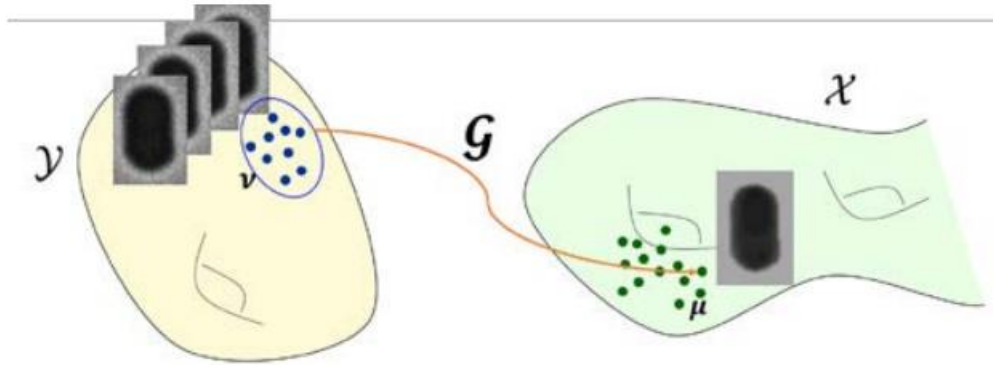
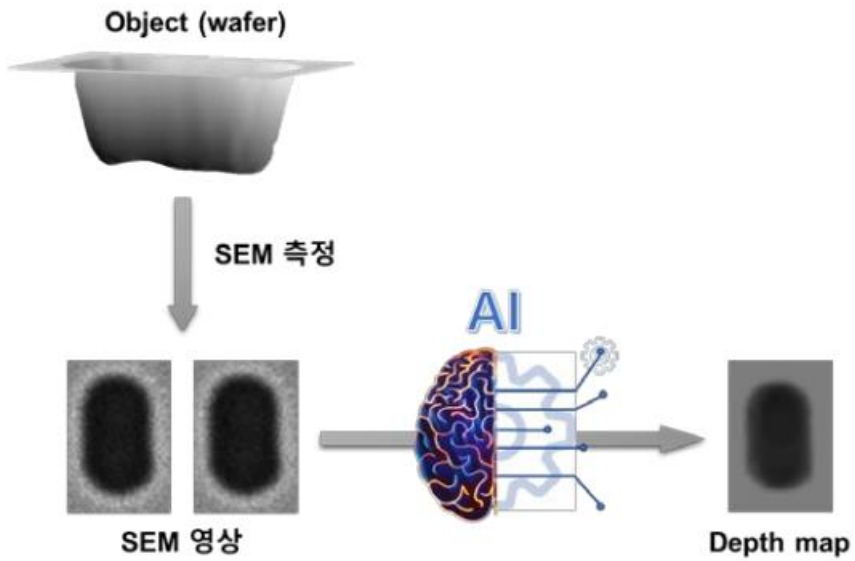
기계지능특강 : 삼성 프로젝트 보고서

2021-21674 권준우, 2021-22028 김수영

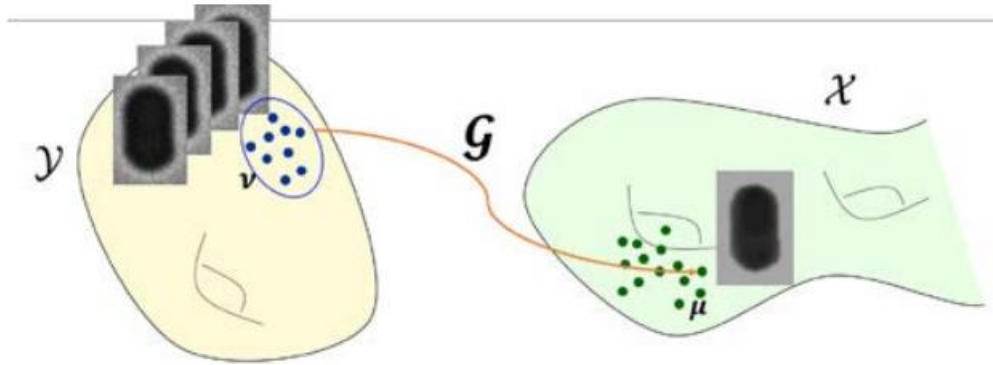
1. Introduction : Project 개요

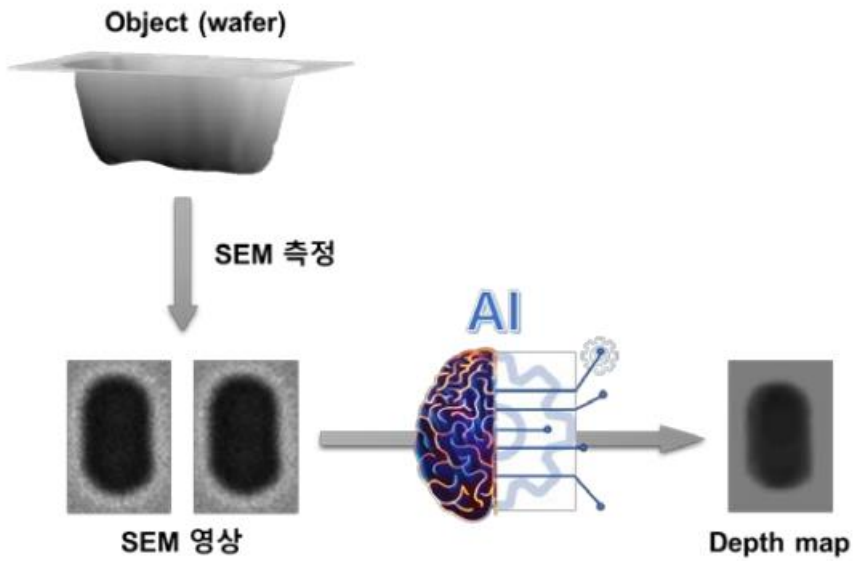




- 반도체 생산 효율을 높이기 위해 공정에서의 오차를 줄이는 것이 중요.
- 이를 확인하려면 생산된 Wafer가 설계된 대로 만들어졌는지 확인할 수 있어야 함.
- 현재는 SEM 영상의 Intensity로 Depth를 예측함.
- 허나 전자의 무작위 움직임으로 인해 동일한 Object (Wafer)에 대해 획득한 SEM 영상 간에도 미묘한 차이가 존재하고 이로 인한 Depth 예측의 오차가 발생.

