

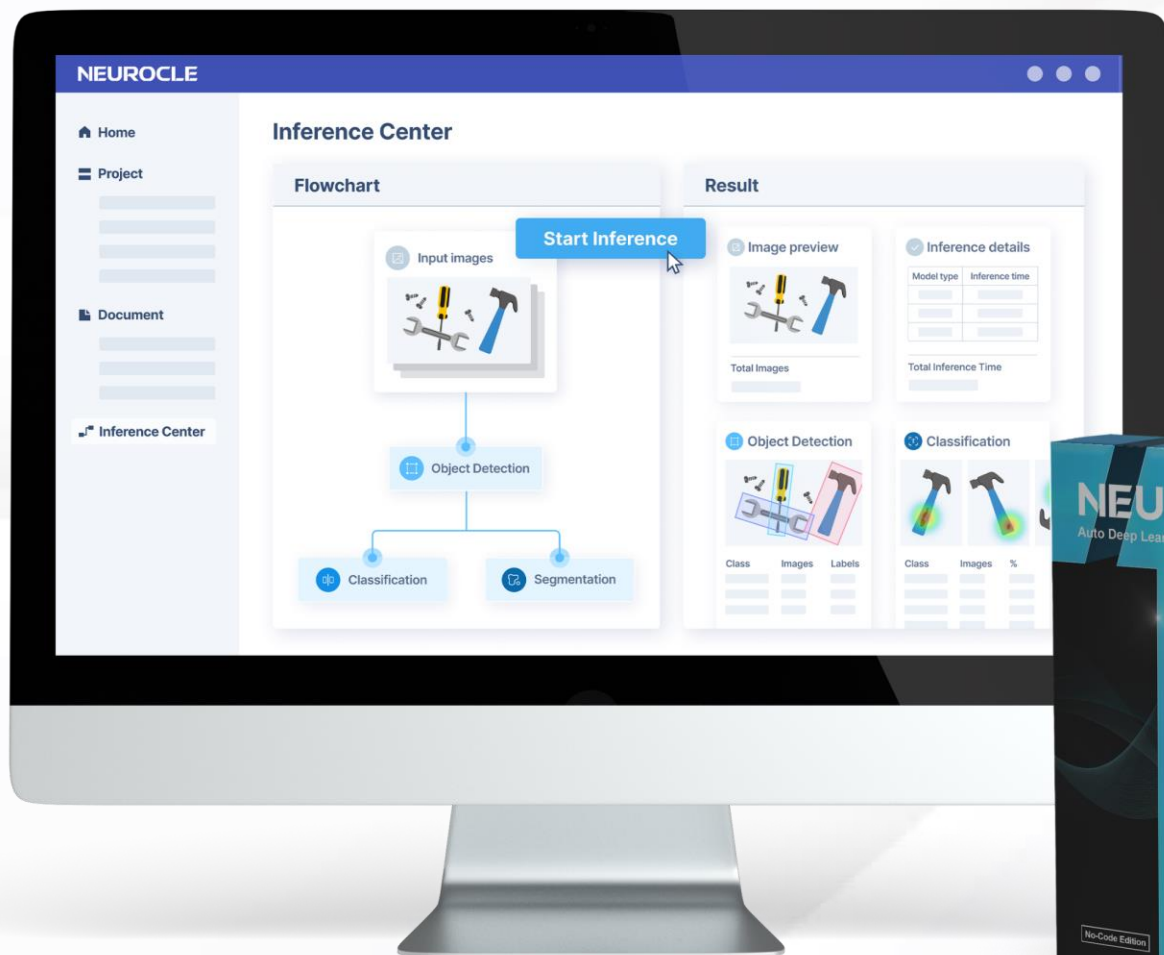
소프트웨어 사용 가이드

## Neuro-R 기본 개념 및 활용 방법 설명

**NEUROCLE**

Deep Learning Vision Software

# 딥러닝 모델 학습용 소프트웨어



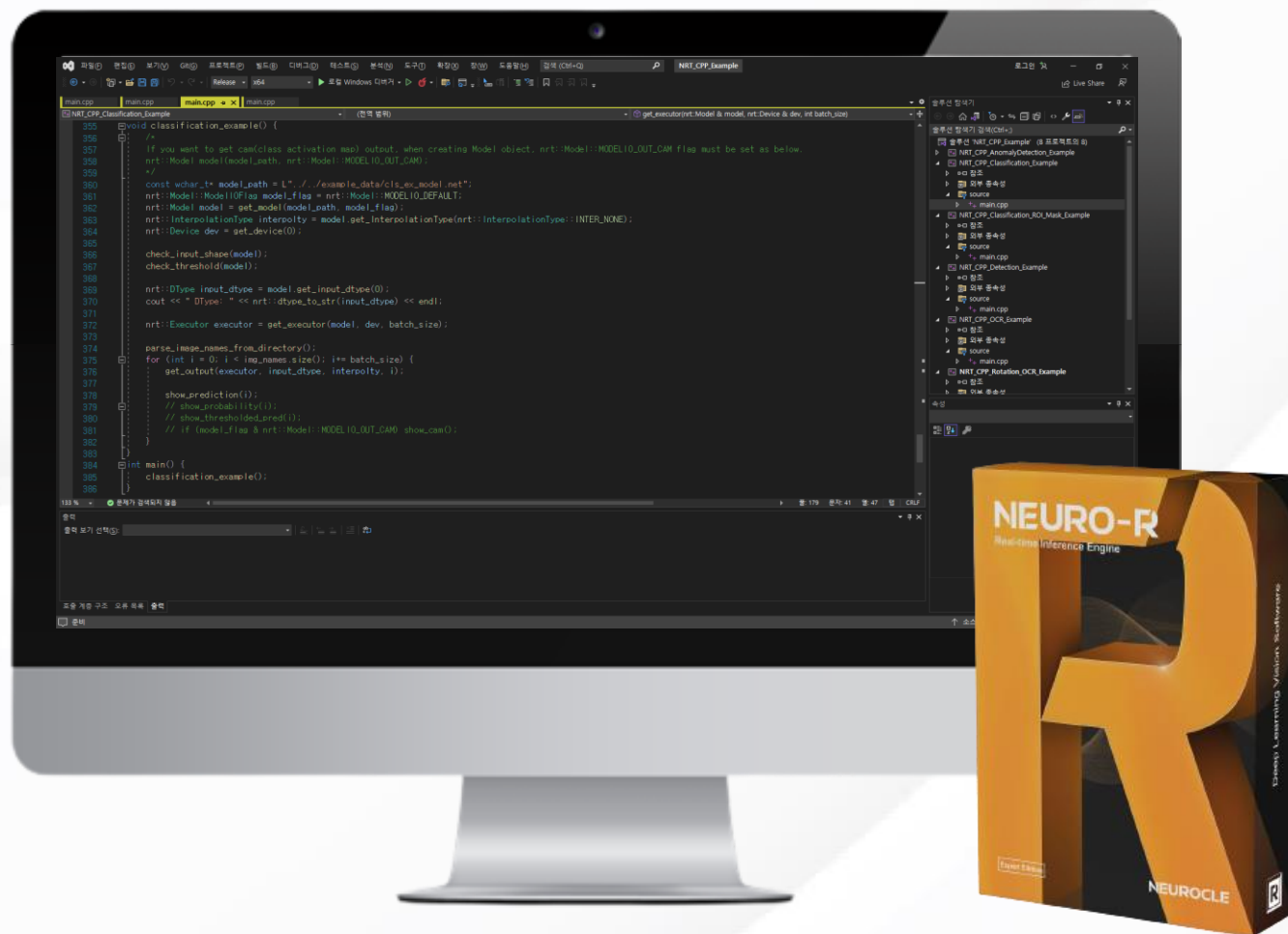
## NEURO-T

GUI 기반의 No-Code 소프트웨어

- 누구나 고성능의 딥러닝 모델을 손쉽게 생성할 수 있는  
오토 딥러닝 알고리즘 제공
- 이미지 편집부터 모델 학습까지 모든 것이 가능한  
올인원 플랫폼



# 런타임 API



## NEURO-R

Neuro-T에서 생성한 모델을 활용하기 위해  
필요한 런타임 API

- 폭넓은 프로그래밍 언어 지원 (C++, C#, Python)
- 임베디드 장비부터 PC까지 다양한 환경을 지원
- 목표 검사 속도 달성이 가능한 **빠른 검사 속도**

