오픈 소스 라이브러리 기반의 딥러닝을 이용한 선체 블록 모형 식별

Ship Block Model Identification using Deep Learning based on Open Source Library

2019 – 05 – 17 / 제주ICC

전해명, 노재규

Contents

1. Introduction

1. Introduction

1.1 Background and Purpose of the Study

연구 배경

- 작업장이 광범위해 사람이 모든 위치를 파악하기 위해서는 상당한 시간과 인력이 소모됨 (Byeong-Wook Nam, et al., 2017).
- 일정변경 등에 의해 블록의 위치는 실시간으로 변경됨 (Jong-Gye Shin, et al., 2006).
- 블록의 종류는 다양하고 개수는 수십 에서 수백 개에 이름.
- 강철에서 전파가 반사되는 특성으로 인해 관련 기술을 적용하기 어려움.

인력의 낭비와 작업자의 편리함을 보완할 수 있도록 선체 블록을 식별하는 방법에 관한 사전 연구로서 선체 블록 모형을 이용한 블록 식별을 수행함

연구 방법 및 절차

선체 블록 모델 인식을 위한 데이터셋 구축

- 좋은 학습 성능에 필요한 특징을 보유한 이미지 필요
- 하나의 모델은 최소 약 80장의 학습을 위한 이미지 필요
- 이미지에 포함된 모델의 위치 데이터 생성을 위한 수작업 요구

선체 블록 모델 분류 Training Model 학습

- 사용할 하드웨어에 알맞은 성능의 학습모델을 선택하여 분류를 수행할 모델의 데이터셋으로 재 학습
- GPU를 이용한 병렬연산 학습으로 시간 감축
- 학습은 Loss Value가 일정한 값에 수렴될 때 까지 진행

학습한 Training Model 평가

- 총 40장의 Accuracy Check용 이미지를 이용해 학습 모델 평가

1. Introduction

1.2 What is Deep Learning?

About Deep-Learning 사람의 신경망을 모방하여 기계가 다층 구조를 통해 학습(높은 수준의 추상화)하도록 만든 기술

- Deep Learning은 1986년 Rina Dechter에 의해 machine learning community에서 소개되었음.
- 1989년, 'Backpropagation Applied to Handwritten Zip Code Recognition(Yann LeCun et al.) 우편물의 우편번호를 인식하는 Deep Neural Network를 소개. (training에 거의 3일이 소요되어 다양한 분야에 적용하기에 한계를 가졌음.)
- 2007년, 'Learning multiple layers of representation(G. E. Hinton et al.)' 딥러닝이라는 용어를 본격적으로 사용하기 시작함.

Machine Learning과 Deep Learning의 차이점

	Machine Learning	Deep Learning
Training dataset	Small	Large
Choose your own feature	Yes	No
Parameters	Many	Few
Training time	Short	Long

< 출처 : Mathworks >

딥러닝을 사용하기 위해 필요한 환경

1. 많은 양의 데이터셋이 필요함.
2. 고사양의 GPU가 필요함.
3. **지도학습이므로, 데이터에 따라 정확도의 의존성이 큼.**

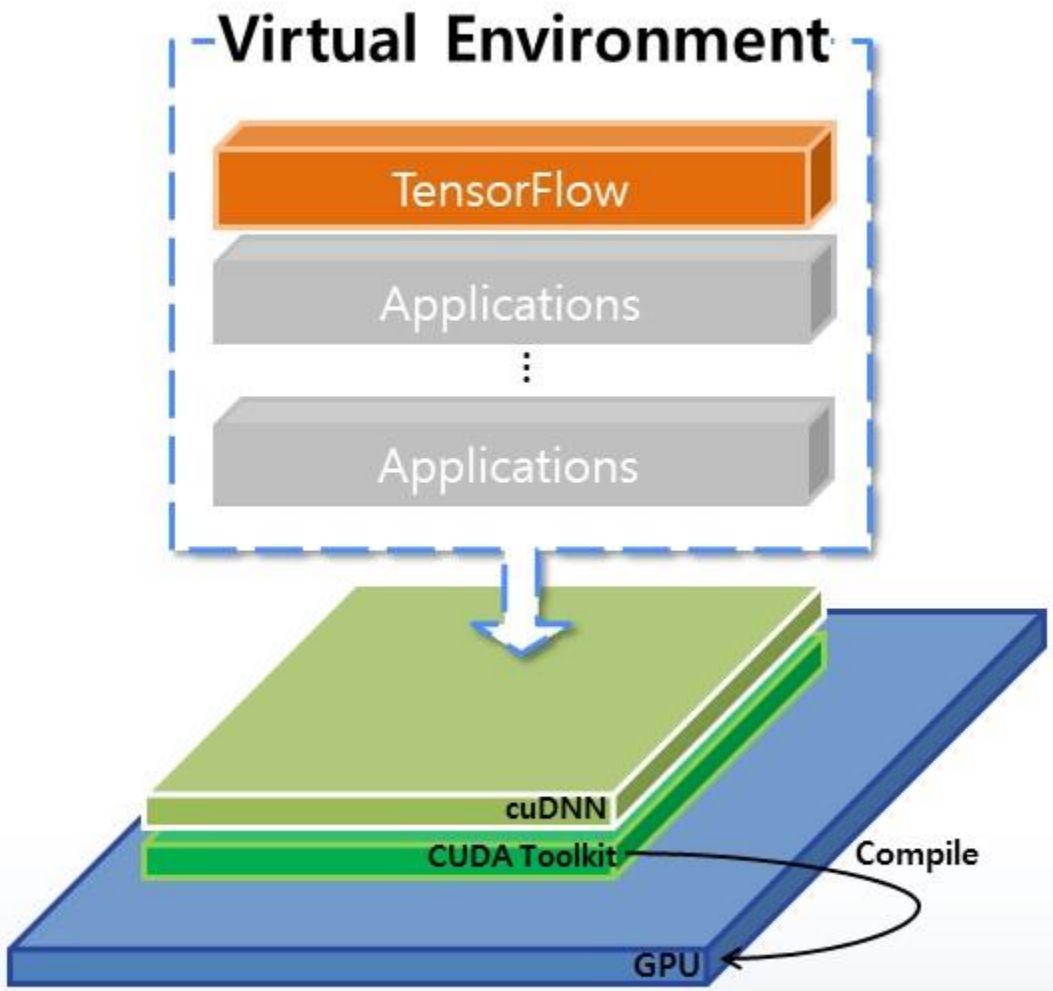
사용자가 이미지에서 관련 특징들을 추출하고, 물체의 특징을 표현하거나 예측하는 모델을 만들어야 함.

사용자가 이미지에서 관련 특징들을 추출할 필요 없이 딥러닝 알고리즘에 직접 제공함으로써 물체를 예측할 수 있음.

