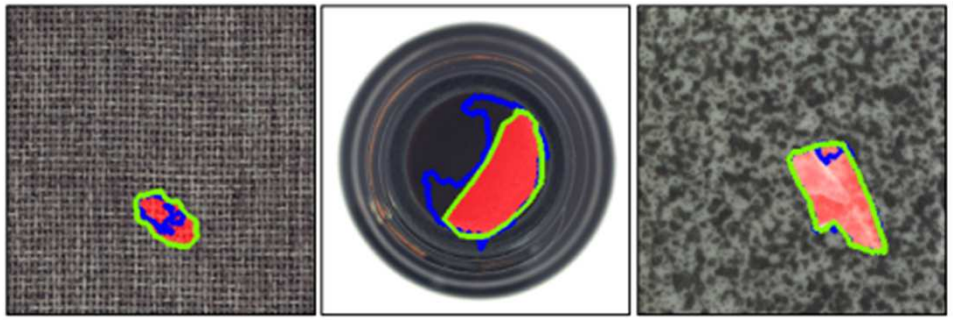
SSPCAB가 결합된 Reconstruction 기반 모델을 이용한 이미지의 재구성과 이상탐지

2024 1학기 산업공학캡스톤 1조

contents

- 1. 논문의 개요
- 2. SSPCAB의 원리
- 3. pytorch를 이용한 SSPCAB 구현
- 4. MVTec AD 데이터셋 소개
- 5. 제품 이미지의 재구성 및 이상탐지 실험
- 6. 결론, 한계 및 개선방안

Part 1, 논문의 개요



- 이미지의 이상탐지는 주로 이상이 있는 원본 이미지와 재구성된 이미지 사이의 오차를 측정한다.
- 정상 데이터를 통해 학습된 reconstruction 모델(CAE, GAN 등)을 통해 비정상 데이터를 재구성하여 비정상 데이터와 재구성된 데이터 사이의 error을 통해 이상을 판별한다.
- 논문에서는 reconstruction 모델을 자가 지도 예측 아키텍처 빌딩 블록에 결합할 것을 제안한다.