# 비전 검사 워크플로우

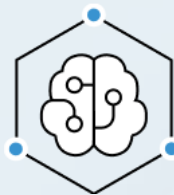
## NEURO-T



데이터 불러오기



데이터 편집 및 관리



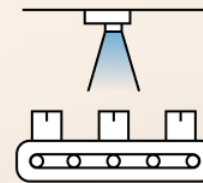
모델 학습



성능 평가



## NEURO-R



검사

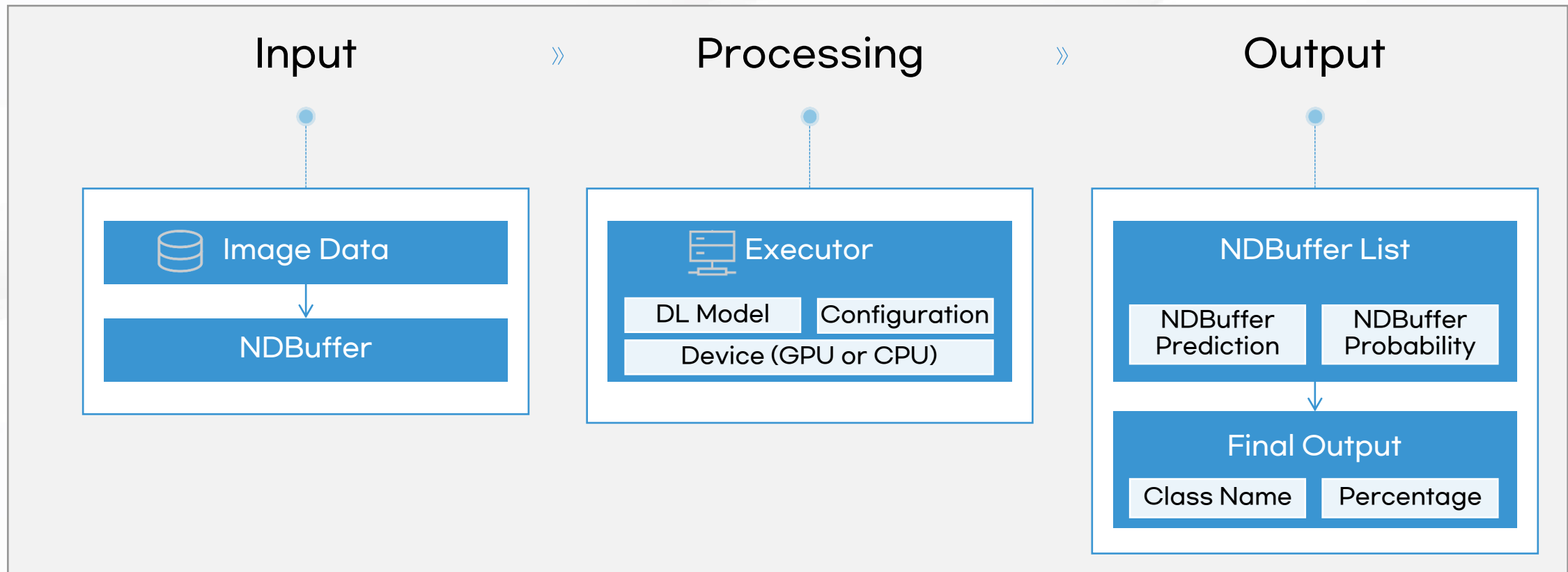
Neuro-T에서 생성된 모델을 효과적으로 활용하기 위해서는 Neuro-R이 반드시 필요합니다.

# 목차

1. 개념 설명
2. 적용 방법
3. 샘플 코드 분석

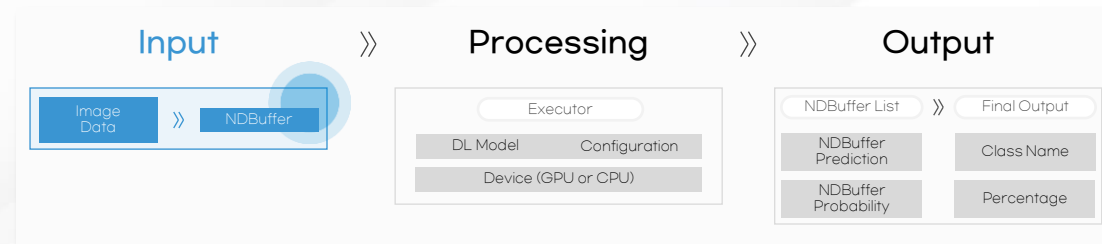
# 01. 개념 설명

# Neuro-R 워크플로우



# Input

이미지를 학습에 사용하기 위해 **NDBuffer** 형태로 변환 및 입력하는 과정입니다.



## NDBuffer란

**NDBuffer**는 OpenCV의 Mat, 혹은 NumPy의 ndarray와 같이 데이터의 모양 및 유형을 포함하고 있는 **이미지 매트릭스 제어 클래스**입니다.



# NDBuffer

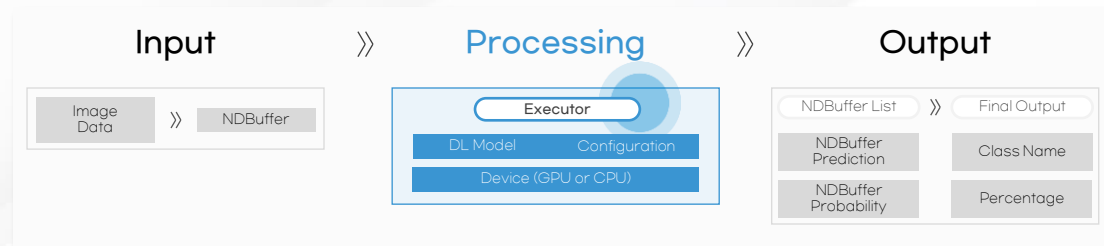
## CV to NDBuffer

```
for (int j = 0; j < local_batch_size; j++) {  
    cv::Mat cur_img = cv::imread(image_base_dir + img_names[start_idx + j], cv::IMREAD_COLOR);  
  
    if (cur_img.channels() != input_c) {  
        cerr << "The input image must be a three channel BGR image." << endl;  
        return;  
    }  
  
    nrt::NDBuffer image_buff(nrt::Shape(local_batch_size, input_h, input_w, input_c), input_dtype);  
    unsigned char* image_buff_ptr = image_buff.get_at_ptr<unsigned char>(j);  
    std::copy(cur_img.data, cur_img.data + input_image_byte_size, image_buff_ptr);  
}
```