2. Ship Block Model Identification Process

2. Ship Block Model Identification Process

2.1 Architecture



설치 환경 / 사용 프로그램, 라이브러리

- **설치 환경**

  OS : Ubuntu 16.04

  GPU : $GTX\ 2080\,TI \times 4$개

  RAM : 16GB

- **사용 프로그램, 라이브러리**

  NVIDIA CUDA

  NVIDIA cuDNN

  ANACONDA Virtual Environment

  Python 3.5

  TensorFlow

  SPYDER

- GPU를 CUDA Toolkit을 이용하여 컴파일하며, CUDA의 cuDNN GPU 가속화 프레임 워크로 적합한 환경을 만듦.
- ANACONDA의 Virtual Environment 를 이용해 TensorFlow Library를 이용한 딥러닝 뉴럴 네트워크 모델을 훈련함

2. Ship Block Model Identification Process

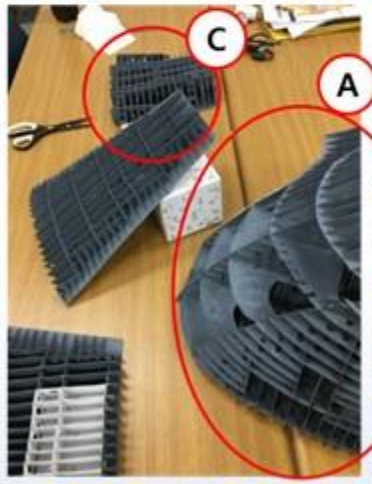
2.2.1 Gather and Labeling pictures

학습에 필요한 이미지의 특징

- TensorFlow는 좋은 이미지 검출 능력을 가지기 위해 수백 개의 이미지가 필요함
- 훈련을 위해 촬영하는 이미지에는 Detection Model과 함께 무작위 물체가 배치해야 함
- 이미지의 배경과 조명 조건이 다양해야 함
- Detection Model은 무작위 물체와 겹치거나 반만 촬영된 이미지가 포함돼 훈련해야 함
- 이미지가 각 200KB 미만, 해상도는 720 * 1280 이상이여야 함

- ***Ship Block Model dataset***의 일부 (총 400장의 이미지 촬영) / iPhone 6s 를 이용하여 촬영 (이미지 크기는 4032 * 3024 / 3024 * 3024)
- 하나의 데이터셋에 크기가 다른 이미지를 사용하여도 Training 가능함
- 학습에 필요한 이미지의 특징을 보유함

A : 반만 보이는 Detection model / B : 무작위 물체와 배치됨 /C : 겹쳐 있는 Detection model



3024 * 3024

4032 * 3024

Block_A Block_B Block_C Block_D

< 사용된 Ship_Block_Model >

KUNSAN NATIONAL UNIVERSITY

국립군산대학교 KUNSAN NATIONAL UNIVERSITY

2. Ship Block Model Identification Process

2.2.1 Gather and Labeling images

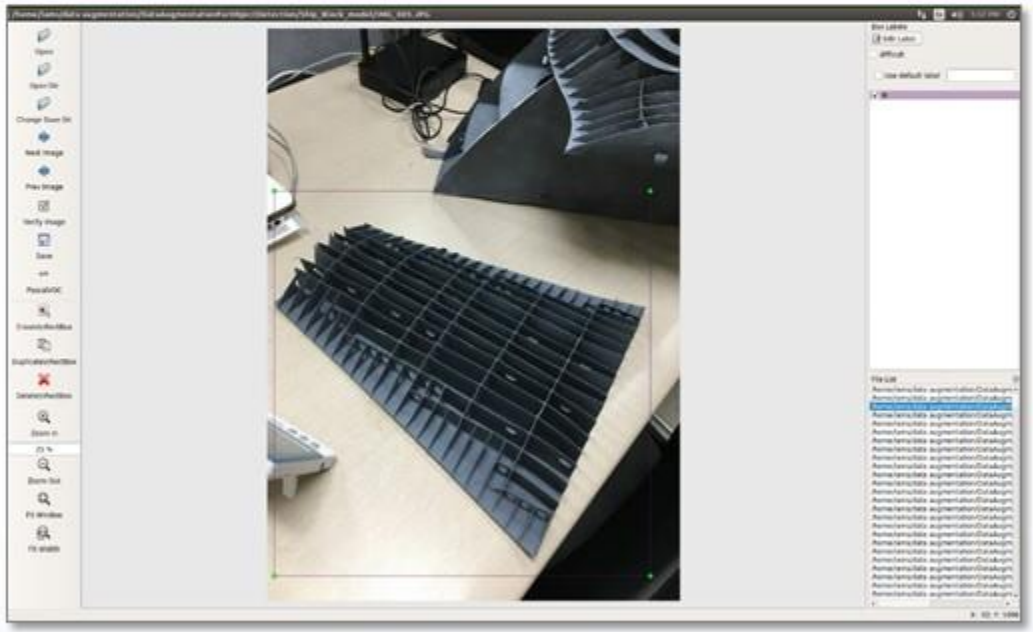
Image XML file

```
<annotation>
  <folder>Ship_Block_model</folder>
  <filename>IMG_003.JPG</filename>
  <path>
    /home/lams/data augmentation/DataAugmentationForObjectDetection/Ship_Block_model/IMG_003.
  </path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>3024</width>
    <height>4032</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>B</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>52</xmin>
      <ymin>1144</ymin>
      <xmax>2808</xmax>
      <ymax>3856</ymax>
    </bndbox>
  </object>
</annotation>
```

< 각 이미지당 생성된 XML file >

Labeling image

1. LabelIMG 파이썬 스크립트를 이용한 XML file 생성
2. 이미지에 객체 위치에 대한 레이블 정보가 포함된 XML 파일을 저장함
   - .XML 파일은 Training에 이용될 TFRecord파일을 생성하는데 사용됨
3. 초기 생성한 이미지의 수가 많을수록 많은 시간이 소요됨 (노동집약적)



< Labelimg 실행 화면 >

< 파일 변환 순서 >

2. Ship Block Model Identification Process

2.2.2 Data Augmentation

About Data Augmentation

① 왜 많은 양의 데이터가 필요한가

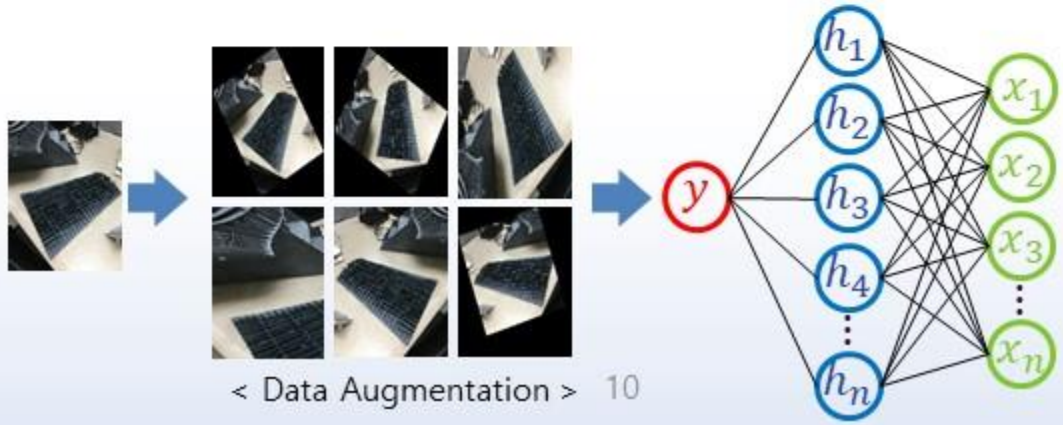
- Neural Network모델을 training할 때, 실제로 일어나는 일은 입력 Data를 일부 출력 Data에 Mapping할 수 있도록 Parameter를 조정하는 것.
- 최종 목표는 모델이 가장 낮은 손실(또는 가장 높은 정확도)을 갖도록 하는 것.

= Parameter들이 올바르게 조정되었을 때 가능함.

- 이미지 데이터를 이용하는 Neural Network들은 전형적으로 많은 Parameter를 가지고 있음.
- Parameter가 많은 경우 우수한 성능을 얻기 위해서는 Parameter와 비례하는 양의 training Data를 이용해 training을 해야함.