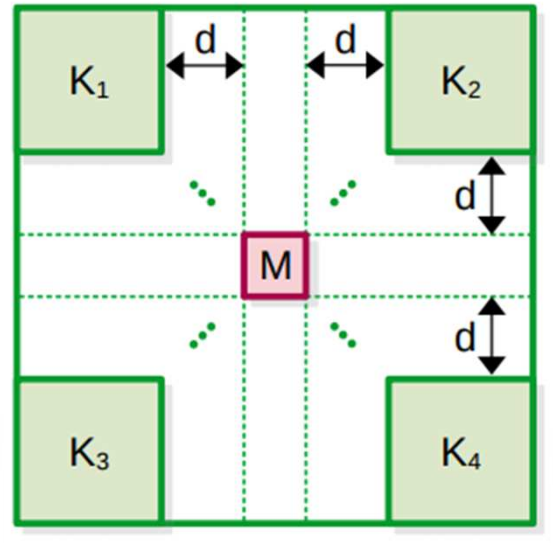
Part 1, 논문의 개요

Venue	Method	Avenue				ShanghaiTech			
		AUC		RBDC	TBDC	AUC		RBDC	TBDC
		Micro	Macro			Micro	Macro		
BMVC 2018	Liu *et al.* [38]	84.4	-	-	-	-	-	-	-
CVPR 2018	Sultani *et al.* [66]	-	-	-	-	-	76.5	-	-
ICASSP 2018	Lee *et al.* [32]	87.2		-	-	76.2		-	-
WACV 2019	Ionescu *et al.* [27]	88.9	-	-	-	-	-	-	-
ICCV 2019	Nguyen *et al.* [47]	86.9	-	-	-	-	-	-	-
CVPR 2019	Ionescu *et al.* [25]	87.4	90.4	15.77	27.01	78.7	84.9	20.65	44.54
TNNLS 2019	Wu *et al.* [73]	86.6		-	-	-	-	-	-
TIP 2019	Lee *et al.* [33]	90.0		-	-	-	-	-	-
ACMMM 2020	Yu *et al.* [77]	89.6	-	-	-	74.8	-	-	-
WACV 2020	Ramachandra *et al.* [50]	72.0		35.80	80.90	-	-	-	-
WACV 2020	Ramachandra *et al.* [51]	87.2		41.20	78.60	-	-	-	-
PRL 2020	Tang *et al.* [69]	85.1		-	-	73.0		-	-
Access 2020	Dong *et al.* [12]	84.9		-	-	73.7		-	-
CVPRW 2020	Doshi *et al.* [13]	86.4		-	-	71.6		-	-
ACMMM 2020	Sun *et al.* [67]	89.6		-	-	74.7		-	-
ACMMM 2020	Wang *et al.* [72]	87.0		-	-	79.3		-	-
ICCVW 2021	Astrid *et al.* [4]	84.7	-	-	-	73.7	-	-	-
BMVC 2021	Astrid *et al.* [3]	87.1	-	-	-	75.9	-	-	-
CVPR 2021	Georgescu *et al.* [17]	91.5	92.8	57.00	58.30	82.4	90.2	42.80	83.90
CVPR 2018	Liu *et al.* [37]	85.1	81.7	19.59	56.01	72.8	80.6	17.03	54.23
CVPR 2022	Liu *et al.* [37] + SSPCAB	**87.3**	**84.5**	**20.13**	**62.30**	**74.5**	**82.9**	**18.51**	**60.22**
CVPR 2020	Park *et al.* [49]	82.8	86.8	-	-	68.3	79.7	-	-
CVPR 2022	Park *et al.* [49] + SSPCAB	**84.8**	**88.6**	-	-	**69.8**	**80.2**	-	-
ICCV 2021	Liu *et al.* [39]	89.9	**93.5**	41.05	86.18	74.2	83.2	44.41	83.86
CVPR 2022	Liu *et al.* [39] + SSPCAB	**90.9**	92.2	**62.27**	**89.28**	**75.5**	**83.7**	**45.45**	**84.50**
TPAMI 2021	Georgescu *et al.* [18]	92.3	90.4	65.05	**66.85**	82.7	89.3	**41.34**	78.79
CVPR 2022	Georgescu *et al.* [18] + SSPCAB	**92.9**	**91.9**	**65.99**	64.91	**83.6**	**89.5**	40.55	**83.46**

- 이미지와 영상 데이터의 이상탐지를 위한 state- of- the- art 모델에 쉽게 결합할 수 있고, 대체로 성능의 향상을 보였다.

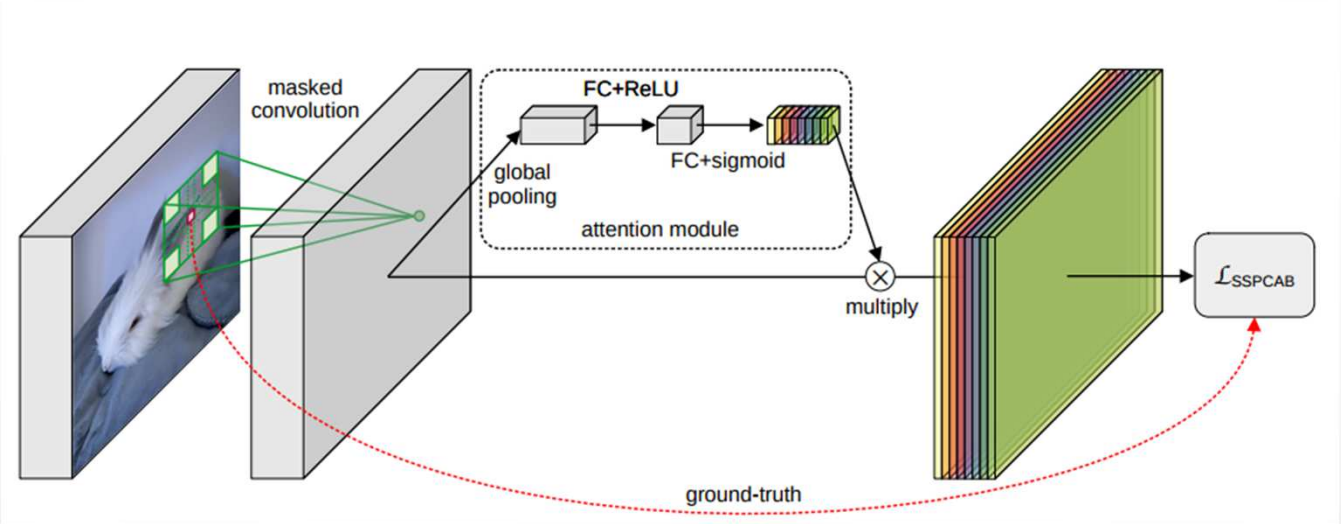
Part 2, SSPCAB의 원리 – Dilated Masked Filters