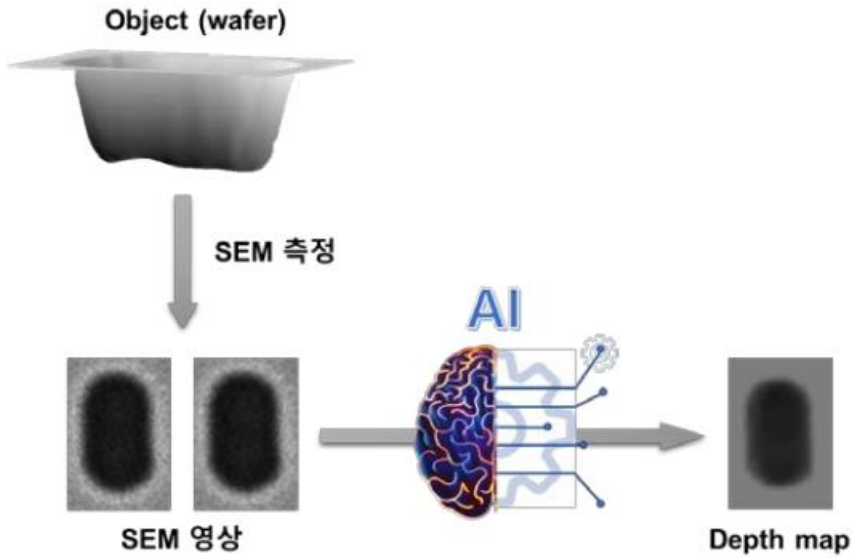
1. Introduction : Project 개요

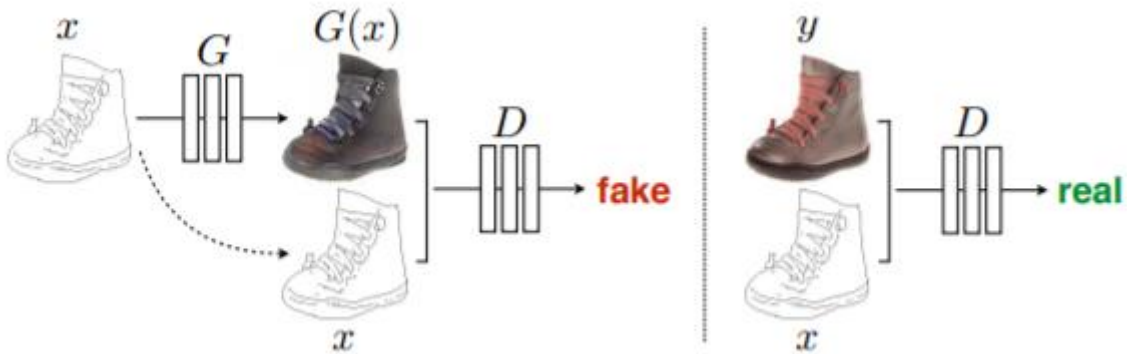




- 또한 현 계측 영상/방식의 특성상 계측 속도와 한계 깊이 간 Trade-off가 존재.
- 2D 영상만으로 계측 한계 깊이 확장 및 계측 속도 향상을 이룩해야 함.
- 따라서 기존 SEM 영상 기반의 방법보다 더 정확한 Depth 예측 방법이 필요.
- **즉, 단일 SEM 영상으로부터 Pixel별로 정확도 높은 Depth를 예측해야 함.**

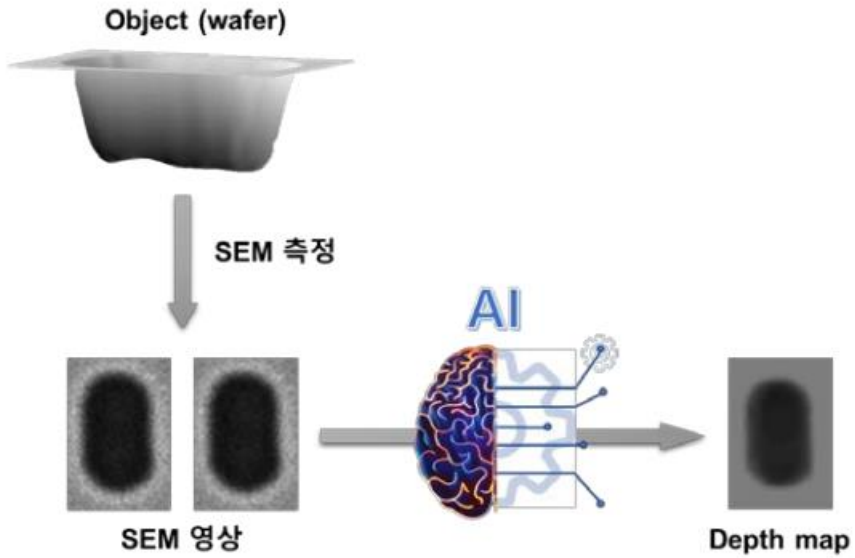
2. Method : 모델 소개

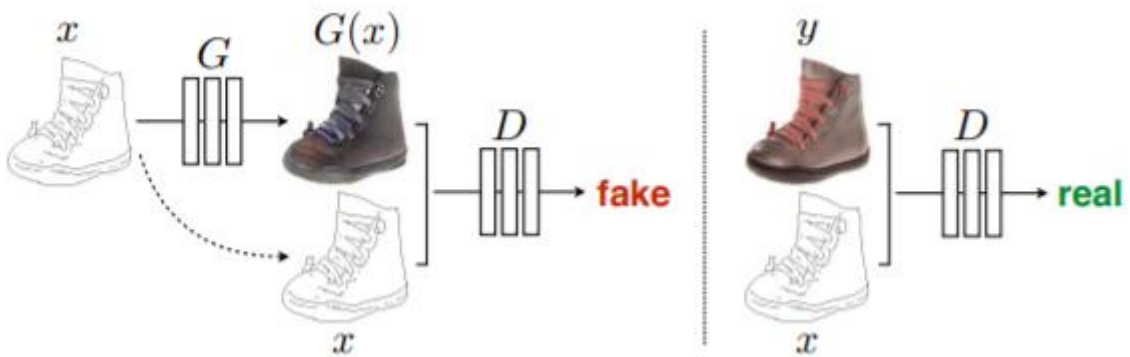




- **단일 SEM 영상으로부터 Pixel별로 정확도 높은 Depth를 예측해야 함 == Depth map을 생성해야 함.**
- 정확한 Depth Map 생성을 위해 image-to-image 방식인 pix2pix 모델을 사용.
- Pix2pix는 image를 image로 변환하도록 generator를 학습.
- Generator는 SEM 영상을 input으로 받아 이 영상을 Depth map으로 Generate함.
- Generator가 만든 이미지를 discriminator가 보고 진짜 depth map인지, generato에 의해 생성된 가짜 depth map 인지 판독할 수 있게 함.
- Generator는 Ground Truth Depth map을 최대한 잘 모방하여 Discriminator가 진짜와 가짜를 구분하지 못하게 학습.
- 반대로 Discriminator는 Generator가 만든 가짜 depth map과 Ground truth depth map을 확실하게 구분할 수 있도록 학습.