② Data Augmentation이 필요한 이유

- 획득할 수 있는 데이터의 양에 한계가 있는 경우, 데이터의 양은 증가시키기 위해 사용함.
- 완벽하게 학습한 Neural Network 모델이 아니라면 Augmentation 후의 데이터를 새로운 데이터로 인식함.
- 간단한 밝기의 조절, 회전, 크기 변형 등으로도 효과적으로 데이터를 증축 시킬 수 있음



< Data Augmentation >

KUNSAN NATIONAL UNIVERSITY

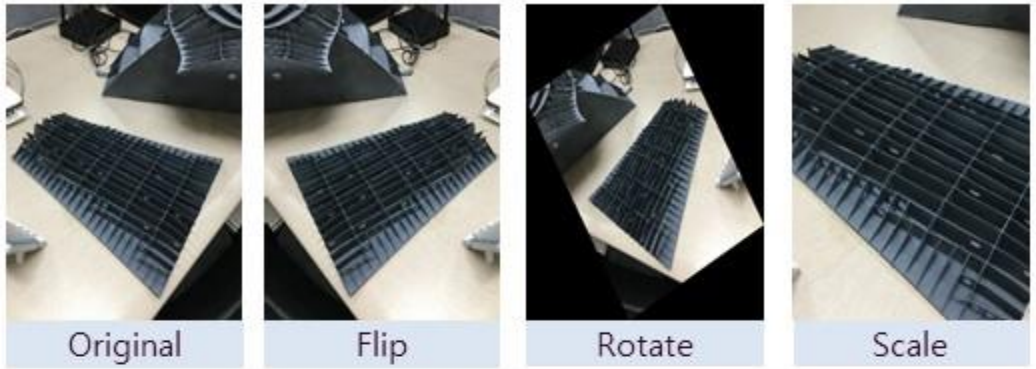
국립군산대학교 KUNSAN NATIONAL UNIVERSITY

2. Ship Block Model Identification Process

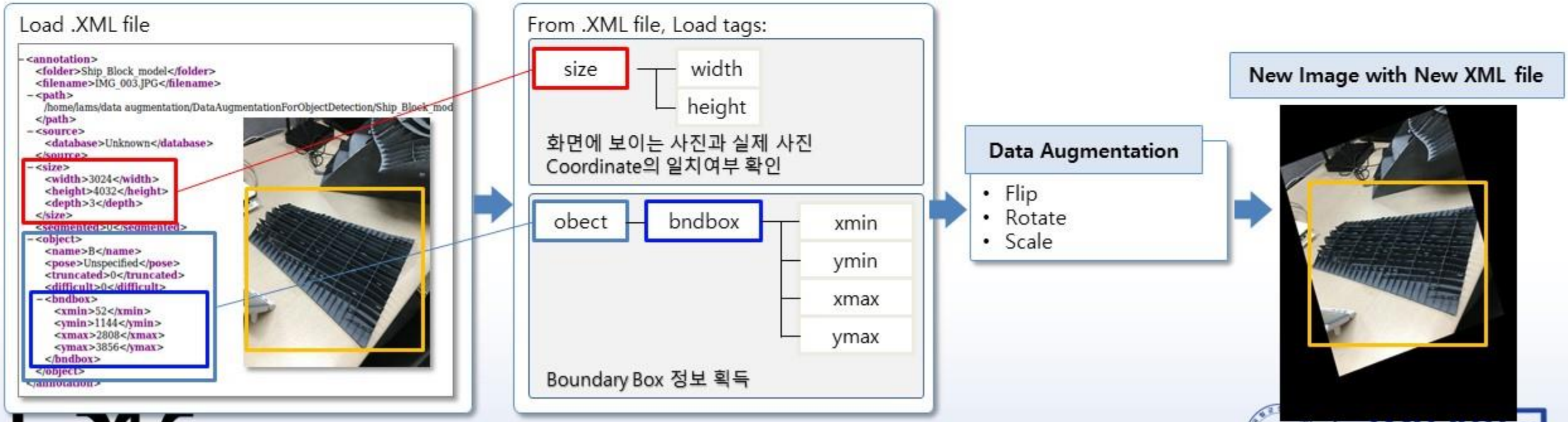
2.2.2 Data Augmentation

Data Augmentation Technique

- Flip – 이미지를 수평 및 수직으로 반전
- Rotate – 원하는 각도만큼 이미지를 회전
- Scale – 이미지를 바깥쪽 또는 안쪽으로 확대
- Random – Flip 여부, Rotate의 각도, Scale의 비율을 무작위로 이미지 변형



Data Augmentation Process



KUNSAN NATIONAL UNIVERSITY

2. Ship Block Model Identification Process

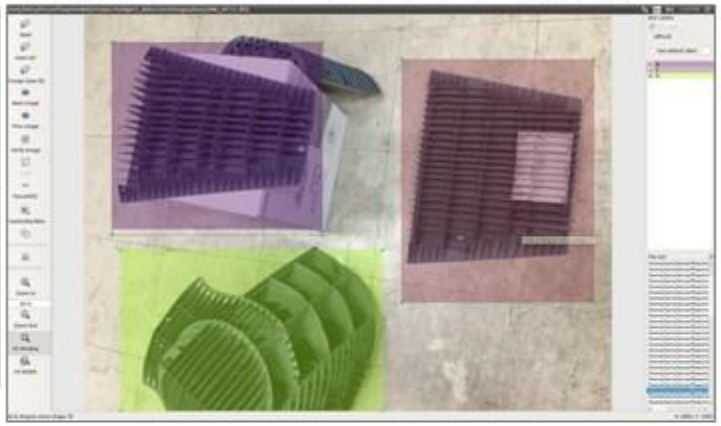
2.2.3 Generating training data

.XML file

.CSV file

**TFRecord file

- XML (Extensible Markup Language)
  인터넷에 연결된 시스템끼리 데이터를 쉽게 주고 받을 수 있게 하여 HTML의 한계를 극복할 목적으로 만들어진 다목적 마크업 언어.



- Labeling이 완료된 Dataset은 하나의 이미지당 하나의 xml로 이루어져 있음.
- "IMG_8712"와 같이 하나의 이미지에 두 개 이상의 모델이 포함되어 있다면, xml 파일에는 각 모델의 위치에 대한 Boundary Box의 데이터를 모델의 개수만큼 나타냄.

- CSV (Comma-Separated Variables)
  몇 가지 필드를 쉼표로 구분한 텍스트 데이터 파일.

- 이미지에 포함된 모델의 위치에 대한 Boundary Box의 정보를 xmin,ymin,xmax,ymax의 순서로 쉼표로 구분해 저장함
- Filename – width – height – class – boundary box 순으로 각 이미지의 XML 파일에 저장되어있는 데이터를 불러와 랜덤 순서로 배치함.
- 랜덤순서로 배치됨으로써 Neural Network의 *배치(batch)에 도움을 줌.

	Standard	Standard	Standard	Standard	Standard	Standard	Standard	Standard
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	IMG_026.JPG	4032	3024	D	338	274	2403	2244
3	IMG_003.JPG	4032	3024	B	1233	34	3818	2804
4	IMG_027.JPG	4032	3024	C	1223	909	2793	2964
5	IMG_078.JPG	4032	3024	C	248	549	1198	1789
6	IMG_078.JPG	4032	3024	B	967	223	1880	1490
7	IMG_078.JPG	4032	3024	A	2244	6	3864	1310
8	IMG_078.JPG	4032	3024	D	1	1520	1453	3022
9	IMG_051.JPG	4032	3024	C	1873	1134	3453	2904
10	IMG_051.JPG	4032	3024	B	1023	2044	2153	3024
11	IMG_051.JPG	4032	3024	D	768	529	1998	2019
12	IMG_051.JPG	4032	3024	A	1107	2586	1599	3024
13	IMG_008.JPG	3024	3024	D	161	313	2486	2695

*배치(batch) : 1 step의 학습에서 모든 데이터에 대한 손실함수를 계산하는 시간을 단축하고자, 데이터 중 일부만을 사용하는 것.
**TFRecord file : TensorFlow의 표준 데이터 파일 포맷 (대용량의 데이터를 효율적으로 처리할 수 있도록 함).

