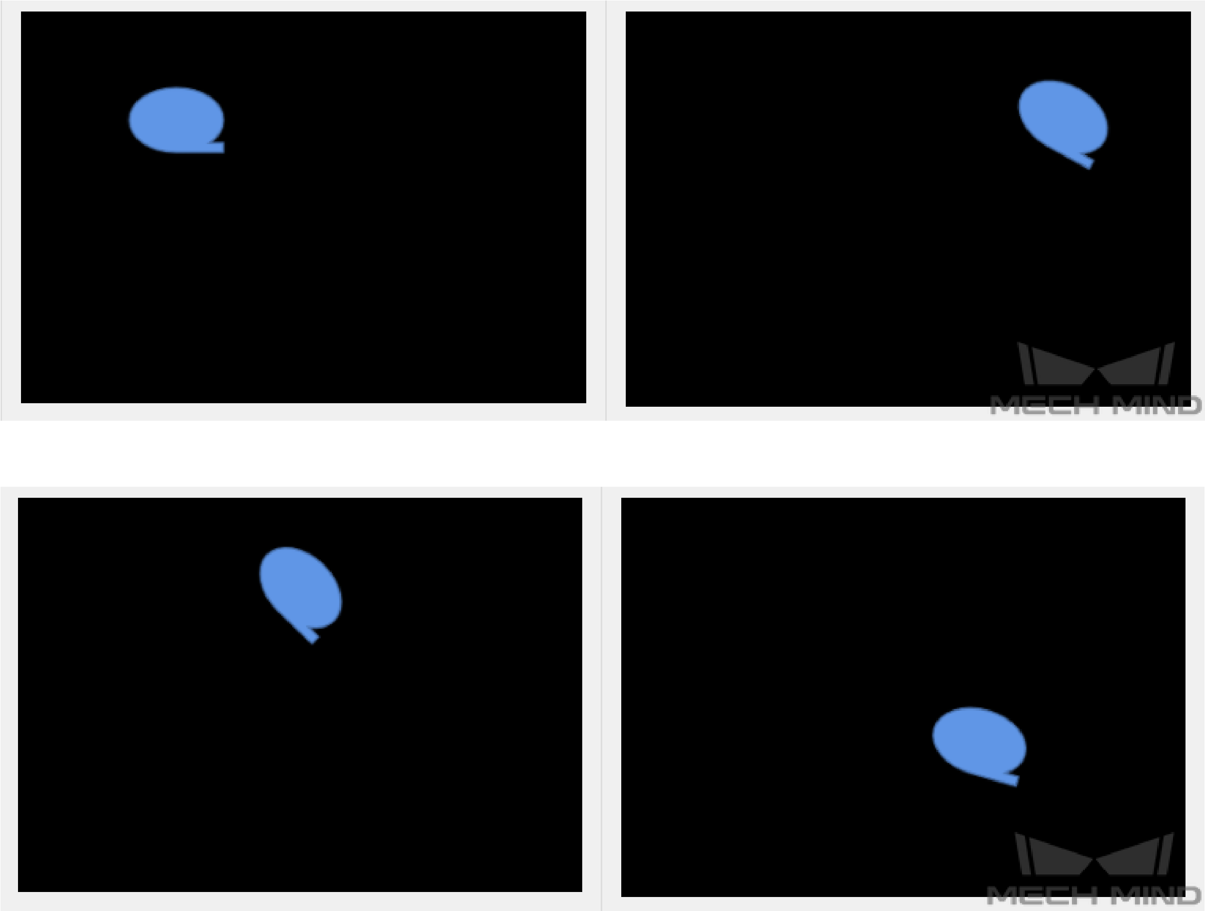


Figure 2. Different positions

Use Mech-Vision to Collect Data

After checking the data collection environment, determining the data quantity to collect, and listing all the possible ways of object placing, please collect the data following the instructions in *Use Mech-Vision to Collect Data*.

Data Collection Examples from Past Projects

Valve, single class

- The front and back faces of the valve need to be distinguished.
- Valves’ positions vary relatively slightly.
- 10 images for the front face and 10 for the back collected.

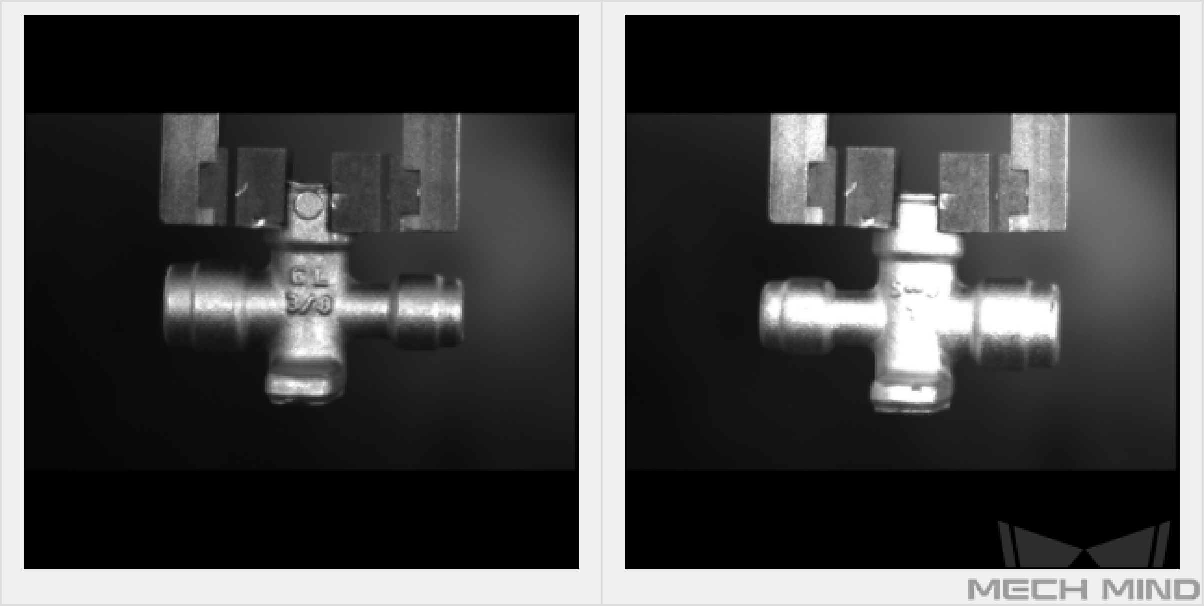


Figure 3. Front and back faces of the valve

Engine valve, single class

- Need to determine whether an engine valve is correctly placed in a slot.
- If not in a slot, an engine valve may be in a variety of poses, so different positions and orientations need to be considered for data collection. Therefore, about 20 images need to be collected for this case.
- If in a slot, an engine valve may be in different positions, but can only be in one of two orientations (as shown in *Figure 4*). Therefore, about 10 images need to be collected for this case.

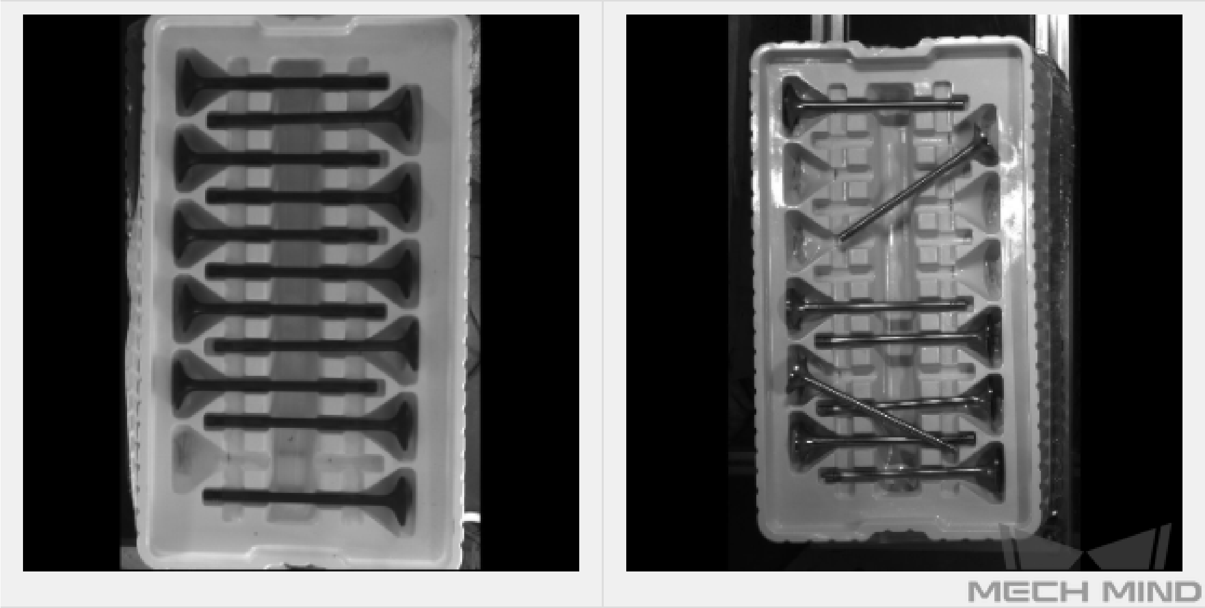


Figure 4. Engine valve in a slot and not in a slot

Sheet metal parts, two classes

- Object size needs to be distinguished.
- Objects may be in different positions and orientations.
- 20 images collected for the front face and 20 for the back.

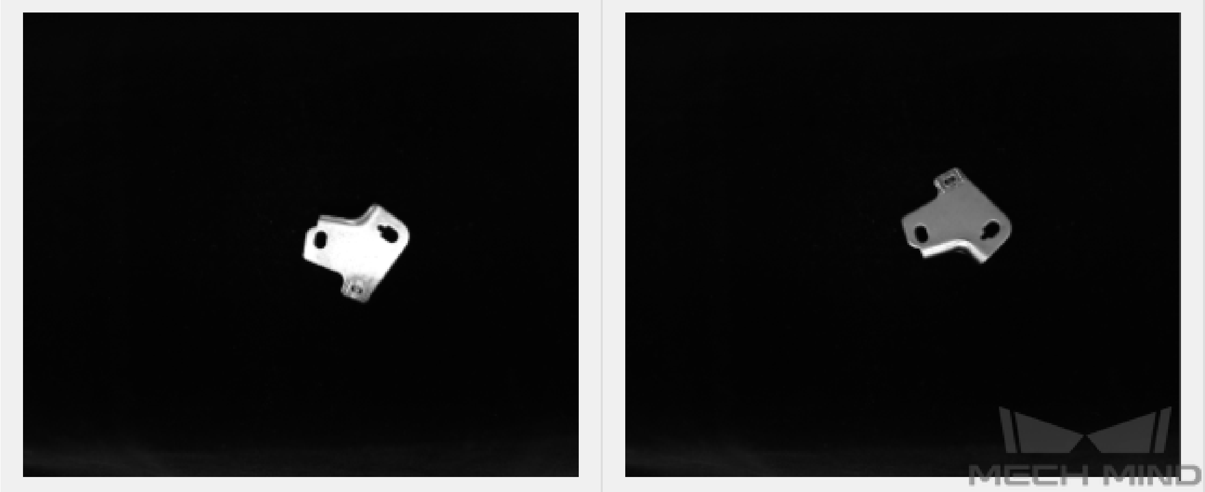


Figure 5. Front & back faces of sheet metal parts

2.2.3 Label the Training Data

Create Label(s)

Please create label(s) based on project needs. For instance, if the project need is distinguishing the front and back faces of a workpiece, please create the labels “front” and “back” .

