

# 2022 Samsung AI Challenge (3D Metrology)

Team 민초맛대흥군

# Contents

## 01 문제 설정 및 접근

- a. 데이터 분석
- b.  $\text{Depth}^S = \text{Depth}^T?$

## 02 전략 및 알고리즘

Architecture overview

- a. Domain Adaptation
- b. Case Classifier
- c. KNN with Cosine-similarity

## 03 제안하는 알고리즘의 차별점

- a. Why cycleGAN + KNN?

## 04 추가 성능 개선 방안

## 05 결론 및 건의사항

# 1. 문제 설정 및 접근 요약

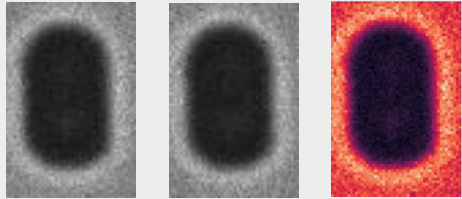
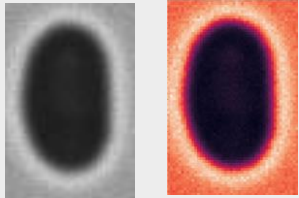
---

- Simulator\_SEM과 Train\_SEM을 각각  $SEM^S, SEM^T$
- $SEM^S, SEM^T$ 에 대응하는 depth map을 각각  $Depth^S, Depth^T$
- $f_s: Depth^S \rightarrow SEM^S$  (Simulator를 통한 Data 생성)
- $f_T: Depth^T \rightarrow SEM^T$  (depth map을 SEM으로 촬영)

가설)  $Depth^S = Depth^T$ 이지만,  $f_s \neq f_T$  이기 때문에  $SEM^S \neq SEM^T$

=> **Unsupervised, Homogenous Domain Adaptation** 문제로 설정하고 접근

# 1-a. 데이터 분석 - $SEM^S$ vs $SEM^T$ : 원본 이미지 + `seaborn.heatmap` 활용 육안 분석

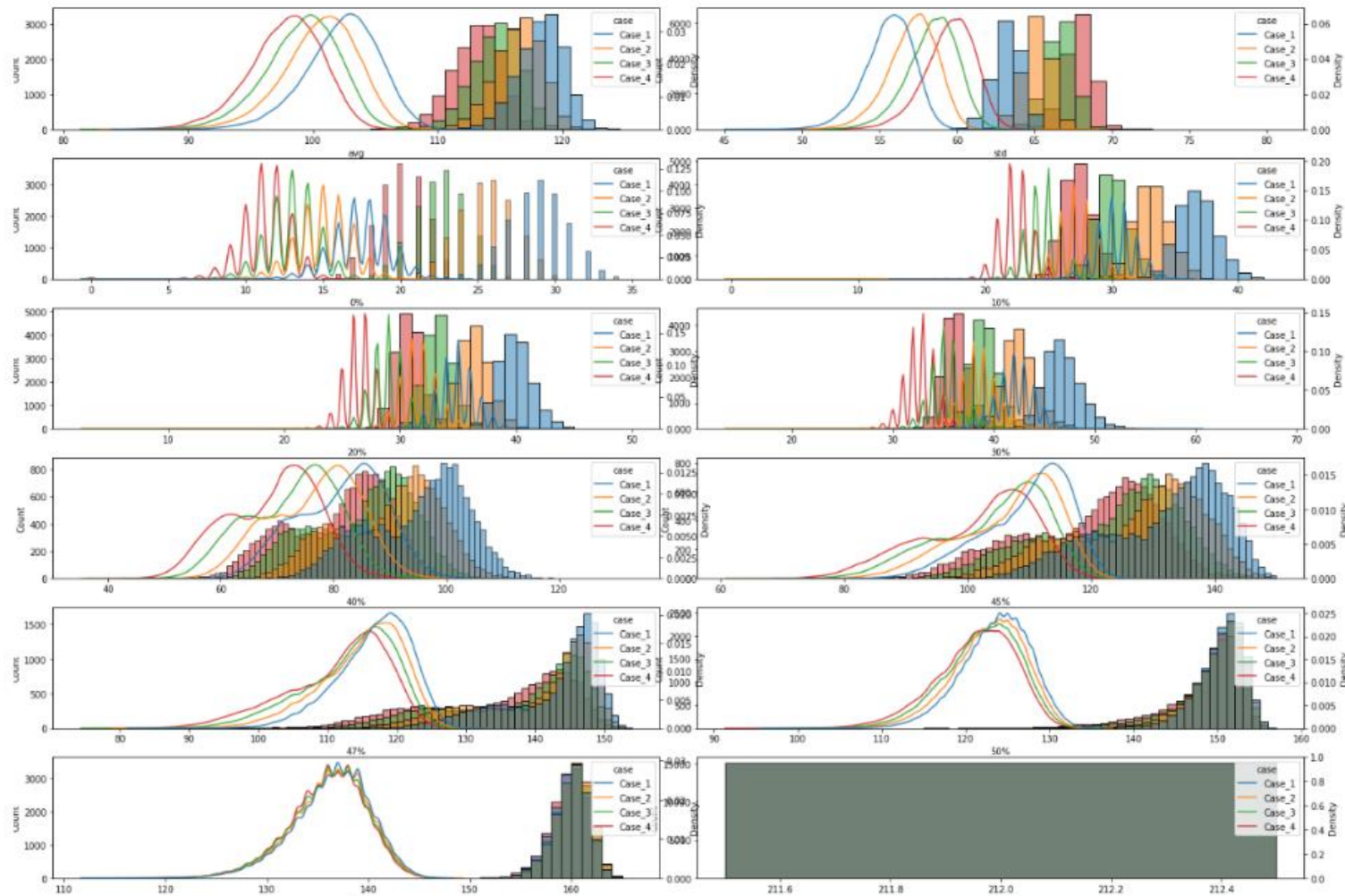
	Simulator_SEM, $SEM^S$	Train_SEM, $SEM^T$
Example		
Noise	itr0, itr1은 random noise 제외 동일, 비교적 <b>강한 noise</b>	비교적 <b>약한 noise</b>
Paired data	하나의 depth map과 두 SEM(itr0, itr1) 매치	각 site내 모든 hole의 depth avg, <b>No paired image</b>
Case	4개의 case로 구성(Case_1~4 & depth_110~140)	
Max value	212 => 항상 Hole 외곽 빛번짐 중 존재	

# 1-a. 데이터 분석 - $SEM^S$ vs $SEM^T$ : Percentile distribution 비교

데이터셋의 특성 - 흑백 이미지, 비교적 단순한 형태

=> 각 image pixel값의 **percentile distribution** 비교가  $SEM^S$ ,  $SEM^T$  두 domain 간 차이 이해에 도움이 된다고 판단