2. Method : 모델 소개





- 이때 Loss function은 다음과 같이 설계함.

```
### backward ###
# Loss - G
self.optimizer_G.zero_grad()
self.G_loss_GAN = self.criterion_GAN(D_fake_out, True)
self.G_loss_L1 = self.criterion_L1(fake_depth, real_depth)

mse = 0
for b in range(sem.shape[0]):
    f_image = ((fake_depth[b, 0, :, :].detach().cpu().numpy() + 1) / 2) * 255.0
    r_image = ((real_depth[b, 0, :, :].detach().cpu().numpy() + 1) / 2) * 255.0
    mse += mean_squared_error(f_image, r_image)
rmse = (mse / sem.shape[0]) ** 0.5

self.G_loss = self.G_loss_GAN + self.config.lambda_L1 * self.G_loss_L1 + rmse
# self.G_loss = self.config.lambda_L1 * self.G_loss_L1 + rmse
self.G_loss.backward(retain_graph=True)
self.optimizer_G.step()

# Loss - D
if idx % 10 == 0:
    self.optimizer_D.zero_grad()
    D_loss_fake = self.criterion_GAN(D_fake_out, False)
    D_loss_real = self.criterion_GAN(D_real_out, True)
    self.D_loss = (D_loss_fake + D_loss_real) / 2
    self.D_loss.backward()
    self.optimizer_D.step()
```

```
# Loss function
class GANLoss(nn.Module):
    def __init__(self, gan_mode, target_real_label=1.0, target_fake_label=0.0):
        super(GANLoss, self).__init__()
        self.register_buffer('real_label', torch.tensor(target_real_label))
        self.register_buffer('fake_label', torch.tensor(target_fake_label))

        self.gan_mode = gan_mode
        if gan_mode == 'lsgan':
            self.loss = nn.MSELoss()
        elif gan_mode == 'vanilla':
            self.loss = nn.BCEWithLogitsLoss()
        elif gan_mode == 'wgangp':
            self.loss = None
        else:
            raise NotImplementedError('gan mode %s not implemented' % gan_mode)

    def get_target_tensor(self, prediction, target_is_real):
        if target_is_real:
            target_tensor = self.real_label
        else:
            target_tensor = self.fake_label
        return target_tensor.expand_as(prediction)

    def __call__(self, prediction, target_is_real):
        if self.gan_mode in ['lsgan', 'vanilla']:
            target_tensor = self.get_target_tensor(prediction, target_is_real)
            loss = self.loss(prediction, target_tensor)
        elif self.gan_mode == 'wgangp':
            if target_is_real:
                loss = -prediction.mean()
            else:
                loss = prediction.mean()
        return loss
```

- Pix2pix는 위 손실 함수에 pixel loss를 추가함.
- 즉, 생성한 가짜 depth map과 진짜 depth map사이의 L1 loss를 계산함으로써 각 pixel별로 값의 차이를 계산하고 이를 줄여나가는 방식으로 학습함.
- 이를 통해 더욱 더 진짜같은 depth map을 구현할 수 있게됨.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

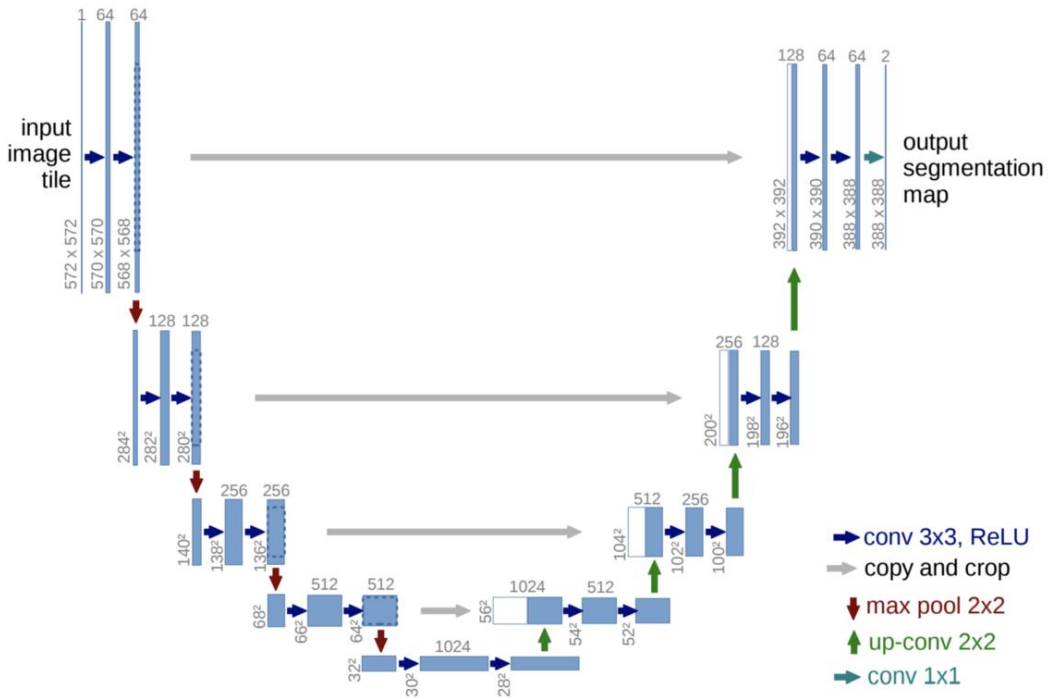
2. Method : 모델 소개

Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see https://phillipi.github.io/pix2pix/ for additional examples.

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

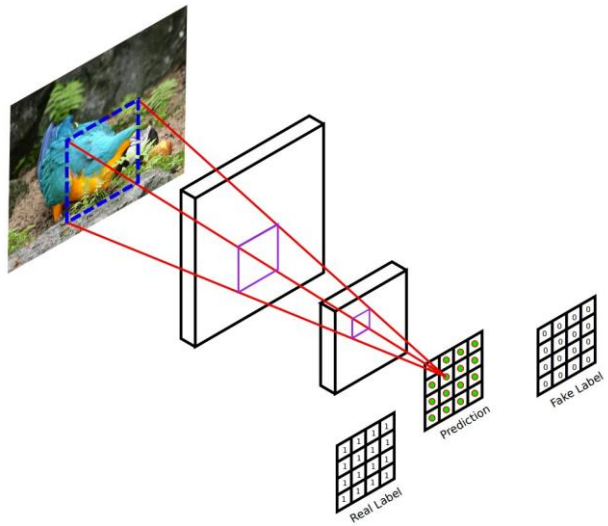
- Loss function에 L1 loss만 사용하면 Generator는 low – frequency 이미지만을 생성함.
- 왼쪽의 그림을 보면 L1 loss만 사용했을 때 low-frequency 성분으로 구성된 blurred된 이미지를 확인할 수 있음.
- 따라서 기존의 GAN loss가 high-frequency 성분을 잘 뽑아낸다는 성질을 이용하여 두 개의 loss를 합쳐서 구해준다면 정확한 이미지를 생성할 수 있음.

