Small Fabric Database for Defect Detection

with only no defect images

Team **SCI**: 구본정, 김지은, 이혜승, { bjgu97, kje5076, 2hyes }@sm.ac.kr

Sookmyung Women's University

1. Introduction

여러 분야에서 결함 검출의 error rate를 낮추기 위한 논의를 지속하고 있다. 결함 검출의 자동화로 인한 인력 비용 절감 효과를 기대하지만, 실제로는 결함을 학습시키기 위한 초기 비용이 높은 경우가 많다. 이는 곧 효율성 감소를 의미한다.
이에 본 프로젝트는 불량 데이터를 임의로 발생시키는 방안보다는 정상 데이터만으로 모델을 학습시켜서 결함을 감지하는 unsupervised learning을 구현한다.

또한 딥러닝에서 좋은 성능을 보이기 어려운 소량의 데이터로 높은 성능의 모델을 구현한다. 고해상도 이미지를 동일한 사이즈의 패치들로 나누어, 학습 데이터의 개수를 늘려 모델을 학습시킨다.

2. Data: How to do preprocess?

프로젝트에서 사용한 데이터 셋은 AITEX사의 Fabric Image Database다. 해당 데이터 셋에는 총 256개의 이미지가 있다. 7개의 fabric 데이터 종류로 구성되어 있으며, 141개의 정상, 105개의 결함 이미지로 구성되어 있다. 결함 또한 12개의 종류로 구분되어 있다.

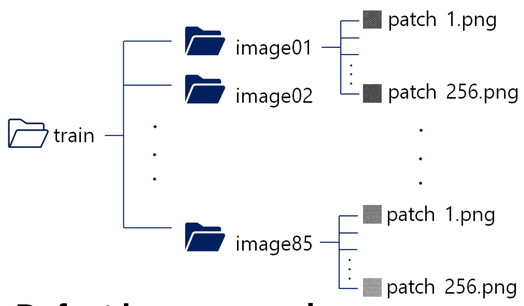
본 프로젝트에서는 fabric의 종류 및 결함의 종류를 구분하지 않고, 결함 여부만을 감지하는 형태로 데이터를 활용한다.

- **데이터 크기:** 256 * 4096 (grayscale이므로 1-channel)

- **How to split into train/validate/test dataset:**

정상 이미지만으로 모델을 학습시키므로, 트레이닝셋에는 정상 데이터만을 갖게 한다. 따라서 정상 이미지는 train : validate : test = 6 : 2 : 2의 비율로, 결함 이미지는 validate : test = 1 : 1의 비율로 분할한다.

- **Methods for solving small size problem:**



Data size가 작은 챌린지 극복을 위해, 이미지를 *64*64 사이즈의 patch 256개*로 분할한다.

결론적으로 트레이닝 데이터셋의 사이즈는 85개에서 *21,760(85x256)개*로 늘어난다.

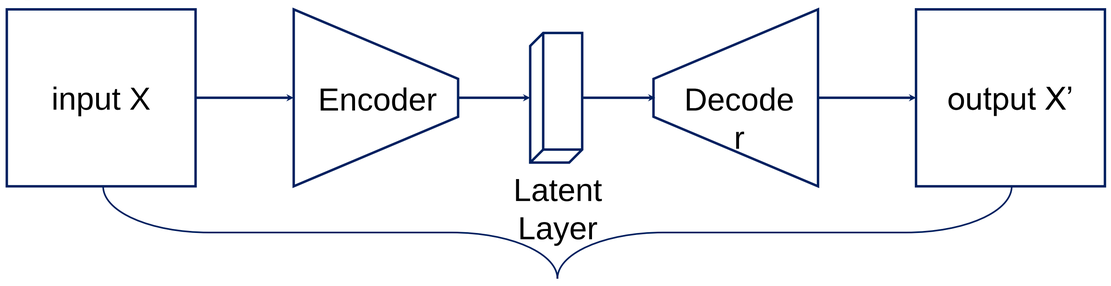
- **Defect image sample**

3. Approach: Step by step

- **Step 1: Split train, validate, test data**
- **Step 2: Divide into patches**

이미지를 patch로 분할하여 (이미지 개수 * 256)개를 독립적인 이미지라 가정한다.

- **Step 3: AutoEncoder**



$$\text{reconstructed error} = \|X - X'\|_2^2$$

Patch를 한 개씩 입력하면, 인코딩 단계에서 feature를 추출하여 압축시킨 후, 디코딩 단계를 거쳐 이미지를 재구성하는 neural network인 **AutoEncoder**를 활용한다. AutoEncoder에서는 Reconstructed error(MSE loss)를 줄여서 이미지를 잘 재구성하는 것(입력 이미지를 최대한 유사하게 출력하는 것)이 목표다.