# Processing

**Executor**를 통해 이미지를 분석하고, 최종 결과 출력에 필요한 값들을 확인하는 단계입니다.

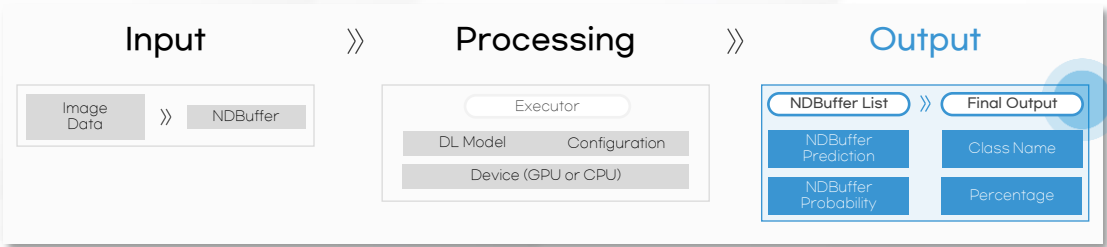


## Executor 구성 요소



# Outputs

이전 단계에서 확인한 분석값을 기반으로 적절한 형태의 최종 분석 결과를 출력합니다.






모델 종류	Classification	Segmentation	Object Detection & OCR	Rotation	Anomaly
NDBuffer List	<div><div>Prediction</div><div>Probability</div><div>CAM</div></div>	<div><div>Prediction</div><div>Probability</div></div>	<div><div>Prediction</div><div>Probability</div></div>	<div><div>Prediction</div></div>	<div><div>Prediction</div><div>Box Info</div><div>Anomaly Score</div></div>
Final Outputs	<div><div>Class Name</div><div>Percentage</div><div>CAM</div></div>	<div><div>Class Name</div><div>Percentage</div></div>	<div><div>Box Info</div><div>Percentage</div></div>	<div><div>Degree</div></div>	<div><div>Class Name</div><div>Box Info</div><div>Anomaly Score</div></div>

# Outputs


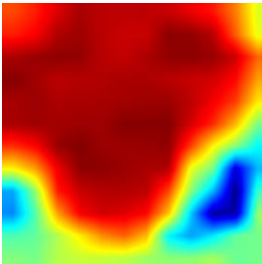
## - Classification



Prediction & Probability

Image			
Prediction (Class)	0 (Good)	0 (Good)	1 (Bad)
Probability	0.9887	0.8830	0.0001
	0.0012	0.1170	0.9999

CAM

Input Image	
CAM Image	

# Outputs

## - Classification

**NDBuffer  
List**

Prediction

Probability

CAM

API

```
nrt::NDBuffer output_cam = outputs.get_at(cam_idx);

nrt::Shape output_cam_shape = output_cam.get_shape();

status = nrt::convert_to_colormap(output_cam, cam_colormap);

nrt::Shape color_map_shape = cam_colormap.get_shape();
for (int j = 0; j < color_map_shape.dims[0]; j++) {

    cv::Mat cam_colormap_mat(
        color_map_shape.dims[1],
        color_map_shape.dims[2],
        CV_8UC3,
        (void*)(cam_colormap.get_data_ptr<char>() + (color_map_shape.dims[1] * color_map_shape.dims[2] * color_map_shape.dims[3]) * j));

    cv::imshow("cam", cam_colormap_mat);
    cv::waitKey(0);
}
```

# Outputs

## - Segmentation

**NDBuffer  
List**

Prediction

Probability

Prediction

픽셀 단위로 Class 출력  
[512, 512] = [input\_height, input\_width]

0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0
2	2	0	0	0	0	0	0
2	2	0	0	0	0	0	0

□ Background

■ Good

■ Bad

Probability

Output prob shape =  
[input\_height, input\_width, num\_classes]

0.03	0.73	0.96	0.99 6	0.99 9	0.5
0.1	0.3	0.4	0.6	1	0.3
0.1	0.98	0.99	0.99	0	0
1	0	0	0	1	1
0.7	0.7	0.65	0.8	0.9	0.9
0	0	0	1	1	0.88

Height

Width

Number of classes

# Outputs

## - Segmentation

**NDBuffer  
List**

Prediction

Probability

API

```
nrt::NDBuffer merged_output = outputs.get_at(pred_idx);  
unsigned char* output_ptr = merged_output.get_at_ptr<unsigned char>(j);  
cv::Mat output_mat(merged_output_shape.dims[1], merged_output_shape.dims[2], CV_8UC1, output_ptr);  
  
cv::Mat img_to_display = output_mat == 1;  
  
cv::imshow("w", img_to_display);  
cv::waitKey(0);
```

# Outputs

## - Object Detection & OCR

### NDBuffer List

Prediction

Probability

#### Prediction

Output pred shape = [objects의 개수, 6]  
 \*각 배열의 값들은 예측한 Class를 의미함

#### BCYXHW

B : Batch\_idx

C : Class

Y, X : Center Y, X

H, W : Height, Width

B	C	Y	X	H	W
..	...	..	...	...	...