Kdeplot:  $SEM^S$   
Histplot:  $SEM^T$

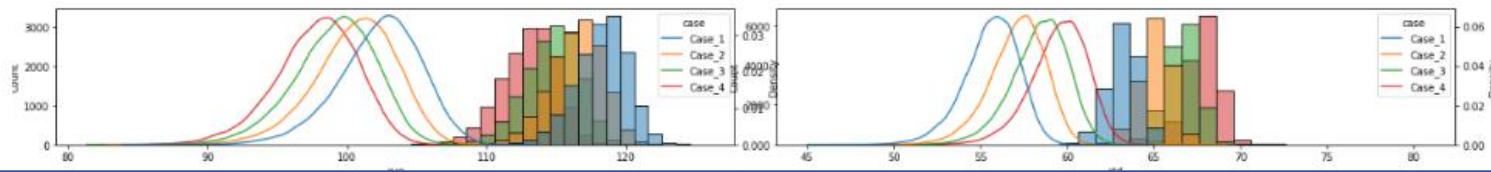


# 1-a. 데이터 분석 - $SEM^S$ vs $SEM^T$ :Percentile distribution 비교

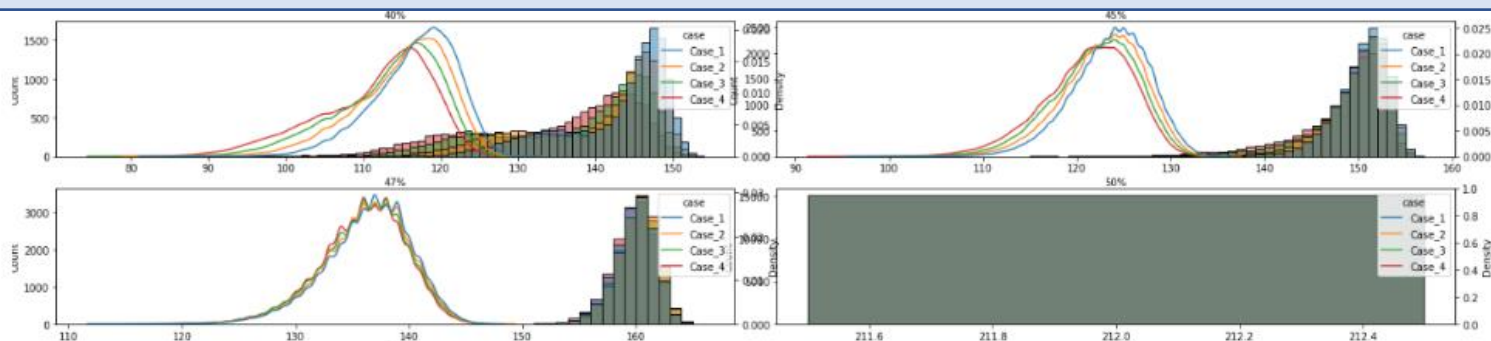
데이터셋의 특성 - 흑백 이미지, 비교적 단순한 형태

=> 각 image pixel값의 **percentile distribution** 비교가  $SEM^S$ ,  $SEM^T$  두 domain 간 차이 이해에 도움이 된다고 판단

Kdeplot:  $SEM^S$   
Hist:  $SEM^T$

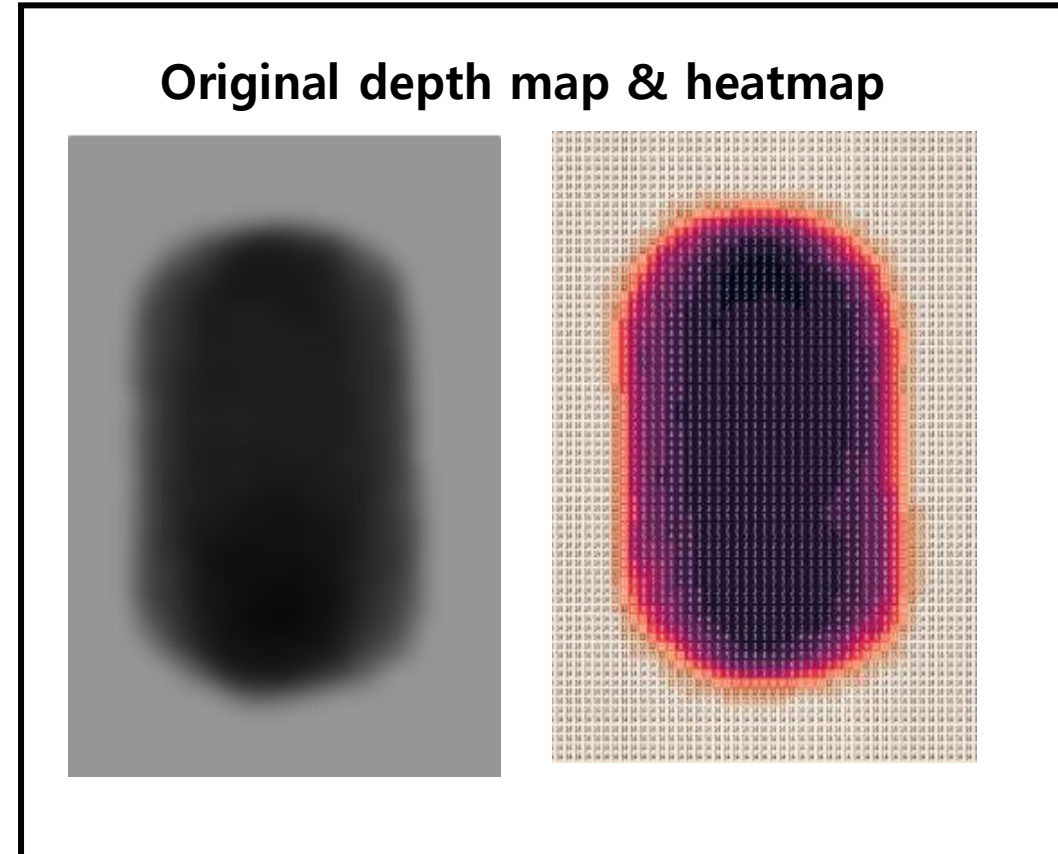
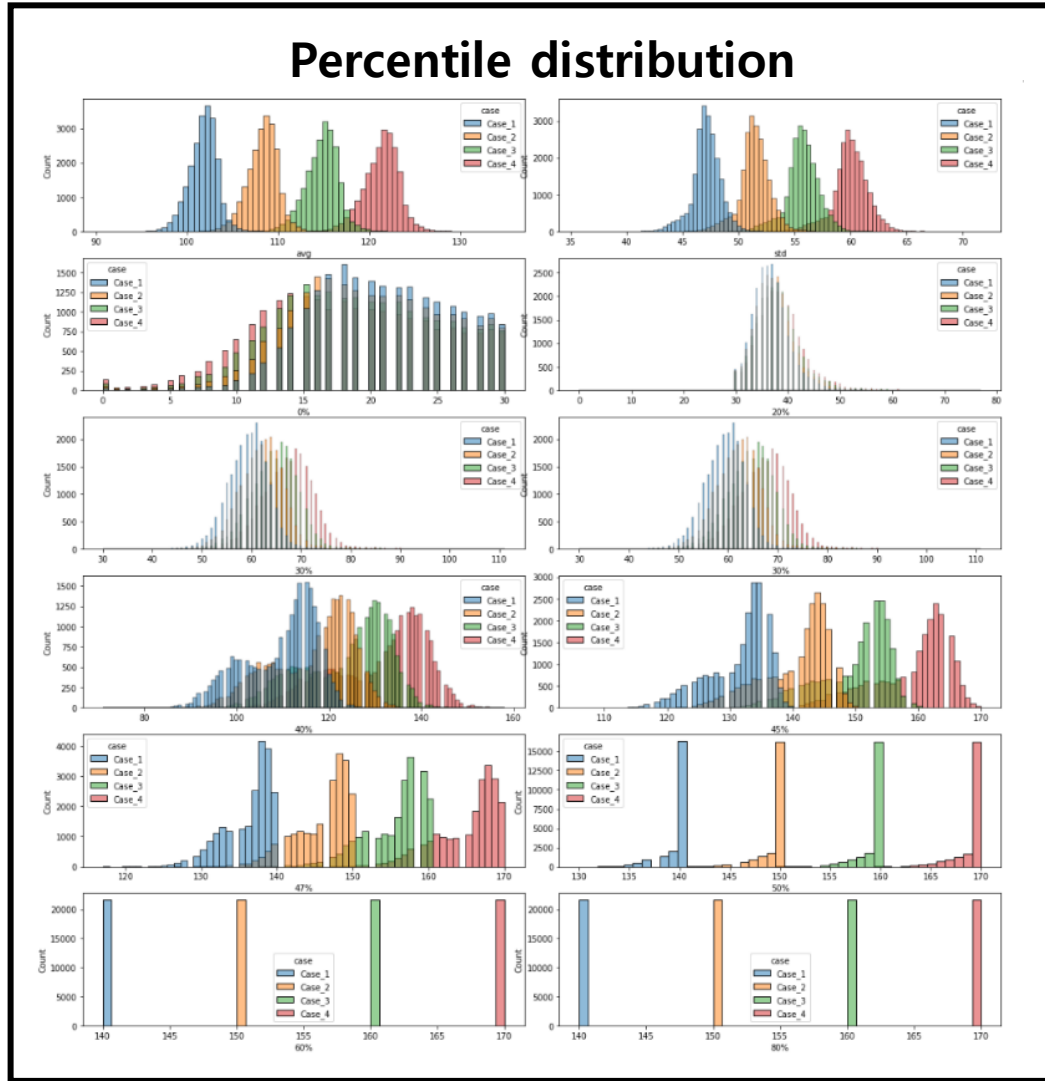