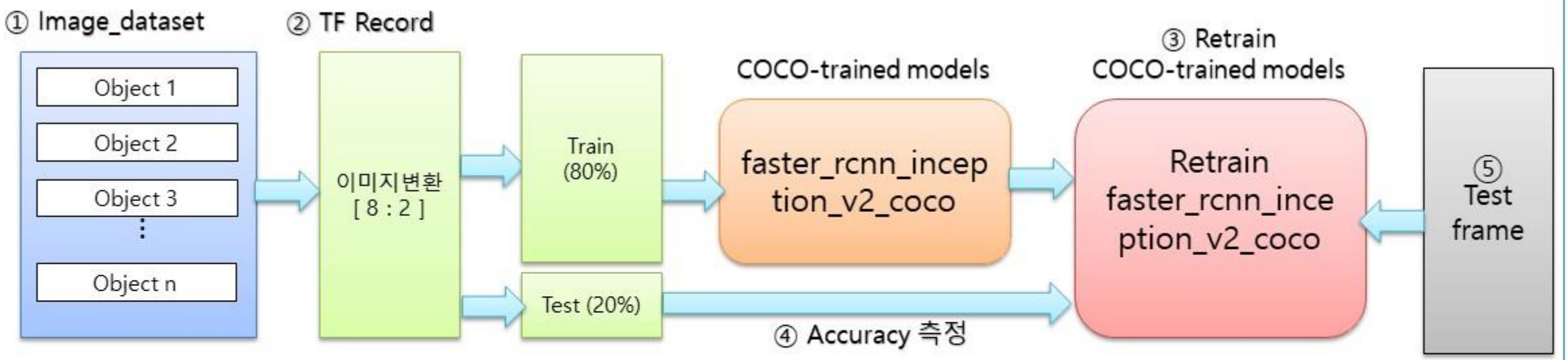
KUNSAN NATIONAL UNIVERSITY

국립군산대학교 KUNSAN NATIONAL UNIVERSITY

2. Ship Block Model Identification Process

2.2.4 Training

Inception_V2_COCO Model을 이용한 training과정



①	n개의 Object로 이루어진 Image_Dataset 제작
②	빠르고 편리한 학습을 위한 TF파일로 변환 (train data : 8 / test data : 2)
③	COCO-trained models의 'faster_rcnn_inception_v2_coco'모델을 제작한 Image Dataset으로 Retrain
④	②번에서 2의 비율로 지정한 test data로 Accuracy 측정
⑤	Retrain된 faster_rcnn_inception_v2_coco모델에 촬영한 Video를 이용하여 Object_Detection수행

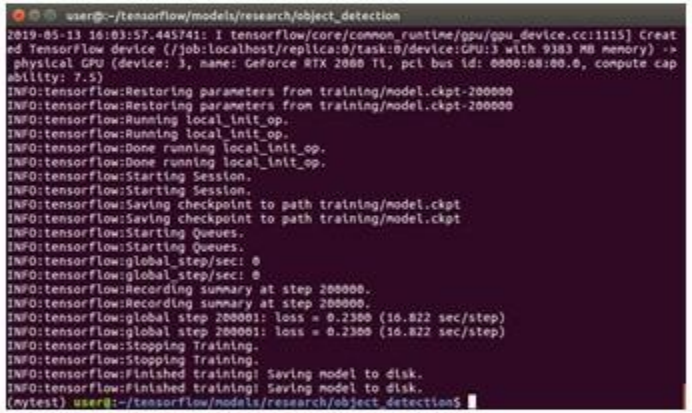
2. Ship Block Model Identification Process

2.2.5 Training

```
lams@lams-desktop: ~
(tensorflow) lams@lams-desktop : ~$ python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_coco.config
```

- 위 명령어와 스크립트를 이용해 Neural Network Model인 Inception_v2_coco의 training을 시작함.
- Training은 다음과 같이 총 4가지 경우로 진행하였음.

Dataset name	Data Augmentation	Step	Dataset 크기 (장)	최종 Loss Value
Ship_Block_Model_80k	X	80,000	400	0.02111
Ship_Block_Model_200k	X	200,000	400	0.02679
Augment_Ship_Block_Model_80k	O	80,000	2,400	0.2719
Augment_Ship_Block_Model_200k	O	200,000	2,400	0.05785



< Training이 진행되는 화면 >

- Training의 step이 증가할 수록 Loss Value는 특정한 값에 수렴하게 됨.
- Loss Value가 0.05이하로 낮아질 때 까지 training의 step 수를 정하는 것이 성능을 향상시킬 수 있음.
  (MobileNet-SSD 모델의 경우 약 20의 Loss Value로 시작하며, 값이 일관되게 2 정도가 될 때까지 training 하는 것이 좋음.)

3. Result

3. Result

3.1 Ship Block model – Object Detection Loss value & Accuracy check

Ship_Block_Model_80k

Loss value	0.1008	0.08796	0.09930	0.02111
Step	$19.93k$	$39.86k$	$59.76k$	$79.70k$
Elapsed time	$39m53s$	$1h19m53s$	$1h59m53s$	$2h39m53s$

Ship_Block_Model_200k

Loss value	0.2309	0.02889	0.01194	0.02679
Step	$19.72k$	$79.93k$	$140.1k$	$199.5k$
Elapsed time	$41m53s$	$2h49m53s$	$4h57m53s$	$7h3m53s$

Augment_Ship_Block_Model_80k

Loss value	2.357	0.1711	0.06228	0.2719
Step	$19.95k$	$39.89k$	$59.83k$	$79.78k$
Elapsed time	$41m53s$	$1h23m53s$	$2h5m53s$	$2h47m53s$

Augment_Ship_Block_Model_200k

Loss value	0.5839	0.1101	0.4394	0.05785
Step	$19.72k$	$79.90k$	$140.1k$	$199.3k$
Elapsed time	$41m53s$	$2h49m53s$	$4h57m53s$	$7h3m53s$