- Sub-kernel(k'), dilation(d)의 하이퍼파라미터를 가짐
- 중심부(M)의 사이즈는 $1 \times 1 \times C$
- 필터의 corner에 (W, H) = (k', k')인 4개의 sub-kerner을 포함
- Width와 Height는 각각 2k'+2d+1
- 이렇게 구성된 커널은 Squeeze and excitation 블록으로 넘어감
- 기본값으로 d = 1, k' = 1을 사용



Part 2, SSPCAB의 원리 – Channel Attentive Module

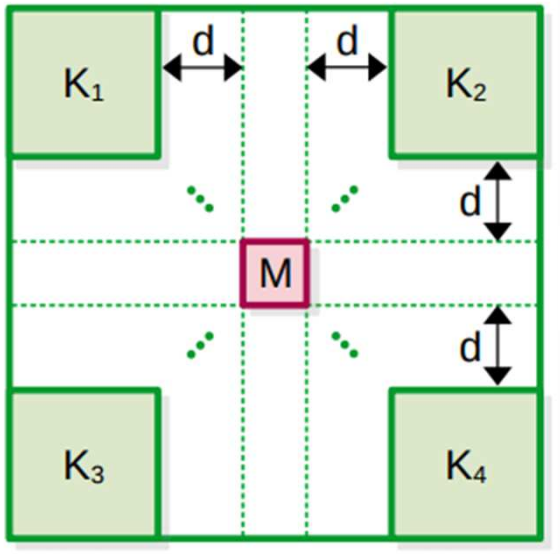
- Dilated Masked Filter의 중심부(M)에 해당하는 부분을 예측하여 masked convolution layer을 생성
- Global (mean) pooling을 통해 (1x1xC)의 텐서 생성
- 이 텐서를 C/reduction_ratio만큼 줄이고 다시 복구시키는 과정을 거쳐 각 channel 별 weight 텐서 생성
- masked convolution layer 과 weight 텐서 간 component wise 연산을 통해 reconstruction map 생성
- Input과 동일한 shape의 Output 생성



Part 3, Pytorch를 이용한 SSPCAB 구현

Dilated Masked Filters

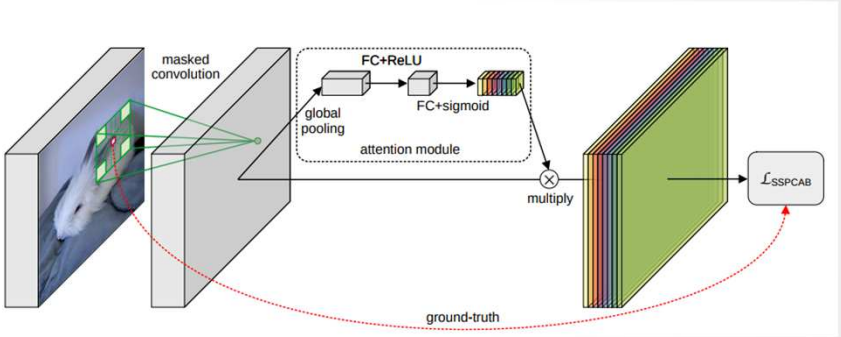
```
class SSPCAB(nn.Module):
    def __init__(self, channels, kernel_dim=1, dilation=1, reduction_ratio=8):
        super(SSPCAB, self).__init__()
        self.pad = kernel_dim + dilation
        self.border_input = kernel_dim + 2*dilation + 1
        self.relu = nn.ReLU()
        self.se = SELayer(channels, reduction_ratio=reduction_ratio)
        self.conv1 = nn.Conv2d(in_channels=channels,
                               out_channels=channels,
                               kernel_size=kernel_dim)
        self.conv2 = nn.Conv2d(in_channels=channels,
                               out_channels=channels,
                               kernel_size=kernel_dim)
        self.conv3 = nn.Conv2d(in_channels=channels,
                               out_channels=channels,
                               kernel_size=kernel_dim)
        self.conv4 = nn.Conv2d(in_channels=channels,
                               out_channels=channels,
                               kernel_size=kernel_dim)

    def forward(self, x):
        x = F.pad(x, (self.pad, self.pad, self.pad, self.pad), "constant", 0)
        x1 = self.conv1(x[:, :, :-self.border_input, :-self.border_input])
        x2 = self.conv2(x[:, :, self.border_input:, :-self.border_input])
        x3 = self.conv3(x[:, :, :-self.border_input, self.border_input:])
        x4 = self.conv4(x[:, :, self.border_input:, self.border_input:])
        x = self.relu(x1 + x2 + x3 + x4)
        x = self.se(x)
        return x
```