두 domain의 각 percentile 그래프의 형태가 **매우 유사**,  
전체적인 **분포에만 차이가 있음**



# 1-a. 데이터 분석 – Simulator depth map

- SEM분석과 마찬가지로 원본 이미지 및 heatmap, percentile distribution 을 통한 분석을 진행함



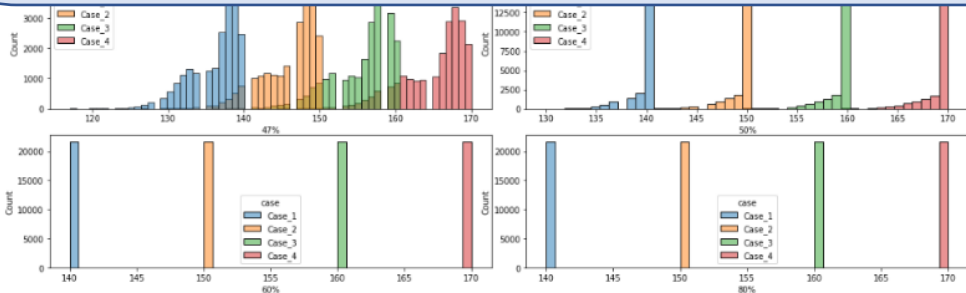
# 1-a. 데이터 분석 – Simulator depth map

- SEM분석과 마찬가지로 원본 이미지 및 heatmap, percentile distribution 을 통한 분석을 진행함

Percentile distribution

Original depth map & heatmap

- 각 Case\_1, 2, 3, 4의 **Hole을 제외한 영역의 pixel value는 모두 동일(각 140, 150, 160, 170 = Max\_value)**
- Max\_value 외의 pixel value 역시 case가 증가함에 따라 증가, max\_value에 가까울수록 증가폭이 크다
- 각 percentile 그래프의 형태는 모든 case에서 유사
- 모든 case의 hole 형태는 매우 유사하고, 각 max\_value에 맞춰 scaling된 것으로 보임





## 1-a. 데이터 분석 - 정리

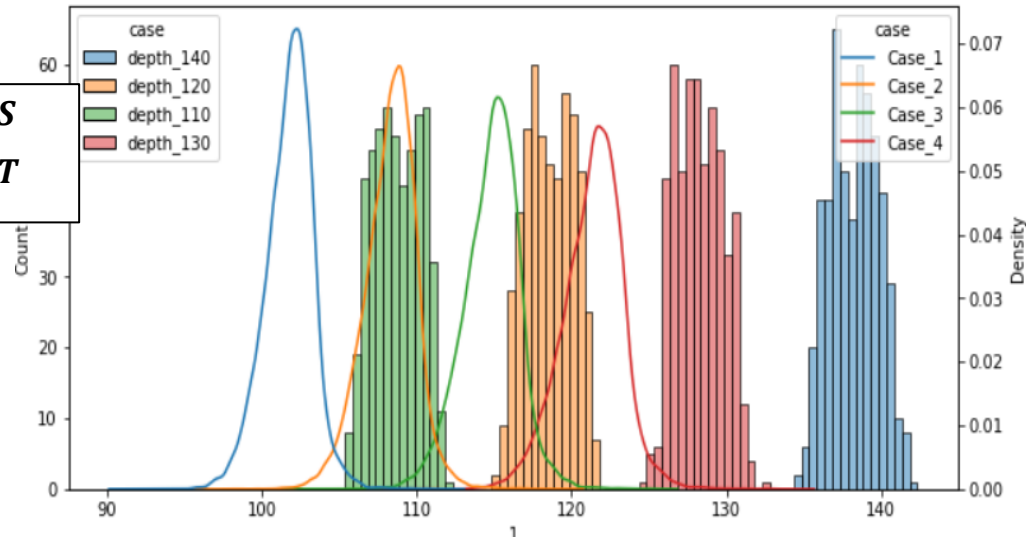
---

1. Noise와 pixel value의 distribution 차이를 제외하면 두 domain간 형태가 매우 유사
2. Inference 대상인  $SEM^T$ 는 paired depth map이 존재하지 않고,  $SEM^S$ 은 존재( $Depth^S$ )
3. Case별 depth map의 형태 자체는 유사하나, 각 Case의 Max\_value를 기준으로 다르게 scaling 됨

=>  $Depth^S = Depth^T$  예상, Case별 분리 학습의 효과 검토

## 1-b. $Depth^S = Depth^T$ ? - $Depth^S$ 와 *average\_depth.csv* 비교를 통한 가설 확인

Kdeplot:  $Depth^S$   
Histplot:  $Depth^T$



- $SEM^S$ 와  $SEM^T$ 에 각각 대응하는 depth map이 같은 domain일 것으로 예상했으나( $Depth^S = Depth^T$ )
- *average\_depth.csv* 속  $Depth^T$ 의 depth average와  $Depth^S$ 의 case별 depth average가 서로 다른 분포를 보임
- 이를 해석하기 위해 다음의 두 가설을 설정하고 진행하였음

1.  $Depth^S \neq Depth^T$

2.  $Depth^S = Depth^T$ 이지만, 다른 요인이 작용해 depth average 분포의 차이가 발생함

## 1-b. $Depth^S = Depth^T$ ? - $Depth^S \neq Depth^T$

---

- 첫 예상과 달리,  $Depth^S \neq Depth^T$  인 경우:

가.  $Depth^S \neq Depth^T$  이지만 여전히 유사성은 존재할 것으로 기대

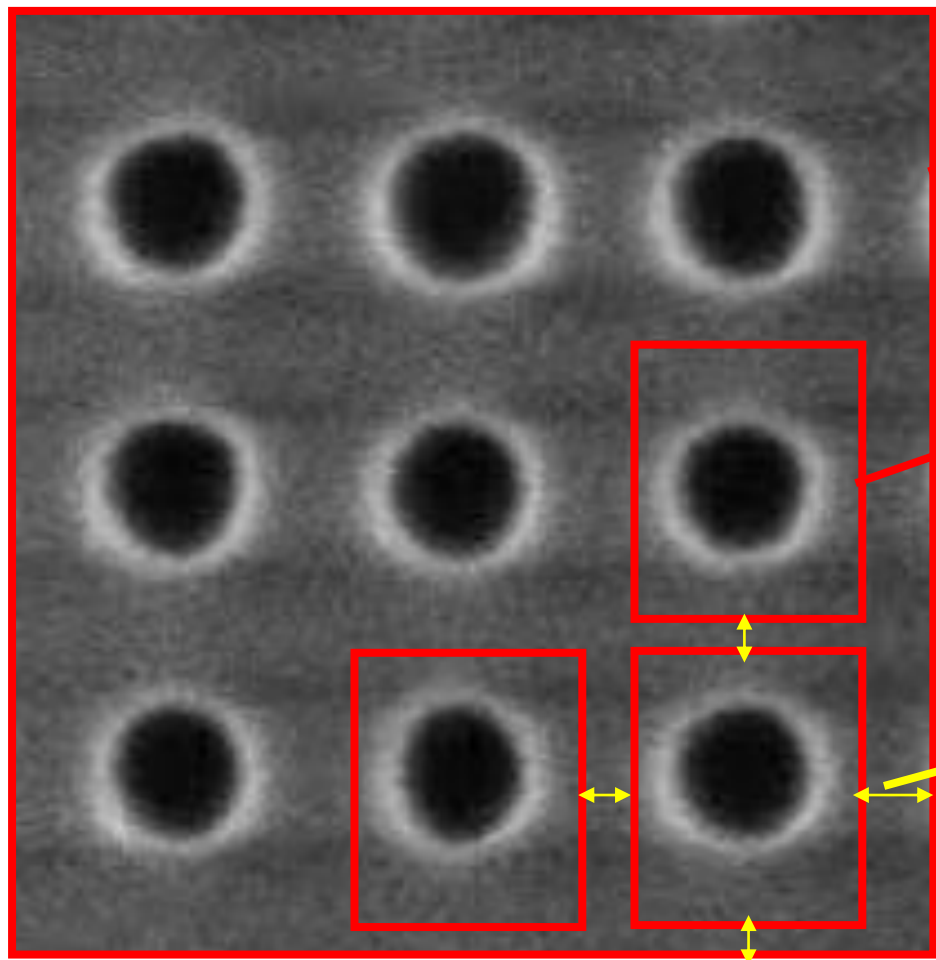
나. case증가에 따라 depth map의 max value나, 기타 statistical 특징들이 어느정도 선형적인 변화를 보임  
다. Avg 분포가 겹치는  $Depth^S$ 의 Case\_2와  $Depth^T$ 의 depth\_110가 같고, 나머지 case는 나.를 활용해 예측

=> 구조가 지나치게 복잡해지고, 성능 또한 좋지 않았음

## 1-b. $Depth^S = Depth^T$ ? - $Depth^S = Depth^T$ : 다른 요인?

- $Depth^S = Depth^T$  이지만 depth avg분포는 다른 이유 설명 필요
- Domain지식을 참고하여 dataset생성 방식을 다음과 같이 예상하였음

Site\_xxxxxx



각 site당 plate 전체의 average depth 계산  
=> average\_depth.csv 생성

Hole단위 crop  
=> Train\_SEM,  $SEM^T$  생성

$SEM^T$ 에는 포함되지 않지만(crop),  
average\_depth.csv에는 포함되는 영역(margin) 존재

## 1-b. $Depth^S = Depth^T$ ? - $Depth^S = Depth^T$ : 다른 요인 = *Margin!*

---

- $Depth^S$ 에 *margin*을 추가하여 avg를 새로 구한 후, average\_depth.csv와 비교

```
def get_margin_avg(avg, max_value):  
    pixel_sum = avg * (72 * 48)          #총 pixel 개수  
    margin_value = max_value * margin    #margin의 pixel 개수 * max_value  
    new_sum = pixel_sum + margin_value  
    new_avg = new_sum / (72*48 + margin) #new_sum / (기존pixel개수 + margin pixel개수)  
    return new_avg
```

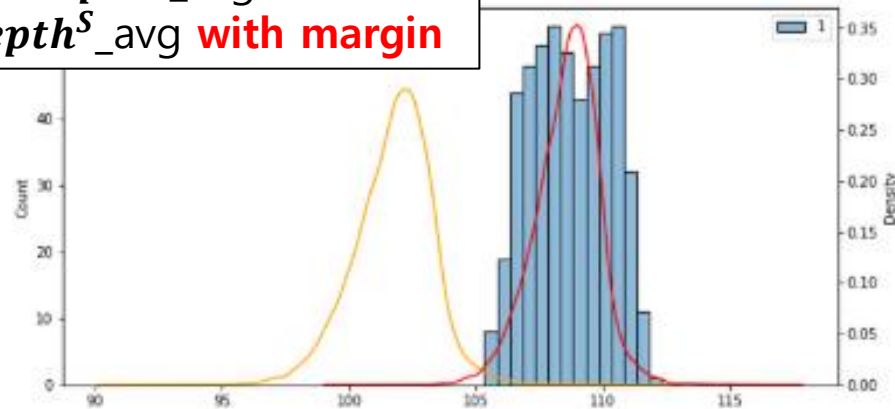
1.  $Depth^S$ 에서 각 hole을 제외한 모든 영역의 값이 해당 case의 max\_value로 같음  
=> *margin의 pixel value들 역시 모두 max\_value일 것으로 유추 가능*
2.  $Depth^S$ 의 각 이미지(72\*48)에 margin이 추가됐을 때 average\_depth.csv의 분포와 유사해지는지 계산  
=> 다음 슬라이드 계속