#### Probability

Output prob shape  
 = [objects의 개수, class의 개수]  
 > 각 Object의 Class별 probability

Object 1	0.011	0.998	0.009
Object 2	0.017	0.003	0.998
Object 3	0.001	0.998	0.019

# Outputs

## - Object Detection & OCR

**NDBuffer  
List**

Prediction

Probability

API

```
nrt::NDBuffer output_boxes = outputs.get_at(boxes_idx);  
for (int box_idx = 0; box_idx < number_of_boxes; ++box_idx)  
{  
    const int* bcyxhw_ptr = output_boxes.get_at_ptr<int>(box_idx);  
  
    BoundingBox bbox = convert_to_bounding_box(bcyxhw_ptr, h_ratio, w_ratio);  
}
```



# Outputs

## - Object Detection & OCR

**NDBuffer  
List**

Prediction

Probability

API

```
BoundingBox convert_to_bounding_box(const int* bcyxhw_ptr, const double h_ratio, const double w_ratio)
{
    BoundingBox bbox;
    bbox.batch_index = bcyxhw_ptr[0];
    bbox.class_number = bcyxhw_ptr[1];
    bbox.box_center_Y = bcyxhw_ptr[2];
    bbox.box_center_X = bcyxhw_ptr[3];
    bbox.box_height = bcyxhw_ptr[4];
    bbox.box_width = bcyxhw_ptr[5];

    bbox.box_center_Y = (double)bbox.box_center_Y / h_ratio;
    bbox.box_center_X = (double)bbox.box_center_X / w_ratio;
    bbox.box_height = (double)bbox.box_height / h_ratio;
    bbox.box_width = (double)bbox.box_width / w_ratio;

    return bbox;
}
```

# Outputs

## - Rotation

**NDBuffer  
List**

Prediction

API

```
nrt::NDBuffer output_rot = outputs.get_at(rot_idx);  
int degree = *output_rot.get_at_ptr<int>(j);  
cout << "Image Name : " << name << " : " << "Predicted rotation degree : " << degree << endl;  
cv::Mat img = cv::imread(image_base_dir + name, cv::IMREAD_COLOR);  
  
auto M = cv::getRotationMatrix2D({ static_cast<float>(input_w) / 2, static_cast<float>(input_h) / 2 }, degree, 1.0);  
cv::warpAffine(img, img, M, { input_w, input_h }, cv::INTER_CUBIC, cv::BORDER_CONSTANT, mean(img));  
  
cv::imshow("rotated img", img);  
cv::waitKey(0);
```

# Outputs

## - Anomaly Detection




NDBuffer  
List

Prediction

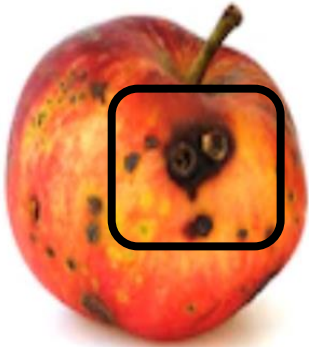
Box Info

Anomaly Score

Prediction & Anomaly Score

Image			
Prediction (Class)	0 (Good)	0 (Good)	1 (Bad)
Probability	0.9887	0.8830	0.0001
	0.0012	0.1170	0.9999

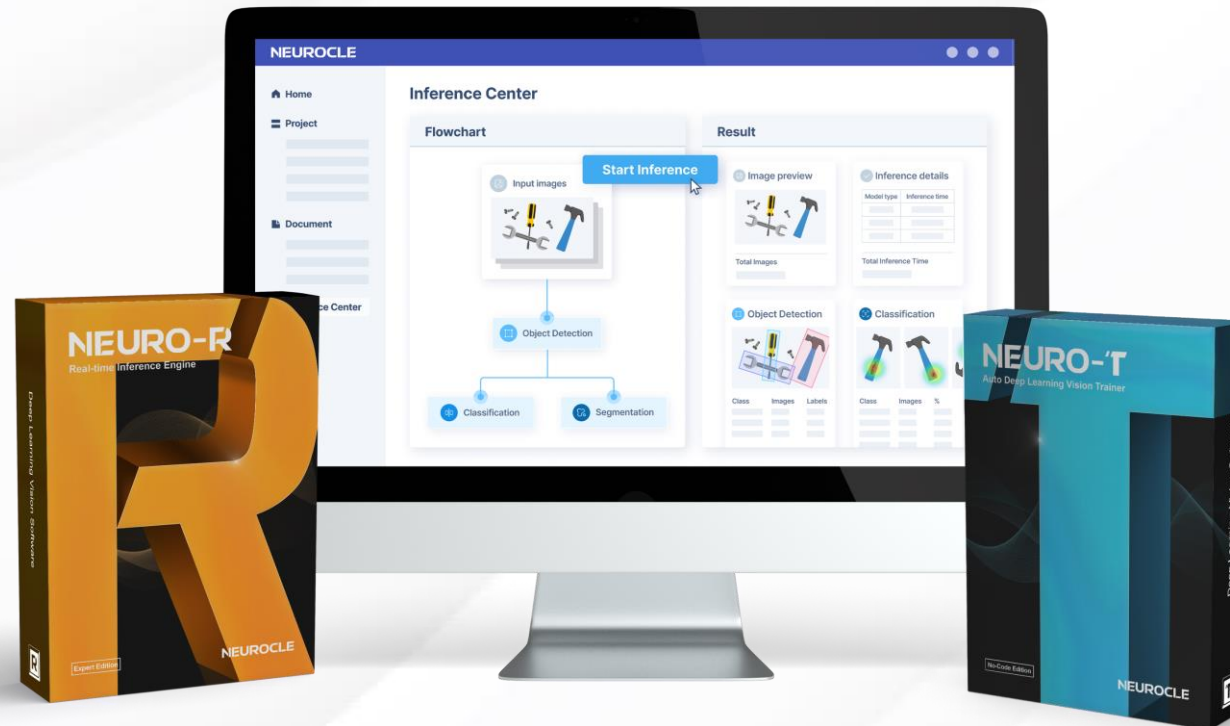
Box Info

Box Info	
----------	--

## 02. 적용 방법

# Neuro-R 적용의 중요 원칙

- Neuro-R을 문제 없이 사용하기 위해서는 Neuro-T에서와 동일한 작업 순서를 따르는 것이 매우 중요합니다.