Part 3, Pytorch를 이용한 SSPCAB 구현

Channel attentive Module

```
class SELayer(nn.Module):
    def __init__(self, num_channels, reduction_ratio=8):
        '''
            num_channels: The number of input channels
            reduction_ratio: The reduction ratio 'r' from the paper
        '''
        super(SELayer, self).__init__()
        num_channels_reduced = num_channels // reduction_ratio
        self.reduction_ratio = reduction_ratio
        self.fc1 = nn.Linear(num_channels, num_channels_reduced, bias=True)
        self.fc2 = nn.Linear(num_channels_reduced, num_channels, bias=True)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, input_tensor):
        batch_size, num_channels, H, W = input_tensor.size()
        squeeze_tensor = input_tensor.view(batch_size, num_channels, -1).mean(dim=2)

        # channel excitation
        fc_out_1 = self.relu(self.fc1(squeeze_tensor))
        fc_out_2 = self.sigmoid(self.fc2(fc_out_1))

        a, b = squeeze_tensor.size()
        output_tensor = torch.mul(input_tensor, fc_out_2.view(a, b, 1, 1))
        return output_tensor
```



Part 3, Pytorch를 이용한 SSPCAB 구현

Hyperparameters

Method	Loss type	d	k'	r	Attention type	AUC Micro	AUC Macro	RBDC	TBDC
Plain auto-encoder	-	-	-	-	-	80.0	83.4	49.98	51.69
	MAE	0	1	-	-	83.3	84.1	47.46	52.11
		1	1	-	-	83.9	84.6	49.05	52.21
		2	1	-	-	83.2	84.3	48.56	52.03
	MSE	0	1	-	-	83.6	84.2	47.86	52.21
		1	1	-	-	84.2	84.9	49.22	52.29
		2	1	-	-	83.6	84.3	48.44	51.98
	MSE	0	2	-	-	83.7	84.0	47.41	53.02
		1	2	-	-	84.0	85.1	48.22	51.84
		2	2	-	-	82.7	83.1	46.94	50.22
	MSE	0	3	-	-	82.6	83.7	48.28	51.91
		1	3	-	-	82.9	84.7	48.13	52.07
		2	3	-	-	83.1	83.8	47.13	49.96
	MSE	1	1	8	CA	**85.9**	**85.6**	53.81	**56.33**
		1	1	-	SA	84.3	84.4	53.31	53.41
		1	1	8	CA+SA	85.7	85.6	**53.98**	54.11
	MSE	1	1	4	CA	85.6	85.3	53.83	55.99
		1	1	16	CA	84.4	84.9	53.28	54.37

4개의 하이퍼파라미터 포함

- channels:
  CNN의 채널 수
- kernel_dim (default : 1)
  sub-kerner의 크기
- dilation (default : 1)
  mask와 sub-kerner의 간격
- reduction_ratio (default : 8)
  채널 수의 감소율

```
class SSPCAB(nn.Module):
    def __init__(self, channels, kernel_dim=1, dilation=1, reduction_ratio=8):
```

