Brightics Deep Learning 플랫폼

삼성SDS AI플랫폼Lab 장진엽 Lab장

AI 개발할 때 Pain Points

1. **Data 준비할때 고생 좀 덜하면 좋겠다**
2. 모델 선정 시행착오를 줄이면 좋겠다
3. GPU들을 공유하면 좋겠다 (Jupyter를 그냥 띄워놓은 상태에서)

1 Data 준비할때 고생 좀 덜하면 좋겠다

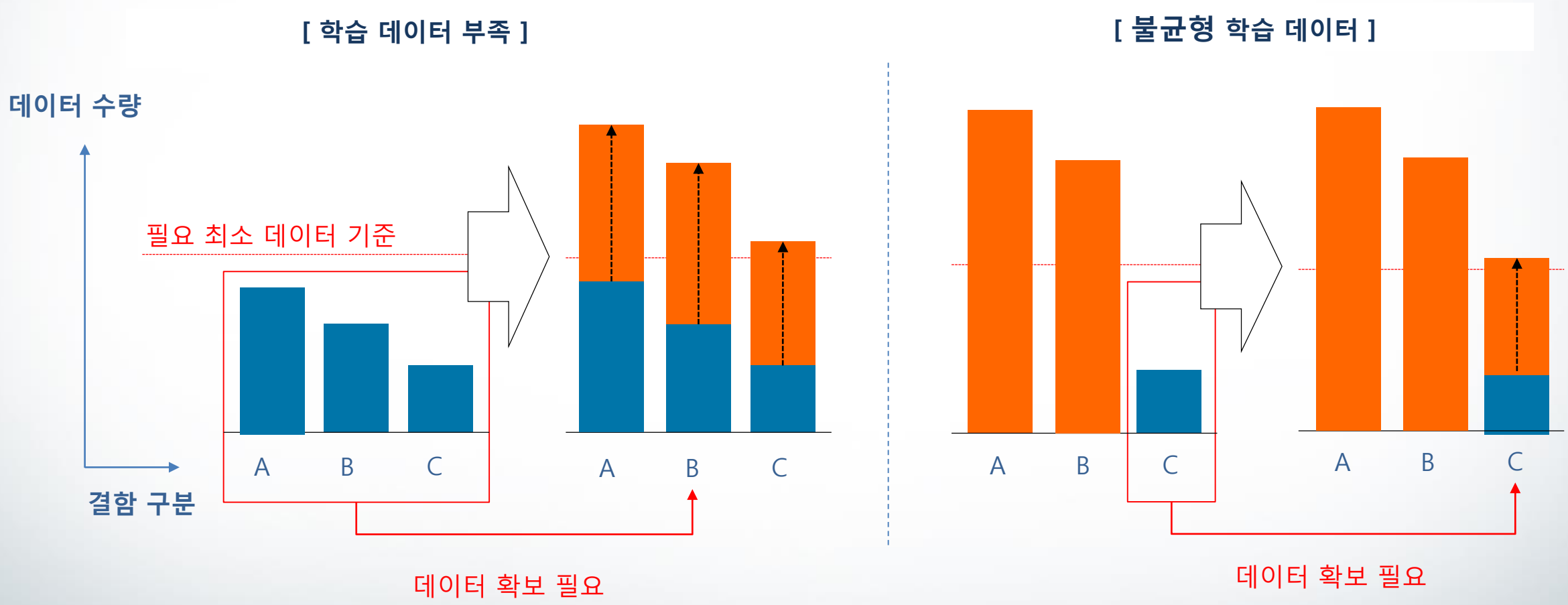
① Data가 너무 적을때 → GAN

② Data가 정리가 안될때 → Visual Search

③ Labeling 비용이 클 때 → Active Learning, Domain Adaptation

① 결함 데이터는 적은게 정상이다

제조 데이터는 대량의 정상건, PPM 수준의 결함 등
데이터의 부족과 불균형 데이터의 분포 → 성능 향상을 위해 해결책 필요



① GAN을 이용한 희소 결함 Class 데이터 증식

X : Real Data for Training

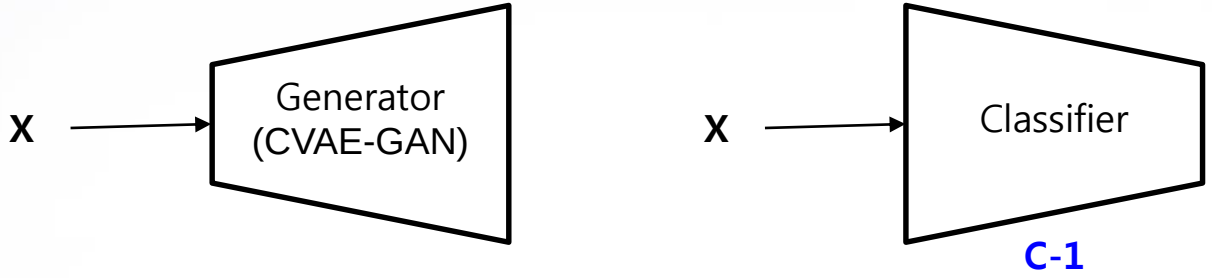
X^1 : Generated Samples

X^2 : High Confidence X^1

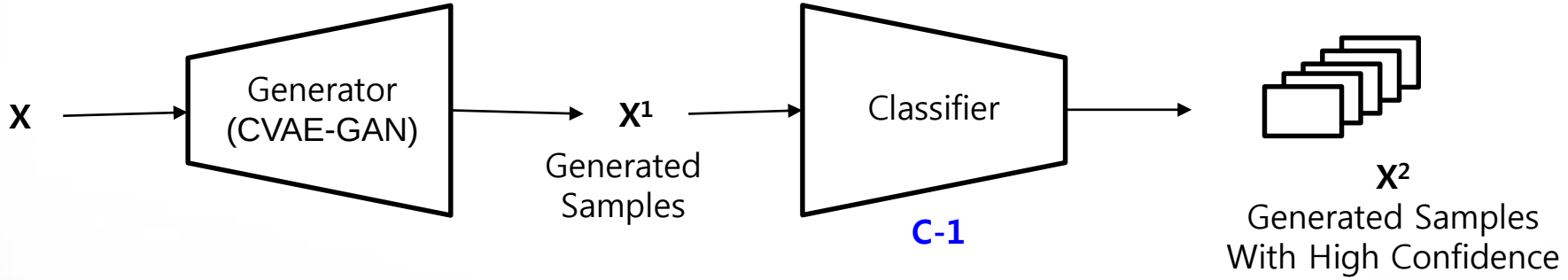
X^{test} : Real Data for Test

Y^1 & Y^2 : Classification Result

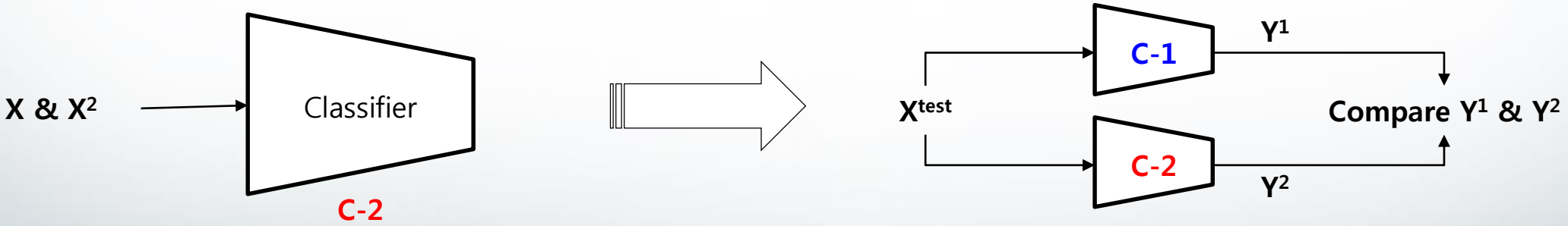
Step-1. Generator & Classifier training



Step-2. Training sample generation



Step-3. Second classifier training & Evaluation test



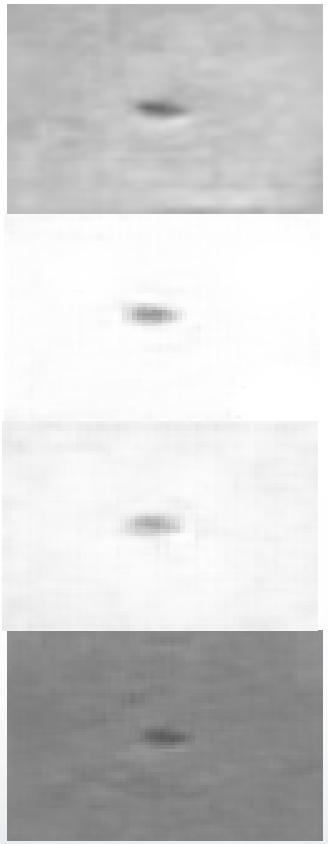
① GAN을 이용한 희소 결함 Class 데이터 증식

결함 원본 이미지와 CVAE-GAN에 의해 생성된 이미지 비교

[Pinhole]