Attention: Label names should be relevant to the objects and easily recognizable. Please do not use meaningless names like a, b, tmp, etc. Label names should only include letters or numbers.

Determine Method of Labeling

1. If the task is to distinguish different parts of a single object, please label the prominent feature for each part with boxes, as shown below.

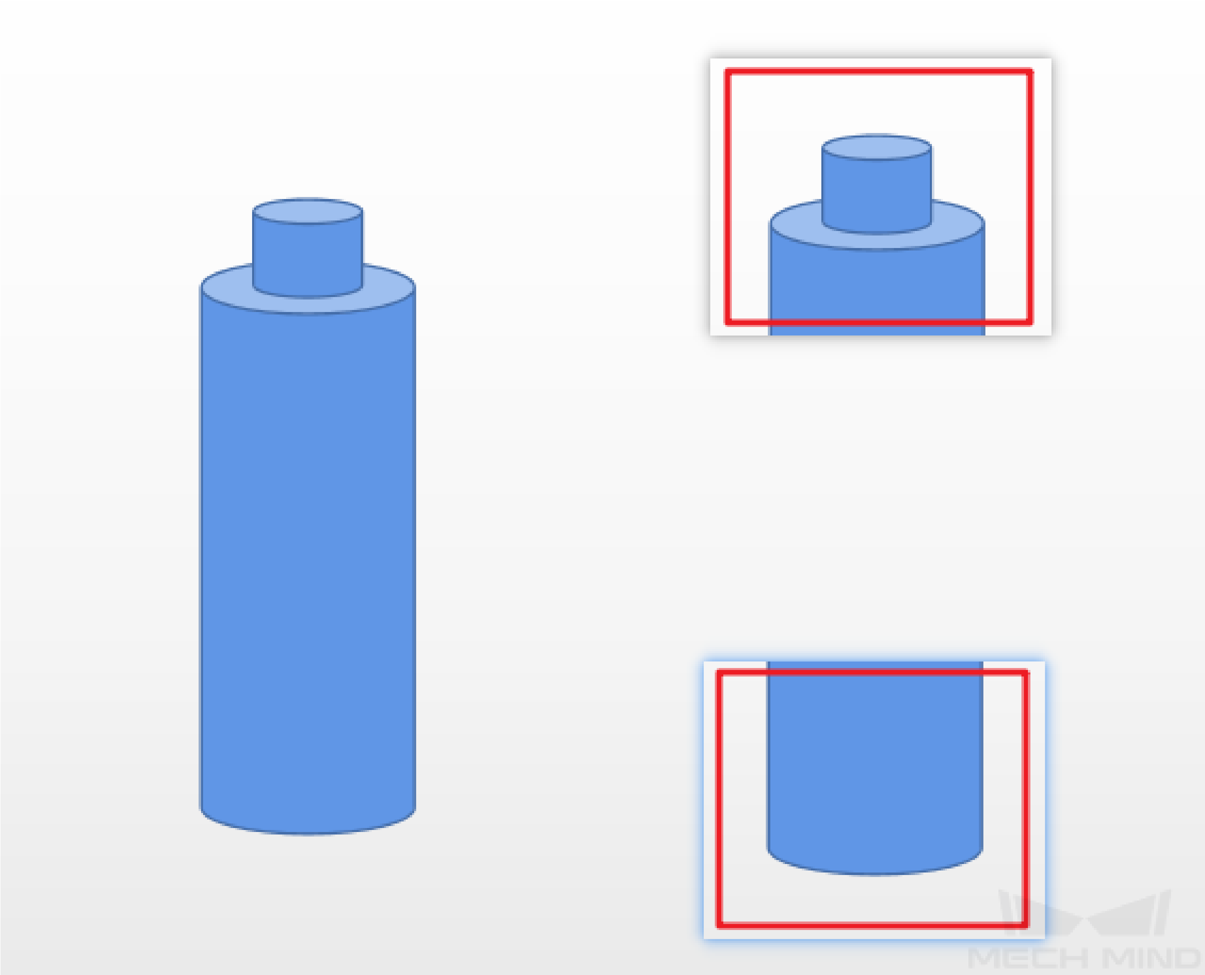


Figure 1. Labeling the prominent features for different parts of a single object

2. If the task is to distinguish objects of different classes, please envelop the entire object with a box, as shown below.

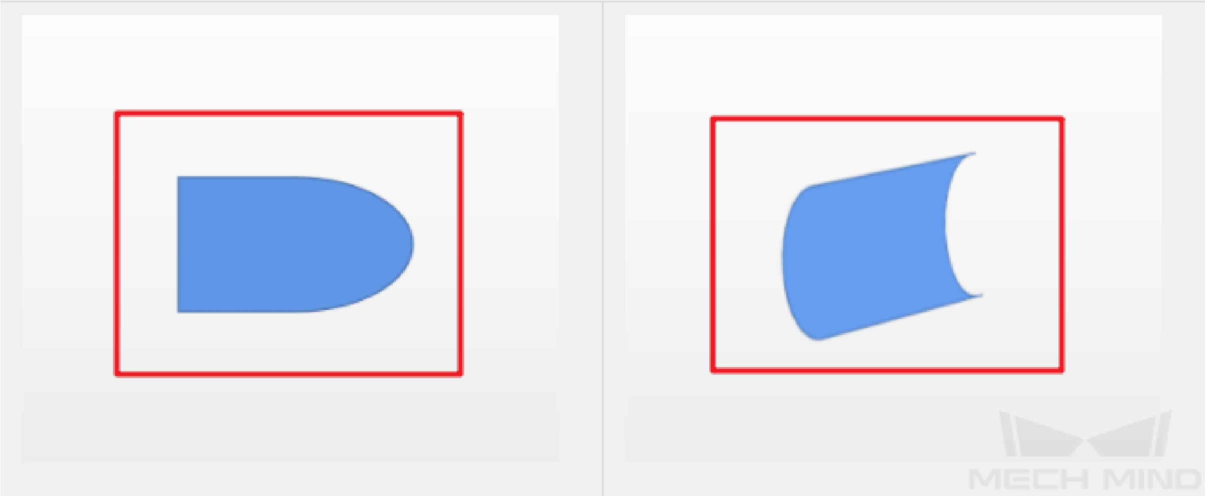


Figure 2. Labeling different objects with boxes

3. If the input images for the actual application have the background removed, please label the entire contour of each object in the training dataset, as shown below.

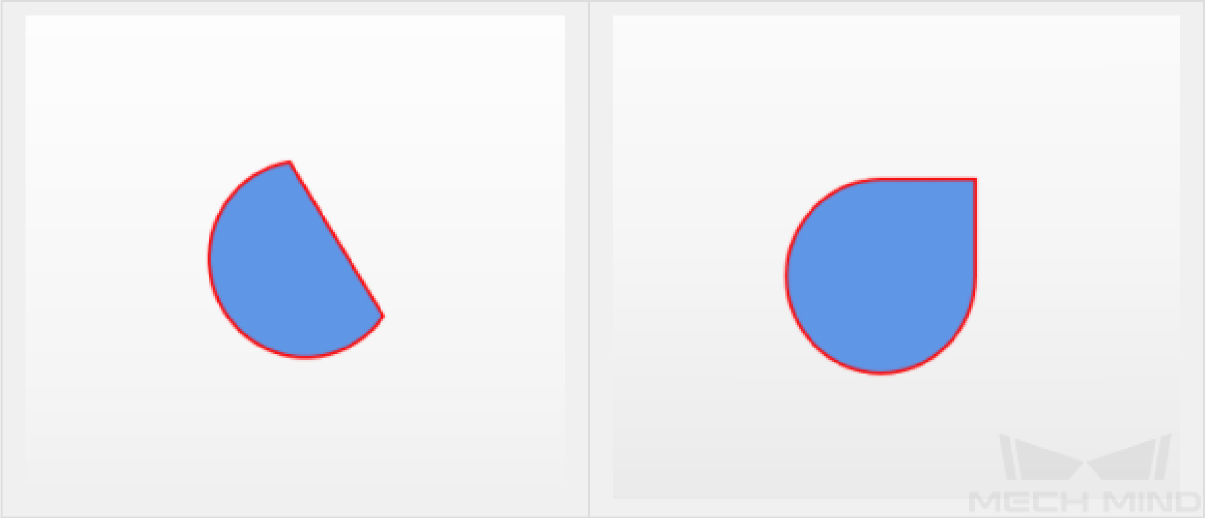


Figure 3. Labeling the entire contours of objects

Attention: Please ensure the quality of labeling. Any incorrect labels will adversely affect model performance. For instance, if in ten images of the workpiece’ s front face, one is labeled as “back” , the classification performance will be severely affected.

2.2.4 Train the Model

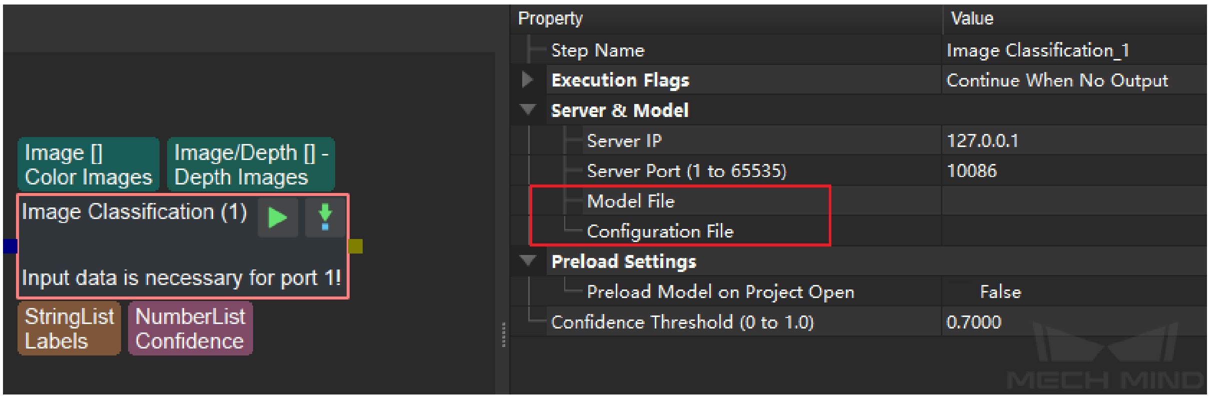
Using Mech-DLK to Train the Model

Please see *Train the Model* for details.

2.2.5 Use the Model

Using Image Classification Model in Mech-Vision

Attention: The model file is in the .pth format and the configuration file is in the .json format. The input data to the **Image Classification** Step should be consistent with the model training data.



Input data is necessary for port 1!

Property	Value
Step Name	Image Classification_1
Execution Flags	Continue When No Output
Server & Model	
Server IP	127.0.0.1
Server Port (1 to 65535)	10086
Model File	
Configuration File	
Preload Settings	
Preload Model on Project Open	False
Confidence Threshold (0 to 1.0)	0.7000

Figure 1. The Image Classification Step in Mech-Vision

CHAPTER

THREE

TYPICAL SCENARIOS

3.1 Box Palletizing/Depalletizing

Box palletizing/depalletizing projects usually utilize *instance segmentation* to segment each box in an image and identify its position.

Mech-Mind Robotics provides a **Super Model** tailored for box palletizing/depalletizing scenarios, which can be used directly in Mech-Vision to segment most box types without training.

> A **Super Model** refers to a generic deep learning model trained on massive amounts of data and applicable to certain types of objects, such as boxes, sacks, shipping packages, etc.

The overall application process is as follows:

1. Use the Super Model and check its performance.

   Please see *Use the Model* for instructions on using an instance segmentation model in Mech-Vision. Check if all box types involved in the project can be correctly segmented. If so, the project can be run using the Super Model, and there is no need to do the following steps; otherwise, please proceed to the next step.