# Neuro-R 활용 예시

- 앞의 내용에 대한 이해를 돕기 위해 Neuro-R 활용 방법의 예시를 살펴보겠습니다.
- Neuro-T와 Neuro-R에서 아래 세 가지 기능을 사용하는 순서에 중점을 두고 설명하겠습니다.

**1****Threshold****2****ROI / Mask****3****Patch Mode**

# Threshold 적용 순서 [Neuro-T]

- Neuro-T에서 Threshold 기능을 사용하기 위해서는, 먼저 모델 학습을 완료해야 합니다.
- 학습 완료 후에 Threshold의 기준이 되는 Probability 값이 생성되기 때문입니다.

## 01. 학습 완료

(LabelSet) 2.0.2_Chips_cla					(Model) 2.0.2_Chips_Cla_512_fast				
Size	ROI	Mask	Label ▾	Set ▾	Rete...	ROI	Mask	Predicted Class ▾	Score Ⓢ
			choco	test				choco	
512			Choco	test				Choco	99.92
512			Choco	test				Choco	99.88
512			Choco	test				Choco	99.84
512			Choco	test				Choco	99.57
512			Choco	test				Choco	99.22
512			Choco	test				Choco	99.11
512			Choco	test				Choco	99.09
512			Choco	test				Choco	98.97
512			Choco	test				Choco	90.06
512			Choco	test				Choco	76.76



## 02. Threshold 적용

Threshold (Segmentation)

☐ scratch

☐ tear

☐ dent

☐ concave

☐ depress

Minimum Size (pixel)

WidthHeight

▼

▼

▼

▼

▼

Probability (%)

/ 100.00

/ 100.00

/ 100.00

/ 100.00

/ 100.00

Cancel

OK

## Threshold 적용 순서 [Neuro-R]

- Neuro-T에서와 마찬가지로, **Neuro-R**에서도 Probability 값이 사전에 준비되어 있어야 합니다.
- 아래 이미지는 Threshold 적용과 관련된 API를 순서대로 보여줍니다.

### API

```
nrt::NDBuffer prob_thres = model.get_prob_threshold();
status = nrt::prob_map_threshold(output_prob, prob_thres, thresholded_pred);

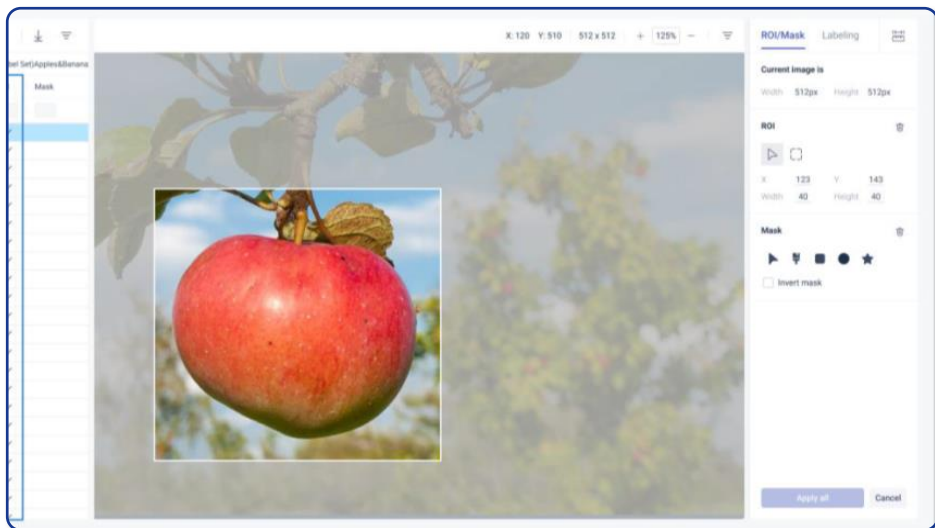
nrt::Shape thresholded_pred_shape = thresholded_pred.get_shape();
for (int j = 0; j < thresholded_pred_shape.dims[0]; j++) {
    int cls = *thresholded_pred.get_at_ptr<int>(j);
    cout << "image name : " << img_names[j] << endl;
    cout << "image - Prediction class index(thresholded): " << cls << ((cls < 0) ? "(Unknown)" : "") << endl;
}
```