Original

Generated



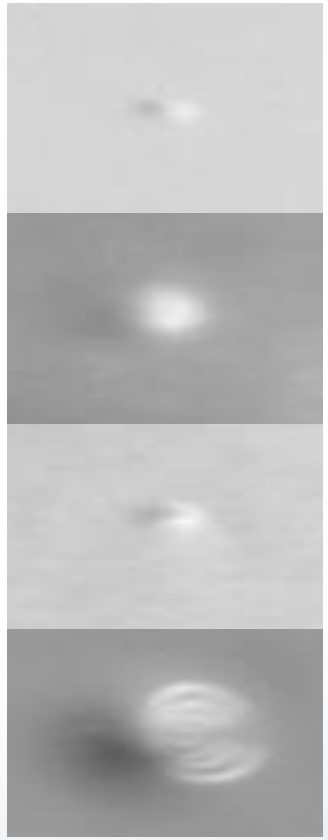
[Crater]

Original

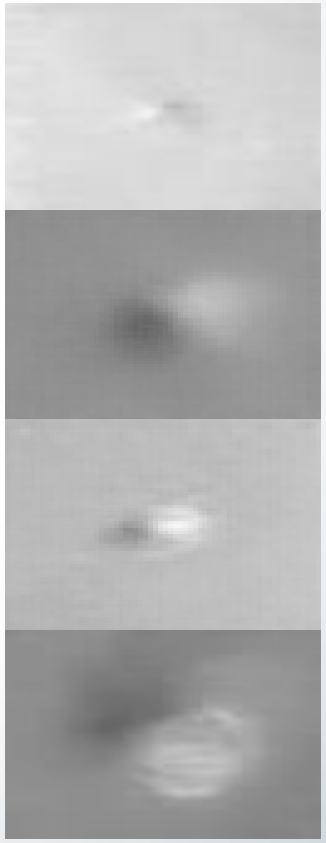
Generated

[Pollution]

Original

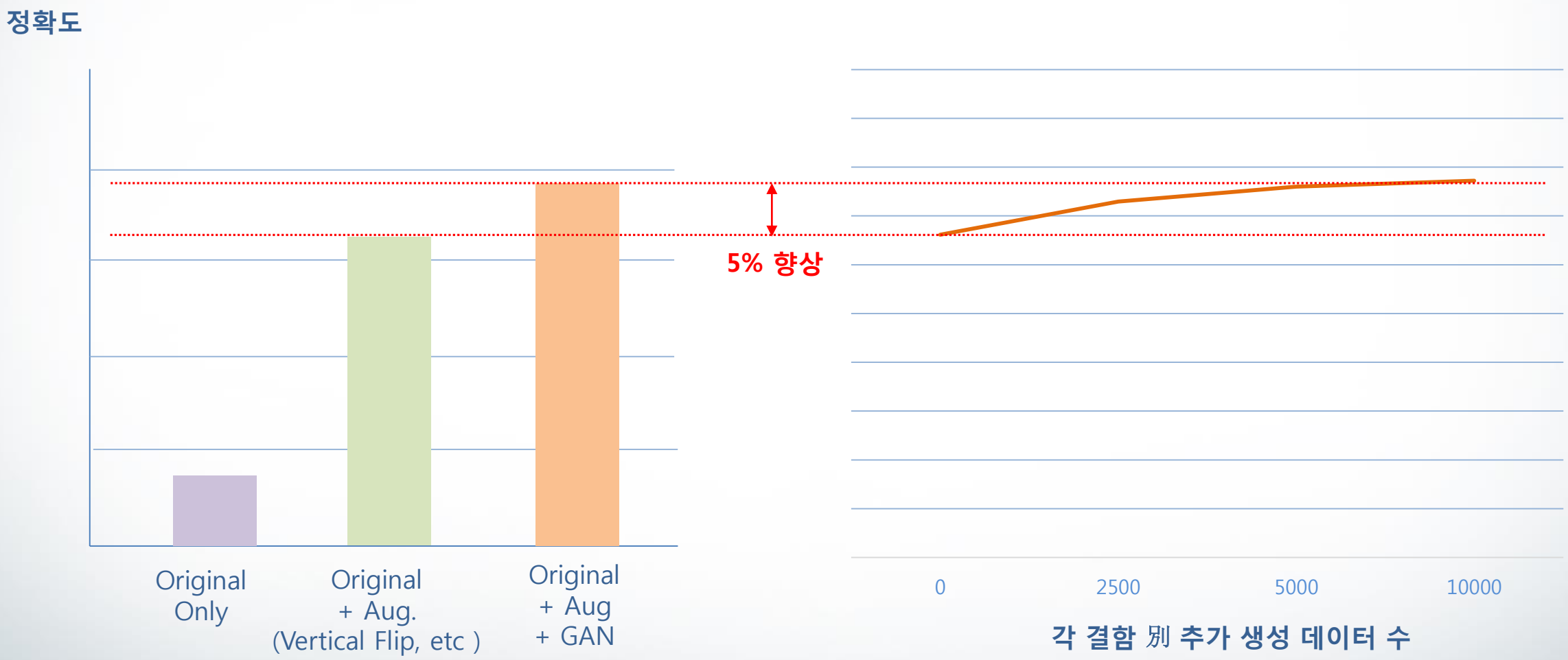


Generated



① GAN 적용 효과

GAN을 이용한 데이터 생성을 통한 학습 효과 : 5%의 분류 정확도 향상을 보임



② 결함 Class의 정의 자체가 모호할 때

Data를 탐색하고 clustering할 수 있는 도구 필요

(유사 이미 검색 동영상 시연)

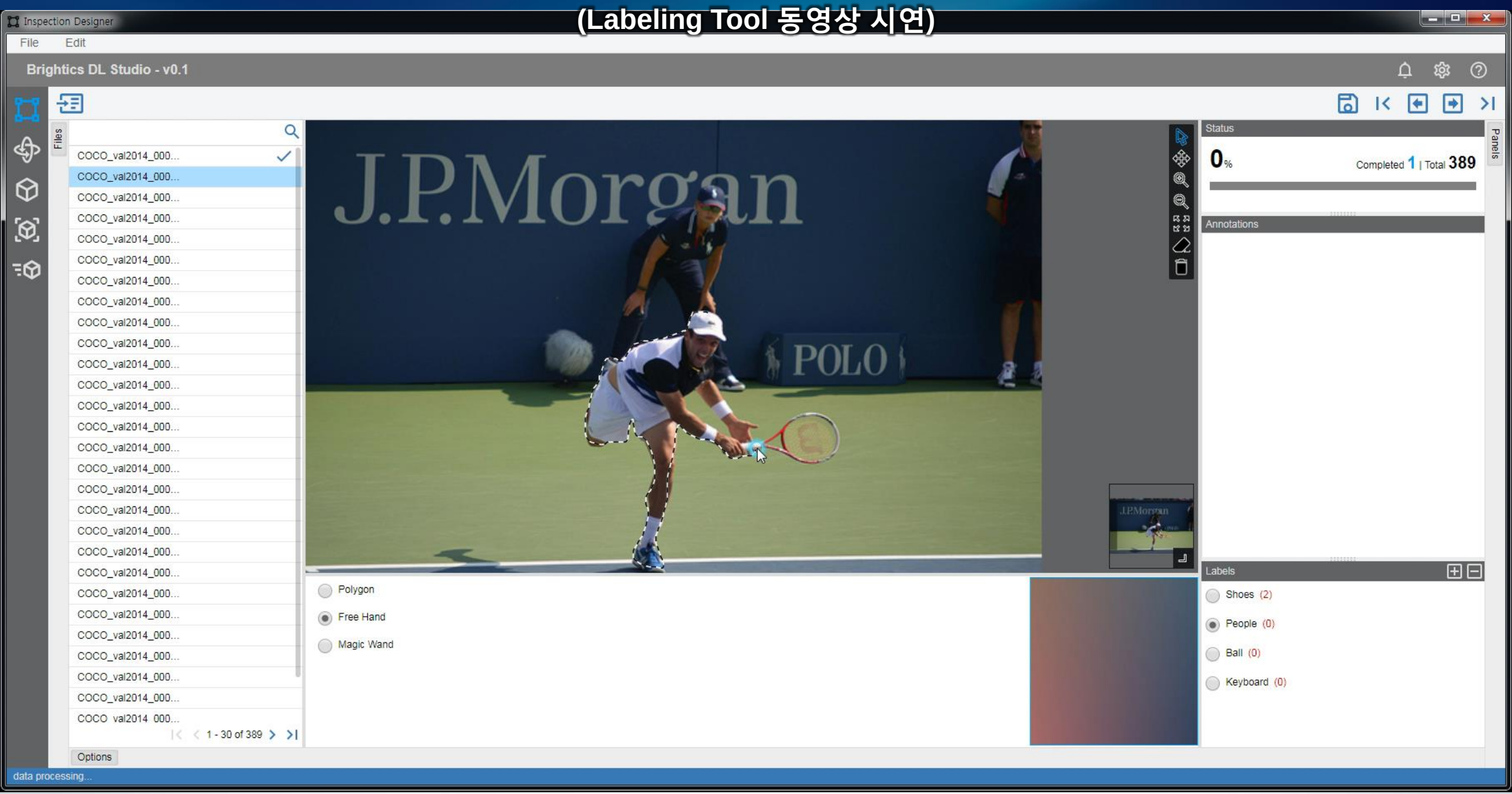
Pdf Retrieval

주의 요함 | 70.70.10.157:8080/pdf/search

쿼리 이미지 (검색 시간: 103.0ms)