   Attention: Regardless of how well the Super Model performs, please keep all testing data for any possible further testings.

2. Collect image data on boxes not correctly segmented.

   In general, the Super Model can recognize most boxes. In rare cases, such as when the boxes are fitted closely together or have complicated surface patterns, segmentation errors may occur, or masks may be incomplete. In such cases, image data of the boxes that are not correctly recognized need to be collected for further training.

   For example, if the Super Model can correctly segment 18 out of 20 box types, then image data on the remaining 2 types need to be collected.

   Another example is that, if all separately placed boxes are correctly segmented, but closely fitted boxes are not, then image data on closely fitted boxes need to be collected.

   - Quantity of images to collect: twenty images for each box type (or type of placing)
   - Data requirements:
     - Collect a total of 10 images of closely fitted boxes at different heights (top, middle, and bottom layers);

- Collect a total of 10 images of full layers, half-full layers, partially-filled layers at different heights (top, middle, and bottom layers).

The following are some examples of images to collect:

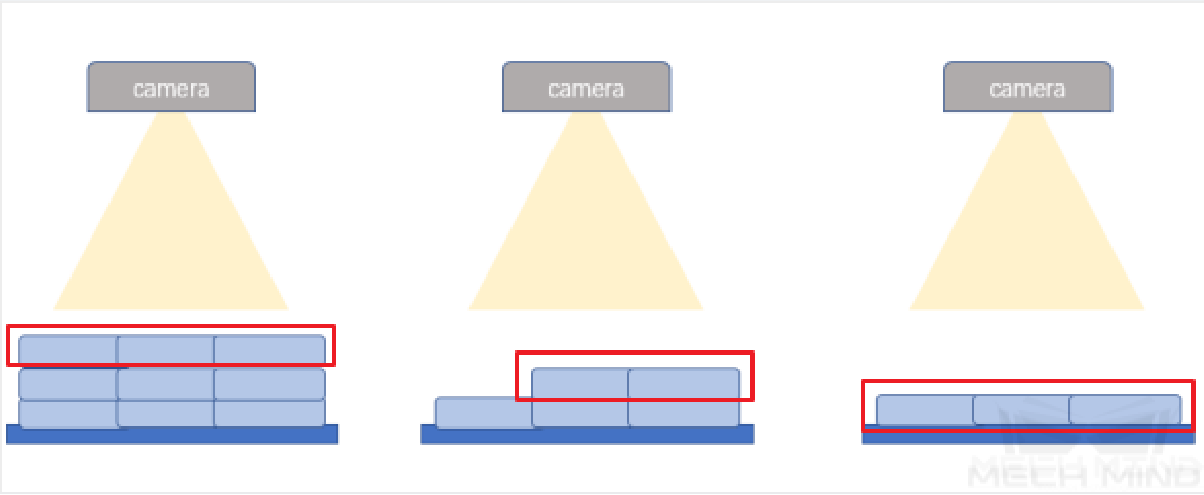


Figure 1. Closely fitted boxes at the top (left), middle (center), and bottom layers (right)

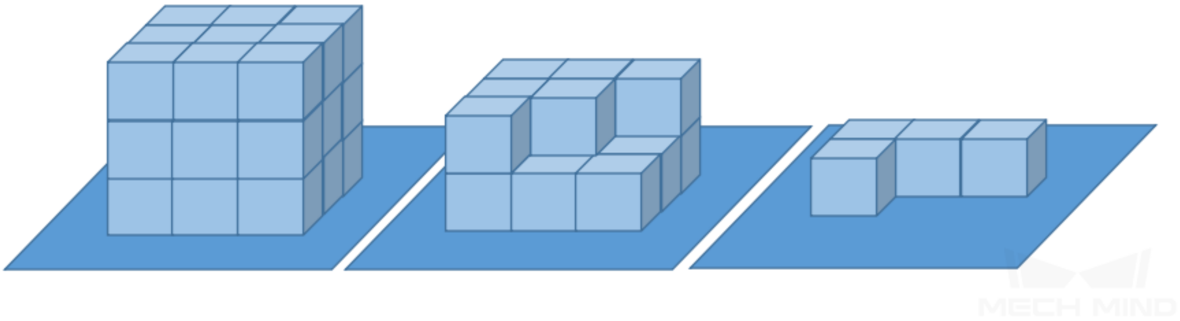


Figure 2. A full top layer (left), a half-full middle layer (center), and a partially-filled bottom layer (right)

3. Remove the background from the images.

   As boxes usually come in stacks on pallets, background removal helps avoid interference in recognition and significantly improves the model' s performance. Background removal can be done in Mech-Vision using the Steps shown in *Figure 3*.

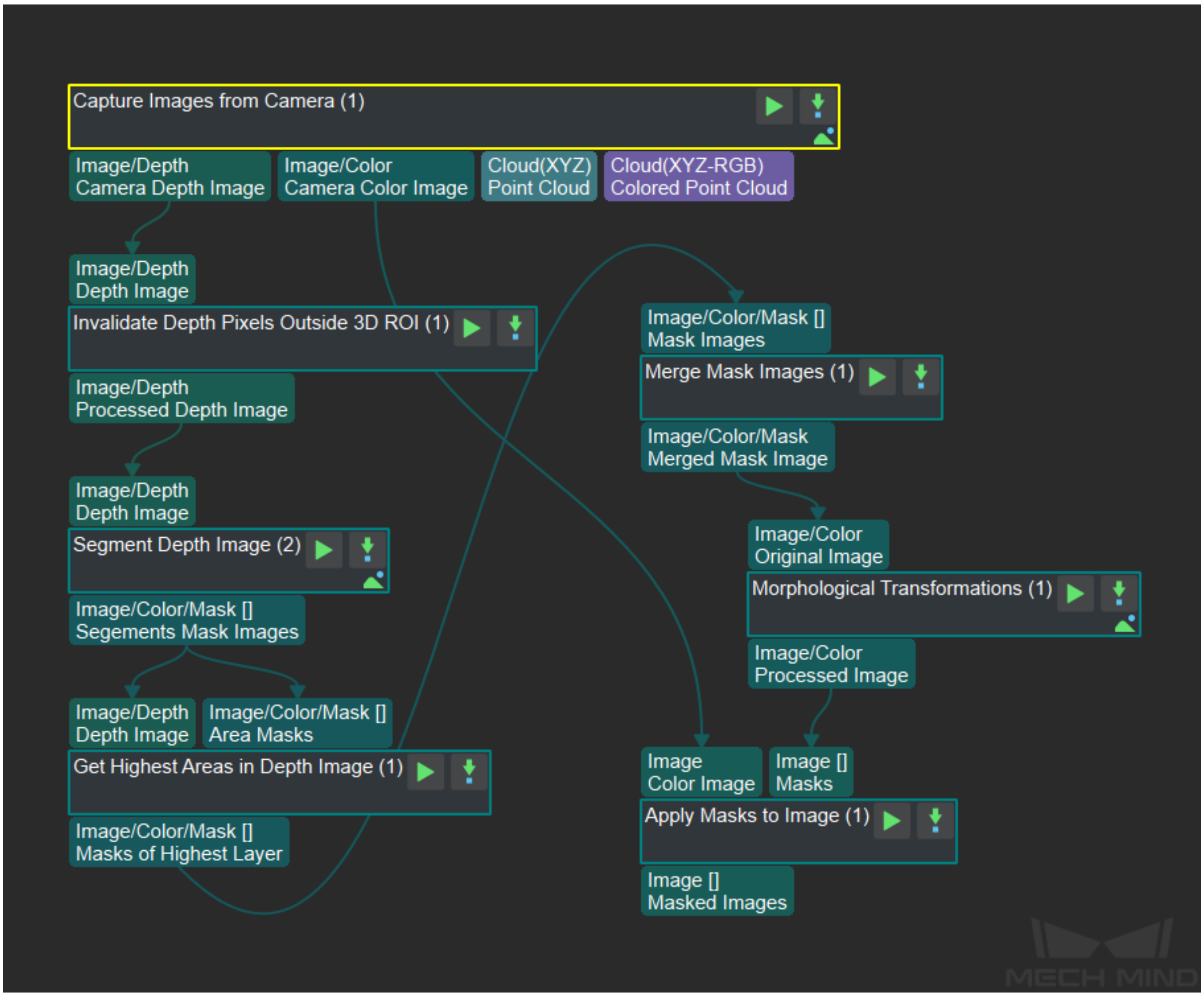


Figure 3. Mech-Vision Steps for removing background from images

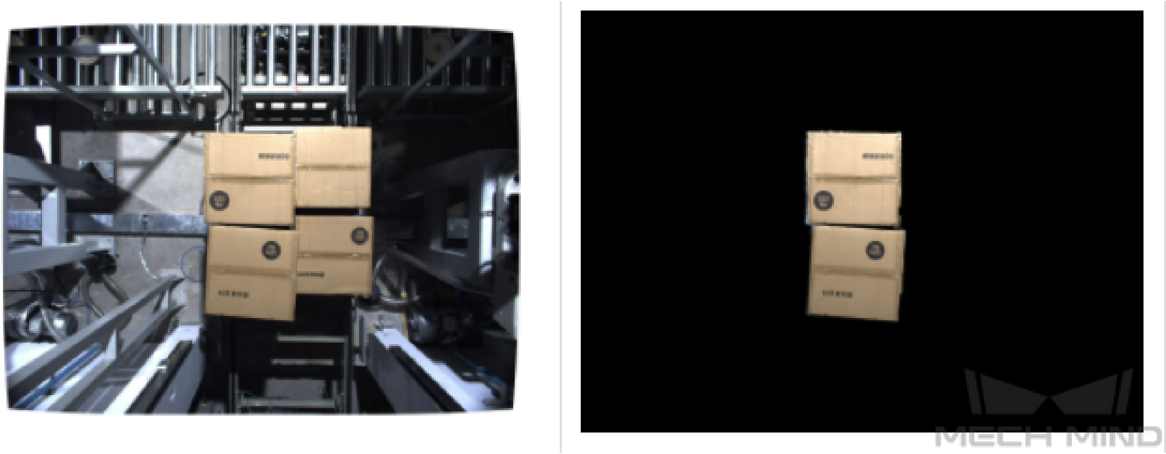


Figure 4. The image before (left) and after (right) background removal

4. Label the training data.

   Please see *Label the Training Data* for instructions on data labeling. Palletizing/depalletizing scenarios only require labeling the upper surfaces of boxes, and only those upper surfaces that are completely exposed need to be labeled.

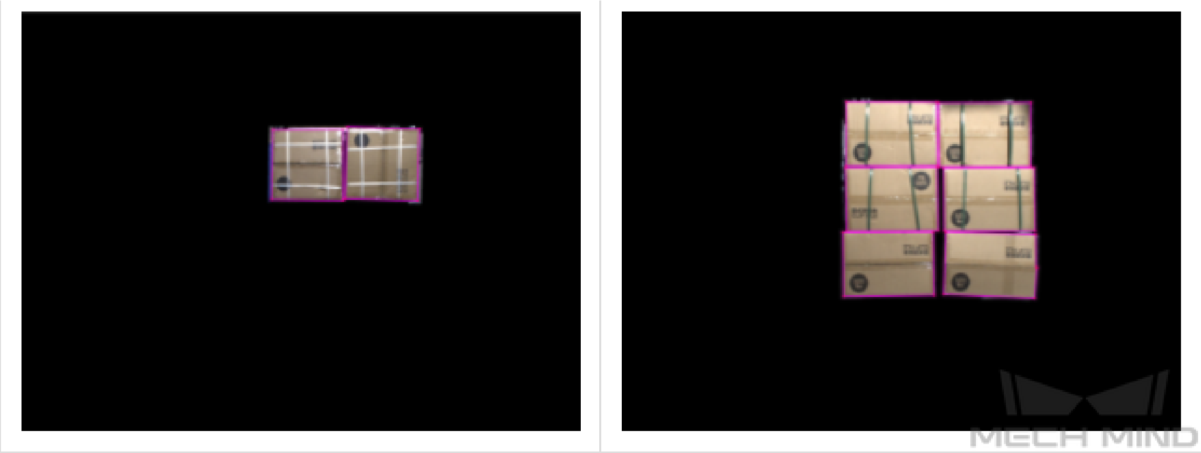


Figure 5. Labeling the contour of the upper surface

5. Train the model.

   Please see *Train the Model* for instructions on model training. The parameter **Total Epochs** should be set to 200, and other parameters should be kept as default.