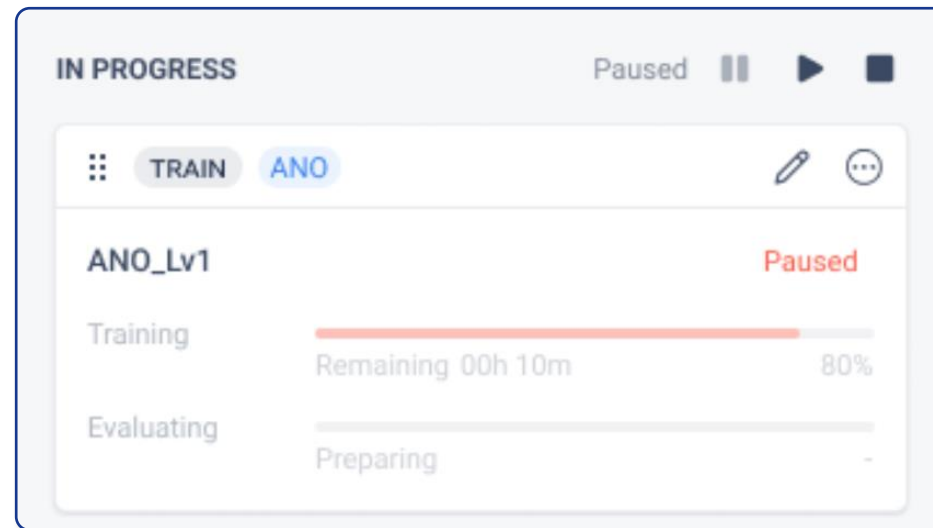
## ROI/Mask 적용 순서 [Neuro-T]

- ROI/Mask는 모델 학습 전에 원본 이미지를 전처리하는 기능입니다.
- 따라서 Neuro-T에서는 원본 영상이 아닌 ROI/Mask 처리된 이미지가 학습용 이미지로 사용됩니다.

### 01. ROI/Mask 적용



### 02. 모델 학습



## ROI/Mask 적용 순서 [Neuro-R]

- Neuro-R에서도 ROI/Mask 처리된 이미지를 사용해야 합니다.
- 아래 이미지는 ROI/Mask 적용과 관련된 API를 순서대로 보여줍니다.

### API

```
roi_info = model.get_roi_info();  
mask_info = model.get_mask_info();  
status = nrt::set_roi_mask(image_buff, roimasked_image_buff, roi_info, mask_info);  
  
status = nrt::resize(roimasked_image_buff, resized_image_buff, input_image_shape, interpoly);  
  
status = executor.execute(resized_image_buff, outputs);
```

## ROI/Mask 적용 순서 [Neuro-R] (For Object Detection)

- Object detection 모델에서는 ROI/Mask 적용에 약간의 차이점이 있습니다.
- Bounding Box의 크기 및 중심 좌표를 ROI 영역에 따라 계산해야 하기 때문입니다.

### API

```
nrt::NDBuffer roi_info = model.get_roi_info();
int* roi_info_ptr = roi_info.get_at_ptr<int>();

int roi_x = roi_info_ptr[2];
int roi_y = roi_info_ptr[3];
int roi_height = roi_info_ptr[4];
int roi_width = roi_info_ptr[5];

double h_ratio = (double)input_image_shape.dims[0] / roi_height;
double w_ratio = (double)input_image_shape.dims[1] / roi_width;

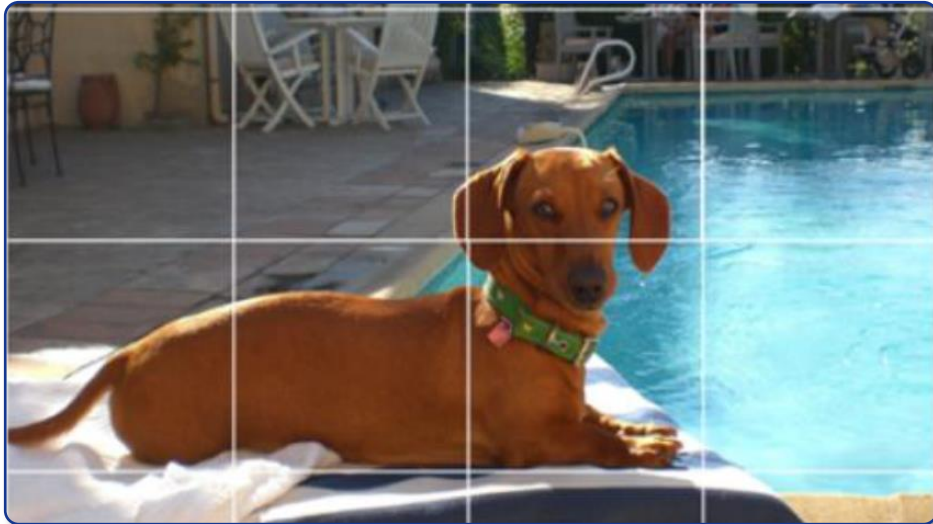
BoundingBox bbox = convert_to_bounding_box(bcyxhw_ptr, h_ratio, w_ratio);

bbox.box_center_Y = (double)bbox.box_center_Y / h_ratio + roi_x;
bbox.box_center_X = (double)bbox.box_center_X / w_ratio + roi_y;
```

# Patch Mode 적용 순서 [Neuro-T] (For Segmentation)

- 다른 모델 유형과 다르게, Segmentation 모델은 이미지를 분할하여 학습에 사용합니다.
- Neuro-T에서는 이미지 분할 크기에 대해 세 가지 옵션을 제공합니다. (128x128, 256x256, 512x512)

## 01. 이미지 분할 크기 선택



## 02. 분할된 이미지로 학습 진행



## Patch Mode 적용 순서 [Neuro-R] (For Segmentation)

- Neuro-R에서도 Segmentation 모델 사용 시, 이미지를 분할해야 합니다.
- 분할된 이미지는 이후 분석 결과 표시를 위해 다시 결합하여 사용됩니다.