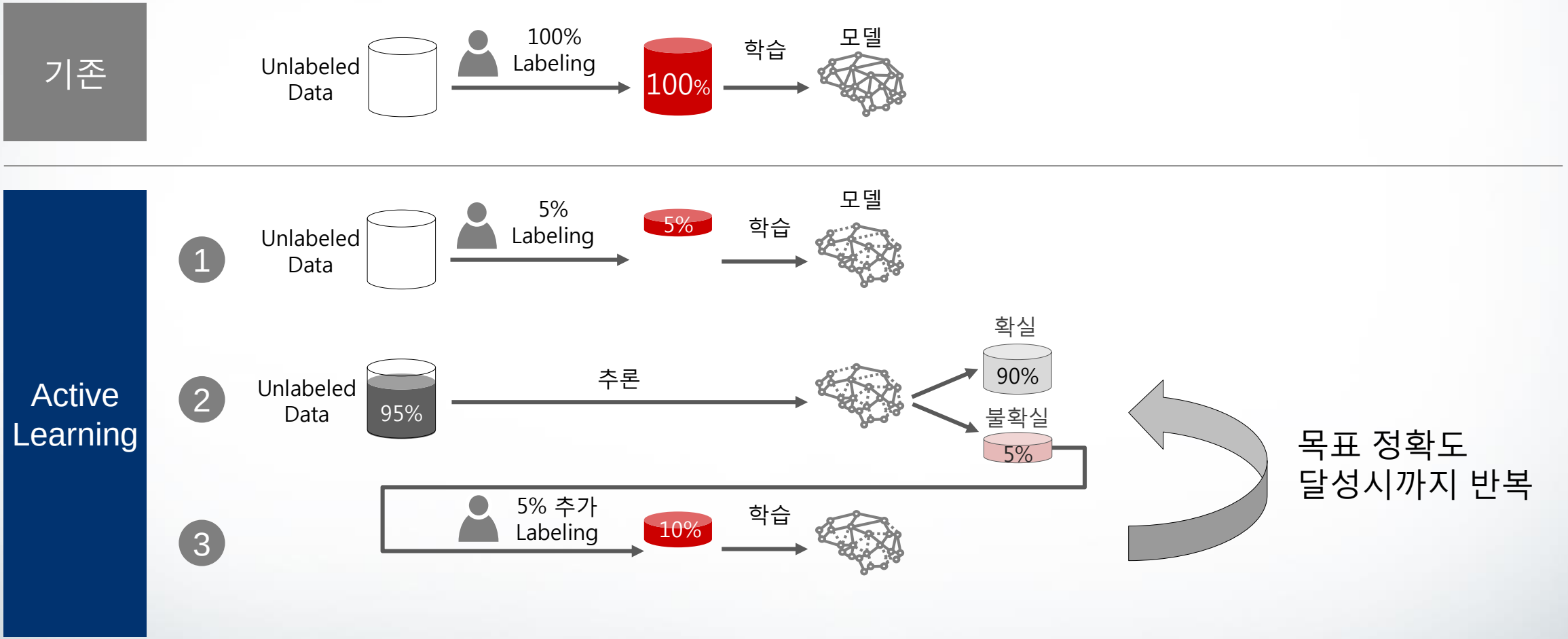
검색 결과

(Labeling Tool 동영상 시연)