6. Use the new model.

   Please see *Use the Model* for instructions on using a model in Mech-Vision.

7. Repeat steps 2 to 6 when necessary.

   Sometimes, not all box types are available during the early stage of a project. Please repeat steps 2 to 6 to update the model with the image data of the new box types.

3.2 Sack Palletizing/Depalletizing

Sack palletizing/depalletizing projects usually use *instance segmentation* to segment each sack in an image.

Mech-Mind Robotics provides a *Super Model* tailored for sack palletizing/depalletizing scenarios, which can segment sacks of most types without training.

The overall application process is as follows:

1. Use the Super Model in the project and check if all sack types involved in the project can be correctly segmented. If so, the project can be run using the Super Model, and there is no need to do the following steps; otherwise, please proceed to the next step.
2. Collect image data on those sacks that cannot be correctly segmented by the model for further training.
3. Label the collected data.

4. Train the model using the labeled data.
5. Use the newly trained model in Mech-Vision and check its performance.
6. Sometimes, not all sack types are available during the early stage of a project. Please repeat steps 2 to 6 to update the model with the image data of the new sack types.

CHAPTER

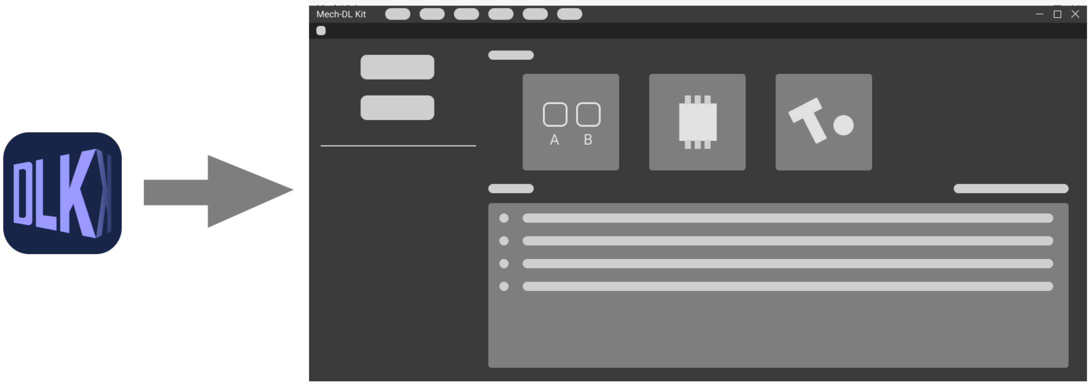
FOUR

MECH-DLK HANDBOOK

4.1 Overview of Mech-DLK

4.1.1 Brief Description

Mech-DLK is a deep learning platform software independently developed by Mech-Mind Robotics. With a variety of built-in industry-leading AI algorithms and through intuitive and simple UI interactions, it helps customers solve complex problems, including overlapping object recognition, highly difficult defect detection, and product grading and classification, etc. It can improve production efficiency, product yield, and reduce production line labor costs, and is suitable for industries including consumer electronics, new energy, automobiles, home appliances, logistics, etc.



4.1.2 Introduction of the Module Feature

Classification: Classify multiple types of objects. Given a limited amount of images of different types of objects, this module can classify different types of objects by deep learning.

Object Detection: Quickly detect objects in images. Given a limited amount of images, this module can detect and locate target objects.

Semantic Segmentation: Detect the defects in images. Given a certain amount of positive (OK) samples and negative (NG) samples, this module can detect the defects in images via deep learning.

Instance Segmentation: Segment and locate the target objects in images and classify them. Given a limited amount of images, this module can locate different objects and classify them via deep learning.

> **Attention:** All of the above modules can be used separately to train deep learning models that meet your needs. If you have customized requirements for a variety of functions, Mech-DLK can also meet your needs by cascading different modules for training.

4.1.3 Custom Development Configuration Requirements

Operating system	Windows 7 and above
Development platform	VS2013, VS2015, VS2017, VS2019 (Recommended)
Programming language	C, C++, C#

4.2 Environment Configuration

4.2.1 Hardware Requirements

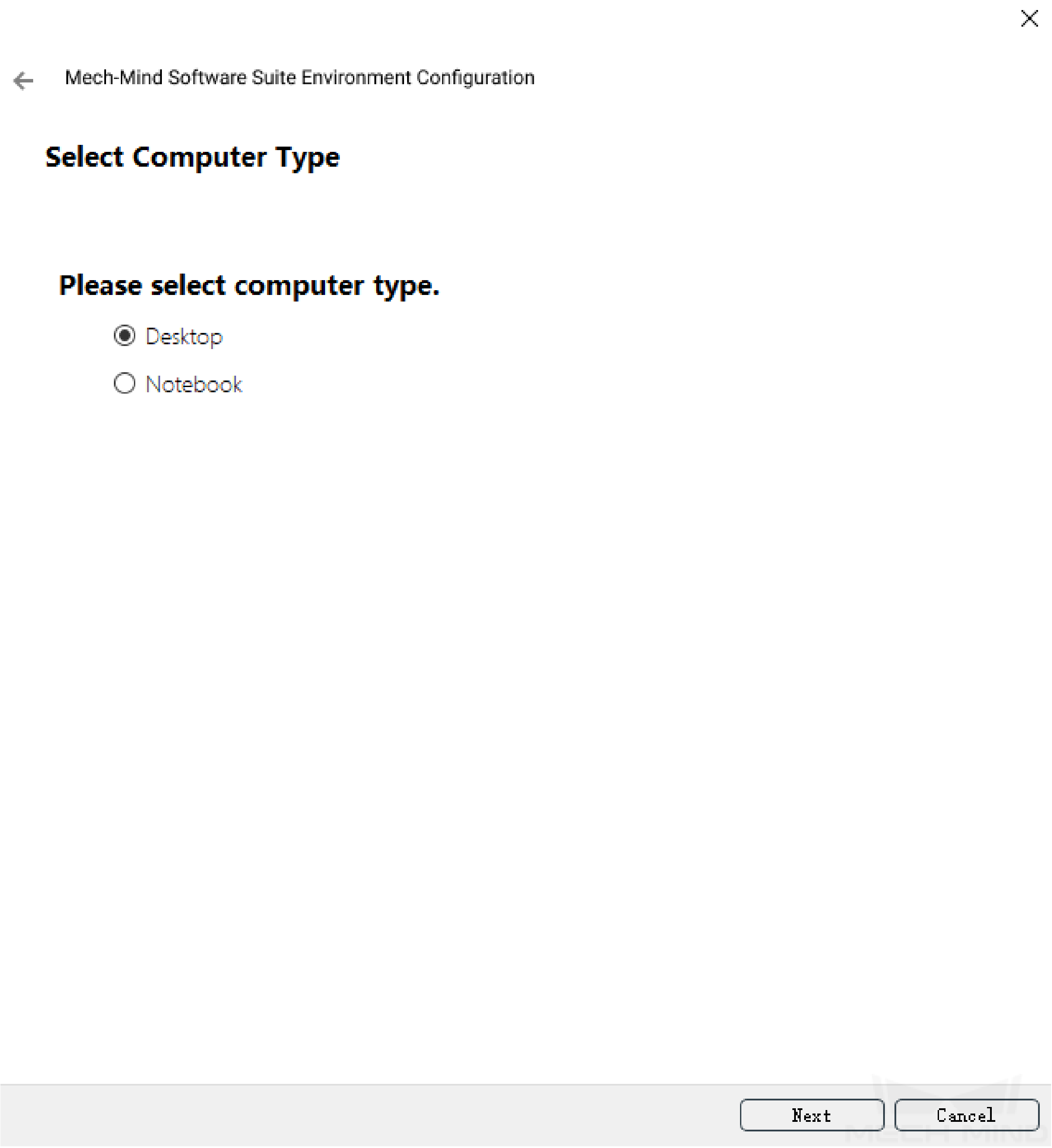
	Mech-DLK Pro-Run	Mech-DLK Pro-Train/Standard
Operating System	Windows 10	
CPU	Core i5 or above	Core i7 or above
RAM	8GB or above	16GB or above
Graphics Card	GeForce GTX 1650(4GB) and above	

Graphics Card Model	Desktop	Laptop
10 Series	NVIDIA GeForce GT 1030 NVIDIA GeForce GTX 1050 NVIDIA GeForce GTX 1050Ti NVIDIA GeForce GTX 1060 NVIDIA GeForce GTX 1070 NVIDIA GeForce GTX 1070Ti NVIDIA GeForce GTX 1080 NVIDIA GeForce GTX 1080Ti NVIDIA GeForce GTX 1650 NVIDIA GeForce GTX 1650 SUPER NVIDIA GeForce GTX 1660 NVIDIA GrForce GTX 1660Ti NVIDIA GeForce GTX 1660 SUPER	NVIDIA GeForce GTX 1050 NVIDIA GeForce GTX 1050Ti NVIDIA GeForce GTX 1060 NVIDIA GeForce GTX 1070 NVIDIA GeForce GTX 1080 NVIDIA GeForce GTX 1650 NVIDIA GeForce GTX 1650Ti NVIDIA GeForce GTX 1660Ti
20 Series	NVIDIA GeForce RTX 2060 NVIDIA GeForce RTX 2060 SUPER NVIDIA GeForce RTX 2070 NVIDIA GeForce RTX 2070 SUPER NVIDIA GeForce RTX 2080 NVIDIA GeForce RTX 2080Ti NVIDIA GeForce RTX 2080 SUPER	NVIDIA GeForce RTX 2060 NVIDIA GeForce RTX 2070 NVIDIA GeForce RTX 2080
30 Series	NVIDIA GeForce RTX 3050 NVIDIA GeForce RTX 3060 NVIDIA GeForce RTX 3060Ti NVIDIA GeForce RTX 3070 NVIDIA GeForce RTX 3070Ti NVIDIA GeForce RTX 3080 NVIDIA GeForce RTX 3080Ti NVIDIA GeForce RTX 3090	NVIDIA GeForce RTX 3050 Laptop GPU NVIDIA GeForce RTX 3060 Laptop GPU NVIDIA GeForce RTX 3070 Laptop GPU NVIDIA GeForce RTX 3080 Laptop GPU

4.2.2 Environment Configuration

You can run the environment configuration program by double-clicking on *Mech_Mind_software_environment_installer.exe*. If an environment of an earlier version is already installed, the program will check if there are any missing components in the current environment and automatically update the environment to the latest version.