# 1-b. $Depth^S = Depth^T$ ? - $Depth^S = Depth^T$ : $\Delta$ 은 $요인 = Margin!$

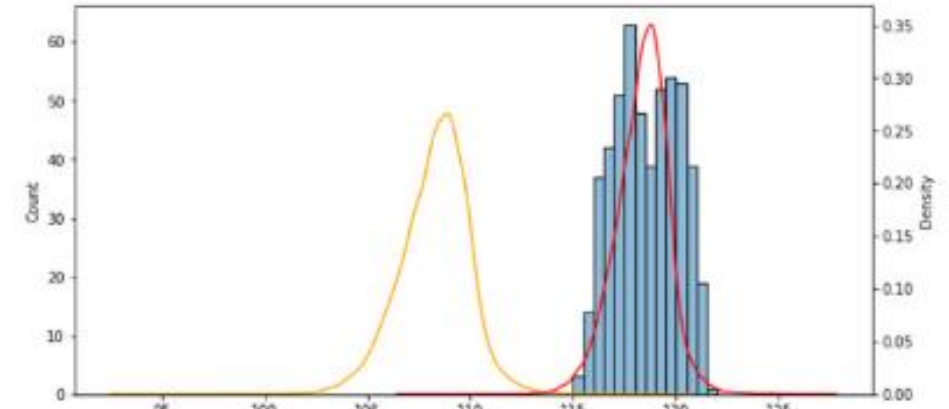
Histplot: average\_depth.csv

Kdeplot\_yellow:  $Depth^S_{avg}$

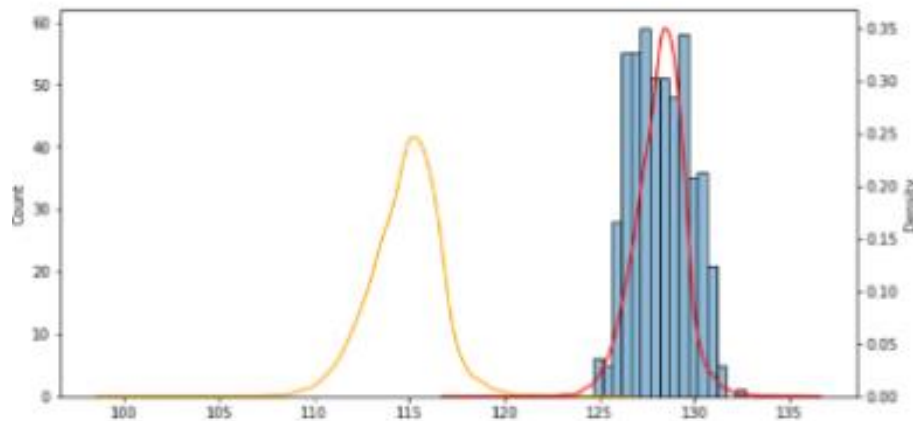
Kdeplot\_red:  $Depth^S_{avg}$  with margin



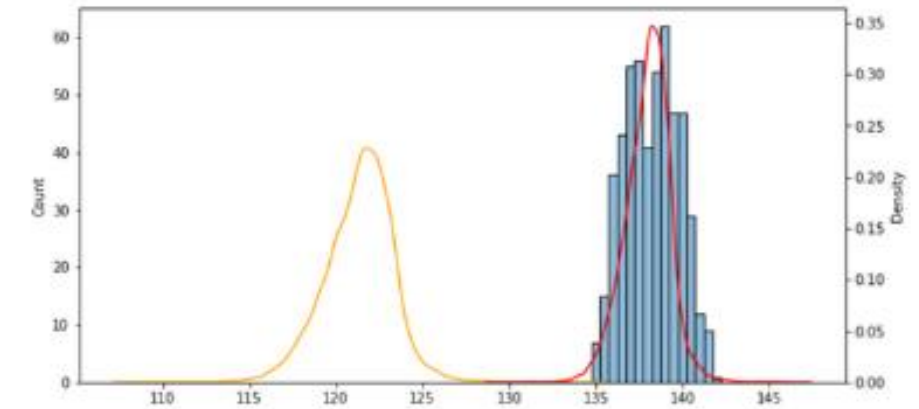
case\_1 vs depth\_110, margin=750



case\_2 vs depth\_120, margin=1100



case\_3 vs depth\_130, margin=1450



case\_4 vs depth\_140, margin=1800

## 1-b. $Depth^S = Depth^T$ ? - $Depth^S = Depth^T$ : 다른 요인 = *Margin!*

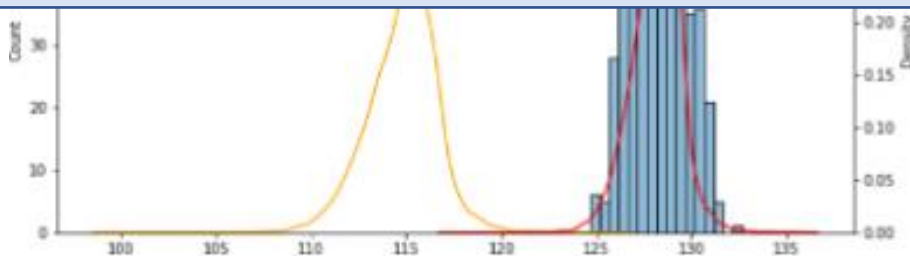
Histplot: average\_depth.csv

Kdeplot\_yellow:  $Depth^S_{avg}$

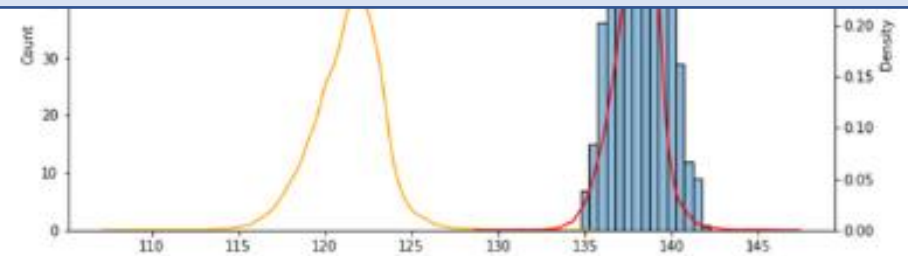
Kdeplot\_red:  $Depth^S_{avg}$  with margin



- case별 margin이 각각 750, 1100, 1450, 1800일 때, average\_depth.csv와 avg 분포가 유사해짐
- case별 margin이 350씩 증가하는 규칙성이 있지만, margin이 증가하는 이유는 여전히 설명 불가
- Margin의 증가를 설명할 수 없으면  $Depth^S = Depth^T$  역시 기대하기 힘들, 다시 domain지식 참고

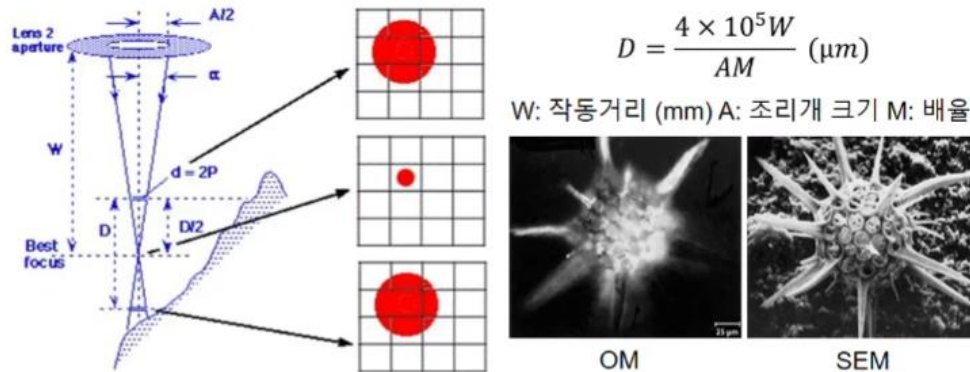


case\_3 vs depth\_130, margin=1450



case\_4 vs depth\_140, margin=1800

# 1-b. $Depth^S = Depth^T$ ? - $Depth^S = Depth^T$ : 다른 요인=Margin & *Depth of Field*



	Magnification	Resolution	Depth of Field
OM	4~1500	0.2 mm	0.5 mm
SEM	10~500,000	1.5 nm	30 mm

- **Depth of Field:** 초점을 유지할 수 있는 최대 깊이
  - 카메라에서의 최대 심도 효과와 유사
  - 작동거리가 길고, 조리개 크기가 작고, 배율이 작을수록 depth of field 증가
  - SEM이 OM보다 큰 depth of field를 보임

이미지 출처: <https://www.youtube.com/watch?v=Dno6x-Ek2xo&t=907s>

- 실제 SEM촬영에서, 목표 초점 깊이에 따라 작동거리, 조리개 크기, 배율을 조정
- Train\_SEM 및 average depth 생성 과정에서, Case가 증가할수록 더 깊은 초점을 유지해야함
- 따라서 margin의 증가 원인을 다음과 같이 유추 가능