③ Active Learning

반쯤 학습한 모델에게 남은 unlabeled data를 추론해보라고 하고, 답이 불확실한 data만 labeling

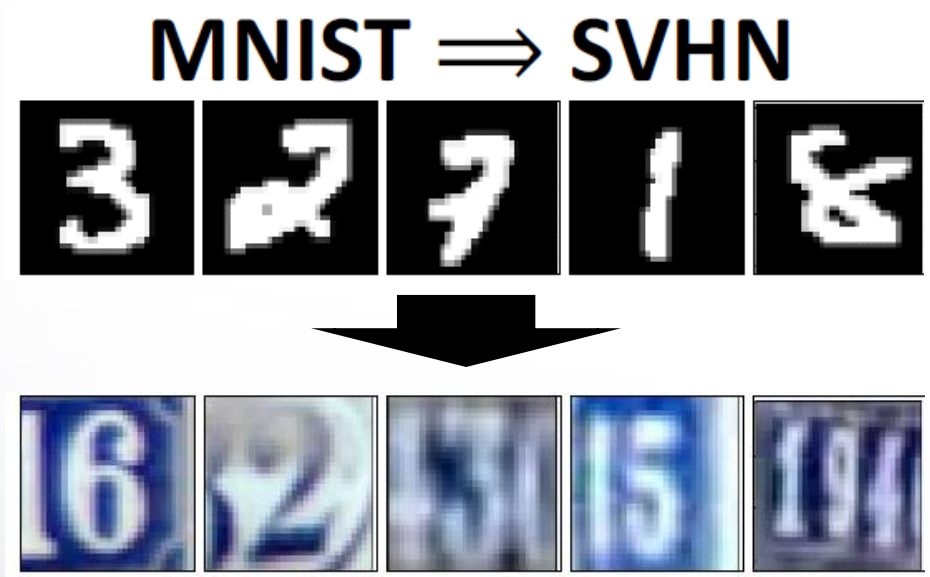


③ Unsupervised Domain Adaptation

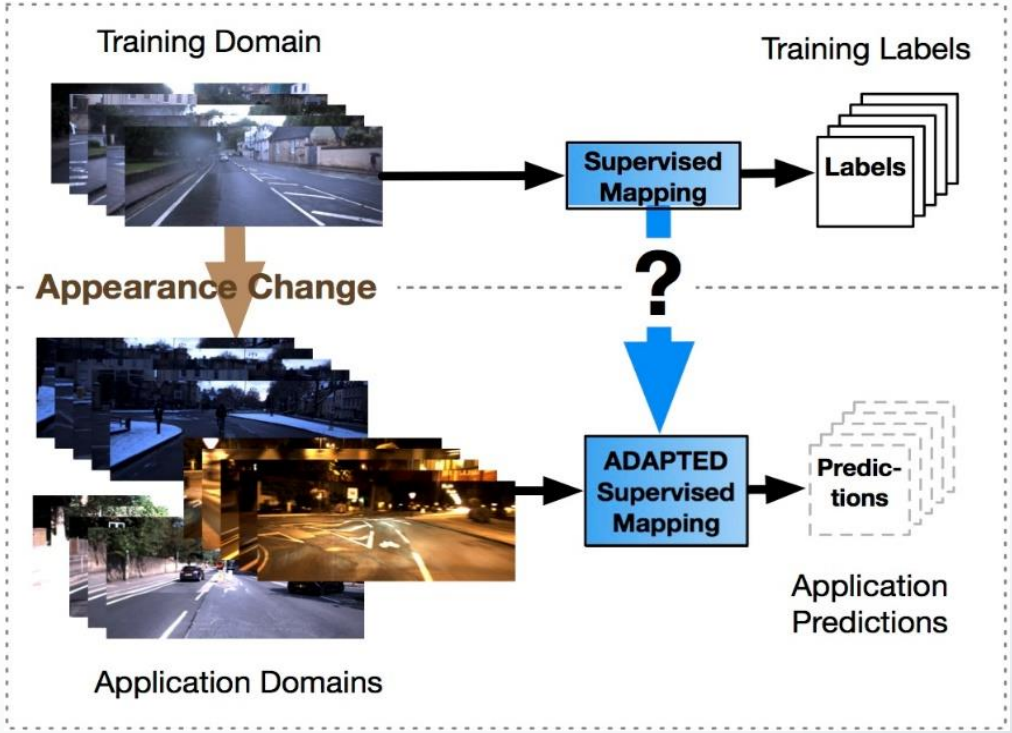
다른 형태/분포의 유사 입력 데이터에 대한 적응력을 학습하는 방법

문제 예시

문자 인식



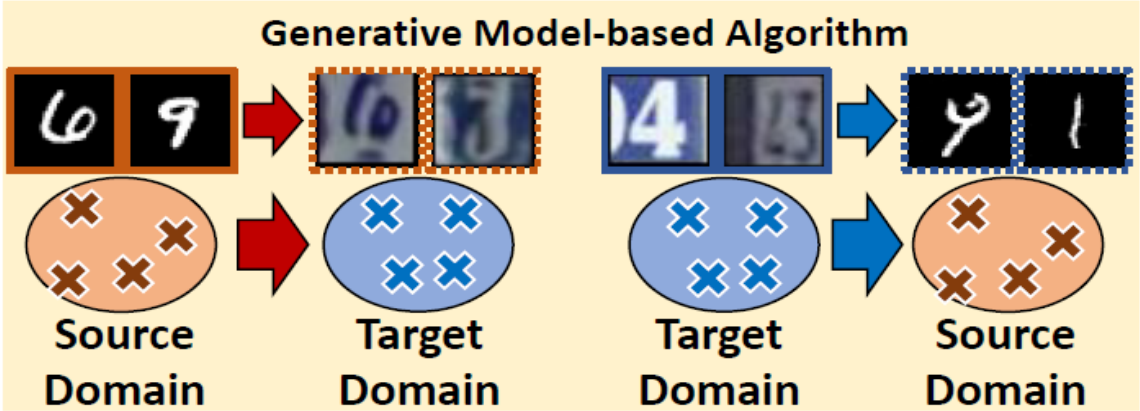
자율 주행



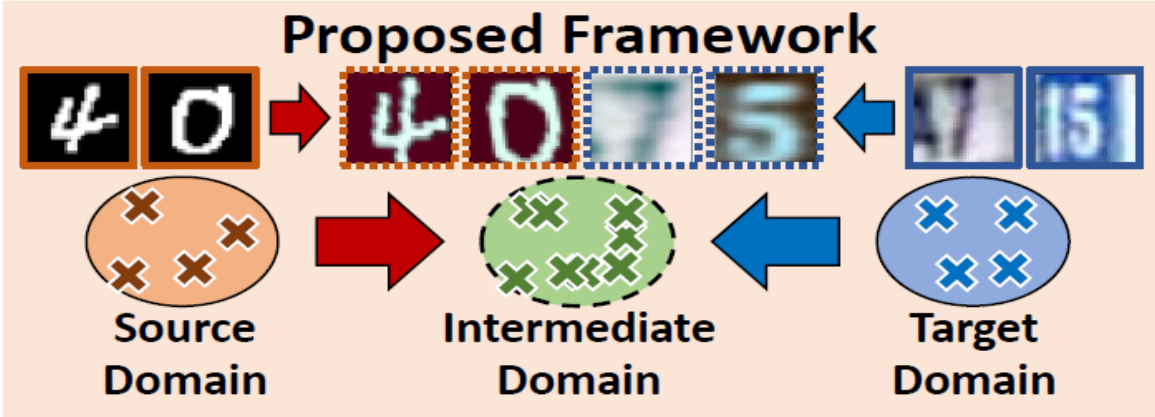
③ Intermediate Domain 기반 Domain Adaptation(DA)

상호 연관성을 가진 중간 형태의 데이터 도메인을 생성하여 적응 학습 과정을 수행

기존



중간 Domain 생성



③ Intermediate Domain 기반 DA 테스트 : 의료 데이터

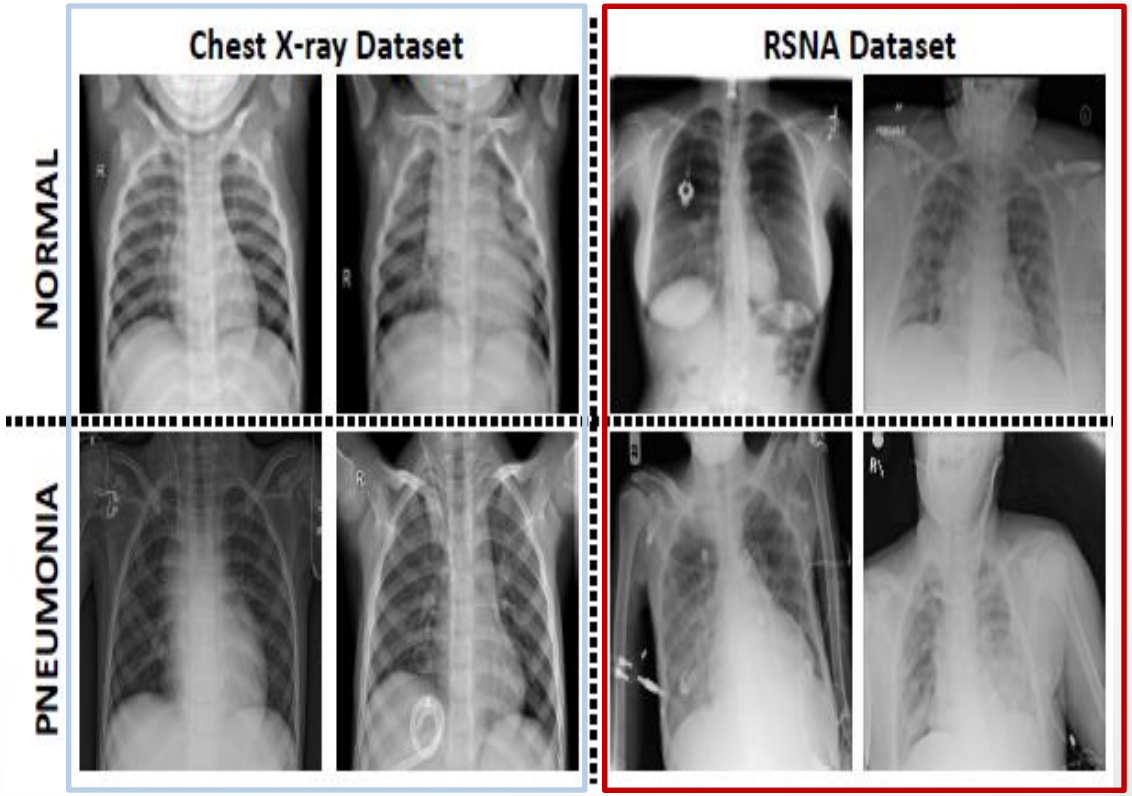
학습 데이터 대비 변형된 신규 데이터에 대한 적응력이 기존 방법 대비 향상