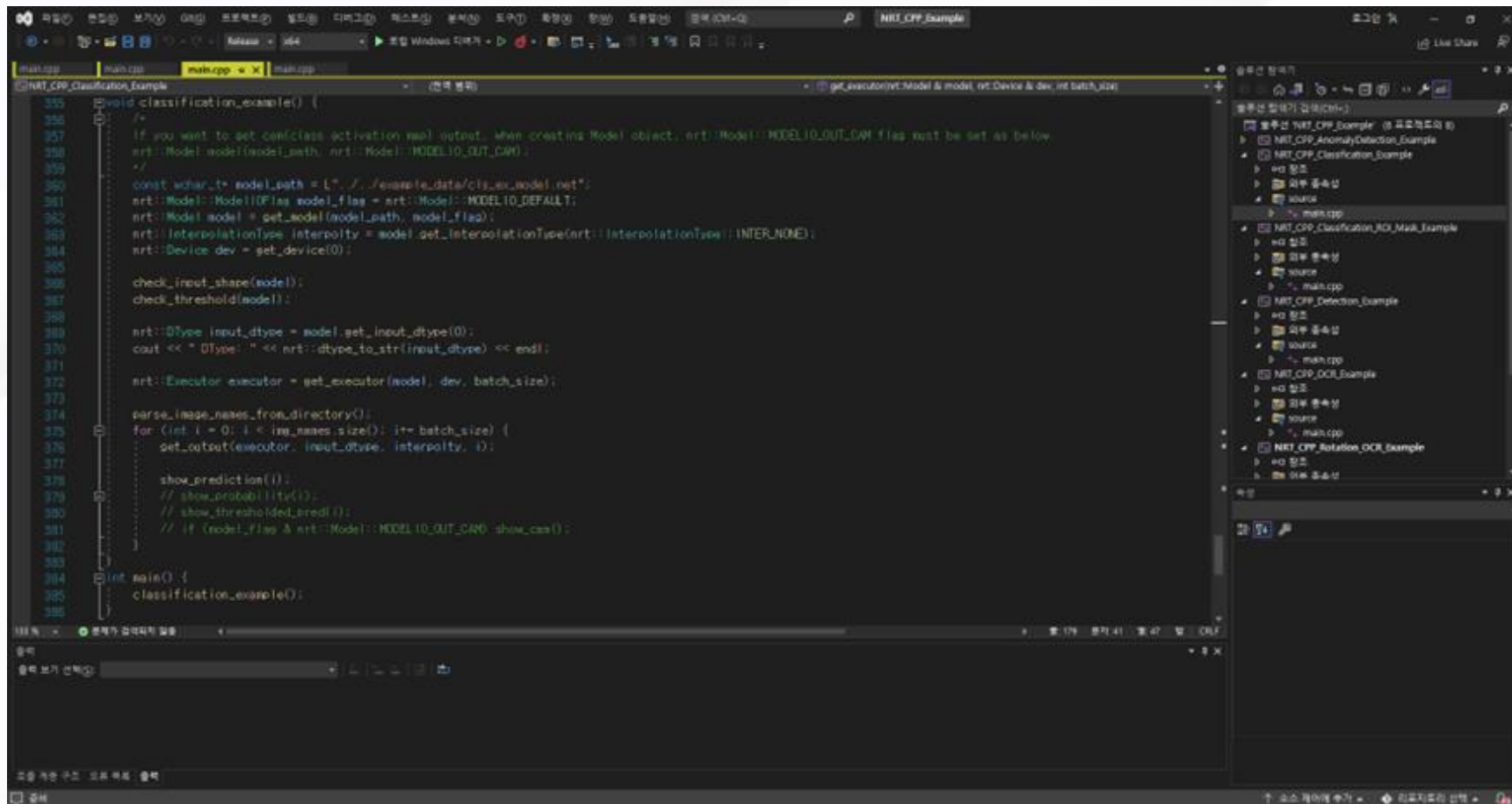
### API

```
status = nrt::resize(image_buff, resized_image_buff, scale_factor, interpoly);  
status = nrt::extract_patches_to_target_shape(resized_image_buff, input_image_shape, image_patch_buff, patch_info);  
status = executor.execute(image_patch_buff, outputs);  
status = nrt::merge_patches_to_orginal_shape(outputs.get_at(pred_idx), patch_info, merged_output);
```

## 03. 샘플 코드 분석



# Visual Studio에서 계속됩니다.



# Q & A



**감사합니다.**