MVTec Anomaly Detection

Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). MVTec AD--A comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9592-9600).

김지원

jeewonkim@g.seoultech.ac.kr

Industrial Artificial Intelligence Lab

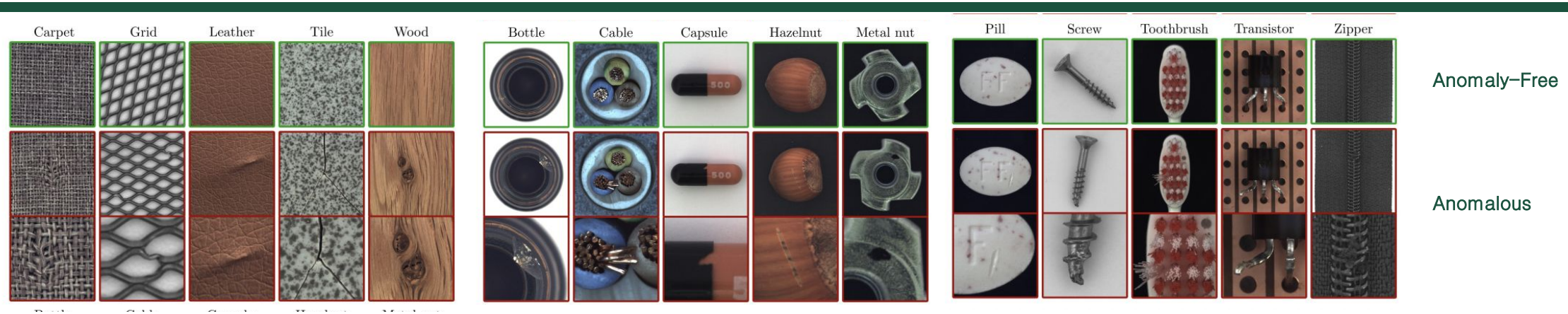
2023.01.1

Contents

Contents

MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection

Abstract



1. MVTec Anomaly Detection(MVTec AD) Dataset

- 5354개의 컬러 이미지 (object + texture)
- 구성 : 정상 이미지 + 결함 이미지(70 종류 이상의 결함)
- Anomaly 에 대한 ground truth 제공

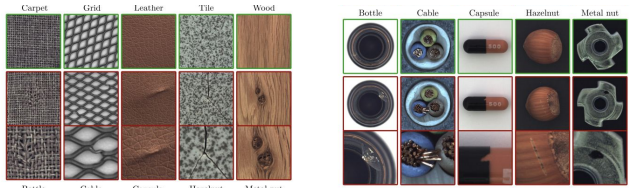
3. Unsupervised Anomaly Detection

- Convolutional Autoencoders
- GAN
- Feature descriptors using pre-trained CNN
- Classical Computer vision methods

Introduction

MVTec Anomaly Detection Dataset

- MNIST, CIFAR10, ImageNet 데이터셋이 컴퓨터 비전 영역에 놀라운 발전을 이끔
- Anomaly는 Machine Learning System에서 인식하기 어려움
- Training Data의 분포와 일치하는가의 여부를 판단하기 위한 다양한 알고리즘이 제안됨 -> 대부분 분류 (Classification)에 초점(outlier detection or one-class-classification)
- 하지만, 분류(Classificatoin)가 Anomaly Detection에서는 맞지 않을 수도 있음 -> Anomaly Region이 매우 작고(small), 감지하기 힘들며(subtle), 국한된 영역(confined region)이기 때문
- 또한, unsupervised anomaly detection을 위한 비교할만한 데이터셋이 존재하지 않음
- 이러한 시나리오에 대한 machine learning models를 발달시키기 위해 **적절한(suitable) 데이터가 필요!**



- 5354 high-resolution images
- 5 textures + 10 objects
- 73 types of anomalies
- Including 1888 of ground truth

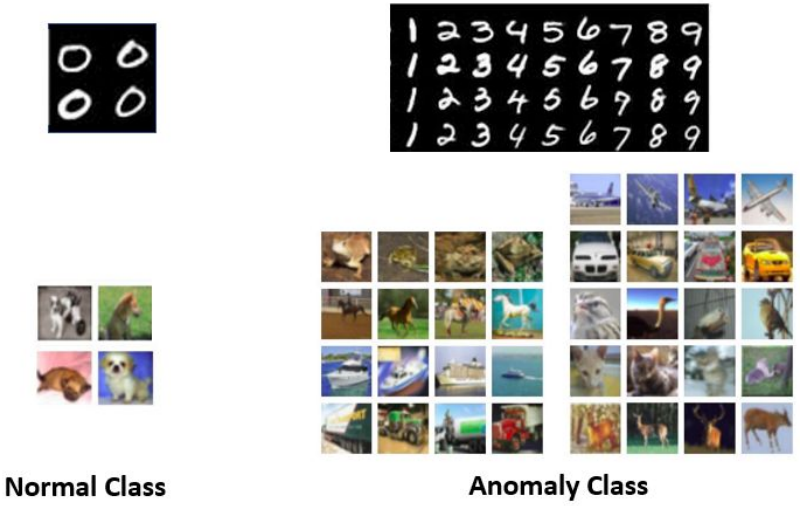
Evaluation

- Convolutional Autoencoders
- GAN
- Feature descriptors using pre-trained CNN
- Classical Computer vision methods

IAI
Industrial Artificial Intelligence Lab

Related Work – Existing Datasets for Anomaly Detection

Classification of Anomalous Images



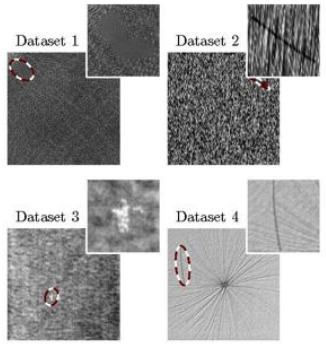
Multiclass classification - 이미 존재하는 데이터셋(ex-MNIST, CIFAR10, ImageNet) 사용

- 장점 : 많은 양의 train, test 데이터 사용 가능
- 단점 : Anomalous samples은 training 분포와 다르며, Test 진행시 training분포와의 차이가 심하지 않은 데이터에 대해 일반화가 불분명함 => Anomaly Detection에 대해 실용성이 떨어짐

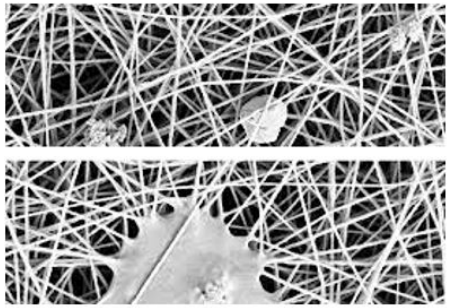
이후, PASCAL VOC dataset이 제시되었으나, Multiclass Classification과 마찬가지로 entire images에 집중하는 경향이 있음. (rather than finding the parts of the images that make them novel or anomalous.)