1. Select the computer type.



Mech-Mind Software Suite Environment Configuration

Select Computer Type

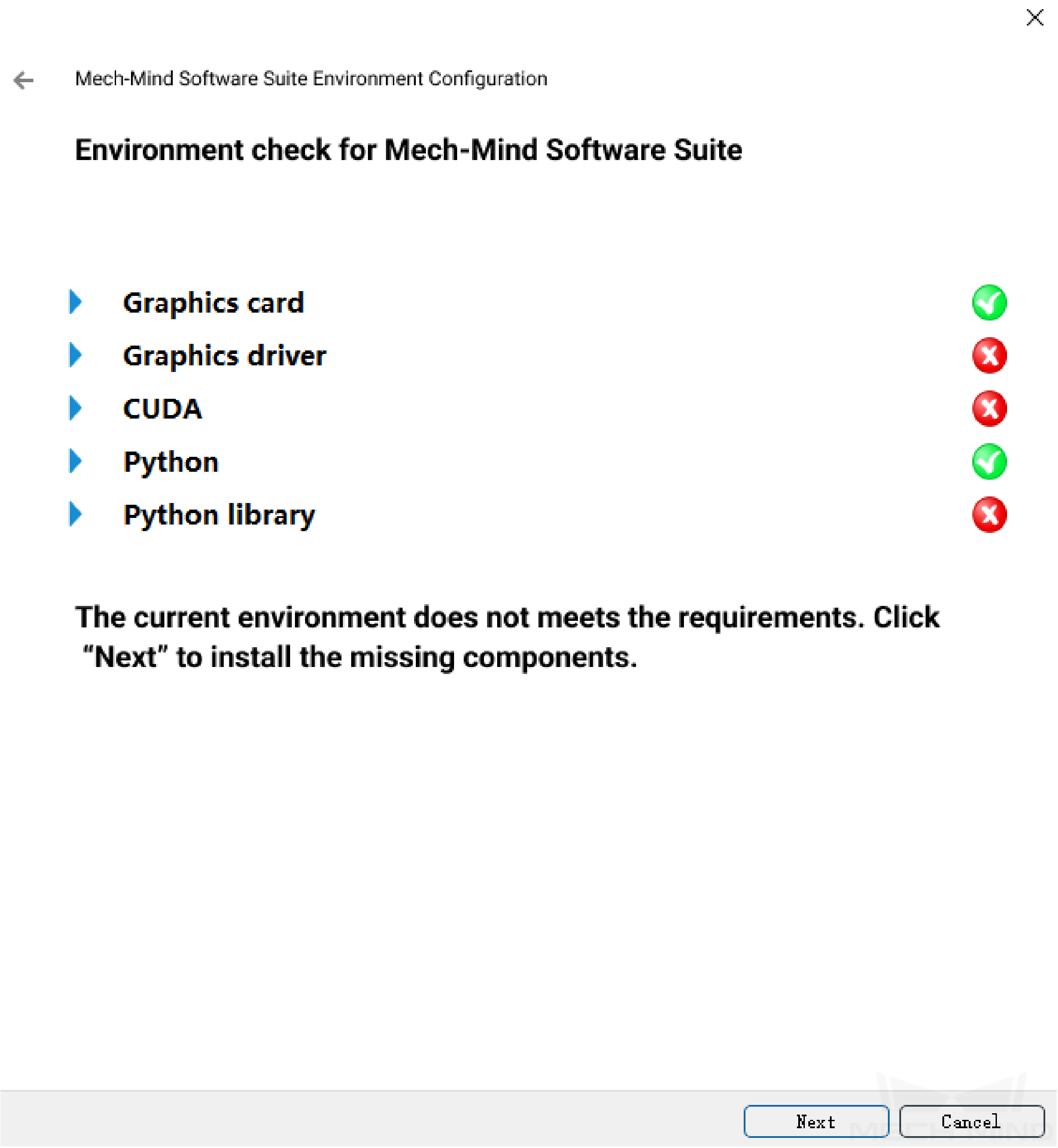
Please select computer type.

● Desktop

○ Notebook

2. The executable program will check the current environment of your computer. Please click on *Next*

to continue the installation. If a window as shown in *Figure 3* appears, the environment is installed successfully.



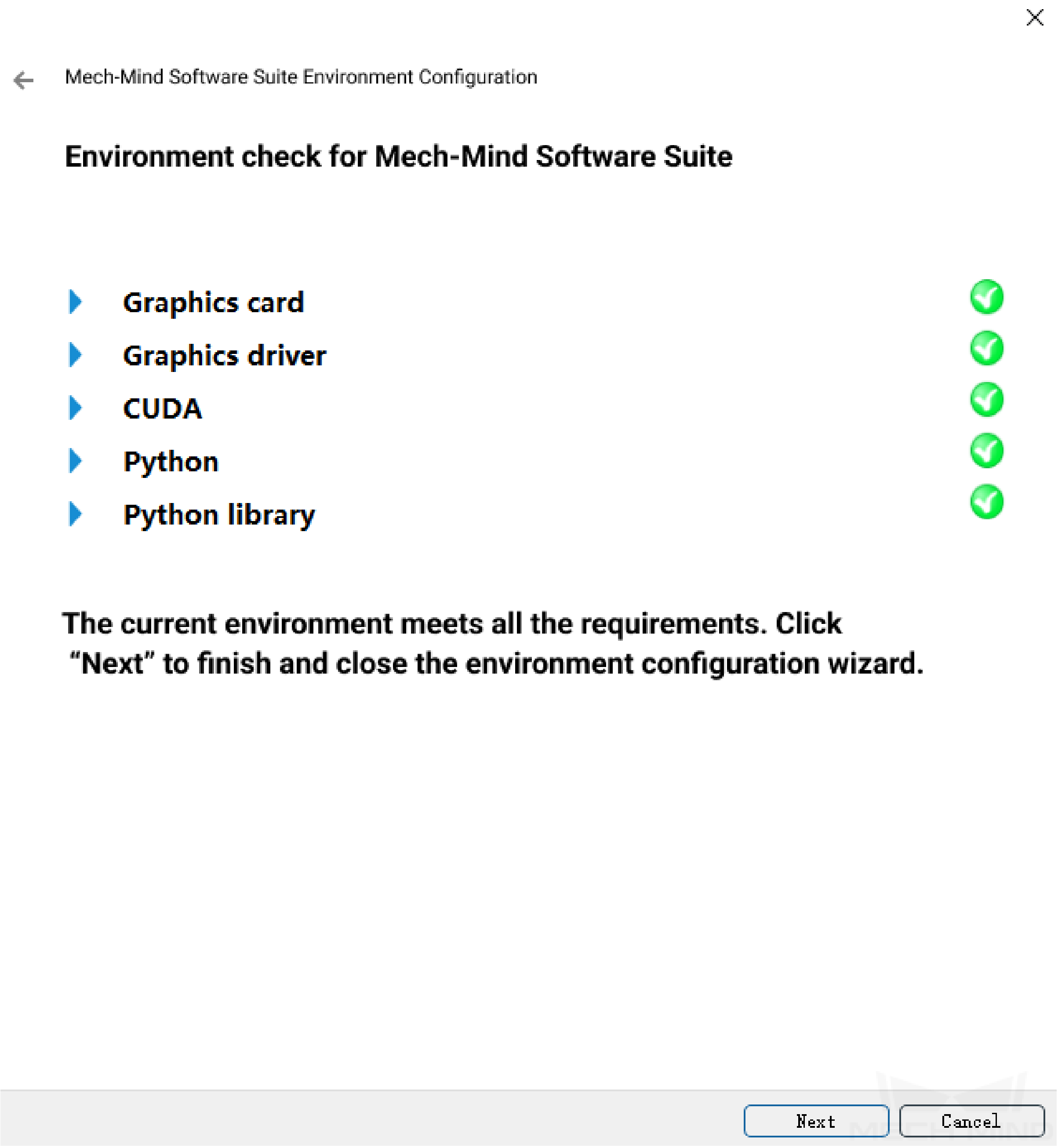
Mech-Mind Software Suite Environment Configuration

Environment check for Mech-Mind Software Suite

- Graphics card ✓
- Graphics driver ✗
- CUDA ✗
- Python ✓
- Python library ✗

The current environment does not meets the requirements. Click "Next" to install the missing components.

Figure 2. Check the DL environment



Mech-Mind Software Suite Environment Configuration

Environment check for Mech-Mind Software Suite

- Graphics card ✓
- Graphics driver ✓
- CUDA ✓
- Python ✓
- Python library ✓

The current environment meets all the requirements. Click "Next" to finish and close the environment configuration wizard.

Figure 3. Installation succeeded

4.3 Mech-DLK Quick Start

- This section shows how to train and export an example model that can be used for defect detection and defect classification.
- You can use a Semantic Segmentation module cascaded with a Classification module in Mech-DLK to make a final trained model.
- Before using Mech-DLK, please ensure that *Deep Learning Environment* and Mech-DLK are both installed successfully.

1. Create a new project

   Open Mech-DLK, click on *New Project* to create a new project, as shown in *Figure 1*.