X-ray Pneumonia Recognition

- 학습 데이터 : **Aligned** X-ray Dataset
- 적용 데이터 : **Unaligned** X-ray Dataset

Table 4. Quantitative Results on X-ray Datasets

Algorithm	Source Only	MCDDA [28]	**Ours**
Accuracy	58.5%	67.21%	**72.37%**

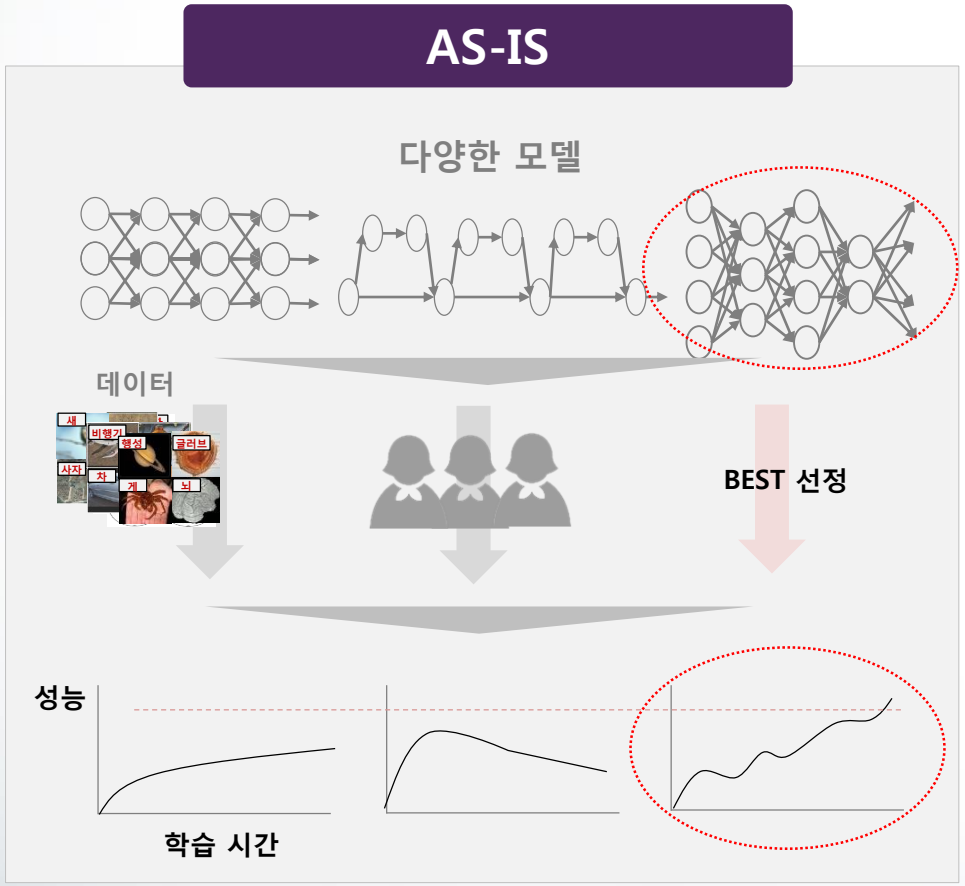


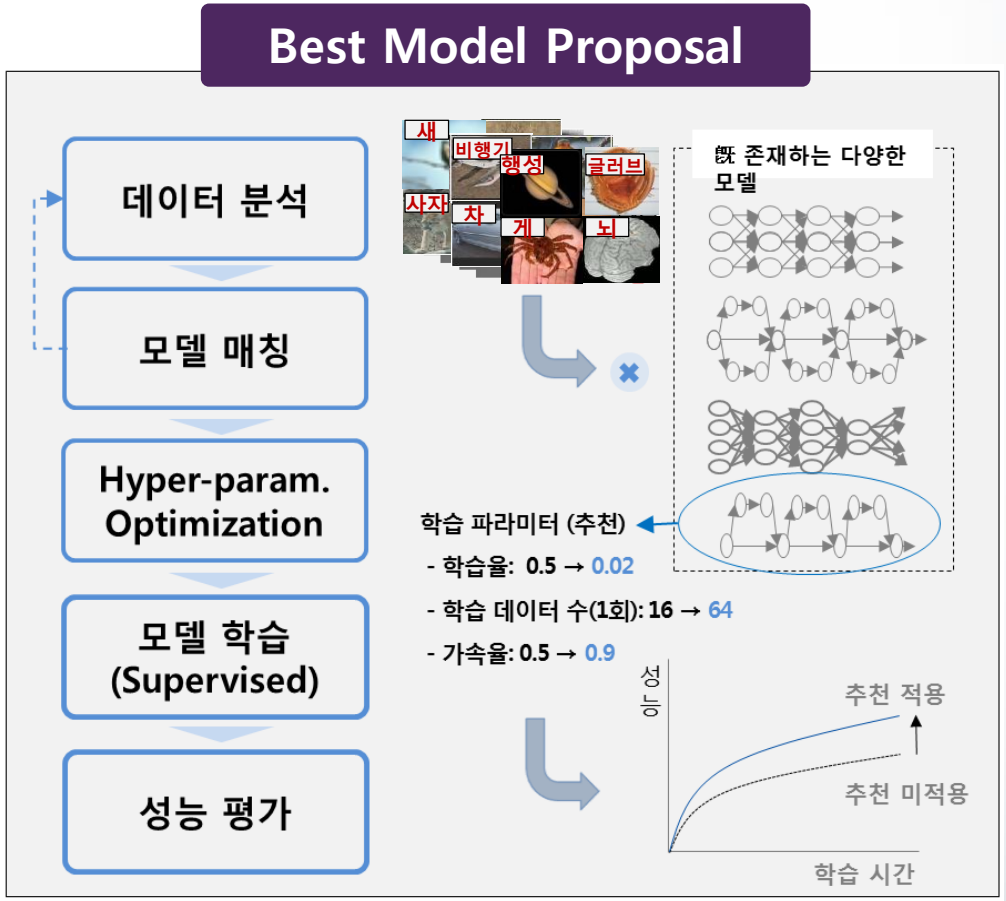
Aligned **Unaligned**

2 모델 선정 시행착오를 줄이면 좋겠다

Pre-trained Model군 중에서 적정한 것을 자동 선정

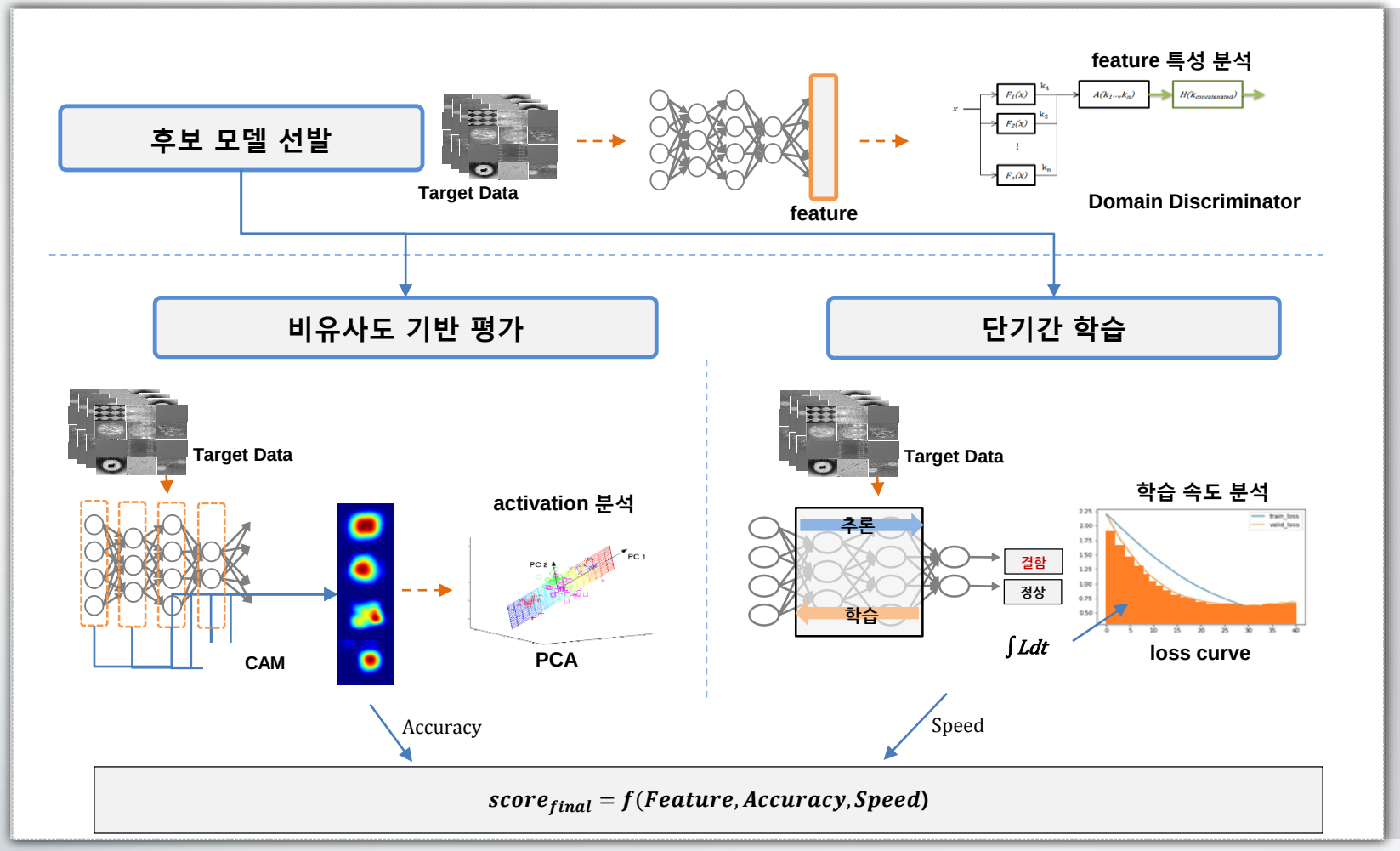
데이터를 빠르고 좋은 성능으로 학습이 가능한 초기(Seed) 모델을 선정하여 학습 효율을 향상





Best Model Proposal 과정

대상 학습 data에 대한 분석으로 1차 후보를 뽑아, CAM 비교와 단기간 학습 loss 평가로 결정



- **후보 모델 선발**
  Domain Discriminator를 활용하여 주어진 데이터의 특성을 분석
  분석된 특성에 기반, 적합한 모델 판단

- **비유사도 기반 평가**
  Activation Map을 통해 각 단계별 Feature를 통한 구분 능력 분석

- **단기간 학습**
  학습 Loss의 변화량 분석

3 GPU들을 공유하면 좋겠다

Kubeflow

Namespace

New Notebook Server

Name

Specify the name of the Notebook Server and the Namespace it will belong to.

Notebook Server's Name

myserver

Namespace

kubeflow

Image

A starter Jupyter Docker Image with a baseline deployment and typical ML packages.

Standard Custom

Image

gcr.io/kubeflow-images-public/tensorflow-1.13.1-notebook-cpu:v0.5.0

CPU

Specify the total amount of CPU reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. `1.5`).

Workspace Volume

Configure the Volume to be mounted as your personal Workspace.

For example, to create an empty Workspace: `New notebook-workspace, 10, /home/jovyan, ReadWriteOnce`

Type	Name	Size (Gi	Mount Path	Access Mode
New	myserver	10	/home/jovyan	ReadWriteOnc

Standard Custom

Image
gcr.io/kubeflow-images-public/tensorflow-1.13.1-notebook-cpu:v0.5.0

CPU

Specify the total amount of CPU reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. `1.5`).

Workspace Volume

Configure the Volume to be mounted as your personal Workspace.

For example, to create an empty Workspace: `New notebook-workspace, 10, /home/jovyan, ReadWriteOnce`

Type	Name	Size (Gi	Mount Path	Access Mode
New	myserver	10	/home/jovyan	ReadWriteOnc

Data Volumes

Configure the Volumes to be mounted as your Datasets.

For example, to create an empty Data Volume: `New, data-volume-1, 5, /home/jovyan/data-volume-1, ReadWriteOnce`