Default 값들은 SSPCAB를 오토 인코더에 결합했을 때 성능을 기준으로 정함

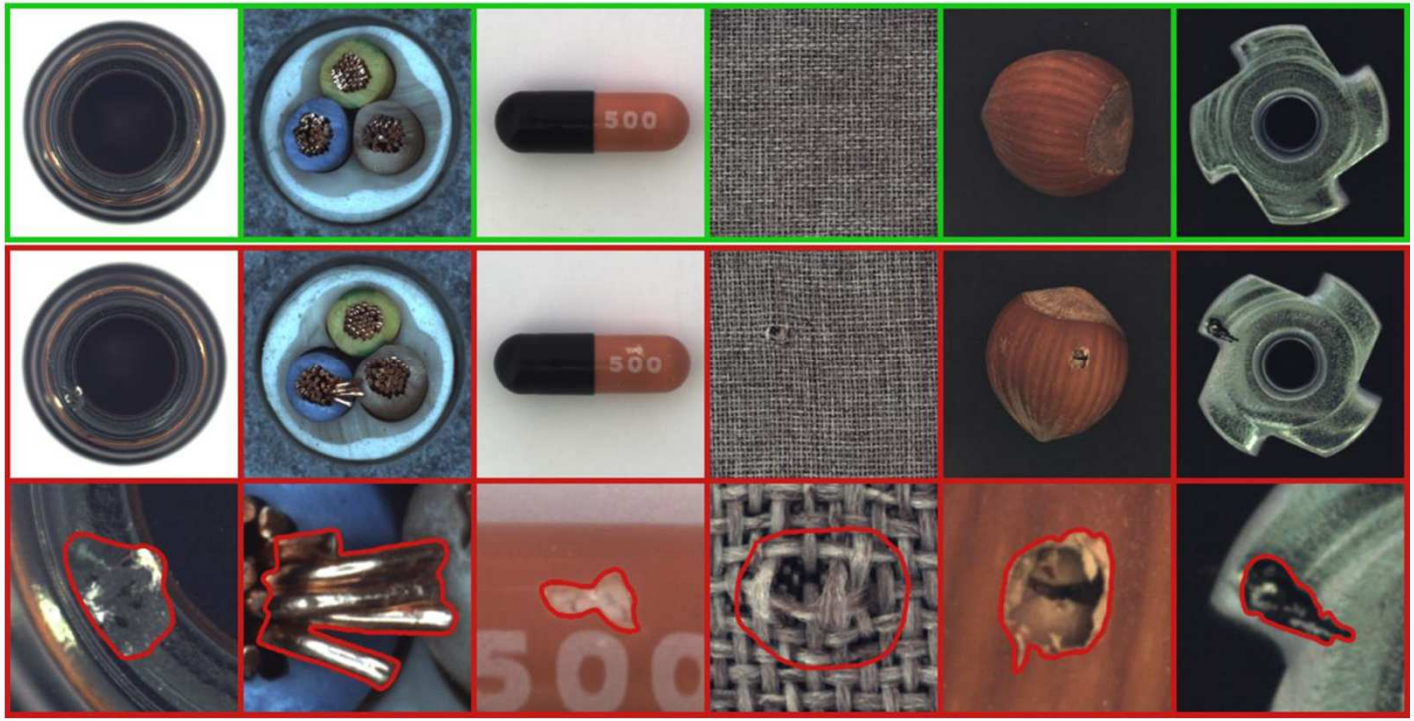
Part 3, Pytorch를 이용한 SSPCAB 구현

활용 예시

```
class Encoder_SSPCAB(nn.Module):
    def __init__(self, encoded_space_dim,fc2_input_dim):
        super().__init__()

        self.encoder_cnn = nn.Sequential(
            nn.Conv2d(1, 8, 3, stride=2, padding=1),

            nn.ReLU(True),
            nn.Conv2d(8, 16, 3, stride=2, padding=1),

            nn.BatchNorm2d(16),
            nn.ReLU(True),
            nn.Conv2d(16, 32, 3, stride=2, padding=0),

            SSPCAB(32),
            nn.ReLU(True)
        )
```

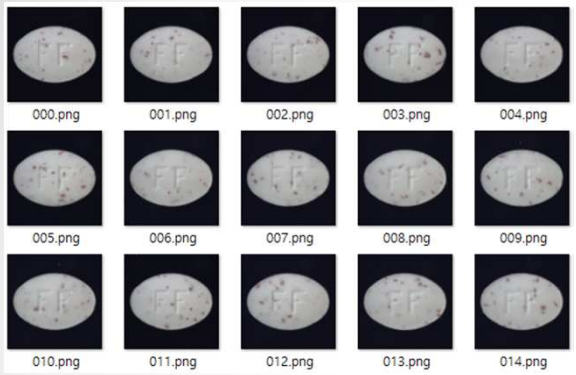
이전 Layer의 채널 수를 input parameter로 한다.