Related Work – Existing Datasets for Anomaly Detection

Segmentation of Anomalous Regions



Defects were generated by similar texture models



Only 5 defect-free images can be used for training

Only focus on the inspection of textured surfaces

NanoTWICE – 45 gray-scale nanofibrous material images

- Training (5 defect-free images), Test (40 images with anomalous regions)
- 단점 : 1가지 종류의 texture => 다른 texture에서의 일반화가 어려움

DAGM 2007 – 인위적으로 만든 10 classes of gray-scale textures, Defect에 원으로 표시가 되어 있음(Annotation)

- Training (1000 defect-free textures), Test (150 defective patches)
- 단점 : Annotation이 coarse함, 비슷한 texture 모델에서 생성되었기 때문 => 다른 texture사이의 variance가 낮음, 인위적으로 만들었기 때문에 실제 데이터의 근사치(approximation)를 보여줌

Related Work – Methods

Generative Adversarial Networks

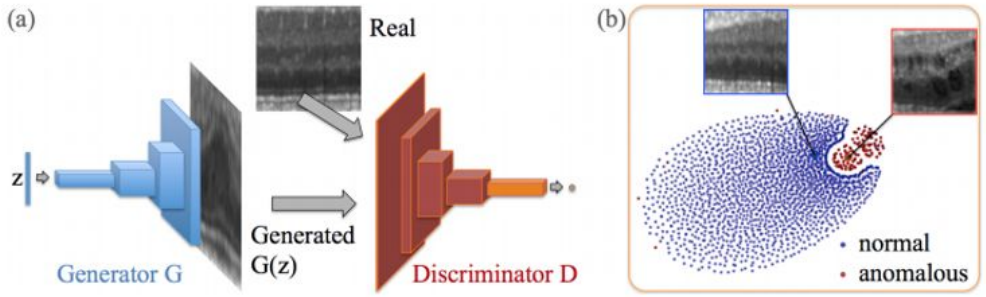


Fig. 2. (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator.

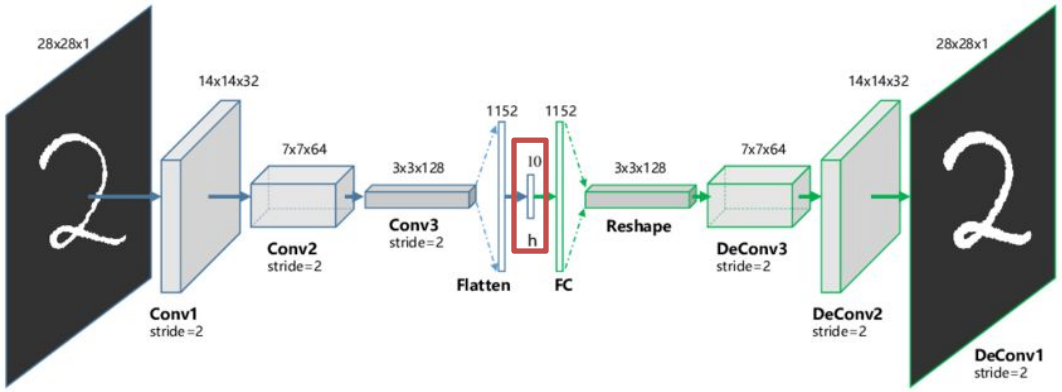
AnoGAN

GAN (AnoGAN)

- Training : 정상 이미지 => Generator가 실제와 같은 이미지를 생성할 수 있음
- Input Image를 잘 표현하는 **Latent Sample**를 찾을 수 있음
- Anomaly Segmentation은 **Reconstructed Image**와 Original Input을 비교하여 얻을 수 있음

Related Work – Methods

Deep Convolutional Autoencoders

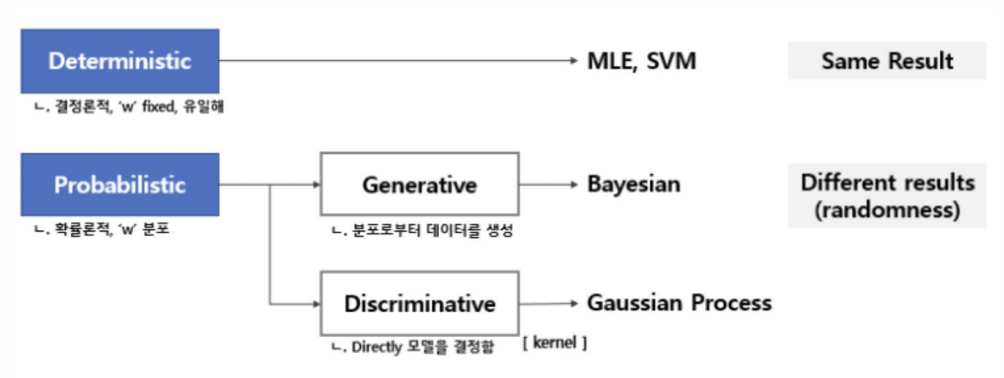


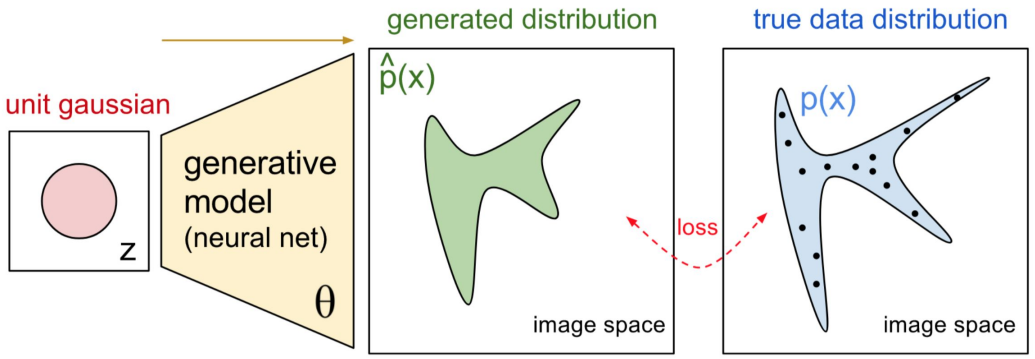
CAE (Convolutional Autoencoders)