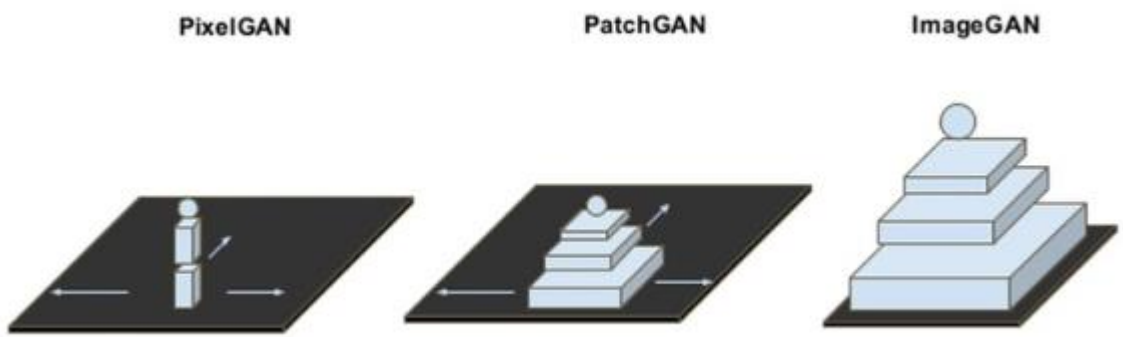
2. Method : Generator 모델 소개



- Generator 모델로는 이미지 생성 model로 탁월한 성능을 보인다고 여겨지는 U-net을 활용
- Unet의 main idea는 아래와 같다.
  - **1) Contracting Path** : 이미지의 context 포착
  - **2) Expansive Path** : 압축된 feature map을 upsampling 하여 원래의 이미지 크기로 생성
  - 3) 이때 1)에서 각 층마다 포착한 feature map을 upsampling 과정에서 skip-connection으로 결합해줌으로써 더 정확한 localization을 할 수 있도록 함.
- 우리의 image size에 맞춰 네트워크 구조를 변경.
  - 이미지 layer를 왼쪽의 5단에서 2, 3, 4단으로 변경
  - 변경 이유는 원본 input 이미지가 66x45 정도 밖에 안돼서 model이 deep 해지면 잘 학습을 못했기 때문.
  - 실제로 shallow model일 때 결과가 가장 좋았음.

2. Method : Discriminator 모델 소개

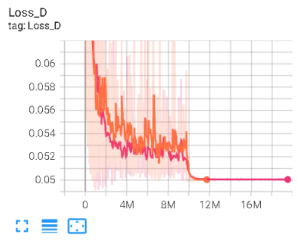


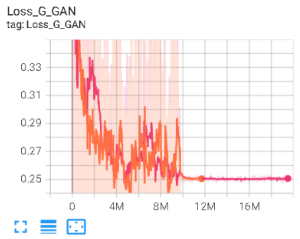


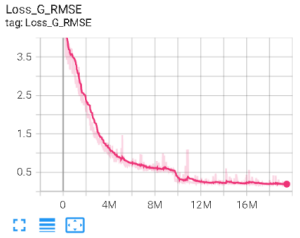
- Discriminator 모델로는 이미지 분류 model로 탁월한 성능을 보인다고 여겨지는 Patch GAN을 활용
- Patch GAN: Image에서 patch단위로 fake / real을 구분하고 그 결과의 평균을 취하는 방식
- Patch GAN은 전체 이미지가 아니라 작은 이미지 패치 단위에 대해 Sliding window가 지나가며 연산을 수행하므로 파라미터 개수가 작음.
- 즉, 연산 속도가 더 빠르고 전체 이미지 크기에 영향을 받지 않아 구조적 관점에서 유연성을 보임.
- 또한 Low-freq에 대해 학습하는 L1 loss와 Local한 영역에서 sharpen한 디테일, 즉 high-freq 영역(edge 정보)에 대해 patch 단위로 학습함으로써 두 방식의 장점을 모두 취할 수 있음.