Part 4, MVTec AD 데이터셋 소개

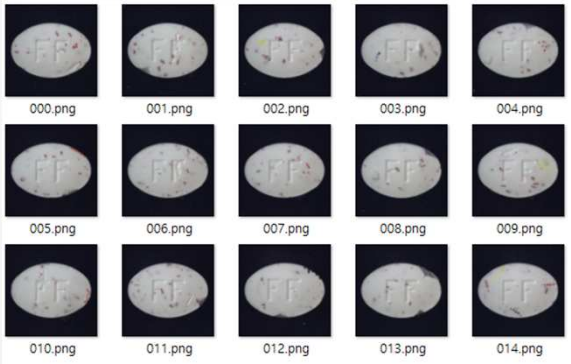


- 산업 검사에 초점을 둔 이상탐지 방법을 벤치마킹하기 위한 데이터셋
- 정상 이미지로 구성된 train set과 비정상 이미지를 포함하는 test set으로 구성
- 연구 및 개발 목적으로 공개된 데이터셋

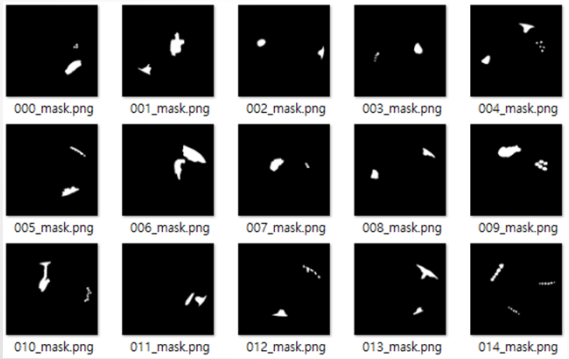
Part 4, MVTec AD 데이터셋 소개



- Train set:
  정상 제품들만으로 구성



- Test set:
  결함이 있는 제품들과 일부 정상품들로 구성



- Ground truth:
  Test set에서 결함에 해당하는 영역을 표시

Part 5, 제품 이미지의 재구성 및 이상탐지 실험

Reconstruction 기반 모델인 Convolutional Auto Encoder 정의
SSPCAB 계층을 추가한 Convolutional Auto Encoder 정의

```python
class Autoencoder(nn.Module):
    def __init__(self, input_shape=(3, 126, 126)):
        super(Autoencoder, self).__init__()

        self.encoder = nn.Sequential(
            self.conv_block(3, 32),
            nn.MaxPool2d(2, stride=2),
            self.conv_block(32, 64),
            nn.MaxPool2d(3, stride=3),
            self.conv_block(64, 128),
        )

        self.bottleneck = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

        self.decoder = nn.Sequential(
            self.conv_block(128, 128),
            nn.ConvTranspose2d(128, 64, kernel_size=3, stride=3),
            self.conv_block(64, 64),
            nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2),
            self.conv_block(32, 32),
            nn.Conv2d(32, 3, kernel_size=1),
            nn.Sigmoid()
        )

    def conv_block(self, in_channels, out_channels):
        return nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

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

Convolutional Auto Encoder

```python
class Autoencoder_SSPCAB(nn.Module):
    def __init__(self, input_shape=(3, 126, 126)):
        super(Autoencoder_SSPCAB, self).__init__()

        self.encoder = nn.Sequential(
            self.conv_block(3, 32),
            nn.MaxPool2d(2, stride=2),
            self.conv_block(32, 64),
            nn.MaxPool2d(3, stride=3),
            self.conv_block(64, 128),
            SSPCAB(128, kernel_dim = 1)   # SSPCAB 계층 추가
        )

        self.bottleneck = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

        self.decoder = nn.Sequential(
            self.conv_block(128, 128),
            nn.ConvTranspose2d(128, 64, kernel_size=3, stride=3),
            self.conv_block(64, 64),
            nn.ConvTranspose2d(64, 32, kernel_size=2, stride=2),
            self.conv_block(32, 32),
            nn.Conv2d(32, 3, kernel_size=1),
            nn.Sigmoid()
        )

    def conv_block(self, in_channels, out_channels):
        return nn.Sequential(
            nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1),
            nn.ReLU(inplace=True)
        )