- Bottleneck(latent space)를 통해 정상 이미지를 복원하고자 함
- 정상 이미지로 Autoencoder를 학습시킨 뒤 Test시 정상 데이터를 넣으면 잘 복원, 결함 데이터를 넣으면 잘 복원하지 못하는 특징 이용
- 주로 Deterministic한 CAE를 사용, VAE(Variational Autoencoder)나 다른 Generative Model들은 정상 데이터들의 true likelihood를 잘 모델링하지 못해 좋지 않은 성능을 보인다는 단점이 있음

Related Work – Methods

Deterministic Model VS Generative Model





Deterministic

- 정확한 수학적 관계식에 의해 예측됨
- 오차(불확실성)를 허용하지 않음
- Decision boundary 학습

Generative

- p(y), p(x|y)를 활용하여 p(y|x)를 간접 계산 (데이터 X가 생성되는 과정을 두 개의 확률모형, 즉 p(Y), p(X|Y)으로 정의하고 베이즈룰을 사용해 p(Y|X)를 간접적으로 도출하는 모델)
- Label 유무에 따라 지도/비지도 학습으로 구분
- 데이터 분포 X' 학습

Deterministic은 '경계선'을, Generative는 '분포'를 학습한다

IAI
Industrial Artificial Intelligence Lab

Related Work – Methods

Features of Pre-trained Convolutional Neural Networks

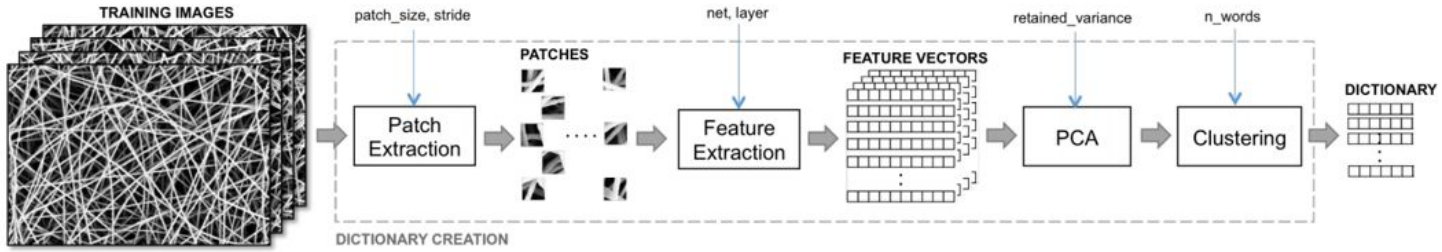


Figure 4. Examples of dictionary achieved considering different patch sizes and different number of subregions.

Feature Dictionary 기반 방법론

Spatial Anomaly Map

- Pre-trained ResNet-18 분류 네트워크를 통해 feature descriptions을 얻고자 함 -> 정상, 비정상 분류 위함, NanoTWICE 데이터셋에 좋은 성능을 보임
- 단점 : Binary Decision만 제공
- 위의 단점을 보완하기 위해선, Spatial Anomaly Map을 얻어야 함. 이상적으로 각 픽셀에 대해 평가해야 하지만 대용량 이미지에 대해서 성능 병목 현상(performance bottleneck)을 가져옴.
- 각 픽셀에 대해 평가하지 않을 경우 coarse한 anomaly map을 얻음

IAI
Industrial Artificial Intelligence Lab

Related Work – Methods

Traditional Methods

- 정상 이미지들로부터 hand-crafted feature를 추출
- Texture 데이터(Texture Inspection Model) : GMM(Gaussian Mixture Model)기반으로 Feature Vector 얻음 -> Anomaly는 Low Probability
- Non-Texture 데이터 : Shaped-based Matching기반의 Variation 모델. 각 픽셀에 대하여 mean과 standard deviation을 구하고 Gray-value(brightness) statistics을 모델링함. 테스팅 과정에서 각 픽셀의 평균에서 deviation을 계산함. -> 이상데이터는 Deviation이 Threshold보다 높음

Dataset Description

큰 도메인
Texture, Object

계

	Category	# Train (훈련 이미지 / 작은 도메인 15 가지 / 정상 이미지)	# Test (good) (테스트 이미지 / 정상 + 비정상)	# Test (defective) (테스트 이미지 / 정상 + 비정상)	# Defect groups (결함 종류)	# Defect regions (결함 픽셀 사이즈 합)	Image side length (이미지 사이즈)
Textures	Carpet	280	28	89	5	97	1024
	Grid	264	21	57	5	170	1024
	Leather	245	32	92	5	99	1024
	Tile	230	33	84	5	86	840
	Wood	247	19	60	5	168	1024
Objects	Bottle	209	20	63	3	68	900
	Cable	224	58	92	8	151	1024
	Capsule	219	23	109	5	114	1000
	Hazelnut	391	40	70	4	136	1024
	Metal Nut	220	22	93	4	132	700
	Pill	267	26	141	7	245	800
	Screw	320	41	119	5	135	1024
	Toothbrush	60	12	30	1	66	1024
	Transistor	213	60	40	4	44	1024
	Zipper	240	32	119	7	177	1024
	Total	3629	467	1258	73	1888	-

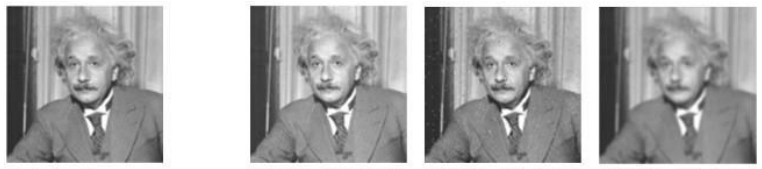
Benchmark – Evaluated Methods

AnoGAN

- Latent Space 차원 64, 생성 이미지 크기 128 * 128
- Object Evaluation : 훈련, 테스트 데이터 128*128
- Texture Evaluation : 512*512 확대 후 128*128로 크롭
- Training : 이미지 증강
- Testing : Patchwise Evaluation of 가로 세로 stride 사이즈 128*128 (작은 stride로 할 수도 있지만 런타임이 길어짐)

L2 and SSIM Autoencoder

- MVTec의 선행 연구인 Improving Unsupervised Defect Segmentation by Applying Structural Similarity to Autoencoders 의 구조를 그대로 사용
- CAE 사용
- Texture Evaluation : 128 * 128 사이즈 30 stride로 patch 복원, 픽셀별로의 L2비교 혹은 SSIM
- Object Evaluation : 256 * 256사이즈로 patch 복원, 픽셀별로의 L2비교 혹은 SSIM
- SSIM이 컬러 이미지에 적용되지 않아 SSIM-autoencoder를 사용할때는 gray-scale로 변환함
- 이미지 증강



SSIM (Structural Similarity Index Map) 이란?