3. Result

3.1 Ship Block model – Object Detection

Ship Block model Object Detection 수행 영상

3. Result

3.1 Ship Block model – Object Detection

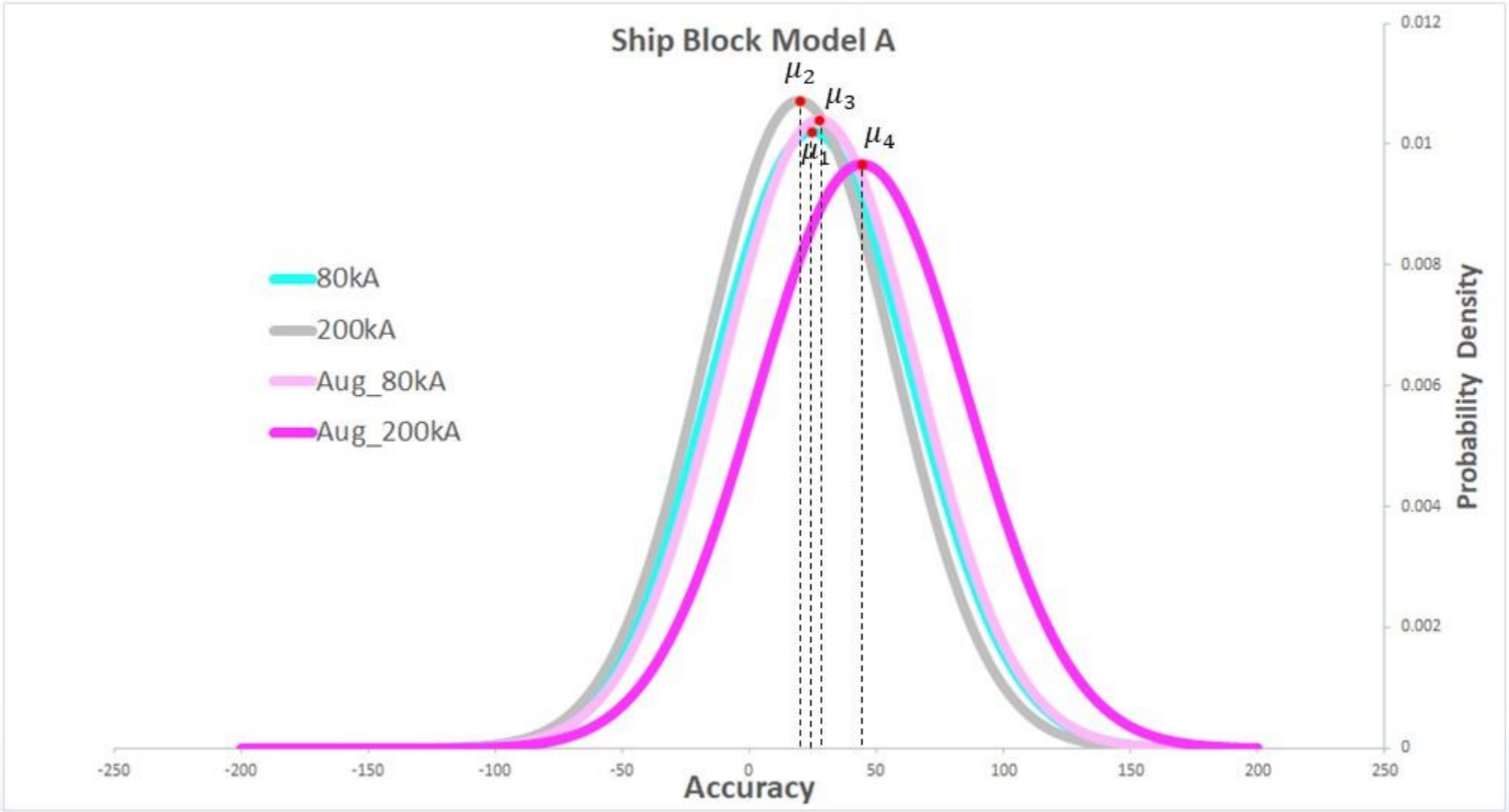
Ship Block model Object Detection Accuracy Check