Extra Resources

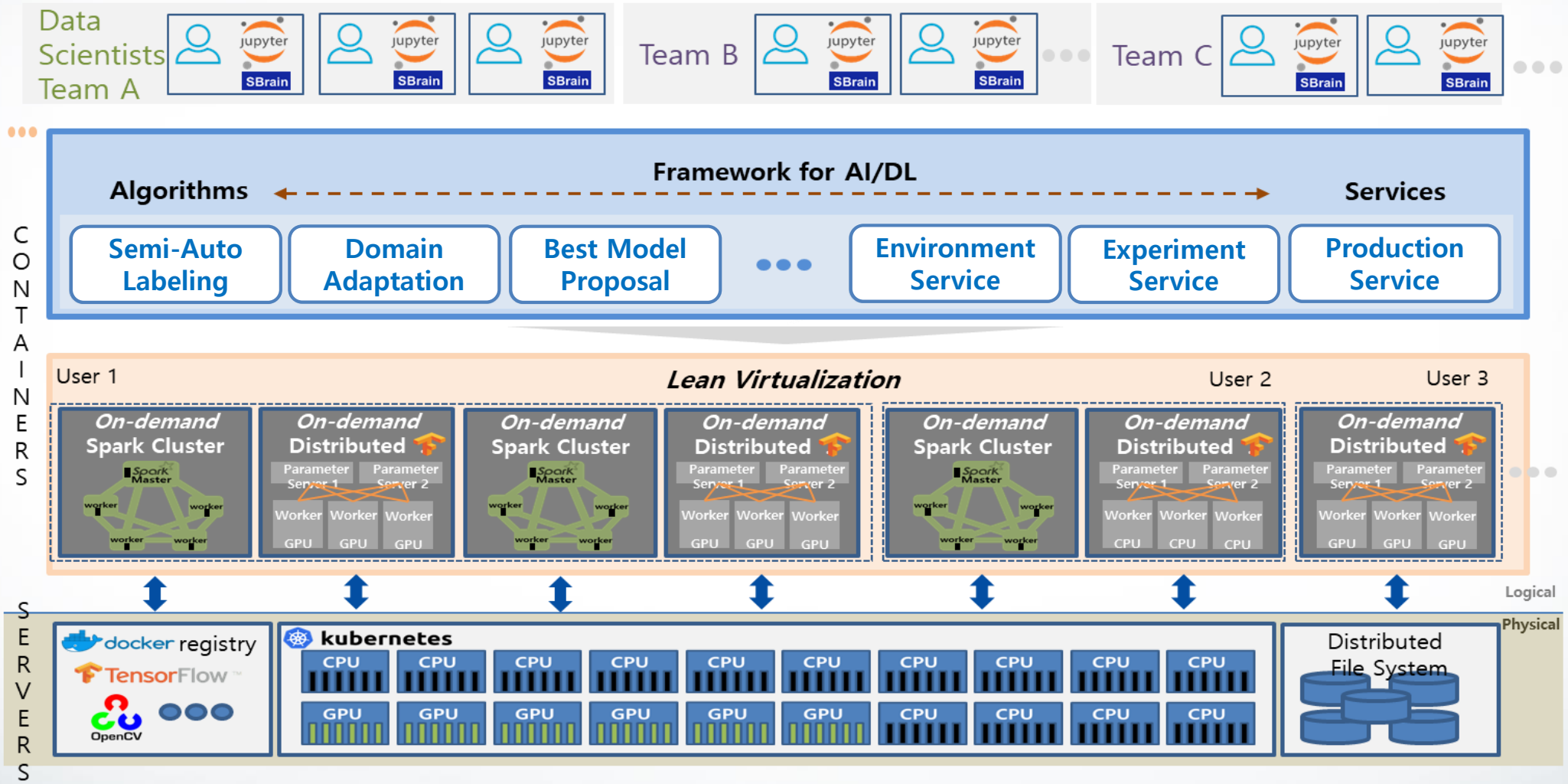
Reserve additional resources.

For example, to reserve 2 GPUs: `{"nvidia.com/gpu": 2}`

Extra Resources
{}

SPAWN CANCEL

AI Platform Architecture



복잡한 분산학습 설정을 한줄로 해결

As-Is 상황

Cluster Provisioning

- Specify CPU, GPU, memory resources of each
- Assign specs for Master, and workers, chief worker
- Use Kubernetes to launch, maintain, and monitor those pods

kubernetes docker APACHE Spark HELM NVIDIA SWAGGER ...

Model Architecture

Define model & input function

TensorFlow

Distributed Training Prep

- Set the Parameters Server, Master, and chief worker (to coordinate the asynchronous training across workers)
- Set where and how to store the logs to view in TensorBoard

TensorFlow

kubernetes

Training

Start the model training

TensorFlow

Distributed software, DevOps,..

Data scientists

복잡한 분산학습 설정을 한줄로 해결

To-Be 상황

Model Architecture	Define model & input function
Training	Start the model training

TensorFlow

```
Experiment.run(
  runconfig=RunConfig(no_of_workers=4))
```

Data scientists

8. 실험 수행 관련 Configuration 설정

클러스터 내에서 분산 학습을 수행하기 위해 필요한 configuration을 설정하는 것으로, 분산 학습의 조정과 weight값을 담당하는 parameter 서버의 개수, 실제 학습을 담당하는 gpu worker 서버의 개수와 tensorboard 상에 metrics 주기, training과 동일한 evaluation 수행 여부 등에 대해 값을 정하는 단계입니다.

- **no_of_ps** : 최소 1이상이 되어야 하며, parameter 서버의 개수를 입력합니다.
- **no_of_workers** : 학습에 사용되는 서버의 수를 입력합니다.
- **summary_save_frequency** : tensorboard 상의 metrics 주기를 입력합니다.
- **run_eval** : True로 할 경우, tensorboard 상에서 evaluation graph는 training graph와 대응되게 그려집니다.
- **use_gpu** : True로 할 경우에만 학습에 사용되는 서버가 GPU로 할당되며, 나머지의 경우 CPU를 사용하게 됩니다.
- **checkpoint_frequency_in_steps** : 체크포인트가 저장되는 주기를 입력합니다.

IMPORTANT

early stop 기능 사용시, run_eval이 반드시 True가 되어야 합니다.

In [13]:
```
run_config = RunConfig(no_of_ps=1, no_of_workers=2, summary_save_frequency=5, run_eval=True, use_gpu=True, chec
```

9. Experiment 수행

unique한 실험명을 선언한 후, 실제 학습을 수행합니다.

- **experiment_name** : 향후 deploy 등을 위한 model과 job에 대한 이름
- **description** : model과 job에 대한 Description
- **estimator** : saved estimator로 이전 cell에서 저장한 estimator

9. Experiment 수행

unique한 실험명을 선언한 후, 실제 학습을 수행합니다.

- **experiment_name** : 향후 deploy 등을 위한 model과 job에 대한 이름
- **description** : model과 job에 대한 Description
- **estimator** : saved estimator로 이전 cell에서 저장한 estimator
- **hyper_parameters** : 특정 hyperparamter 값을 사용시(선택 1)
- **hparams_search_settings** : grid 방식으로 여러 hyperparamter 값들의 조합 사용시(선택 2)
- **run_config** : 실험 수행에 관련된 confiuration 설정
- **input_function** : 정의된 input_function

[주의사항]

- 앞서 작성된 input_function에 error가 있을 경우 실행이 되지 않을 수 있습니다.

In [14]:
```
experiment_name = uniquify("Resnet_Flower102_model")
```

In [15]:
```
experiment = Experiment.run(experiment_name=experiment_name,
                            description="ResNet Model trained on Flower data",
                            estimator=saved_estimator,
                            hyper_parameters=hyper_parameters,
#                           hparams_search_settings=grid,
                            run_config=run_config,
                            dataset_version_split=None,
                            input_function=input_function)
```

10.2.1 실험 정보 및 완료 여부 확인 ¶

In [19]:
```
experiment.report_status()
experiment.list_jobs()
experiment.wait_until_finish()
```

```
|                |          | 0.01, 'iter |              |
|                |          | ations':    |              |
|                |          | 3000, 'learn |             |
|                |          | ing_rate':  |              |
|                |          | 0.01, 'weigh |             |
|                |          | t_decay':   |              |
|                |          | 0.0002, 'bat |             |
|                |          | ch_size':   |              |
|                |          | 100, 'epoch_ |             |
|                |          | staged_lr':  |              |
|                |          | [100, 200], |              |
|                |          | 'batch_norm_ |             |
|                |          | epsilon':   |              |
|                |          | 1e-05}      |              |

Waitin
g.....................................................................................

Completed..
```

10.2.2 job 완료 및 결과 확인

job이 완료된 다음(experiment.has_finished()이 True 인 경우) 학습이 성공/실패 여부를 확인할 수 있습니다.

```
Completed..
```

10.2.2 job 완료 및 결과 확인

job이 완료된 다음(experiment.has_finished()이 True 인 경우) 학습이 성공/실패 여부를 확인할 수 있습니다.
early stop을 사용한 경우 선정한 iterations 보다 적은 학습횟수로 완료된 것을 확인할 수 있습니다.

In [20]:
```
experiment.has_finished()
```

Out[20]:
```
True
```

In [21]:
```
experiment.list_jobs_with_metrics()
```

```
Learning Job Id |              Job Name              |          Eval Metrics
================+====================================+=============================
56              | Resnet_Flower102_model_BDL_okj     | {"accuracy":
                | 1561939964101456                   | 0.6483516693115234, "loss":
                |                                    | 1.694976925849914 6,
                |                                    | "global_step": 3000}
```

eval metrics으로 실험에 대한 learning jobs를 리스트 형태로 확인할 수 있습니다.
NOTE : 학습이 시작된 후 evaluation이 시작된 다음, metrics 정보가 표시됩니다.

SAMSUNG SDS Brightics AI Deep Learning

Input Function

- Input Functions
- Augmenter

Model Function

- Model Functions
- Network
- Train Operations

Training

- Training Jobs
- Models

Inference

- Endpoints
- Simulation

Training Jobs > Create Training Job

Cancel | Create

Model Functions

Basic Settings | User-Defined Function

Model Function For: Image/Classification | Load Settings | Show Advanced Options

Network *

Inception ResNet V2

Inception ResNet V2 Network

image + 3

Reset | Change

Number Of Classes *

1000

Number Of Classes

Fine Tune

Fine Tune

Trainable Scope

InceptionResnetV2/Logits,InceptionResnetV2/AuxLogits

Scopes to be trained

Loss *

※ 지속적으로 개선중인 화면임

SAMSUNG SDS Brightics AI Deep Learning

Input Function

- Input Functions
- Augmenter

Model Function

- Model Functions
- Network
- Train Operations

Training

- Training Jobs
- Models

Inference

- Endpoints
- Simulation

Training Jobs > Create Training Job

Cancel Create

Run Configurations

Run Configuration For: Run Configuration - Sbrain

Iteration *

2000

Iteration description

Number of Parameter Servers *

1

Number of parameter servers description.