휘도/대비/구조 측면에서 수치적인 에러가 아닌 인간의 시각적 화질 차이를 평가하기 위해 고안된 방법. (이미지 품질 측정법)

SSIM 값이 높을수록 원본 영상의 품질에 가깝다.

IAI
Industrial Artificial Intelligence Lab

Benchmark – Evaluated Methods

CNN Feature Dictionary

- Pre-trained ResNet-18의 512-dimensional avgpool layer 사용하여 Feature 추출
- 16 * 16 사이즈 patches 추출, 4 pixels stride로 coarse anomaly map 생성
- Object Evaluation : 256 * 256 사이즈
- Texture Evaluation : 512 * 512 사이즈
- ResNet feature extraction : Gray-scale 이미지는 세번 진행 (feature extractor는 3채널에서만 작동)

GMM-Based Texture Inspection Model, Variation Model

- HALCON machine vision library
- Library의 최적화를 통해 Anomaly Map을 얻음
- 자동적으로 Threshold 얻음

Benchmark – Data Augmentation

Texture

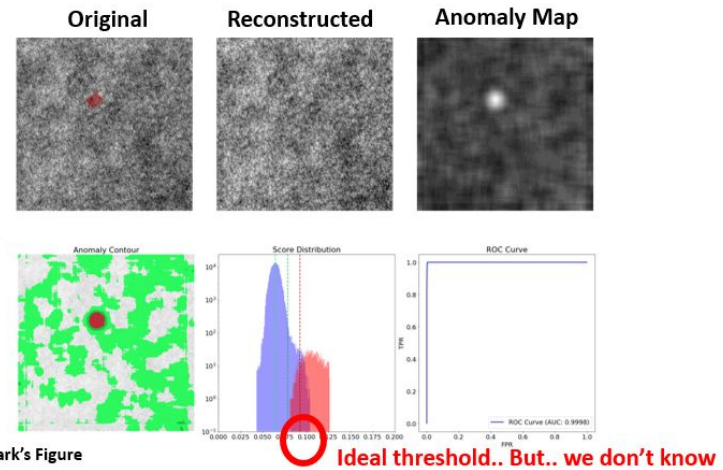
- 고정된 크기의 patch를 random crop하여 추출한 뒤, random rotation 적용

Object

- 이미지의 특성이 바뀌지 않는 선에서 random translation과 random rotation, random flip등을 적용. Object별로 어떤 증강을 적용했는지는 언급 X

- 각 카테고리별로 100,000 training patches 생성

Benchmark – Evaluation Metric



Reference: SUALAB Research Team Park's Figure

- 결과물 : One channel-spatial map(anomaly map), 각 픽셀에 대해서 binary decision 필요 -> Threshold가 결정되어야 함
- Threshold : 각 카테고리마다 결함의 **최소 크기**를 정의한 뒤, 학습에 쓰지 않은 정상 validation set에서 점점 threshold를 키워준 뒤, connected component들이 **최소 크기** 이하가 되는 시점의 threshold를 구하여 사용

1. **Classification**

- Threshold dependent, Threshold를 이용하여 분류하였을 때의 Accuracy

2. **Segmentation**

- Threshold independent, relative per-region overlap과 ROC AUC, (TPR – anomaly라고 잘 구분한 %, FPR – anomaly라고 잘못 구분한 %)

Benchmark – Results

	Category	AE (SSIM)	AE (L2)	AnoGAN	CNN Feature Dictionary	Texture Inspection	Variation Model
Textures	Carpet	**0.43**	0.57	0.82	0.89	0.57	-
		0.90	0.42	0.16	0.36	0.61	
	Grid	0.38	**0.57**	0.90	0.57	1.00	-
		1.00	**0.98**	0.12	0.33	0.05	
	Leather	0.00	0.06	0.91	**0.63**	0.00	-
		0.92	0.82	0.12	**0.71**	0.99	
	Tile	1.00	**1.00**	0.97	0.97	1.00	-
		0.04	**0.54**	0.05	0.44	0.43	
	Wood	0.84	1.00	0.89	**0.79**	0.42	-
		0.82	0.47	0.47	**0.88**	1.00	
Objects	Bottle	**0.85**	0.70	0.95	1.00	-	1.00
		0.90	0.89	0.43	0.06		0.13
	Cable	**0.74**	0.93	0.98	0.97	-	-
		0.48	0.18	0.07	0.24		
	Capsule	0.78	**1.00**	0.96	0.78	-	1.00
		0.43	**0.24**	0.20	0.03		0.03
	Hazelnut	1.00	**0.93**	0.83	0.90	-	-
		0.07	**0.84**	0.16	0.07		
	Metal nut	1.00	**0.68**	0.86	0.55	-	0.32
		0.08	**0.77**	0.13	0.74		0.83
	Pill	0.92	1.00	**1.00**	0.85	-	1.00
		0.28	0.23	**0.24**	0.06		0.13
	Screw	0.95	**0.98**	0.41	0.73	-	1.00
		0.06	**0.39**	0.28	0.13		0.10
	Toothbrush	0.75	**1.00**	1.00	1.00	-	1.00
		0.73	**0.97**	0.13	0.03		0.60
	Transistor	1.00	**0.97**	0.98	1.00	-	-
		0.03	**0.45**	0.35	0.15		
	Zipper	**1.00**	**0.97**	0.78	0.78	-	-
		0.60	**0.63**	0.40	0.29		

Classification
Top row – correctly classified for normal
Bottom row – correctly classified for anomaly