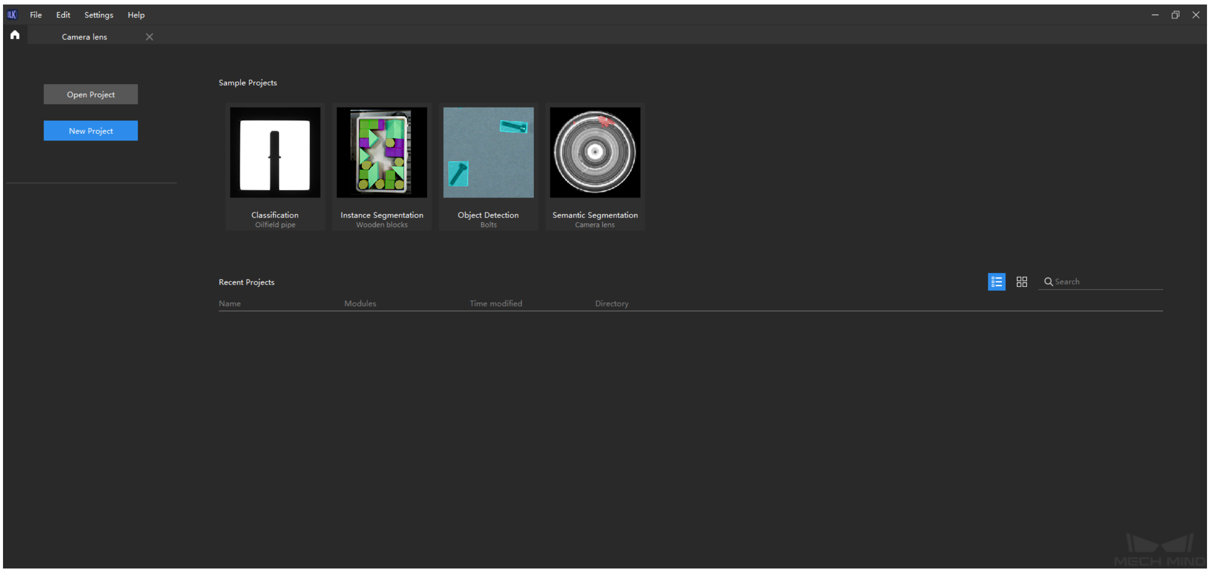
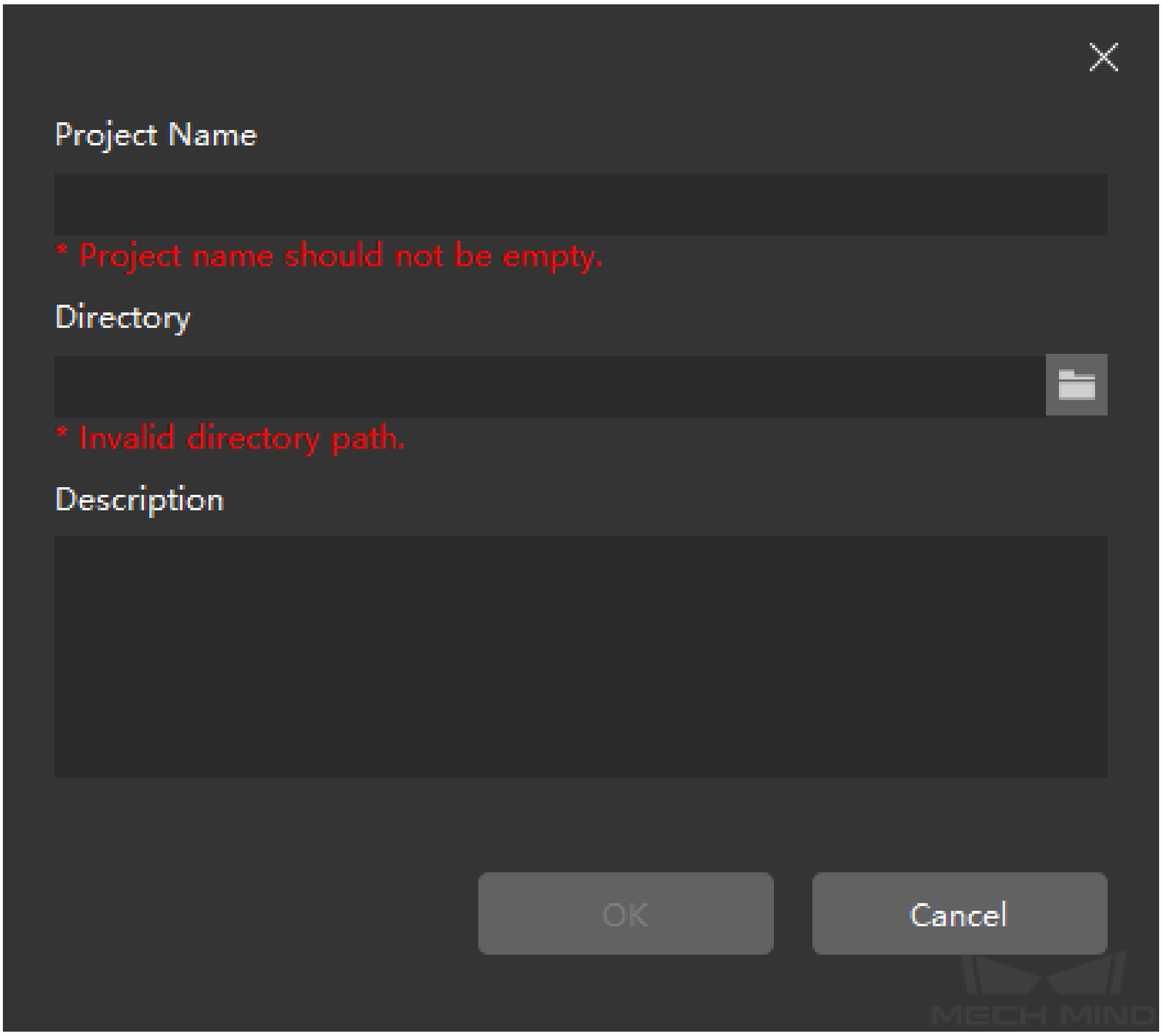


Figure 1. New project

2. Name and save the project

   Name the project, select a folder to save the project, and add descriptions if necessary. Then click on *OK* to finish setting, as shown below.



Project Name

* Project name should not be empty.

Directory

* Invalid directory path.

Description

Figure 2. Project settings

3. Select a module

   Add a module: Click on + on the right side of the panel *Modules*, and then select *Semantic Segmentation*. Click on *OK* to finish setting.

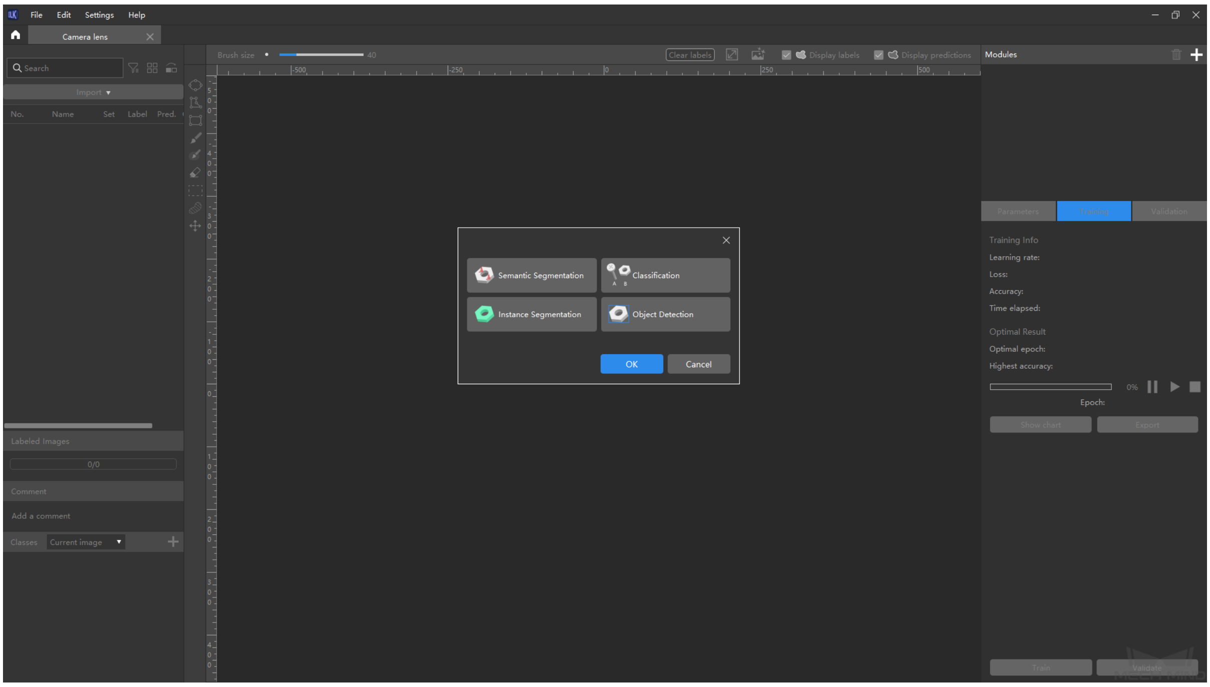
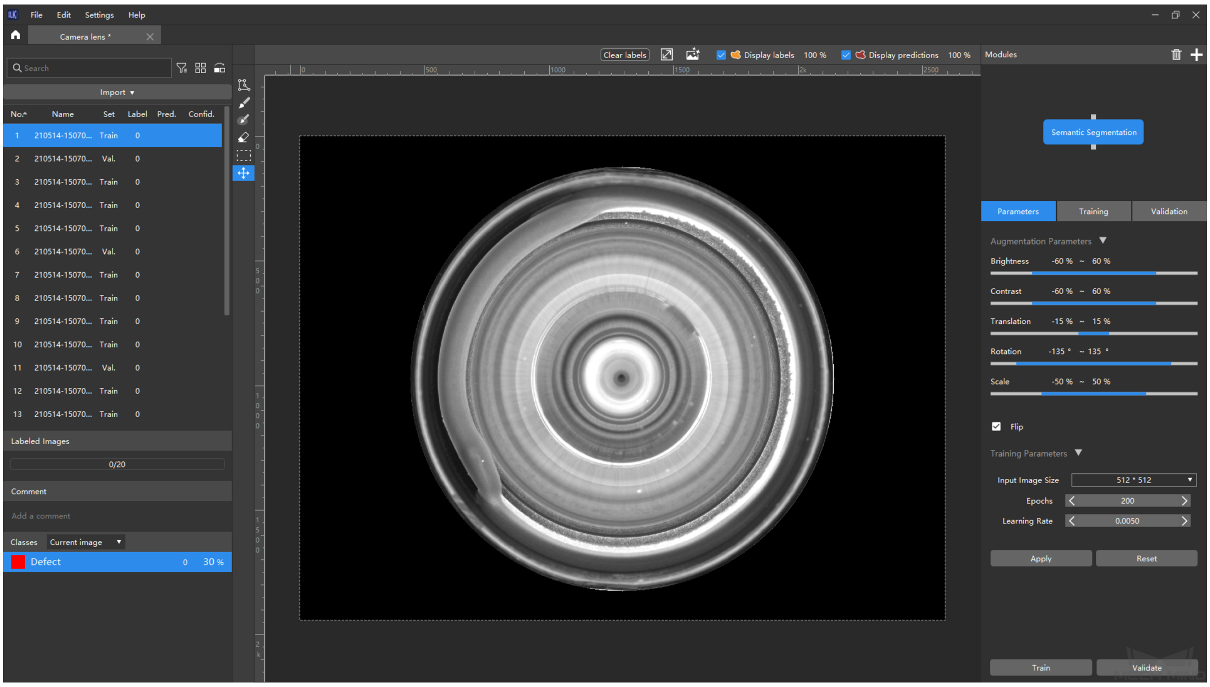


Figure 3. Add module

4. Import data

Click Import ▼ in the upper left corner to import local images.



No.	Name	Set	Label	Pred.	Confid.
1	210514-15070...	Train	0		
2	210514-15070...	Val.	0		
3	210514-15070...	Train	0		
4	210514-15070...	Train	0		
5	210514-15070...	Train	0		
6	210514-15070...	Val.	0		
7	210514-15070...	Train	0		
8	210514-15070...	Train	0		
9	210514-15070...	Train	0		
10	210514-15070...	Train	0		
11	210514-15070...	Val.	0		
12	210514-15070...	Train	0		
13	210514-15070...	Train	0		

Labeled Images 0/20

Comment

Add a comment

Classes: Current image

Defect 0 30 %

Augmentation Parameters ▼

Brightness -60 % ~ 60 %

Contrast -60 % ~ 60 %

Translation -15 % ~ 15 %

Rotation -135 ° ~ 135 °

Scale -50 % ~ 50 %

☑ Flip

Training Parameters ▼

Input Image Size 512 * 512

Epochs 200

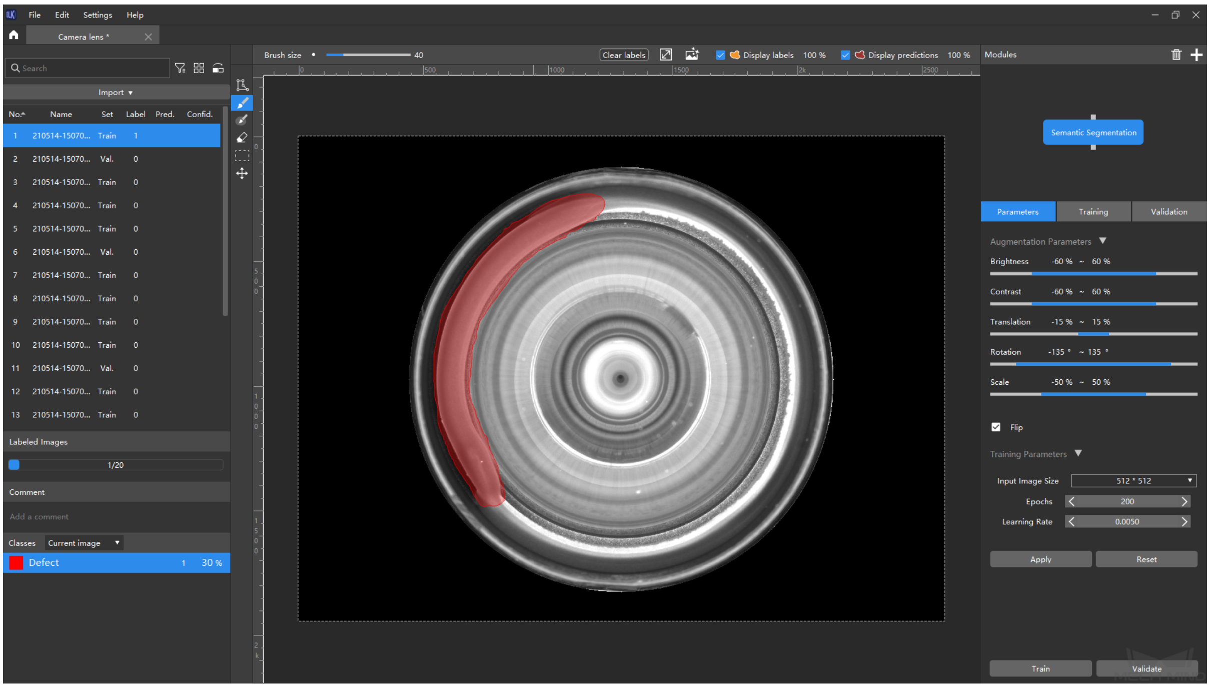
Learning Rate 0.0050

Figure 4. Import data

> **Attention:** Images imported into the Semantic Segmentation module must include defect-free images, which should also be included in both the validation set and the training set, or else an alert will be displayed and the training cannot continue.