**Case 증가 => depth of field 확보 필요**

**=> 작동거리를 늘리고, 조리개 크기와 배율을 줄임**

**=> 촬영 결과에 포함되는 영역 증가**

**=> Margin역시 함께 증가**



## 1-b. $Depth^S = Depth^T$ ? - 결론

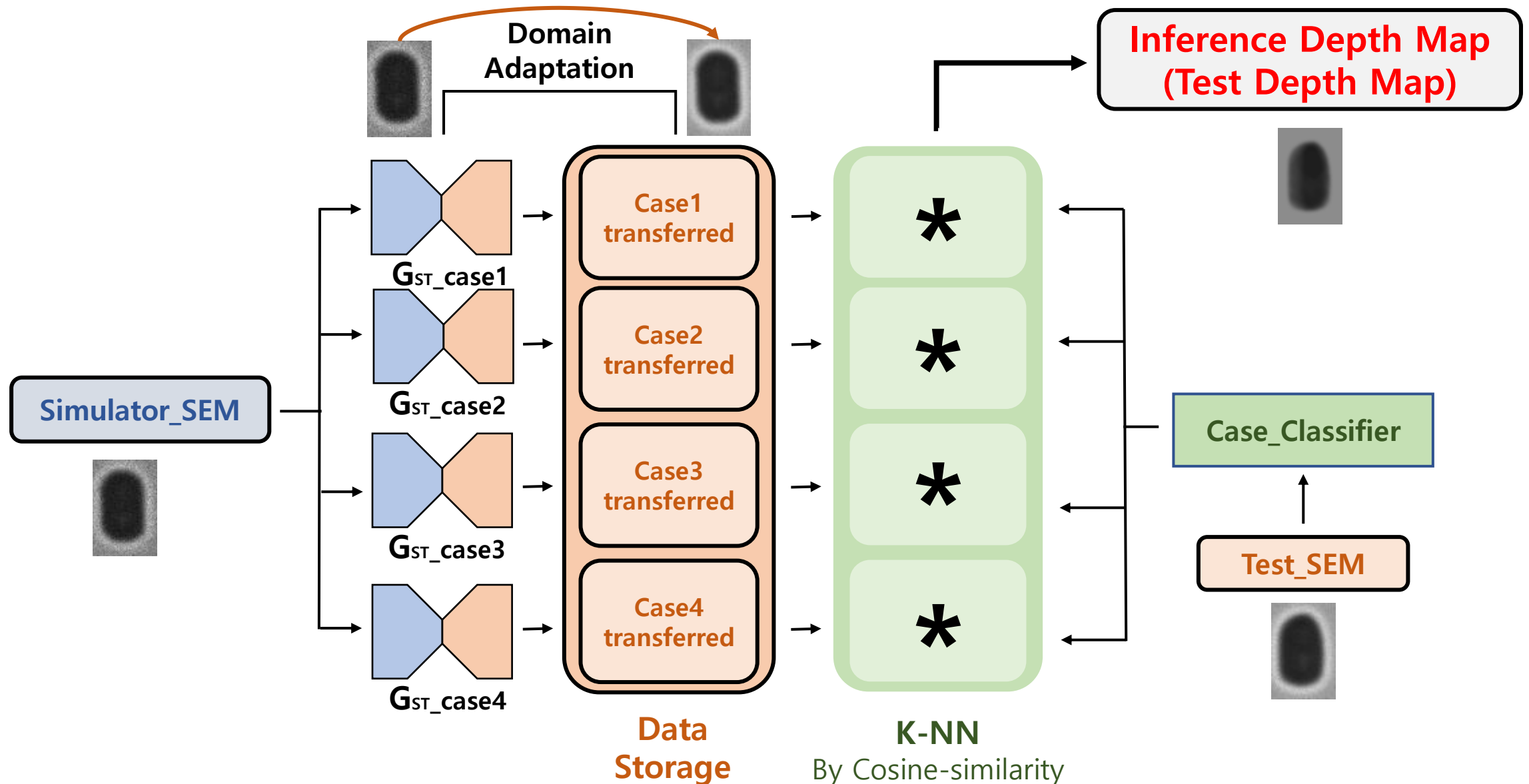
---

1.  $Depth^S$ 의 average와 average depth.csv 분포가 다르지만,
2. Margin과 depth of field를 고려하면  $Depth^S = Depth^T$ 이더라도 average 분포가 달라질 수 있음
3. 구체적인 데이터셋 생성 방식을 알 수 없어 average 분포 차이의 정확한 원인은 알 수 없지만,
4. 기타 데이터 분석 결과를 통해  $Depth^S = Depth^T$  가능성이 충분하다고 판단.

$SEM^S \neq SEM^T$ , but  $Depth^S = Depth^T$ , 이를 근거로

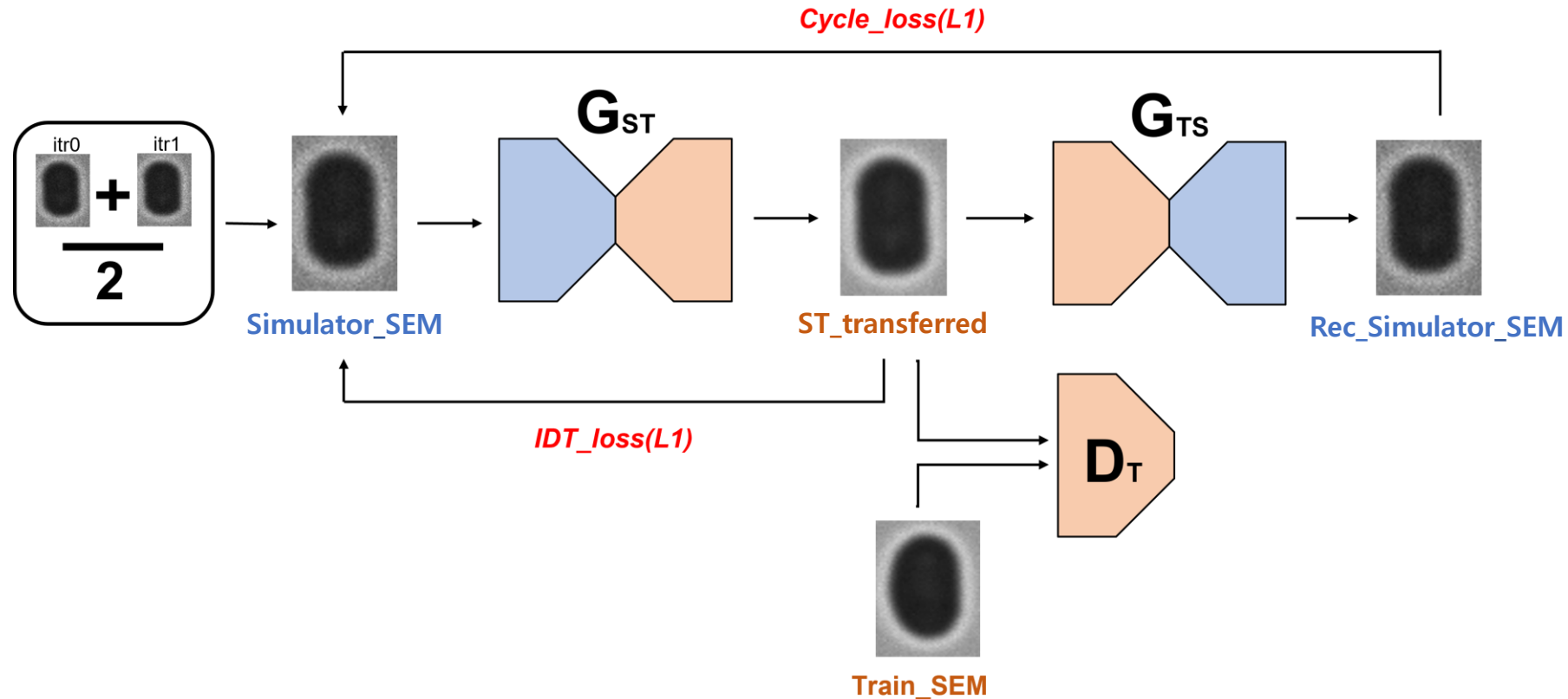
- 
- $SEM^S$ 과  $SEM^T$ 을 서로 유사하게 transfer(Domain Adaptation)
  - DA를 거친 새로운 domain  $DA(SEM) \rightarrow Depth$ 를 통해
  - $SEM^T \rightarrow Depth$ 를 구하는 전략 설정