Ship Block Model Detecting Model

Accuracy Check

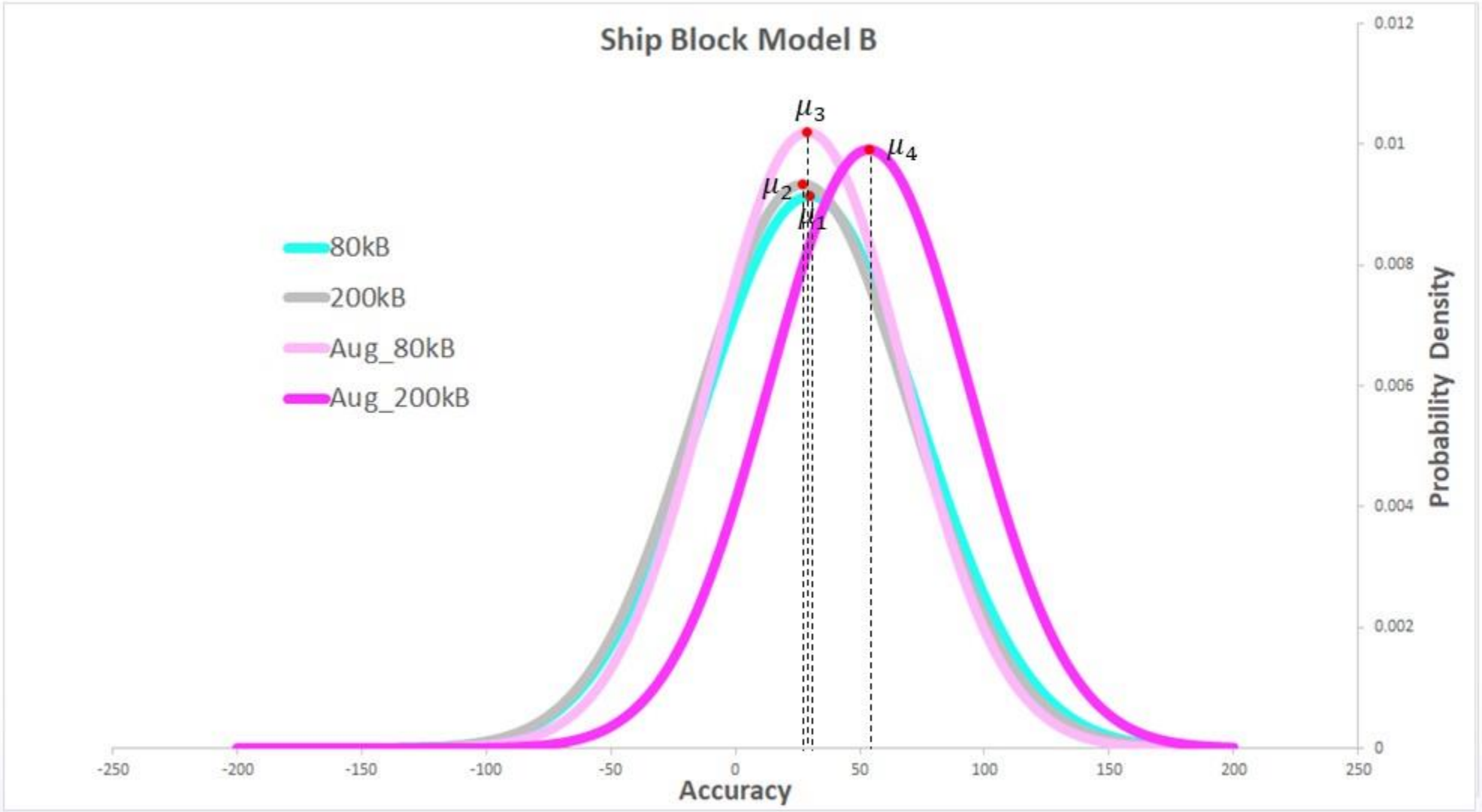
3. Result

3.1 Ship Block model – Object Detection



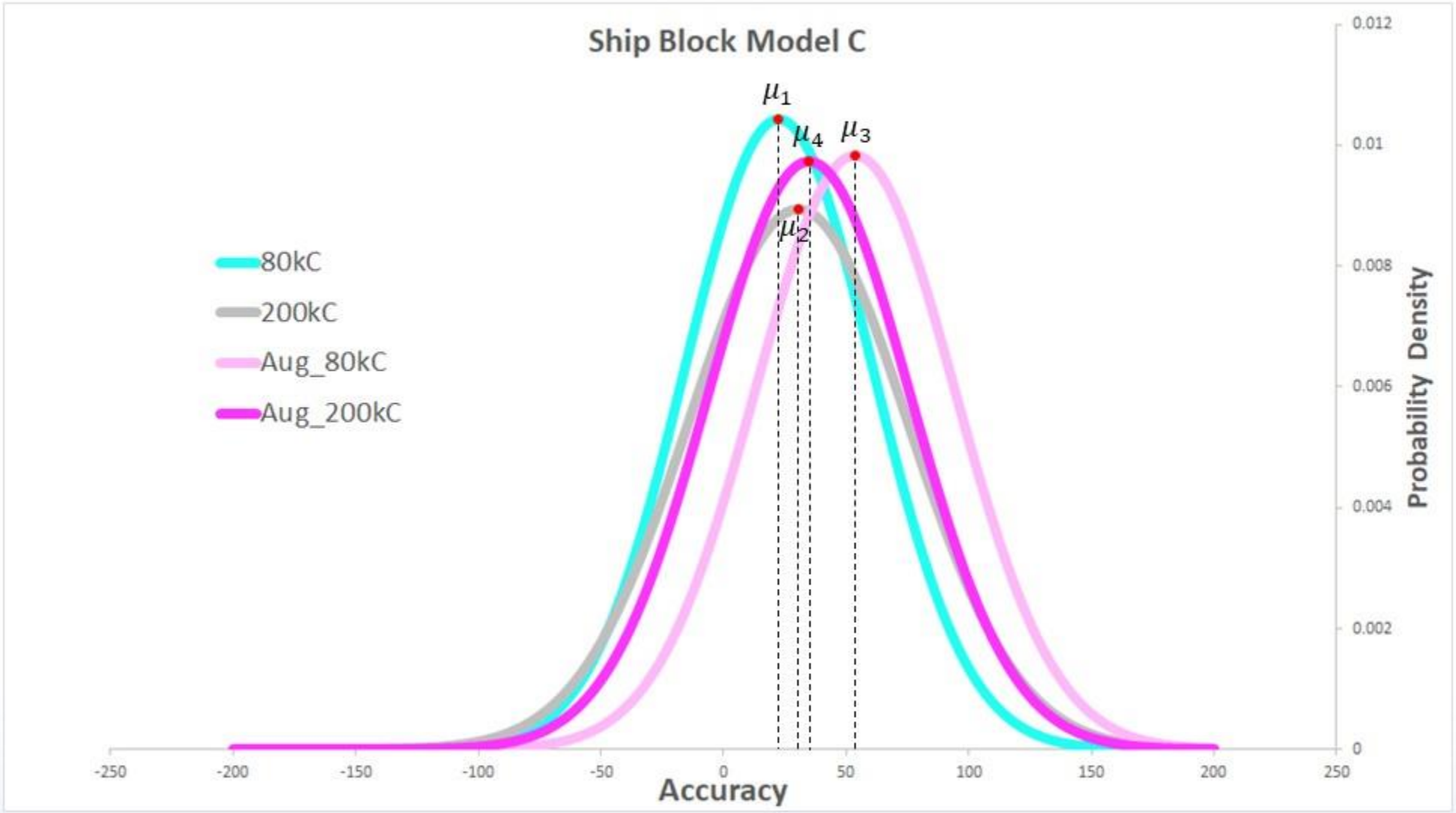
3. Result

3.1 Ship Block model – Object Detection



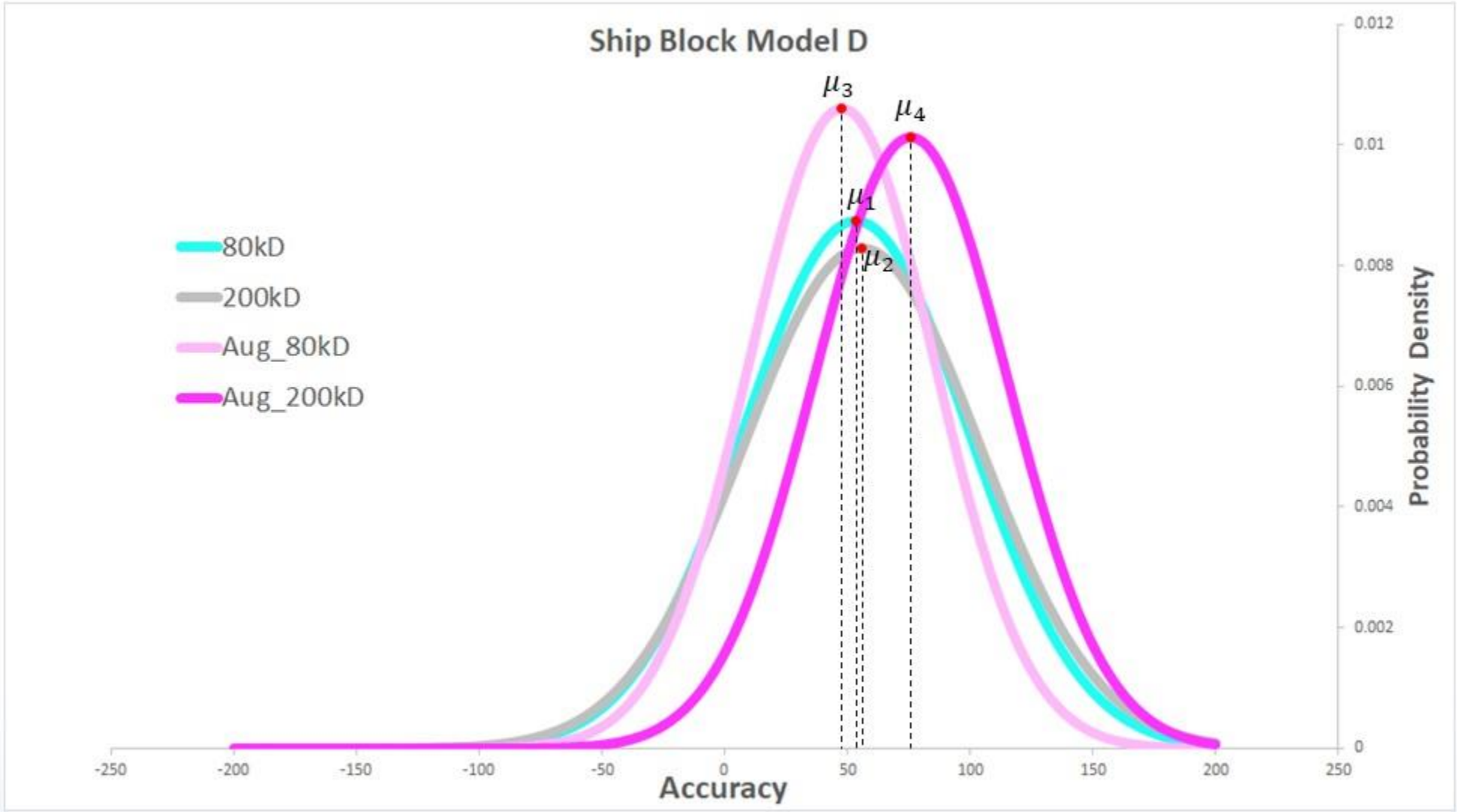
3. Result

3.1 Ship Block model – Object Detection



3. Result

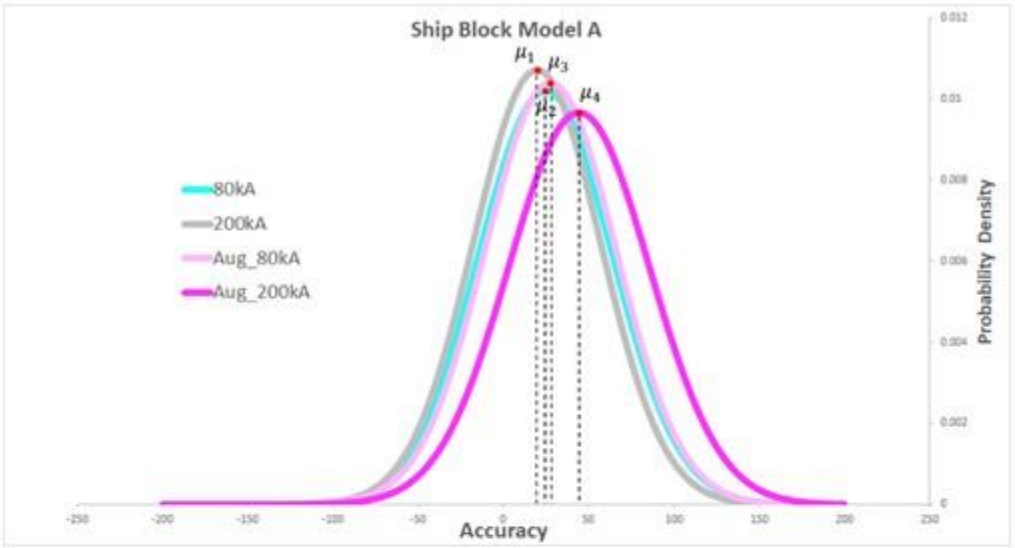
3.1 Ship Block model – Object Detection

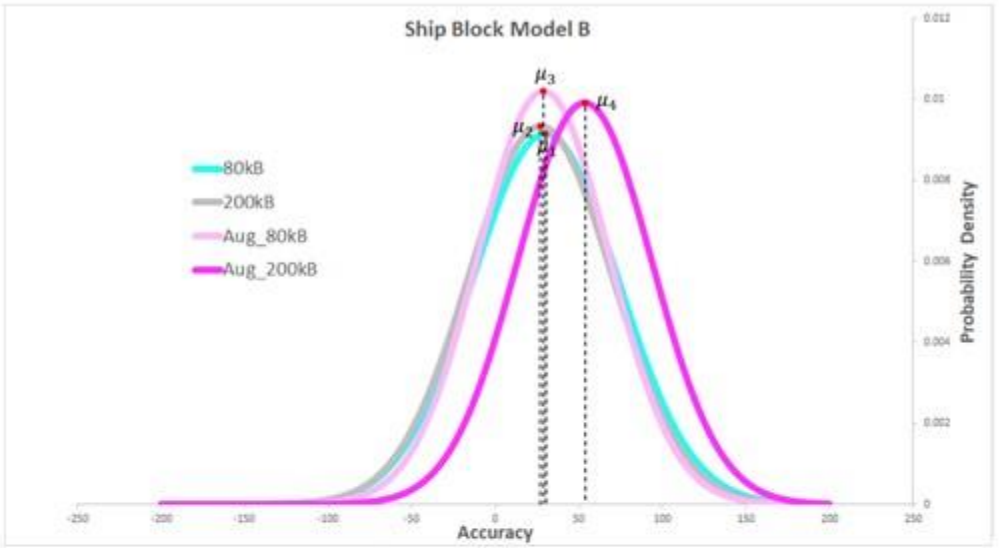


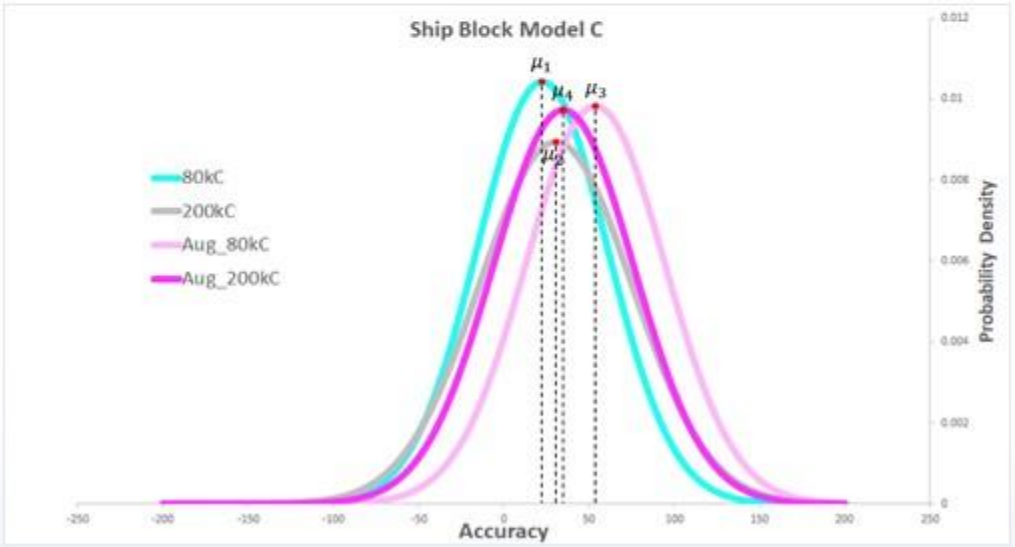