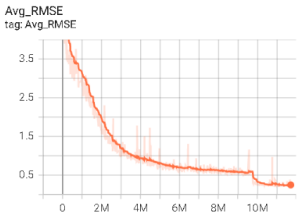
3. Result

MSE

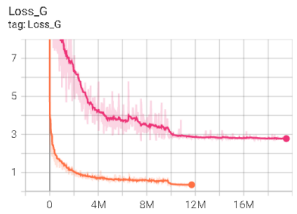




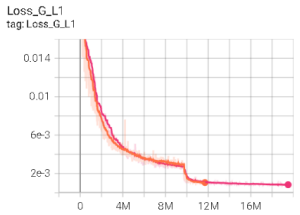




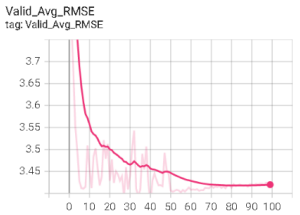
Loss_G



Loss_G_L1

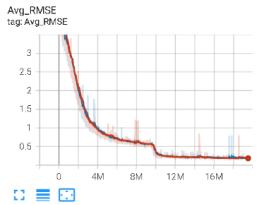


Valid_Avg_RMSE

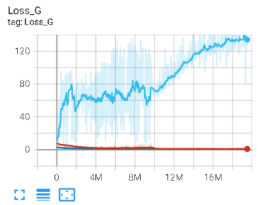


BCE

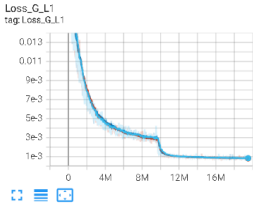
Avg_RMSE



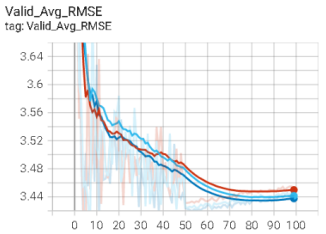
Loss_G



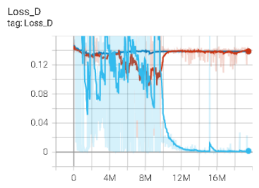
Loss_G_L1



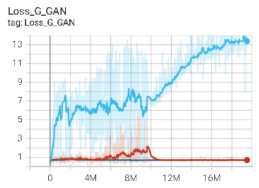
Valid_Avg_RMSE



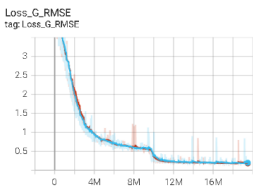
Loss_D



Loss_G_GAN

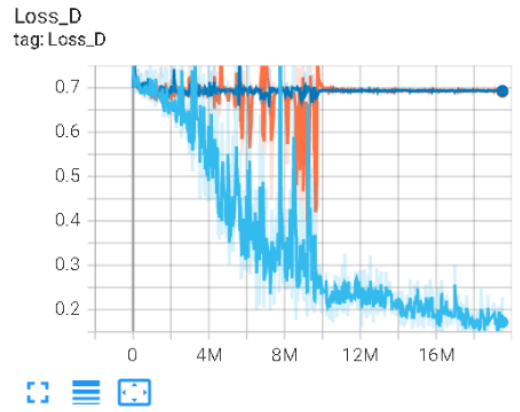


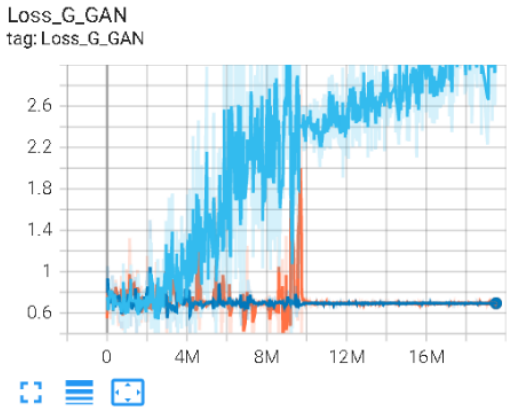
Loss_G_RMSE

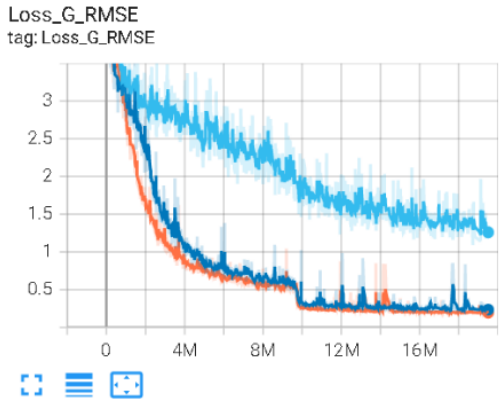


3. Result

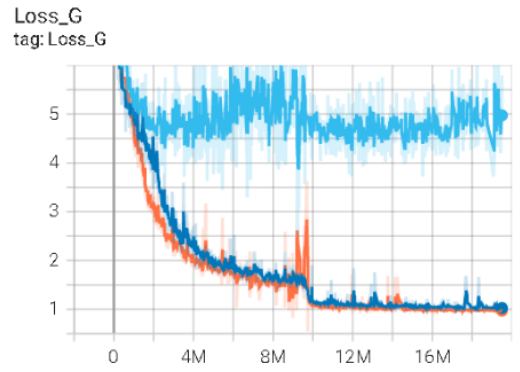
shallow layer model



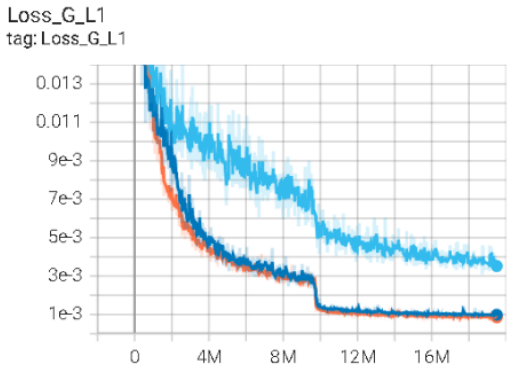