## 2. Architecture overview – cycleGAN+KNN



## 2-a. Domain Adaptation - overview

- $SEM^T$ 가  $SEM^S$ 보다 노이즈가 적기 때문에,  $SEM^S \rightarrow SEM^T$ 로 Domain adaptation
- Adversarial한 방식으로 **cycleGAN**을 활용해 진행
- 이 때, feature 손실을 방지하기 위해  $SEM^S$ 의  $(itr0+itr1)/2$ 를 제외한 별도의 denoising 없이 진행



## 2-a. Domain Adaptation - detail

---

### 1. WGAN-GP loss 사용

- 초기에는 LSGAN으로 진행하였으나 학습의 안정성을 위해 WGAN-GP를 사용함

### 2. Resblock 구조의 Generator

- Encoder->Resblock->Decoder(Transposed convolution)

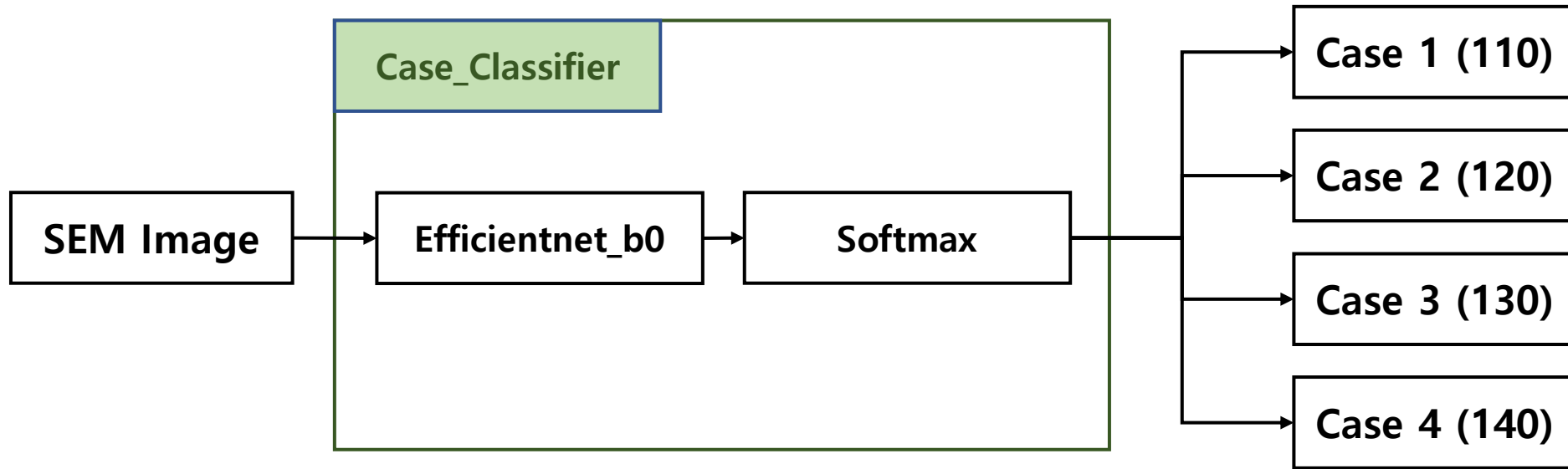
### 3. Case마다 각각의 ST 학습, 저장

- **SEM**의 각 case마다 약간의 차이가 존재, 성능 향상을 위해 별도로 ST 수행
- Inference에서는 Case-Classifier를 활용해 각 case와 매칭

### 4. 학습 시간 단축을 위해 먼저 학습이 완료된 Case의 ST모델을 사용 (transfer learning)

- Case\_1의 학습을 끝낸 후(300 epoch), 이후 case들은 Case\_1의 ST모델 weights로 학습 시작(각 100 epoch)