3. Result

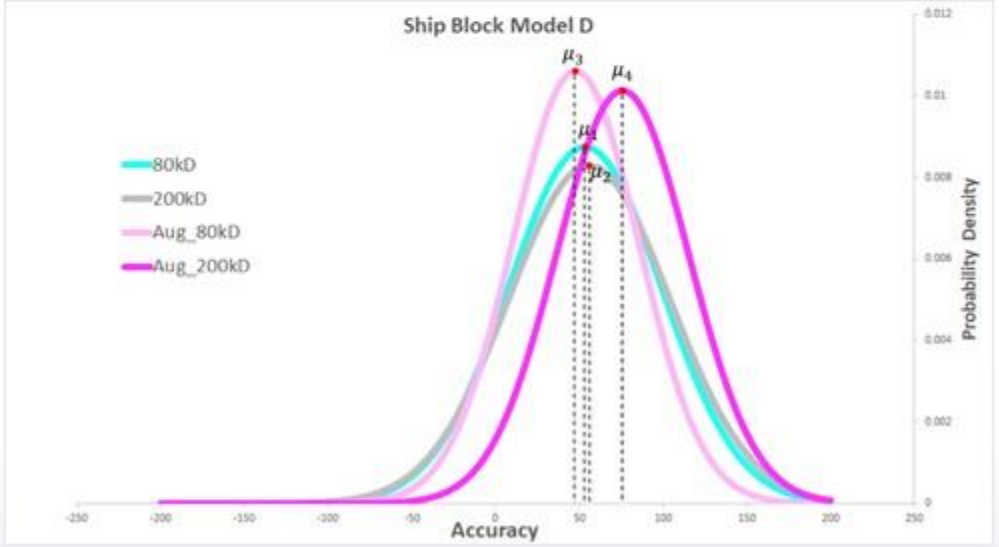
3.1 Ship Block model – Object Detection

Ship Block model Object Detection Accuracy Check









3. Result

3.2 Total Analysis

	Step	Elapsed Time	Final Loss value	Dataset Size (the number of)	Accuracy (%)
Ship Block Model 80k	$80k$	$2h39m53s$	0.02111	400	32.675
Ship Block Model 200k	$200k$	$7h3m53s$	0.02679	400	33.225
Augment Ship Block Model 80k	$80k$	$2h47m53s$	0.2719	2,400	39.45
Augment Ship Block Model 200k	$200k$	$7h3m53s$	0.05785	2,400	52.1167