- **Step 4: Set threshold & best model**

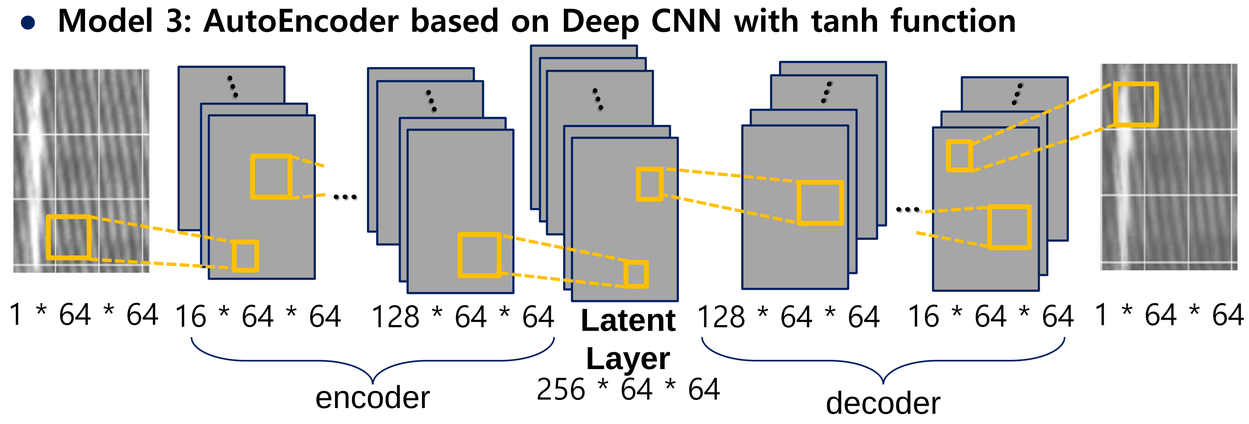
Reconstructed error가 '설정한 threshold'보다 높으면 해당 패치를 defect로 판단한다. 한 개의 이미지에서 defect로 판단된 패치의 개수가 k(threshold)개 이상이면, 해당 이미지를 defect 이미지로 판단한다.
Image defect detection을 위해 '*patch를 detect하기 위한 loss threshold*'와 '*이미지의 결함 여부를 판단하기 위한 threshold*'의 설정이 필요하다. 2개의 threshold를 설정하고, 가장 좋은 성능을 내는 모델을 선택하기 위해 grid search방식을 사용한다.
Patch threshold와 image threshold를 각각 3개로 설정한 후, 모델마다 각 9개의 F1 score를 얻는다. 총 36개의 조합 중 F1 score가 가장 높은 threshold 조합 및 모델을 최종적으로 선정한 후, 테스트를 진행한다.

4. Model: AutoEncoder

- **Model 1: AutoEncoder with linear dimension reduction**
  - Encoder: 64*64 패치를 linear function으로 4*4로 차원 축소하여 특징 추출한다.
  - Decoder: 4*4에서 다시 64*64 차원으로 복원한다.
  - Linear dimension reduction은 PCA의 원리와 유사하게 차원을 축소해 축약한다.
- **Model 2: AutoEncoder with nonlinear(PReLU) dimension reduction**

$$f(y_i) = max(0, y_i) + a_i min(0, y_i)$$

  - Encoder: 64*64 차원인 패치를 PReLU activation function을 활용하여 4*4로 차원 축소하여 특징을 추출한다.
  - Decoder: 4*4 차원에서 다시 64*64 차원으로 복원한다.
- **Model 3: AutoEncoder based on Deep CNN with tanh function**



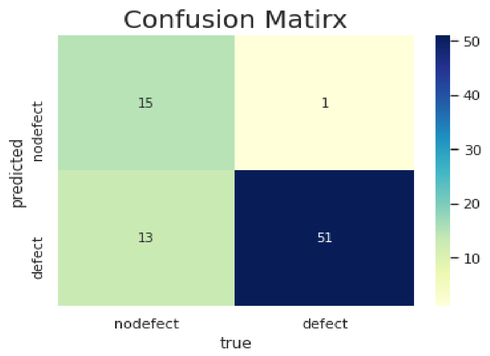
  - Image의 공간정보를 담고있는 CNN 기반으로 encoding을 진행한 후, transpose하여 decoding을 진행한다.
- **Model 4: AutoEncoder based on Deep CNN with Sigmoid function**
  - Model 3에서 decoding 과정의 마지막 activation function만 Sigmoid로 변경한다.