1. Labeling

   The Semantic Segmentation algorithm will have an automatically generated Defect label. You can use the tools , , and on the left to label the images.



No.	Name	Set	Label	Pred.	Confid.
1	210514-15070...	Train	1		
2	210514-15070...	Val.	0		
3	210514-15070...	Train	0		
4	210514-15070...	Train	0		
5	210514-15070...	Train	0		
6	210514-15070...	Val.	0		
7	210514-15070...	Train	0		
8	210514-15070...	Train	0		
9	210514-15070...	Train	0		
10	210514-15070...	Train	0		
11	210514-15070...	Val.	0		
12	210514-15070...	Train	0		
13	210514-15070...	Train	0		

Labeled Images 1/20

Comment: Add a comment

Classes: Current image

🟥 Defect 1 30 %

Augmentation Parameters

Brightness -60 % ~ 60 %

Contrast -60 % ~ 60 %

Translation -15 % ~ 15 %

Rotation -135 ° ~ 135 °

Scale -50 % ~ 50 %

☑ Flip

Training Parameters

Input Image Size 512 * 512

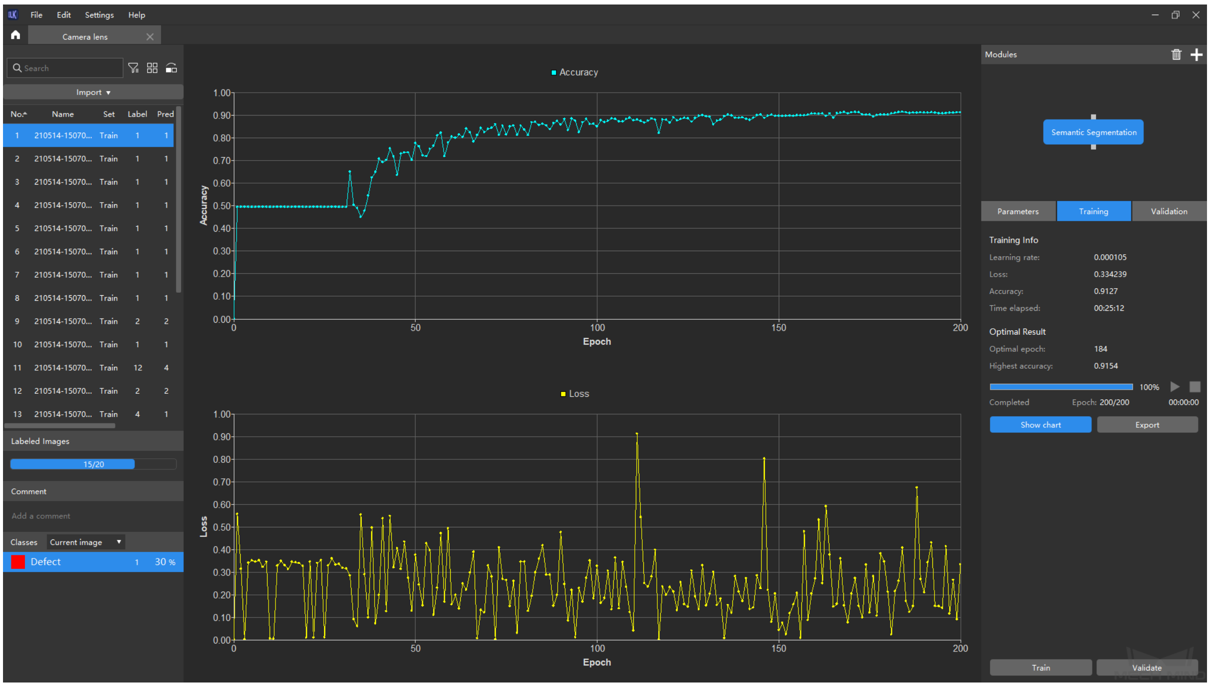
Epochs 200

Learning Rate 0.0050

Figure 5. Labeling in Semantic Segmentation

2. Train the model

   Click on *Train* in the lower right corner to start training. Click on *Show chart* to check the accuracy and loss during the training.



No.	Name	Set	Label	Pred
1	210514-15070...	Train	1	1
2	210514-15070...	Train	1	1
3	210514-15070...	Train	1	1
4	210514-15070...	Train	1	1
5	210514-15070...	Train	1	1
6	210514-15070...	Train	1	1
7	210514-15070...	Train	1	1
8	210514-15070...	Train	1	1
9	210514-15070...	Train	2	2
10	210514-15070...	Train	1	1
11	210514-15070...	Train	12	4
12	210514-15070...	Train	2	2
13	210514-15070...	Train	4	1

Labeled Images 15/20

Comment

Add a comment

Classes: Current image

🟥 Defect	1	30 %

Training Info

Learning rate: 0.000105

Loss: 0.334239

Accuracy: 0.9127

Time elapsed: 00:25:12

Optimal Result

Optimal epoch: 184

Highest accuracy: 0.9154

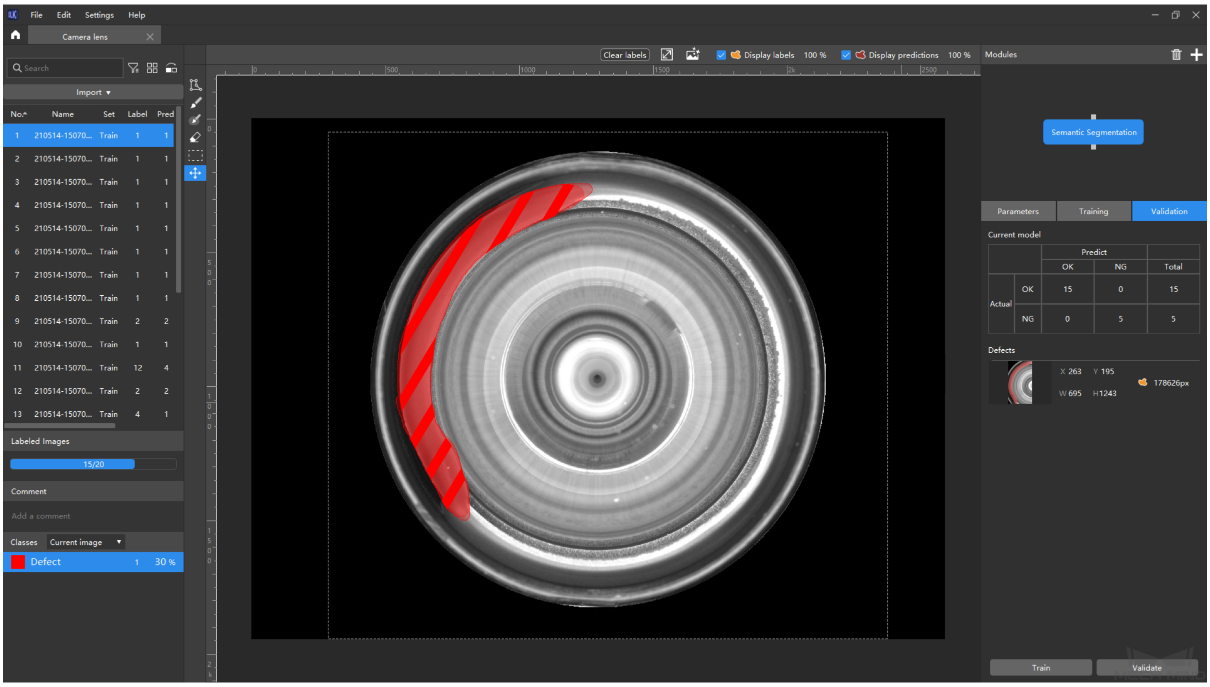
100%

Completed Epoch: 200/200 00:00:00

Figure 6. Learning curves in the chart

3. Validate the model

   After training the Semantic Segmentation model, click on *Validate* to validate the results. A training result will be saved after the validation.



No.	Name	Set	Label	Pred
1	210514-15070...	Train	1	1
2	210514-15070...	Train	1	1
3	210514-15070...	Train	1	1
4	210514-15070...	Train	1	1
5	210514-15070...	Train	1	1
6	210514-15070...	Train	1	1
7	210514-15070...	Train	1	1
8	210514-15070...	Train	1	1
9	210514-15070...	Train	2	2
10	210514-15070...	Train	1	1
11	210514-15070...	Train	12	4
12	210514-15070...	Train	2	2
13	210514-15070...	Train	4	1

Labeled Images 15/20

Comment

Add a comment

Classes: Current image

🟥 Defect	1	30 %

Current model

		Predict		
		OK	NG	Total
Actual	OK	15	0	15
	NG	0	5	5

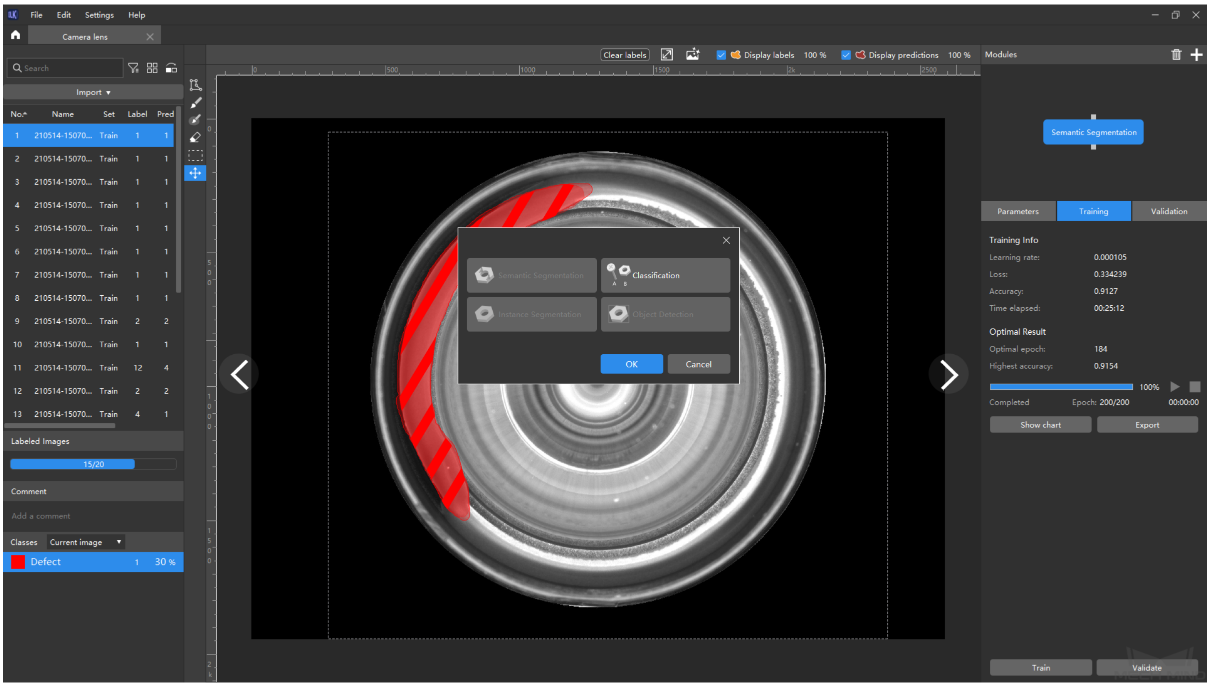
Defects

X 263 Y 195 W 695 H1243 178626px

Figure 7. Validation

4. Add another module

   After confirming that the model training is completed and satisfactory, click on + in the upper right corner to add the Classification module.