	Category	AE (SSIM)	AE (L2)	AnoGAN	CNN Feature Dictionary	Texture Inspection	Variation Model
Textures	Carpet	**0.69**	0.38	0.34	0.20	0.29	-
		0.87	0.59	0.54	0.72	**0.88**	
	Grid	**0.88**	0.83	0.04	0.02	0.01	-
		0.94	0.90	0.58	0.59	0.72	
	Leather	0.71	0.67	0.34	0.74	**0.98**	-
		0.78	0.75	0.64	0.87	**0.97**	
	Tile	0.04	**0.23**	0.08	0.14	0.11	-
		0.59	0.51	0.50	**0.93**	0.41	
	Wood	0.36	0.29	0.14	0.47	**0.51**	-
		0.73	0.73	0.62	**0.91**	0.78	
Objects	Bottle	0.15	**0.22**	0.05	0.07	-	0.03
		0.93	0.86	0.86	0.78		0.82
	Cable	0.01	0.05	0.01	**0.13**	-	-
		0.82	**0.86**	0.78	0.79		
	Capsule	0.09	**0.11**	0.04	0.00	-	0.01
		0.94	0.88	0.84	0.84		0.76
	Hazelnut	0.00	**0.41**	0.02	0.00	-	-
		0.97	0.95	0.87	0.72		
	Metal Nut	0.01	**0.26**	0.00	0.13	-	0.19
		0.89	0.86	0.76	0.82		0.60
	Pill	0.07	**0.25**	0.17	0.00	-	0.13
		0.91	0.85	0.87	0.68		0.83
	Screw	0.03	**0.34**	0.01	0.00	-	0.12
		0.96	**0.96**	0.80	0.87		0.94
	Toothbrush	0.08	**0.51**	0.07	0.00	-	0.24
		0.92	**0.93**	0.90	0.77		0.68
	Transistor	0.01	**0.22**	0.08	0.03	-	-
		0.90	0.86	0.80	0.66		
	Zipper	0.10	**0.13**	0.01	0.00	-	-
		0.88	0.77	0.78	0.76		

Segmentation
Top row – Relative per-region overlap
Bottom row – ROC AUC

1. **Texture image**

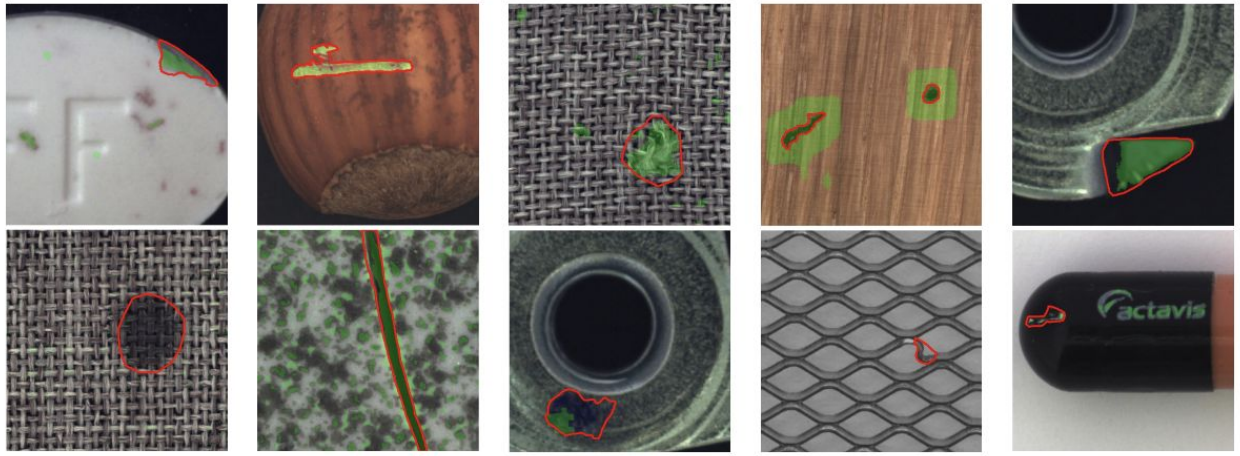
- 대체로 Autoencoder와 CNN Feature Dictionary 방법들이 좋은 성능을 보임

2. **Object image**

- Autoencoder가 다른 방법들에 비해 우수한 성능을 보이며, 특히 Relative per-region overlap 관점에서 보면 SSIM AE보다 L2 AE가 좋은 성능을 보임

Industrial Artificial Intelligence Lab

Benchmark – Results Example



(a) (b) (c) (d) (e)

(a) AnoGAN : Mode Collapse를 자주 겪었으며, 데이터셋의 shape, orientation등의 변화가 많은 데이터셋에 취약한 모습을 보였음

(b) Autoencoder : 대체로 안정적으로 학습되었고 성능도 좋았음. 하지만, 일부 category에 대해선 작은 디테일을 잘 복원하지 못하게 흐릿하게 복원을 하였고 (Autoencoder의 고질적인 문제), high-frequency 영역에 대해서도 잘 복원하지 못한(False Positive) 문제가 있음

(c) CNN Feature Dictionary : 원래 texture image에 적용하기 위해 제안되어 texture 데이터셋에서는 좋은 성능을 보였지만, object 데이터셋에서는 취약한 모습

(d) Texture Inspection Model : 전체적으로 낮은 성능

(e) Variation Model : 전체적으로 낮은 성능

Conclusions

Conclusion

- Unsupervised Anomaly Detection을 위한 잘 정의된, Real-World와 유사한 최초의 데이터셋을 제작하였다는 점에서 큰 의미를 가짐
- Threshold를 정하는 부분에선 아직까지 사람의 개입 존재하는 등 아쉬운 점이 있음

Future Work

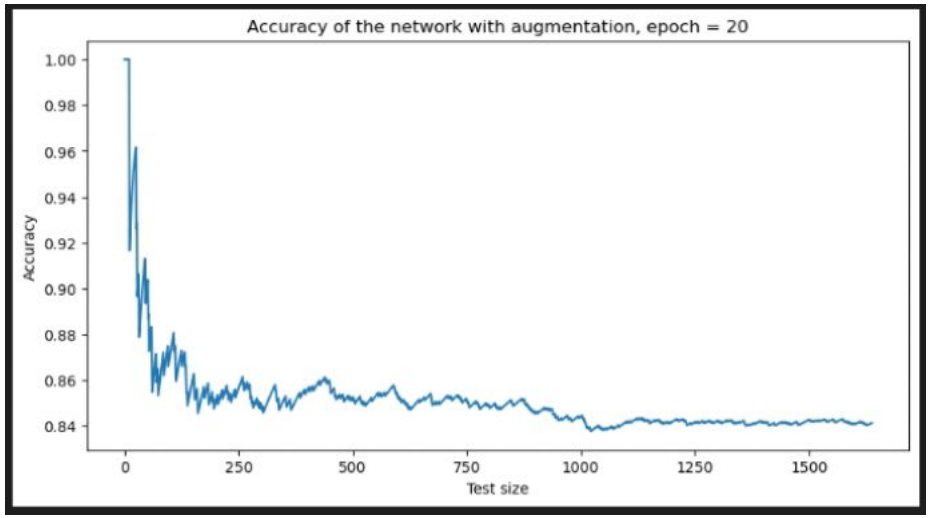
- 논문에서 제시된 데이터셋을 활용한 더 나은 unsupervised anomaly detection의 발달된 방법