5. Results

- **Compare models with F1 score**

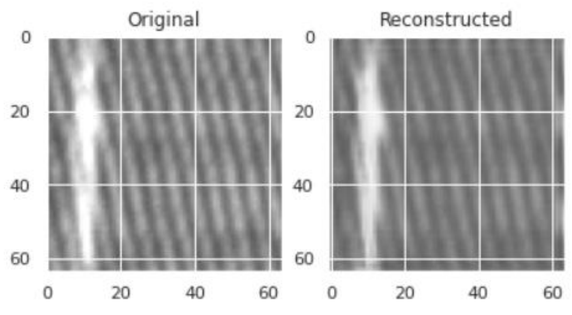
본 프로젝트에서 활용한 데이터는 불균형이므로, 모델의 성능을 정확하게 평가하기 위해서 성능지표로 precision(T로 분류한 것 중 실제 True)과 recall(실제 True 중 T로 분류)의 조화평균인 ***F1 score***를 활용한다.

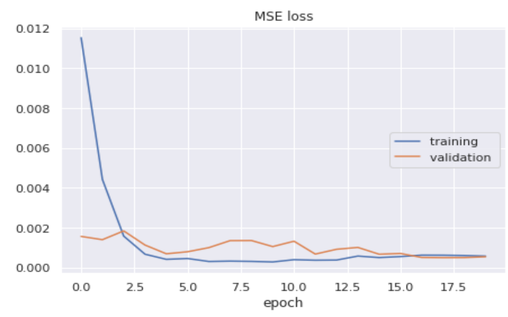
	Image threshold	Patch threshold	F1 score
Model1	5, 7	0.01	0.8824
Model2	5	0.005	0.8276
Model3	3, 4	0.01	0.8067
Model4	**3, 5, 7**	**0.0028**	**0.8793**



Model4(CNN with Sigmoid)가 이미지 재구성 및 성능 평가에서 가장 우수한 결과를 보였다. 이에 본 프로젝트의 모델로 model4를 최종 모델로 활용한다. Validation과 test를 할 때의 image threshold와 patch threshold는 각각 3, 0.0028로 설정한다.

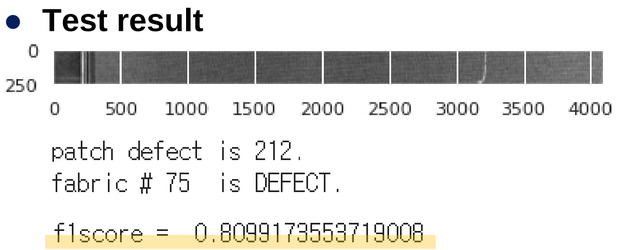
- **Original and Reconstructed image example**





Model1,2로 훈련된 reconstructed 이미지를 출력했을 때, 제대로 재구성되지 않음이 확인되었다. Model4로 train한 결과 이미지 재구성이 잘 되었으며, training단계에서 loss값이 낮아져서 제대로 fitting되었음을 확인할 수 있다.

- **Test result**



테스트 단계에서 실제 결함 이미지를 결함으로 판단한 예시이다.
53개의 결함 이미지 중에 49개를 결함으로 검출해내어 error rate를 낮춘다. 그러나 정상 이미지 28개 중 9개 만을 정상으로 판단하여, 정상에 대해서는 좋은 성능을 갖추지 못한다고 판단된다.

6. Discussion & Future work

1. Fabric 이미지의 특성상, Fabric 종류별 feature 추출이 필요하다. 본 프로젝트 데이터에서는 종류 별 이미지 수가 너무 적어서 불가능하다. 추가적인 데이터 수집을 통해 더 정확한 feature 추출이 가능할 것으로 보인다.
2. Real world에서는 결함 데이터가 훨씬 적지만, 본 프로젝트 데이터에서는 결함 데이터의 개수가 더 많다. 아키텍처 및 알고리즘은 그대로 활용할 수 있을 것이라 생각되지만, data augmentation을 통해 정상 데이터의 수를 더 늘려보고자 한다.
3. AutoEncoder 자체의 성능 한계로 인해 큰 어려움을 겪었다. VAE 등의 더 발전된 unsupervised 모델을 활용하고자 한다.

7. Reference

[1] A Public Fabric Database for Defect Detection Methods and Results, Silvestre-Blanes, Javier & Albero-Albero, Teresa & Miralles, Ignacio & Pérez-Llorens, Ruben & Moreno, Jorge, 2019Autex Research Journal. 19, 2019

[2] Fabric defect detection using deep learning, A. Şeker, K. A. Peker, A. G. Yüksek and E. Delibaş, Signal Processing and Communication Application Conference (SIU), Zonguldak, 2016