No.	Name	Set	Label	Pred
1	210514-15070...	Train	1	1
2	210514-15070...	Train	1	1
3	210514-15070...	Train	1	1
4	210514-15070...	Train	1	1
5	210514-15070...	Train	1	1
6	210514-15070...	Train	1	1
7	210514-15070...	Train	1	1
8	210514-15070...	Train	1	1
9	210514-15070...	Train	2	2
10	210514-15070...	Train	1	1
11	210514-15070...	Train	12	4
12	210514-15070...	Train	2	2
13	210514-15070...	Train	4	1

Labeled Images: 15/20

Classes: Current image ▾

🟥 Defect 1 30 %

Training Info

Learning rate: 0.000105

Loss: 0.334239

Accuracy: 0.9127

Time elapsed: 00:25:12

Optimal Result

Optimal epoch: 184

Highest accuracy: 0.9154

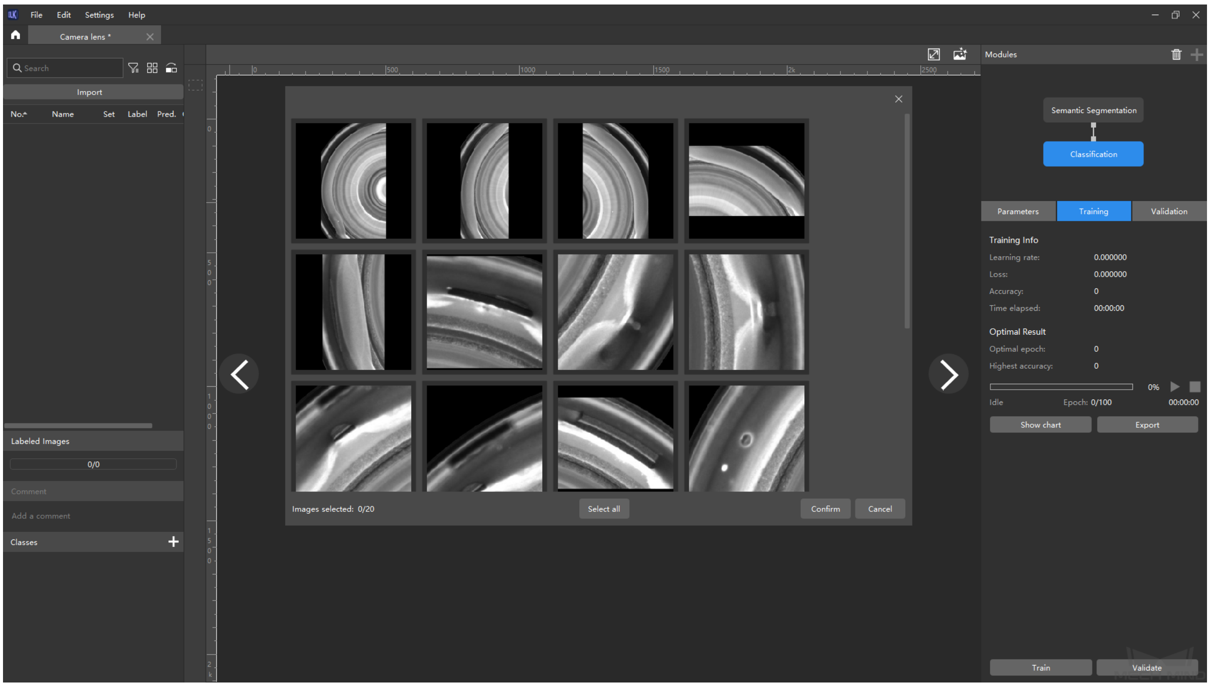
100%

Completed Epoch: 200/200 00:00:00

Figure 8. Add another module

5. Import the data to the Classification module

   The training result of the previous model will be imported as the source data to the Classification module. Click on Import ▾ to display all selectable images, and then manually select the images needed for training/validation of classification.

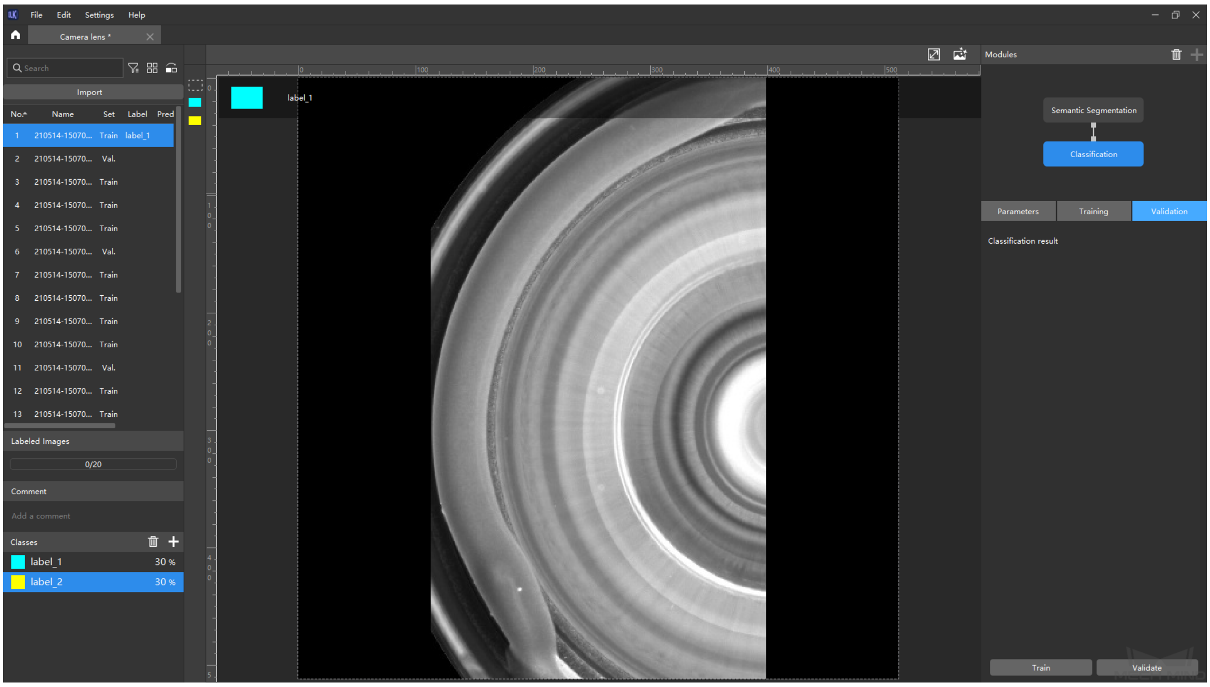


Training Info
Learning rate: 0.000000
Loss: 0.000000
Accuracy: 0
Time elapsed: 00:00:00
Optimal Result
Optimal epoch: 0
Highest accuracy: 0
0%
Idle Epoch: 0/100 00:00:00

Figure 9. Import source data

6. Label images of different classes

   Before labeling images of different classes, you need to click on + in the lower left part of the panel Classes to create different labels for different classes. After creating the labels, you can label the images by clicking on 🟦 or 🟨.



No.	Name	Set	Label	Pred
1	210514-15070...	Train	label_1	
2	210514-15070...	Val.		
3	210514-15070...	Train		
4	210514-15070...	Train		
5	210514-15070...	Train		
6	210514-15070...	Val.		
7	210514-15070...	Train		
8	210514-15070...	Train		
9	210514-15070...	Train		
10	210514-15070...	Train		
11	210514-15070...	Val.		
12	210514-15070...	Train		
13	210514-15070...	Train		

Classes	
🟦 label_1	30 %
🟨 label_2	30 %

Figure 10. Classification and labeling

7. Train the model

   Click on *Train* in the lower right corner to start training. Click on *Show chart* to check the accuracy and loss of the training.

8. Export the final model

   After the training of the model is completed, click on *Export* to export the trained model. You can select a folder to save the final model, and the model file is as shown below.

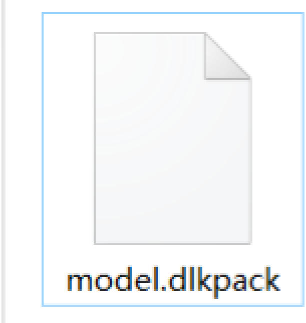


Figure 11. Model file

4.4 Terminology

Annotate:

Manually select target objects in images and add labels to them.

Label:

The tag added to an image after annotation to identify its class.

Dataset:

The .dlkdb file containing annotated data exported by Mech-DLK.

Labeled:

The image data status of having been annotated manually.

Unlabeled:

The image data status of having not been annotated manually.

Training Set:

An image data set that has been annotated manually and is used to train the model.

Validation Set:

An image data set that has been annotated manually and is used to validate the training effect of the model.

OK Image:

A defect-free image.

NG Image:

An image with object defect.

Train:

The process of using a training set to train a deep learning model.

Validate:

The process of using a trained model to predict on the validation set and comparing the results with the validation set labels.

Accuracy:

The ratio of the number of correctly predicted samples to the total number of samples when the model predicts on a validation set.

Loss:

The degree of inconsistency between the validation set result labels from model prediction and the actual labels.

Epoch:

The number of passes of the entire training set the machine learning algorithm has completed for training.

4.5 Shortcuts

No.	Feature	Shortcut Key	Comment
1	Create a new project	Ctrl + n	
2	Save the project	Ctrl + s	
3	Open the Project	Ctrl + o	
4	Undo the labeling	Ctrl + y	
5	Redo the labeling	Ctrl + z	
6	Copy the label	Ctrl + c	
7	Paste the label	Ctrl + v	
8	Select all labels	Ctrl + a	
9	Delete the label	Delete	Select the label in the labeling area first
10	Label Tool -> Ellipse	l	
11	Label Tool -> Polygon	g	
12	Label Tool -> Rectangle	r	
13	Label Tool -> Brush	b	
14	Label Tool -> Auto fill by contour	a	
15	Label Tool -> Eraser	e	
16	Label Tool -> Mask	m	
17	Label Tool -> Select	s	
18	Clear all labels in labeling area	Ctrl + l	
19	Delete image in the data set	Delete	Select the image in the data set section first
20	Switch between items in a list / drag to scroll	↑ ↓ → ←	Select an image in the data set section first

CHAPTER

FIVE

FAQ

1. **Is it feasible 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. Is it feasible 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. Please see *Box Palletizing/Depalletizing* for details.

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

    No.