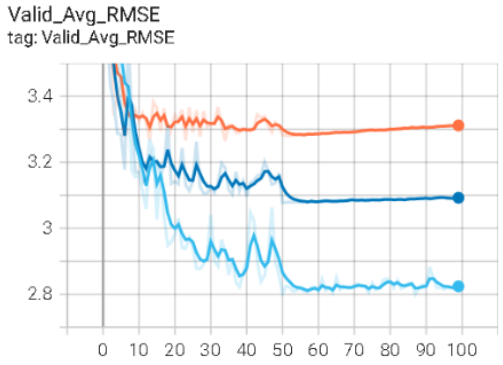
Loss_G



Loss_G_L1



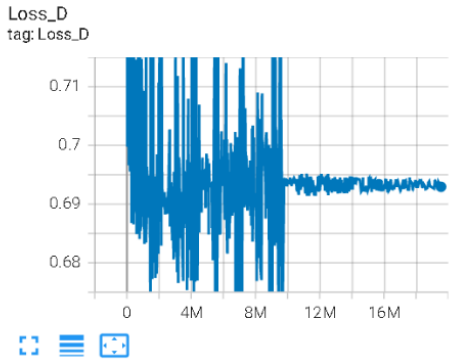
Valid_Avg_RMSE

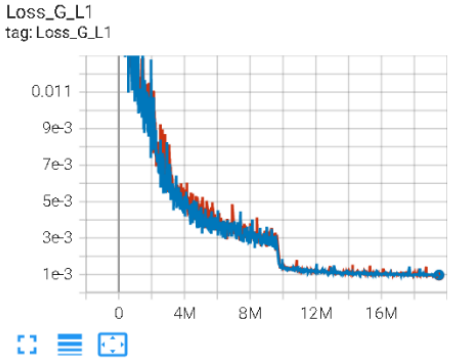


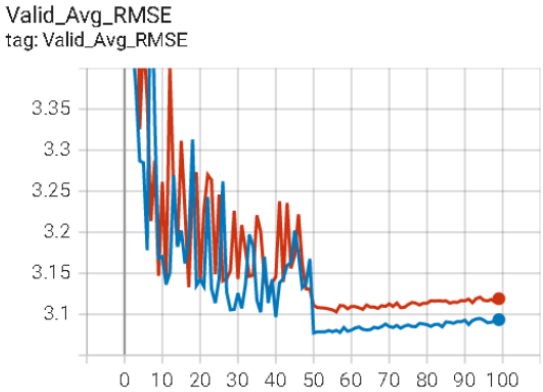
- : 2 layers
- : 3 layers
- : 4 layers

3. Result

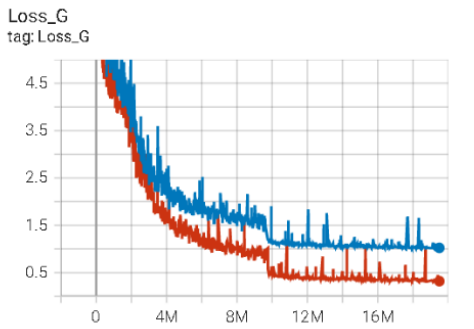
[3 layers] Unet



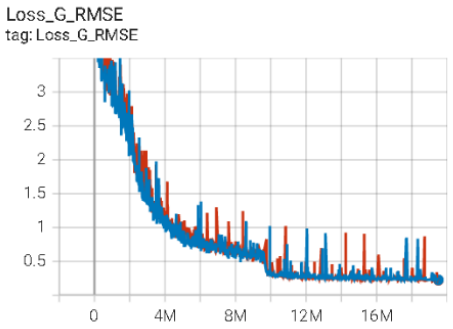




Loss_G



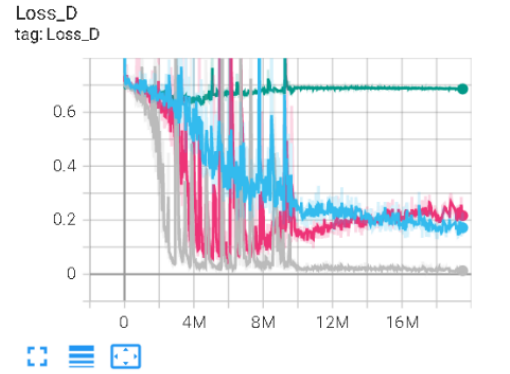
Loss_G_RMSE

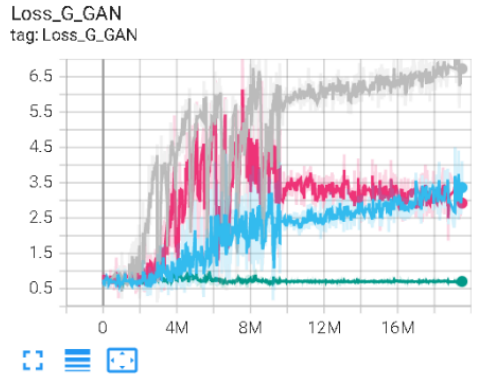


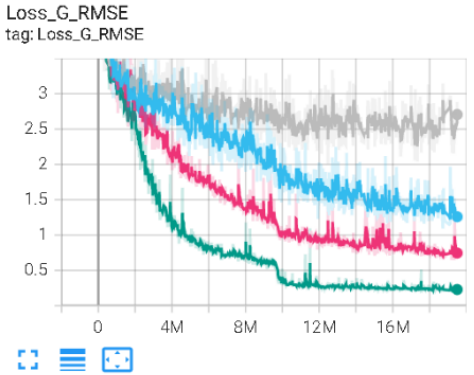
- : 3 layers
- : 3 layers Unet

3. Result

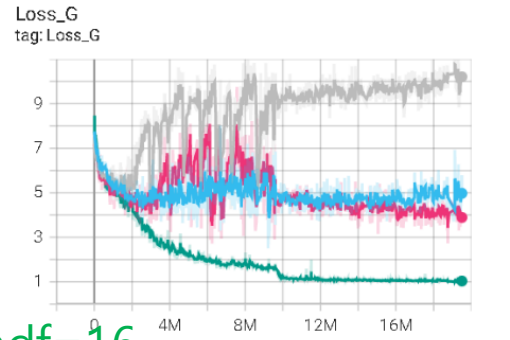
shallow layer model 비교



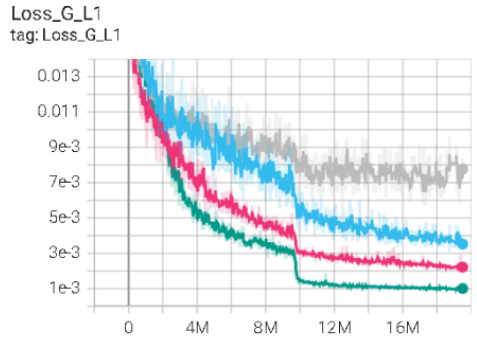




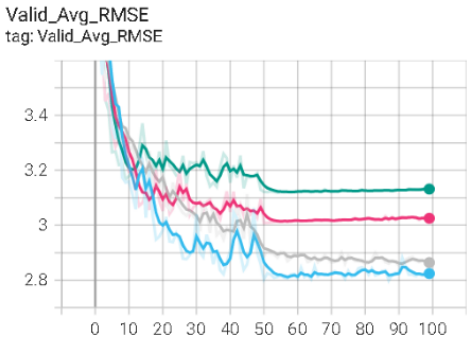
Loss_G



Loss_G_L1



Valid_Avg_RMSE



- : 2 layers
- : 2 layers with ndf=16
- : 3 layers
- : 3 layers with deeper Discriminator