Number of Workers *

1

Number of workers description.

Summary Save Frequency In Step.

50

Summary Save Frequency description

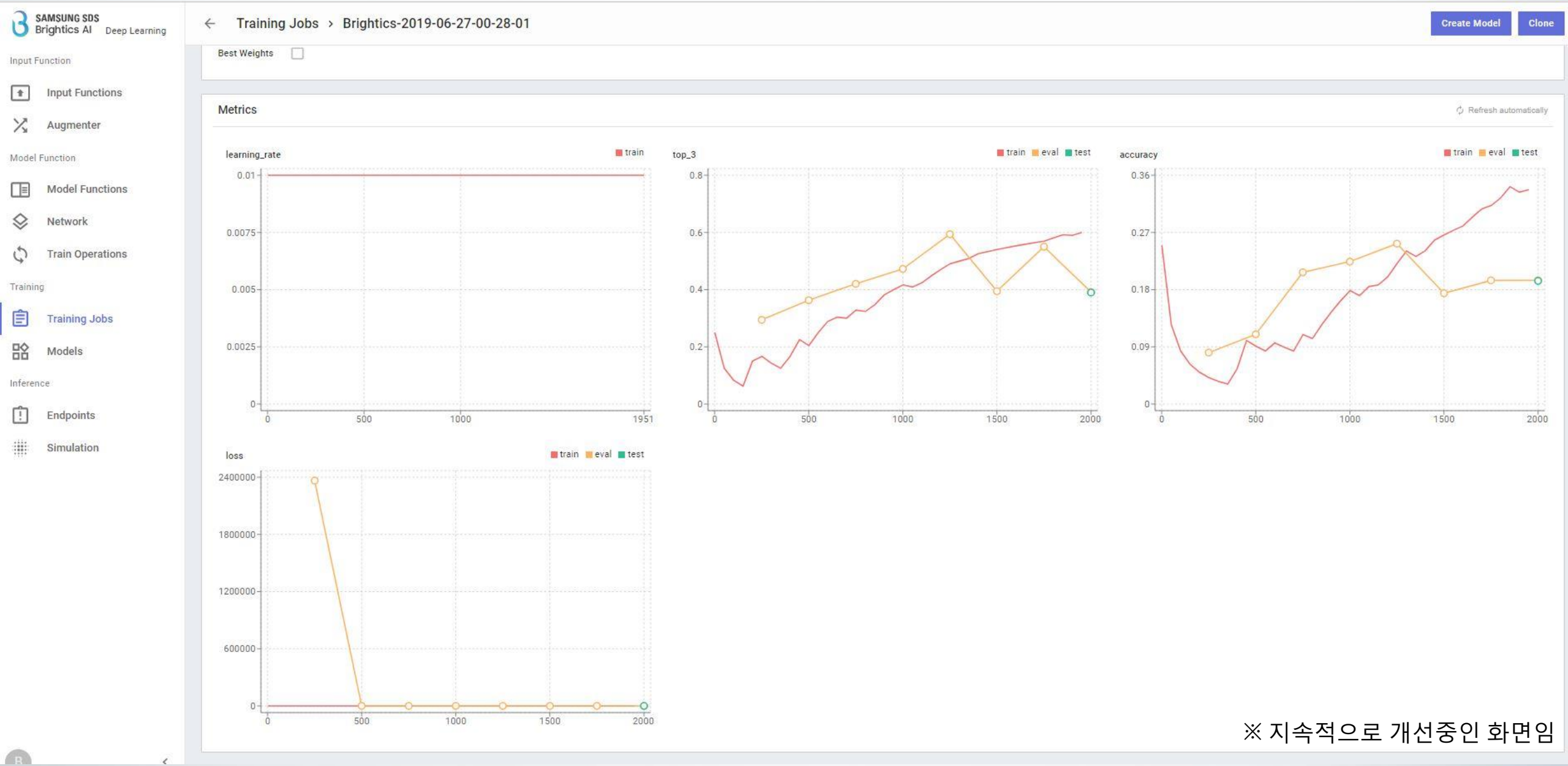
Checkpoint Frequency in Steps.

1000

Checkpoint frequency in steps description

Use GPU * ☑

※ 지속적으로 개선중인 화면임



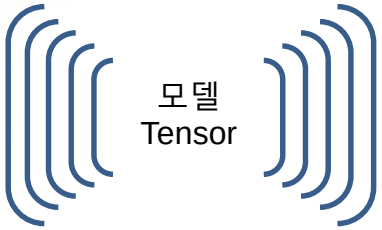
※ 지속적으로 개선중인 화면임

4 추론 가속 (on Edge)

TensorRT

추론 성능 2배 이상 개선, GPU 메모리 사용율 50%이상 절감 가능

Tensorflow, Pytorch 등
개발/학습된 딥러닝 모델



추론 가속기
(Inference Acceleration Algorithm)

모델구조 변경 없이 간소화 수행

Classification/Detection 성능개선
→ 속도 2배 이상 개선,
GPU 메모리 50%이상 절감

[AS-IS 문제점]

"느린 추론 속도"

- 대부분 모델에서 동일 문제 존재
- 원인
  - 최적화 부족
  - 다중 중첩(Wrapper) 구조
  - Tensorflow → C 언어 변환 문제
  - CPU ↔ GPU 병목
  - 메모리 비효율적 사용
  - Tensor(다차원 행렬) 비효율 연산
  - GPU/메모리 간접 핸들링

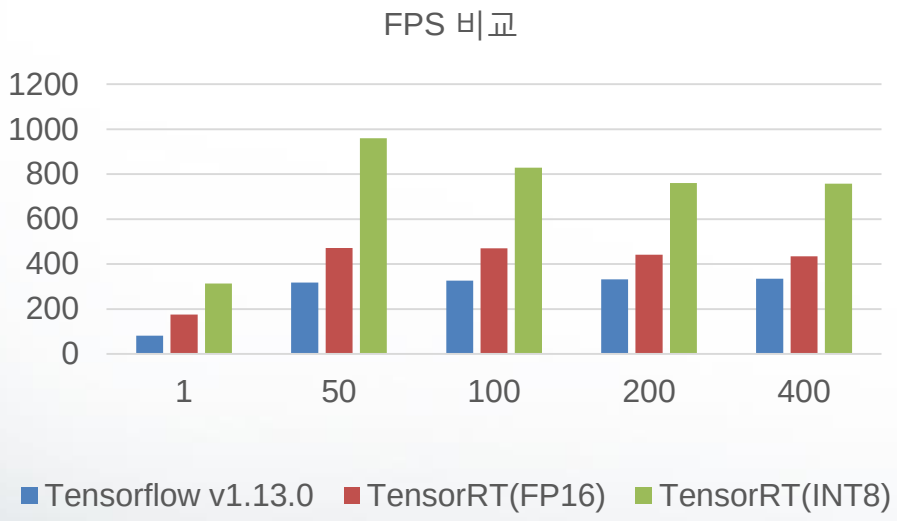
정밀도 간소화 Float32 → Int8	• 정확도 유지하고 연산은 간소화 느린 부동소수 연산을 INT로 양자화하여 처리량 극대화
Tensor Fusion GPU 대역폭 최적화	• GPU ↔ GPU 메모리 간 대역폭 최적화, 병목 개선 GPU 커널 노드를 융합, 직접 Handling
Dynamic Tensor Memory 메모리 사용 효율화	• 메모리 사용량 최적화(적게 사용) 텐서(L x M x N, 3차원 행렬)을 위한 메모리 공간 배정
Parallel-Stream Processing 스트림 병렬 처리	• 기존 단일 입력 → 병렬입력 구조로 개선, 속도개선 단일 입력 스트림을 병렬 입력 스트림으로 변환
Low-rank approximation 행렬 연산 차원 축소	• 행렬 연산 간소화, 고속화 행렬연산 시 동일한 차수 행렬부터 수행함으로써 행렬 연산 간소화
가지치기 알고리즘 불필요한 신경망 가중치 제거	• 중요하지 않은 가중치와 노드를 제거, 탐색속도 개선 불필요한 신경노드 탐색 최소화