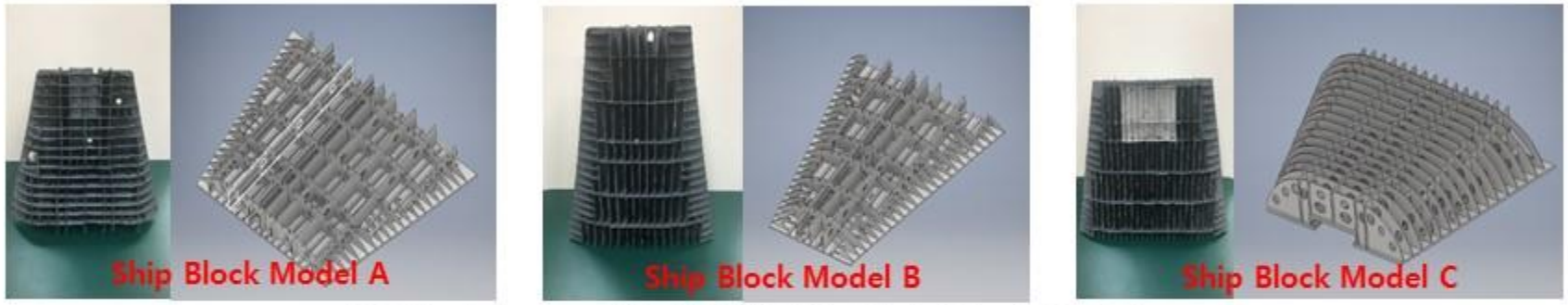
- **Dataset Augmentation을 통해 Dataset의 크기 증가 후 학습한 Neural Network 모델의 성능이 향상됨.**
- **Dataset의 Size 증가가 정확도를 더욱 향상시킴.**

Neural Network 모델이 분류해야 하는 모델의 종류가 많아질수록 Elapsed Time은 매우 증가할 것으로 예상됨. 이를 보완하기 위해 2차원 이미지를 이용한 학습방식이 아닌 다른 방식을 고안해 보고자 함.

4. Future work

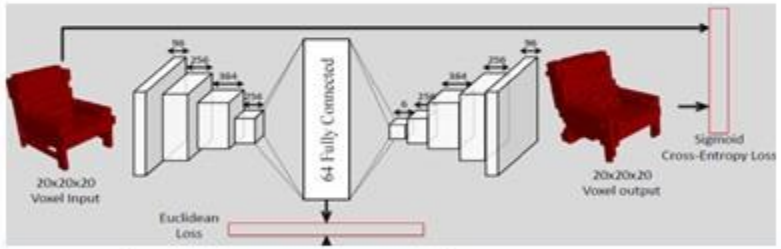
4. Future work

1. **3D CAD 모델을 이용한 다중 뷰 이미지 Nerual Network Model Training 방식 시도.**
   - 가상공간(simulation)에서 3D Modeling 된 Ship Block Model들을 배치하여 이미지 생성.
   - 기존 방식보다 이미지를 쉽게 수집할 수 있음.
   - 여러 Condition들을 필요에 따라 자유롭게 조절할 수 있음.



< 3D Modeling tool을 이용해 만든 Ship Block Model >

2. **3D CAD 모델을 이용한 볼륨 모델(VOXEL) Neural Network Model Training 방식 시도.**
   - 3D 볼륨 모델(volume model)을 복셀(voxel)형태로 단순화시킨 후 이를 CNN학습에 이용하는 것. (Byung-Chul Kim., 2018)



< 자료> R. Girdhar et al., in European Conference on Computer Vision(ECCV), 2016.

3. **실제 실무에 적용 가능한 Object Detecting Device 개발**
   - JETSON TX 2 등의 마이크로 컴퓨터를 이용하여 휴대 가능하며 사용하기 편리한 Device의 개발

Q & A

Reference

- TensorFlow Object Detecting API. *https://github.com/tensorflow/models*
- Data Augmentation source. *https://blog.paperspace.com/data-augmentation-for-bounding-boxes/*
- Edit XML file source. *https://docs.python.org/2/library/xml.etree.elementtree.html*
- Cyki, G. T., 2017. *Deep Learning from scratch Python.* Hanbit meaid.Inc.
- Nikhil, B. 2017. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms.* Hanbit meaid.Inc.
- Shin, J.G. & Lee, J.H., 2006. Prototype of Block Tracing System for Pre-Erection Area using PDA and GPS *Journalal of the Society of Naval Architects of Korea*, 43(1), pp.87-95
- Nam, B.W. Lee, K.H. Lee, J.J. & Mun, S.H., 2017 A Study on Selection of Block Stockyard Applying Decision Tree Learning Algorithm *Journalal of the Society of Naval Architects of Korea,* 54(5), pp.421-429
- Lee, Y.S. & Moon, P.J., 2017. A Comparison and Analysis of Deep Learning Framework *Journalal of the KIECS*, 12(1), pp.115-122.
- Alex, K. Ilya S. & Geoffrey E. H., 2012. ImageNet Classification with Deep Convolutional Neural Networks *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, Volume1, pp.1097-1105
- Matthew, D. Z.& Rob, Fergus., 2013. Visualizing and Understanding Convolutional Networks *European Conference on Computer Vision,* volume8689, pp.818-833
- Karen, S. & Andrew, z., 2014. Very Deep Convolutional Networks For Large-Scale Image Recognition *Published as a conference paper at ICLR 2015,*
- Jason, W. & Luis, P., 2017, The Effectiveness of Data Augmentation in Image Classification using Deep Learning *European Conference on Computer Vision,*

2. Ship Block Model Identification Process

2.2.1 Set up ANACONDA Virtual Environment

```
lams@lams-desktop: ~
lams@lams-desktop: ~$  conda create –n tensorflow pip python=3.5
```

1. conda create 명령어를 입력해 tensorflow라는 이름의 python 3.5버전 가상환경을 만듦.

```
lams@lams-desktop: ~
lams@lams-desktop: ~$  conda activate tensorflow
(tensorflow) lams@lams-desktop: ~$ pip install –ignore-installed –upgrade tensorflow-gpu
```

2. tensorflow 가상환경을 실행 시 위와 같이 입력창에 (tensorflow)라는 마크를 확인 가능함, pip을 이용해 TensorFlow-GPU를 설치함.
   CUDA와 cuDNN은 TensorFlow-GPU를 설치하기 전 설치가 필요함.

```
lams@lams-desktop: ~
(tensorflow) lams@lams-desktop: ~$ pip install "Libraryname"
```

3. pip을 이용해 필요한 다른 라이브러리들을 설치함.
   필요한 라이브러리 : Pillow, lXML, Cython, jupyter, matplotlib, pandas, opencv-python etc...

2. Ship Block Model Identification Process

2.2.1 Set up ANACONDA Virtual Environment

```
lams@lams-desktop: ~
(tensorflow) lams@lams-desktop : ~$   git clone "tensorflow models~"
```

4. TensorFlow git에서 Object_detection API를 다운로드해줌.

< 출처: TensorFlwo 예제 >

COCO-trained models

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes

5. TensorFlow에서 제공하는 Object detection model 중 하드웨어의 성능과 Train dataset에 적합한 COCO-trained model을 선택하여 적용함
   - 전력이 낮은 장치(스마트폰, 라즈베리 파이 등)에서 사용 시 SDD-MobileNet Model을 사용.
   - 전력의 제한이 없는 장치(Laptop, Desktop 등)에서 사용 시 RCNN Model을 사용.

- Object_detection에 사용될 하드웨어의 성능을 고려하여 'faster_rcnn_inception_v2_coco'를 선택함
- 호환성을 위해 TensorFlow에서 제공하는 라이브러리와 COCO-trained models의 Version을 항상 최신버전으로 사용해야 하는 유의점이 있음