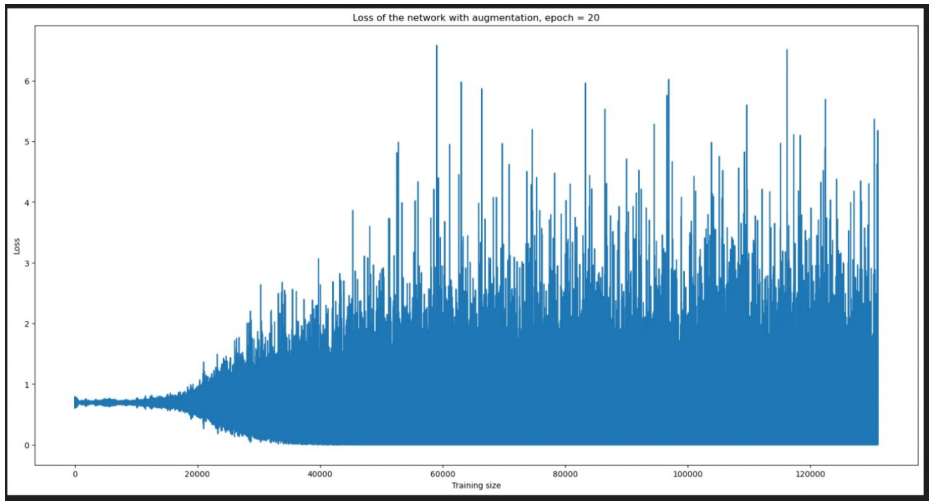
Code

Supervised – Classification

```python
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv2d(3,6,5)
        self.conv2 = nn.Conv2d(6,16,5)
        self.conv3 = nn.Conv2d(16,32,5)
        self.pool1 = nn.MaxPool2d(4,4)
        self.pool2 = nn.MaxPool2d(5,5)
        self.fc1 = nn.Linear(32*9*9, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 2)

    def forward(self,x):
        x = self.pool1(F.relu(self.conv1(x)))
        x = self.pool2(F.relu(self.conv2(x)))
        x = self.pool2(F.relu(self.conv3(x)))
        x = x.view(-1, 32*9*9)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

```python
model = ConvNet().to(device)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = learning_rate)
```





CNN - Augmentation 전

- Epoch 1 Accuracy : 78.86100386100387 %
- Epoch 20 Accuracy : 77.02702702702703 %

CNN - Augmentation 한 이미지 저장 후

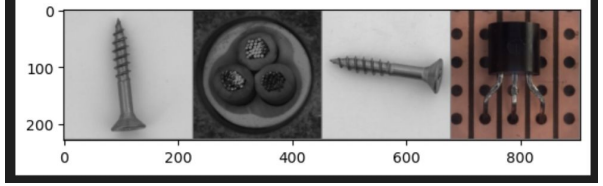
- Epoch 1 Accuracy : 49.35936546674802 %
- Epoch 10 Accuracy : 81.1470408785845
- Epoch 20 Accuracy : 84.1366687004271 %

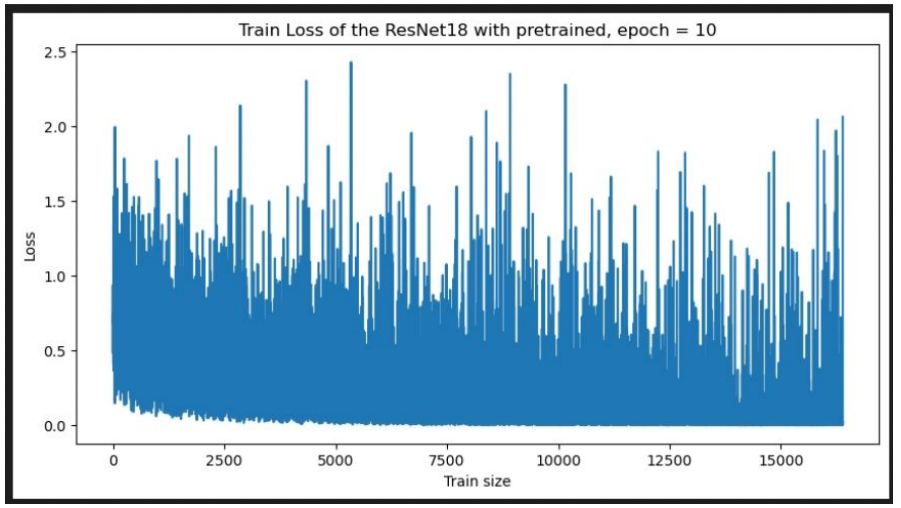
IAI
Industrial Artificial Intelligence Lab

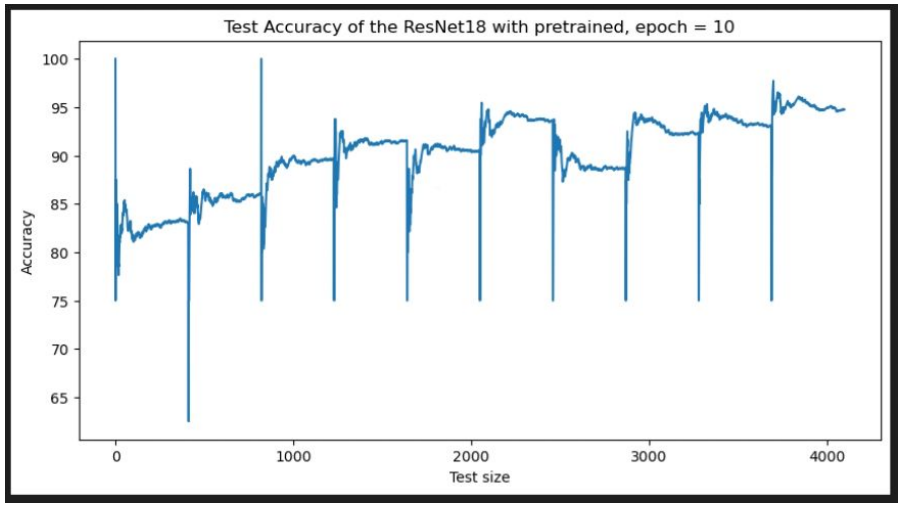
Supervised – Classification