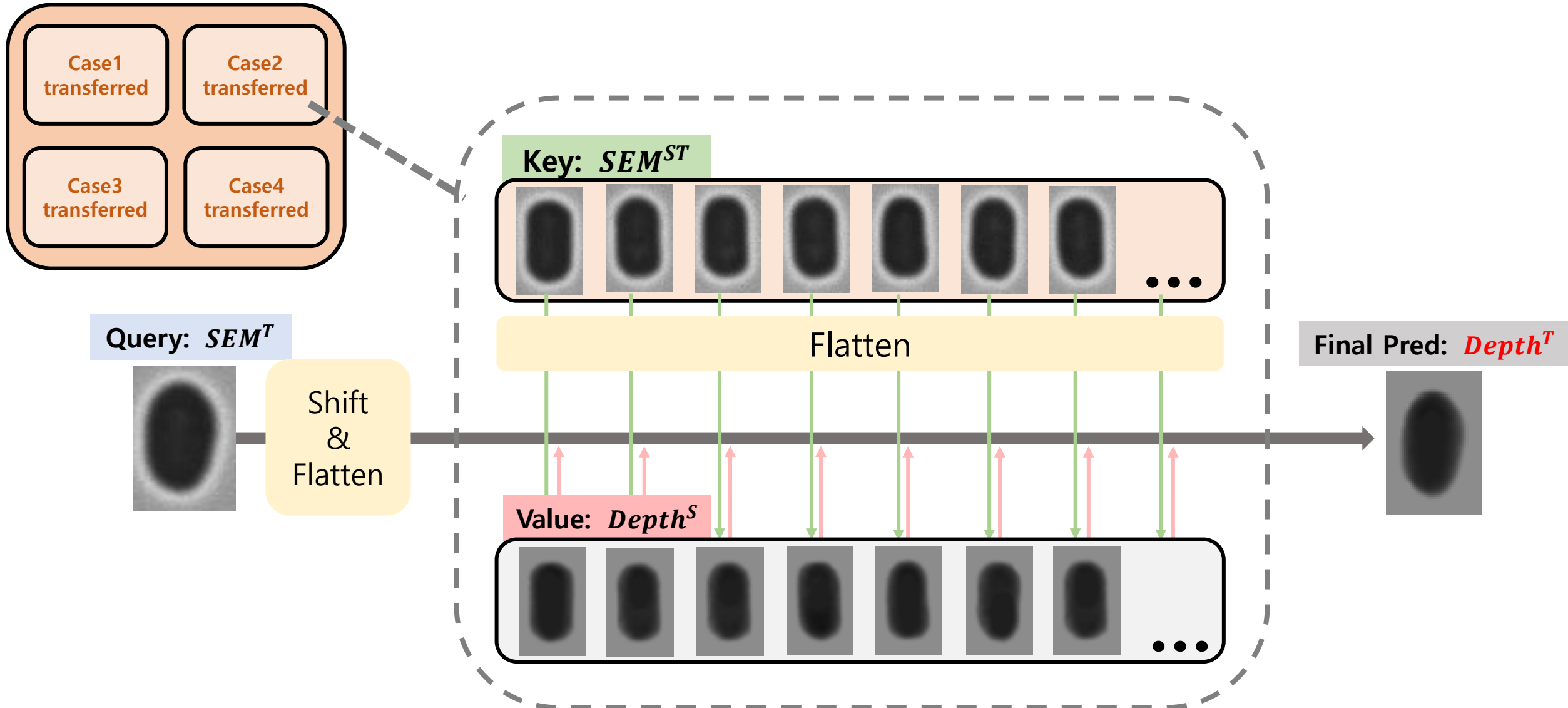
## 2-b. Case Classifier



- SEM image에 따른 **depth map의 max value별 case** 정보를 사용하여 inference시 예측하는 depth image의 max value를 더욱 정확하게 예측할 수 있도록 함.
- 몇몇 Classifier를 학습해본 결과, case 구분이 어려운 작업이 아니기 때문에 가벼운 모델도 성능이 대부분 **약 99%** 정도에 수렴하는 것을 확인함.
- 최종 Classifier 모델로는 성능이 좋고 학습 시간이 오래 걸리지 않는 **tf\_efficientnet\_b0\_ns pretrained** 모델을 선택하게 됨.
- 학습단계에서는 **Train SEM Image**를 이용하여 SEM 이미지에 따른 case 구분을 학습하며, Inference 단계에서는 Test Image의 case 구분에 사용함.

## 2-c. KNN with Cosine-similarity

### Data Storage



## 2-c. KNN - detail

---

### 1. 각 이미지를 flatten, z-score normalize하여 query, key로 사용

- 벡터 내적을 위한 flatten, normalize를 통해 특정 값(외곽 빛번짐)들이 지나치게 큰 영향을 갖는 것을 방지

### 2. Cosine-similarity 사용

### 3. 각 shift에 따른 별도의 Cosine-similarity 계산

- Hole의 위치에 Cosine-similarity가 영향을 받음, best-similarity sample을 찾기 위해 shift를 진행하면서 탐색

# 3-a. Why cycleGAN + KNN?

---

cycleGAN + KNN은 다음과 같은 차별점을 가진다

## 1. 단순한 구조

타 Domain Adaptation approach의 경우:

- domain-invariant feature representation을 학습하기 위한 discrepancy-base 방식
- Adversarial 방식으로 feature extractor 학습시키는 방식
- cycleGAN에 task성능 유지를 위한 별도의 구조를 더한 방식 등,

DA에서 general한 task 성능을 얻기 위해 복잡한 구조가 사용된다.

하지만 cycleGAN + KNN의 경우, 비교적 간단한 cycleGAN만이 DA를 위해 사용되고 이어지는 KNN은 추가적인 학습을 필요로 하지 않기 때문에 구조가 단순하다.

단순한 구조에도 불구하고 좋은 성능을 얻을 수 있었던 이유는 다음과 같다.

1. Data가 hole 형태의 단순한 grayscale 이미지이다.
2. DA 전에도 두 domain  $SEM^S$ ,  $SEM^T$ 이 이미 상당히 유사

이로 인해 cycleGAN의 cycle-consistency loss와 idt-loss만으로도 feature 손실을 줄이면서 ST가 가능했다.



# 3-a. Why cycleGAN + KNN?

---

cycleGAN + KNN은 다음과 같은 차별점을 가진다

## 2. KNN의 Robustness

DA에서 **Domain**간 차이를 해결하는 모델과, 주 **task**를 위한 모델로 나누어 생각해볼 때  
(본 알고리즘에서는 각각 cycleGAN(ST)와 KNN이 해당된다.)

앞선 Domain간 차이를 해결하는 과정에서:

1. 완전히 해결되지 않은 Domain간 차이
2. Feature 손실

이 발생하고, 이를 완전히 제거하는 것은 매우 어려운 작업이다.

또한, 위 문제에 **Task모델의 학습이나 성능이 크게 영향**을 받을 수 밖에 없다.

이는 기존 DA에서 주로 다룬 Classification이 아닌 Regression 문제에서 더욱 불안정해지는 것으로 판단.

실제로 KNN 사용 이전에 Encoder-Decoder 구조의 Task 모델을 사용했을 때,  
앞선 ST과정의 불완전성은 Task모델 성능이 불안정으로 이어졌고, 이를 예측하는 것 역시 매우 어려웠음.

하지만, **KNN의 경우 정해진 metric에 따라 실제 data를 hard하게 사용하므로 앞서 언급한 문제에 Robust**하다  
특히, 해당 챌린지의  **$Depth^S = Depth^T$** 이고, **depth map이 단순하며, 충분한 sample 수가 확보된 상황** 속에서는  
단순한 KNN으로도 뛰어난 성능을 보장할 수 있다.

## 4. 추가 성능 개선 방안

---

### 1. cycleGAN 학습 시 epoch의 수 증가.

현재는 case1에 대해서 300 epoch, case 2, 3, 4에 대해서는 case 1으로부터 transfer learning을 진행해 100 epoch씩 학습했는데, 전반적인 학습 epoch 수를 늘리면 성능 개선이 일어날 것 같다. 실제로 전체 케이스에 대해서 200 epoch씩 진행한 결과물에 비해서 300 epoch의 성능이 20%정도 향상되었기 때문이다.

### 2. 더욱 많은 simulation data.

우리 모델은 simulation data, 즉 paired data 수에 비례하여 성능이 개선되기 때문에 많은 양의 데이터가 있다면 더 좋은 성능을 보일 수 있다. 즉, 다시 말해 **data storage 기반 KNN 모델**을 사용하기 때문에 data의 수에 따라 모델의 성능이 증가하는 비례관계를 가진다.

### 3. KNN시 사용되는 가중치 함수 변경

유사도 값의 차이가 커짐에 따라 큰 차이를 발생시키도록 SoftMax 함수를 사용하였으나, 다른 알고리즘을 사용해 가중치를 더욱 효과적으로 부여하는 방향으로 개선.

### 4. Hole Center Align

현재는 similarity를 확인할 때 모든 방향으로 shift를 진행하여 비교하지만, 데이터 자체의 홀 중앙을 맞추고 유사도 확인을 진행한다면 더 빠른 추론 속도와 정확성을 가질 수 있음.

## 5. 결론 및 건의사항

---

### 결론

- cycleGAN을 이용하여 Simulation SEM Image를 Train SEM Domain으로 Transfer하여 Test SEM Image와의 Domain Gap을 줄였다.
- 또한 KNN을 사용한 data storage 기반 depth map estimation으로 depth map을 중요한 feature의 손실 없이 예측할 수 있었다.

### 건의사항

- 데이터에 대한 도메인 지식이 많이 필요했던 대회였기 때문에 다음에 비슷한 대회가 열린다면 사전 설명회와 같은 방식으로 도메인 지식을 제공받을 수 있는 기회가 있었으면 좋을 것 같다.