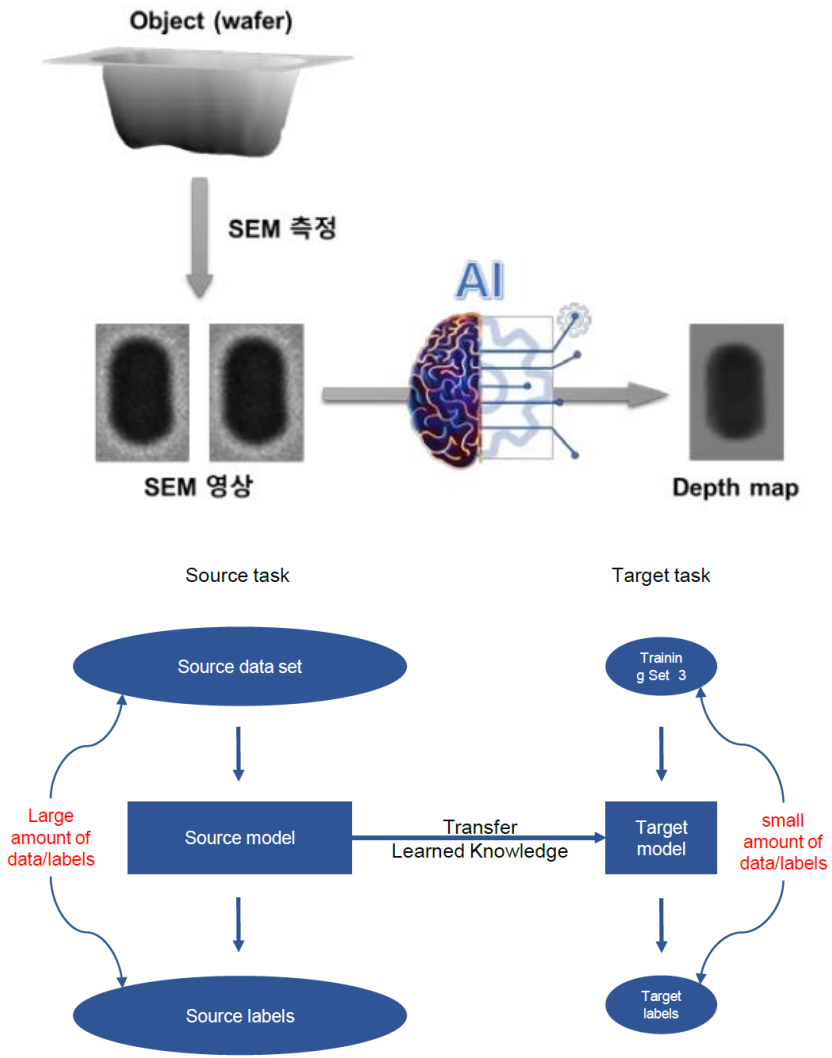
3. Result

Loss	D loss	G loss	Valid RMSE
BCE	0.1386	0.7783	3.424
MSE	0.05005	2.765	3.3768

# of Layers	D loss	G loss	Valid RMSE
2 layers	0.1469	4.843	2.799
3 layers	0.6962	1.022	3.077
4 layers	0.6901	0.9659	3.26

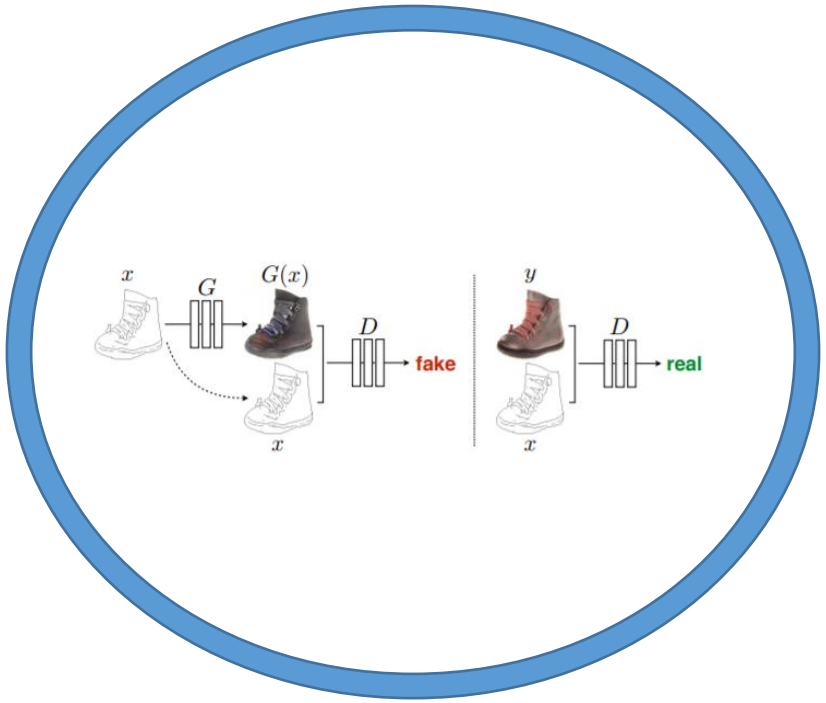
Model	D loss	G loss	Valid RMSE
Pix2Pix	0.6926	1.022	3.077
Unet	x	0.3188	3.111

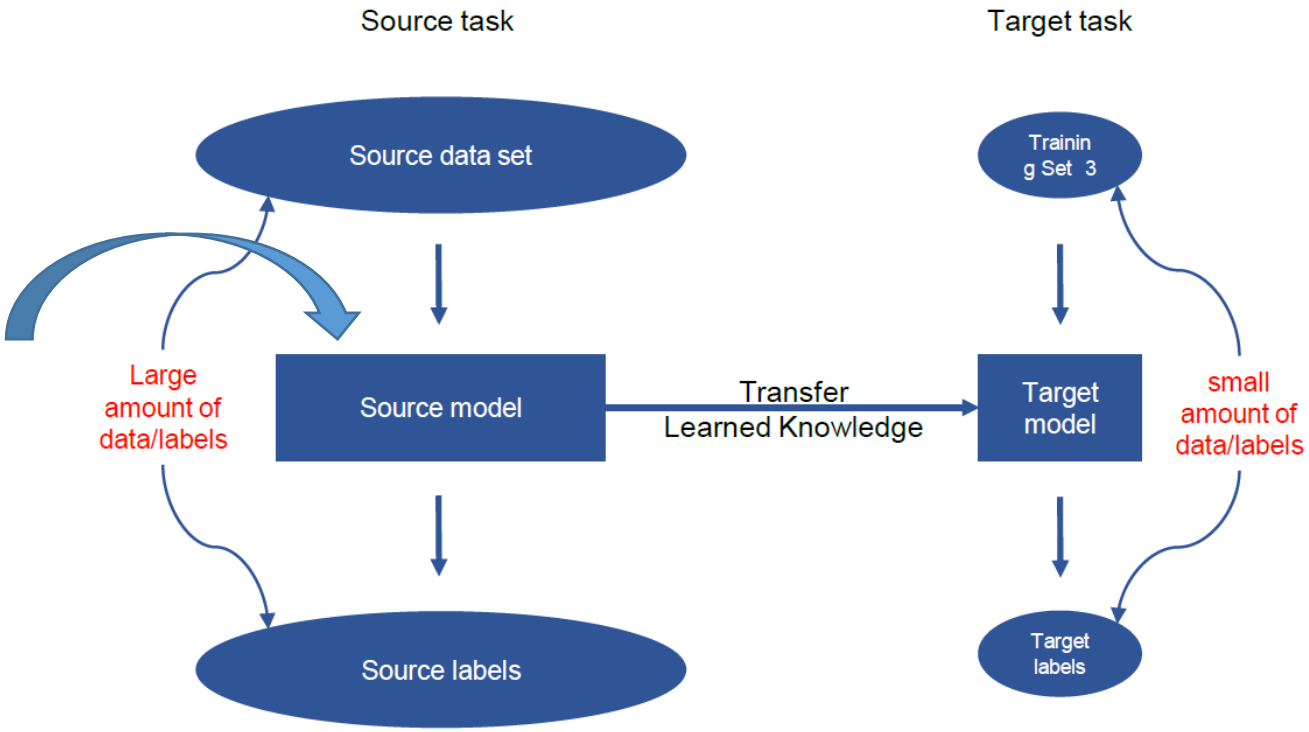
4. Initial Ideas….