```
model = models.resnet18(pretrained=False).cuda()
num_ftrs = model.fc.in_features
model.fc = nn.Linear(num_ftrs, 2)
model = model.cuda()
```

```
dataiter = iter(test_loader)
images, labels = next(dataiter)

imshow(torchvision.utils.make_grid(images))
print(images.shape)
```







Pre-trained 적용 전

- ResNet18 Epoch 1 Accuracy : 64.8462 %
- ResNet18 Epoch 10 Accuracy : 85.6010 %
- ResNet18 Epoch 10 Accuracy : 85.6010 %
- ResNet18 Epoch 20 Accuracy : 86.3331 %
- ResNet18 Epoch 50 Accuracy : 90.8481 %

Pre-trained 적용 후

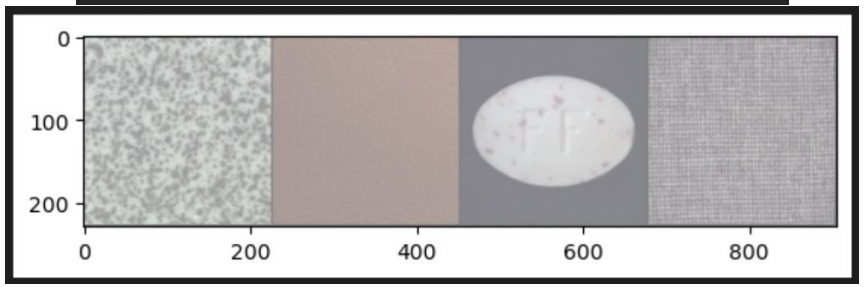
- ResNet18 Epoch 10 Accuracy : 94.7529 %
- VGG16 Epoch 10 Accuracy : 82.0622 %

Industrial Artificial Intelligence Lab

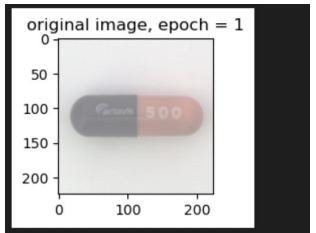
Unsupervised – CAE

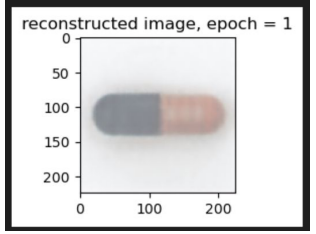
```python
# CNN Autoencoder

class CNN_Autoencoder(nn.Module):
  def __init__(self):
    super().__init__()
    # N, 3, 224, 224
    self.encoder = nn.Sequential(
        nn.Conv2d(3, 16, 4, stride = 2, padding = 1), # N, 16, 112, 112
        nn.ReLU(),
        nn.Conv2d(16, 32, 4, stride = 2, padding = 1), # N, 32, 56, 56
        nn.ReLU(),
        nn.Conv2d(32, 64, 4, stride = 2, padding = 1), # N, 64, 28, 28
        nn.ReLU(),
        nn.Conv2d(64, 128, 4, stride = 2, padding = 1), # N, 128, 14, 14
        nn.ReLU(),
        nn.Conv2d(128, 256, 14) # N, 256, 1, 1
    )

    self.decoder = nn.Sequential(
        nn.ConvTranspose2d(256, 128, 14),
        nn.ReLU(),
        nn.ConvTranspose2d(128, 64, 4, stride = 2, padding = 1),
        nn.ReLU(),
        nn.ConvTranspose2d(64, 32, 4, stride = 2, padding = 1),
        nn.ReLU(),
        nn.ConvTranspose2d(32, 16, 4, stride = 2, padding = 1),
        nn.ReLU(),
        nn.ConvTranspose2d(16, 3, 4, stride = 2, padding = 1),
        nn.Sigmoid()
    )

  def forward(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

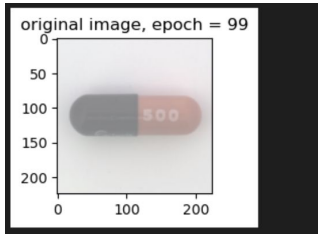
```python
transform = transforms.Compose(
    [transforms.Resize((224,224)),
    # 0~1 normalization
    transforms.ToTensor(),
    #  transforms.Normalize((0.5,0.5,0.5),(0.5,0.5,0.5))
     ]
)
```