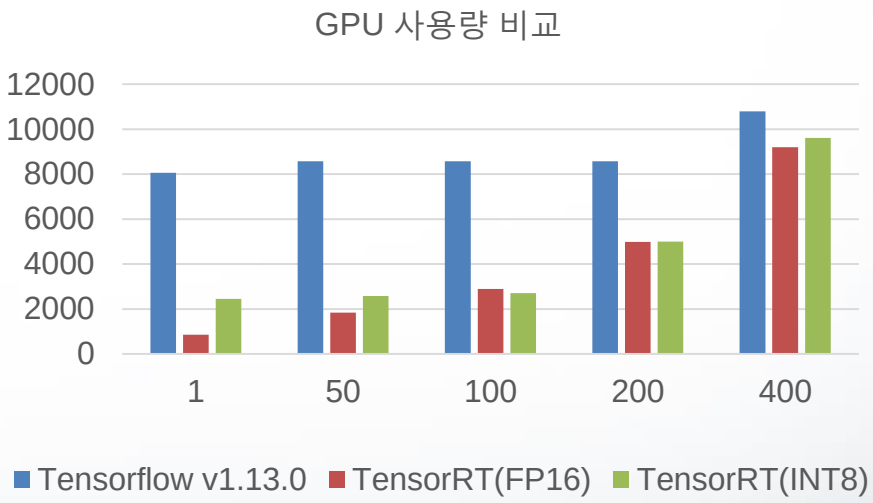
TensorRT

Resnet101 FPS 및 GPU 메모리 사용량 비교

※ NVIDIA Korea 최적화 미적용 수치

	Tensorflow v1.13.0		TensorRT FP16		TensorRT INT8	
BATCH	**FPS**	**GPU(MB)**	**FPS**	**GPU(MB)**	**FPS**	**GPU(MB)**
1	82	8061	175	858	313	2448
50	317	8573	471	1834	960	2574
100	326	8575	470	2886	829	2702
200	332	8573	441	4986	761	5006
400	335	10793	435	9204	758	9614



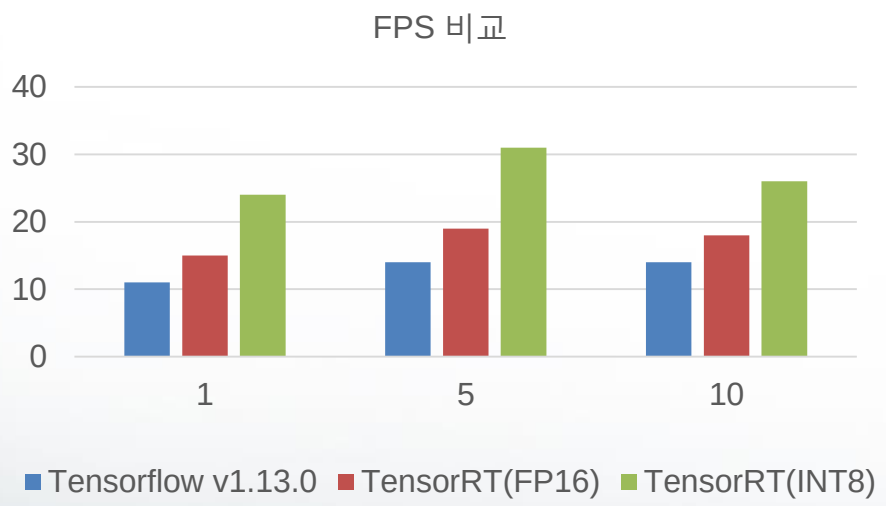


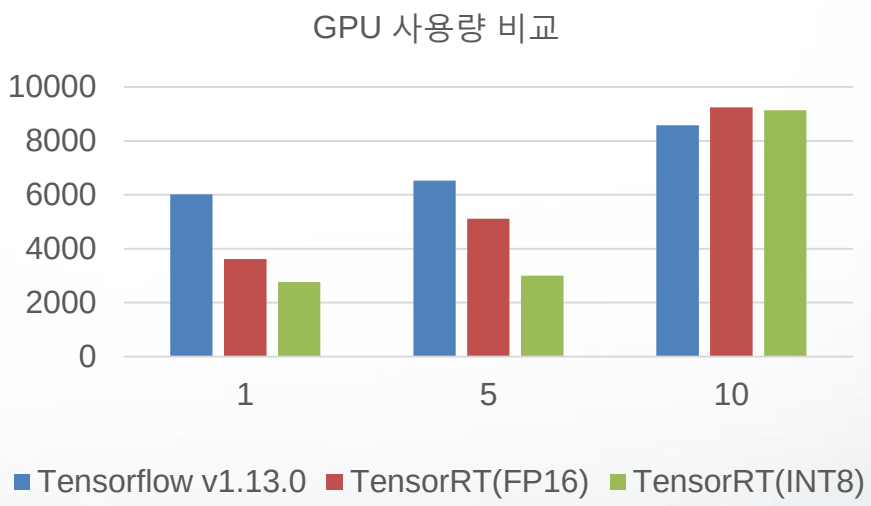
TensorRT

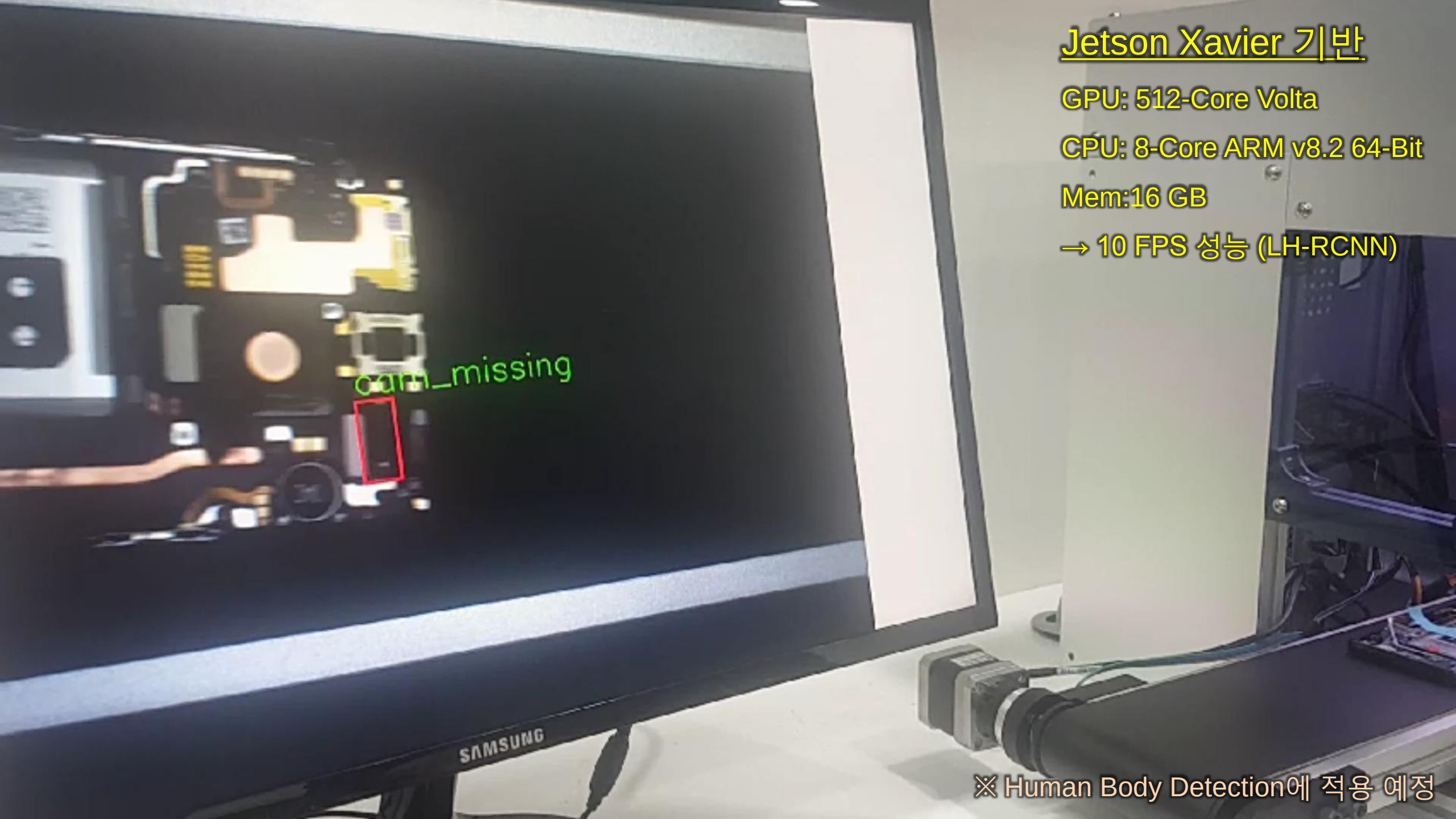
Light-Head R-CNN FPS 및 GPU 메모리 사용량 비교

※ NVIDIA Korea 최적화 미적용 수치

	Tensorflow v1.13.0		TensorRT(FP16)		TensorRT(INT8)	
BATCH	**FPS**	**GPU(MB)**	**FPS**	**GPU(MB)**	**FPS**	**GPU(MB)**
1	11	6015	15	3623	24	2759
5	14	6527	19	5111	31	2999
10	14	8575	18	9246	26	9143







Jetson Xavier 기반

GPU: 512-Core Volta

CPU: 8-Core ARM v8.2 64-Bit

Mem:16 GB

→ 10 FPS 성능 (LH-RCNN)

※ Human Body Detection에 적용 예정

Q & A

발표를 도와주신 분들께 감사드립니다!

NVIDIA Korea 정소영 상무, 이종환 과장

삼성SDS 민승재 마스터님과 조성호, 이재영, 최종원, 오경진, 이상학, 김종성, 박재선, 이권호, 황봉규 프로님

오래 기다려주신 Waywai 팀

문의처 : brighticsdl@samsung.com