- 전자의 무작위 움직임으로 인해 동일한 Object (Wafer)에 대해 획득한 SEM 영상 간에도 미묘한 차이가 존재하고 이로 인한 Depth 예측의 오차가 발생.
- 이런 Unstable한 Image Modality의 한계를 극복하고 Ground Truth가 구하기 힘든 경우에서도 안정적으로 좋은 결과를 얻기 위해 초기엔 Pix2Pix 모델을 우선 Supervised learning으로 학습 시키고 이를 Transfer learning 하여 Self-supervised 학습을 하고자 했었음.
- 실제로 코드도 완성한 상태 (github 참고)

Two networks Combined

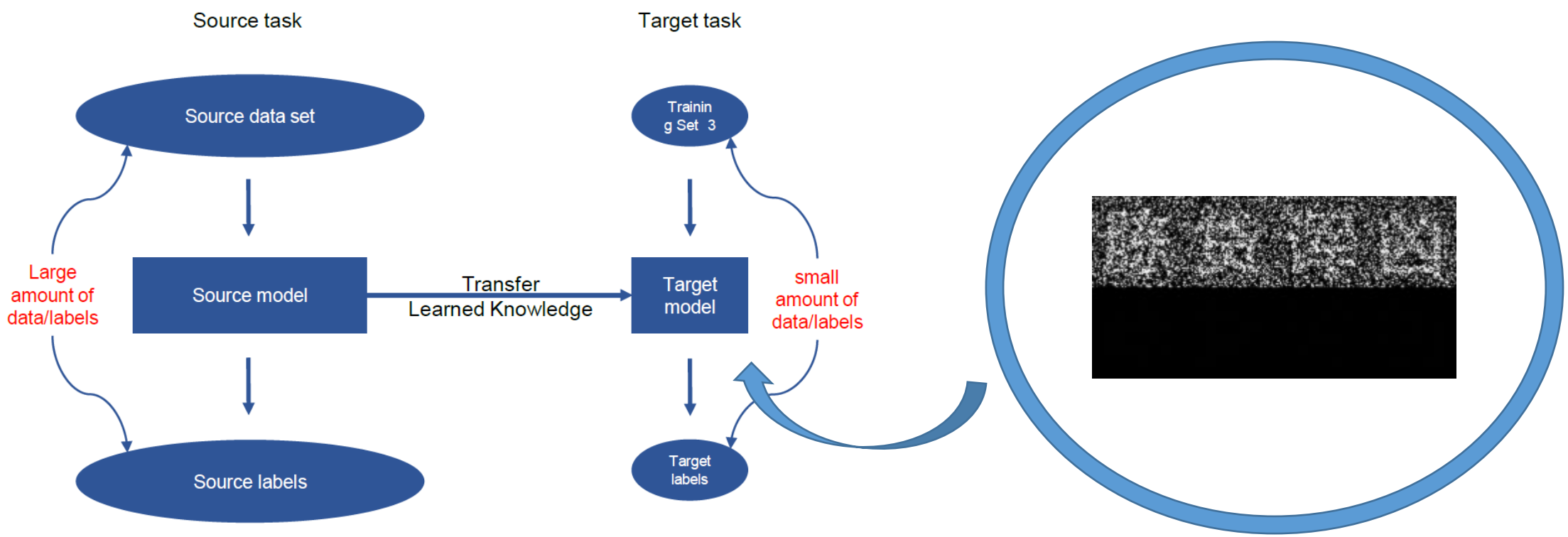




- 즉, Pix2pix를 Supervised learning으로 학습시킴.
- 그 후, 이 학습된 모델을 Transfer learning 시킴.

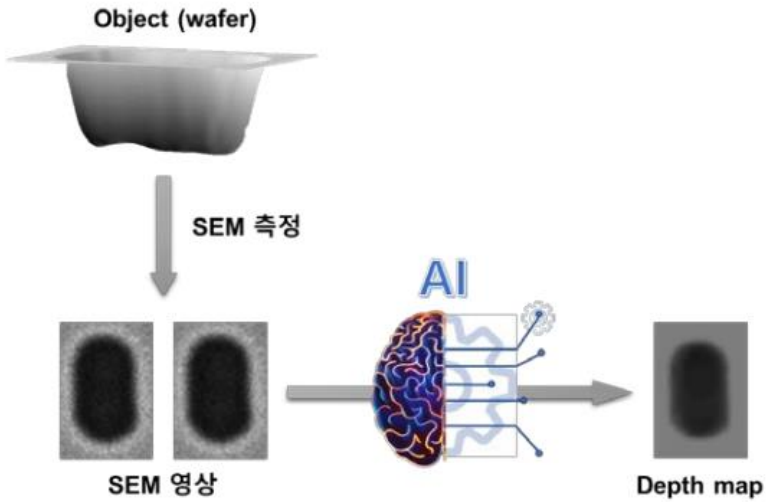
Two networks Combined

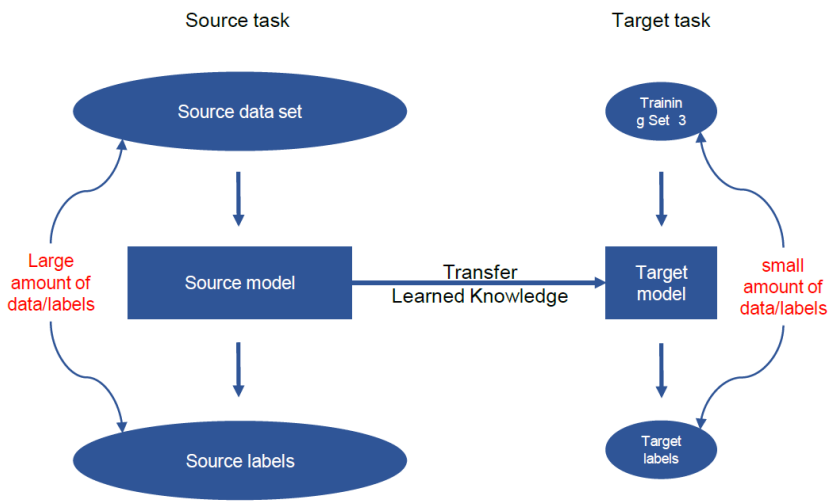
- 그 후, 이미지 디노이즈 알고리즘인 Noise2Self를 이용하여 추가적인 이미지 resolution 향상을 이룩하고자 했음.



Noise2Self: Blind Denoising by Self-Supervision, Batson et al. CVPR 2019

4. Initial Ideas….





- Self-supervised learning을 통해 다양한 이미지 획득 환경에서도 견고하게 작동하는 이미지 모델을 만들고자 함.
- 허나 이를 통해 나온 결과가 단순히 Pix2Pix를 통해 Inference된 것보다 안좋게 나옴. (RMSE 결과)
- 따라서 최종 제출은 Pix2Pix만 통과한 결과만 제출함.
- 추후에 self-supervised learning 부분을 더 fine-tuning하여 더 좋은 결과를 얻어볼 예정.

5. Github 주소

https://github.com/Sooyyoungg/SAIT.git

결과 정리 링크 :
https://docs.google.com/presentation/d/1rEttq4fKeAeZj2otnZy2LAB6ImVaMh05kgz3KmVTLWU/edit?usp=sharing