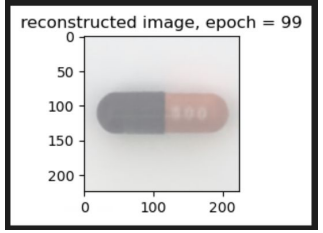


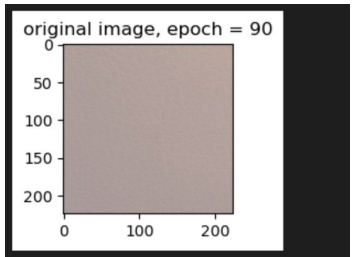
입력

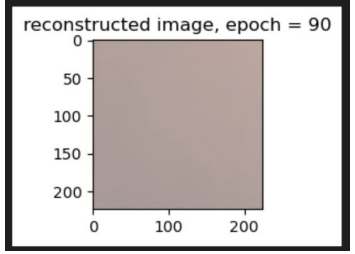


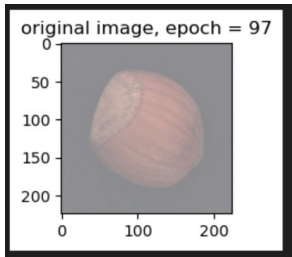


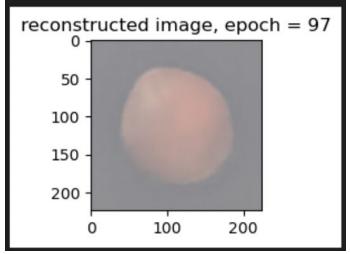












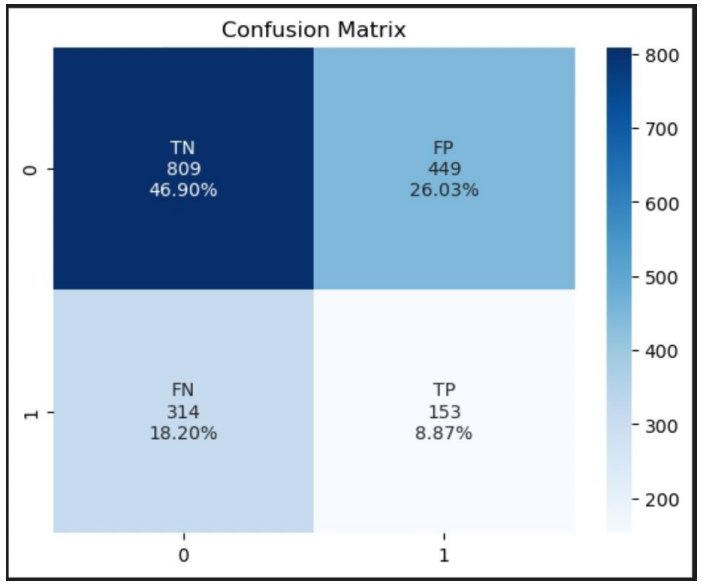
복원 결과

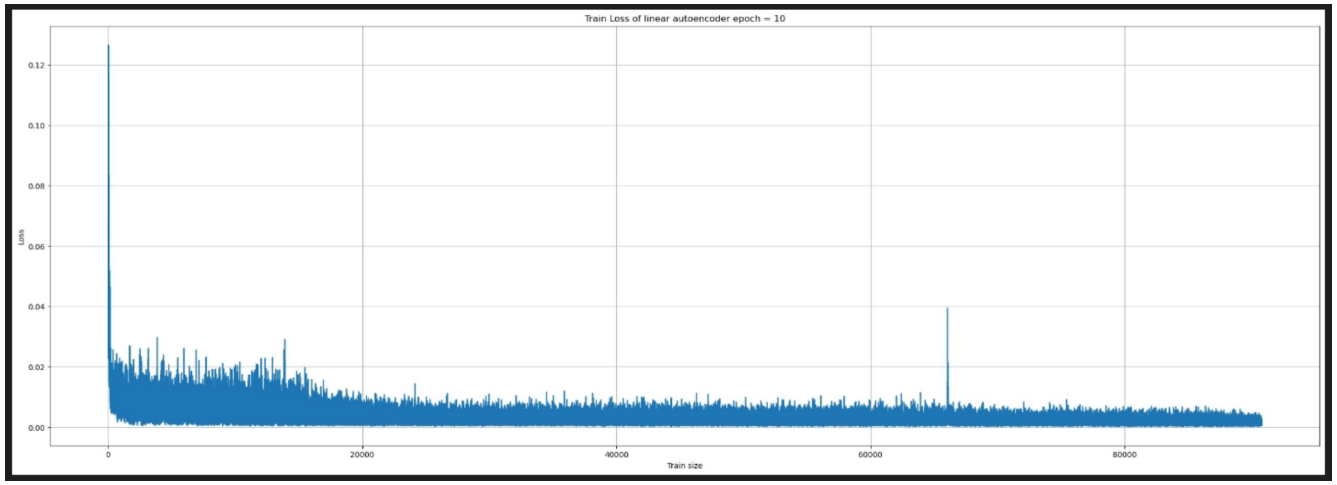
Unsupervised – CAE

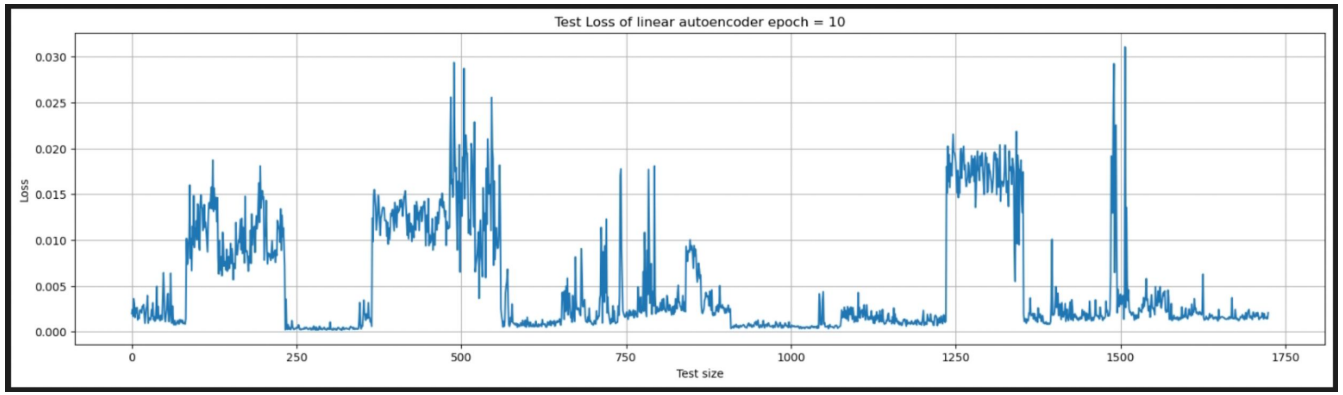
제일 좋았던 성능

```
# threshold = 25

threshold = np.percentile(df_trainloss['train_loss'], 25)
```

```
Accuracy is 0.5576811594202898
Precision is 0.7203918076580588
Recall is 0.6430842607313195
F1 score is 0.6795464090718185
```







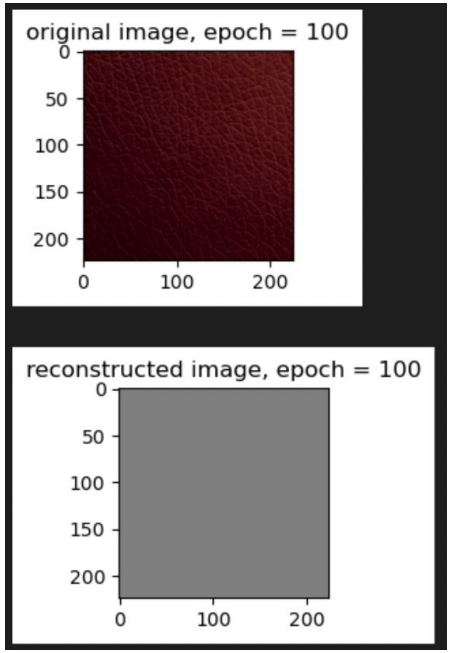
Industrial Artificial Intelligence Lab

Unsupervised – CAE

```python
transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize(mean_train, std_train),
])
```

```python
transform = transforms.Compose(
    [
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
    ]
)
```

정규화를 했지만 성능이 좋지 않았다



예시

감사합니다 ☺

Industrial Artificial Intelligence Lab