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

Convolutional Auto Encoder
With SSPCAB

Part 5, 제품 이미지의 재구성 및 이상탐지 실험

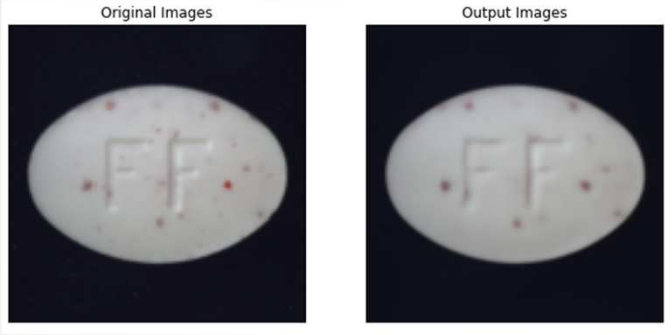
정상 데이터로 구성된 train set으로 모델 학습

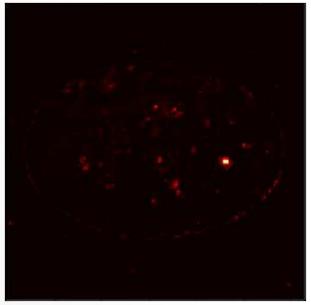
```
epochs = 500
learning_rate = 0.001

model = Autoencoder()
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

train_losses = []

for epoch in range(epochs):
    running_loss = 0.0

    for data in train_loader:
      img = data
      img = img.view(img.size(0), 3, 126, 126)
      output = model(img)
      loss = criterion(output, img)
      running_loss += loss.item()

      train_losses.append(loss.item())


      optimizer.zero_grad()
      loss.backward()
      optimizer.step()

    if (epoch+1) % 10 == 0 or epoch == 0:
        print(f'Epoch: {epoch+1}, Loss: {running_loss / len(train_loader)}')
```

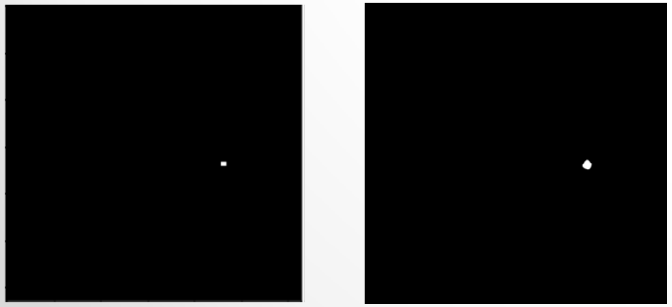
- Loss Function : MSE
- Optimizer : Adam
- Epoch : 500

Part 5, 제품 이미지의 재구성 및 이상탐지 실험

재구성된 이미지와 원본 이미지와의 차이를 확인하고
이상 영역 탐지







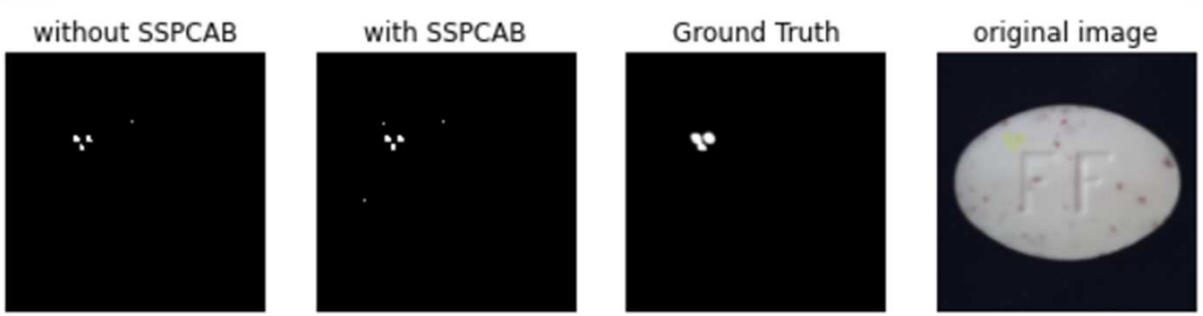
- 이상이 있는 이미지(왼쪽)을 모델에 입력하여 재구성 이미지를 구함(오른쪽)
- 두 이미지 간의 차이를 확인
- 차이가 임계를 넘어가는 지점을 표시(왼쪽)하고, ground truth(오른쪽)와 비교

Part 5, 제품 이미지의 재구성 및 이상탐지 실험

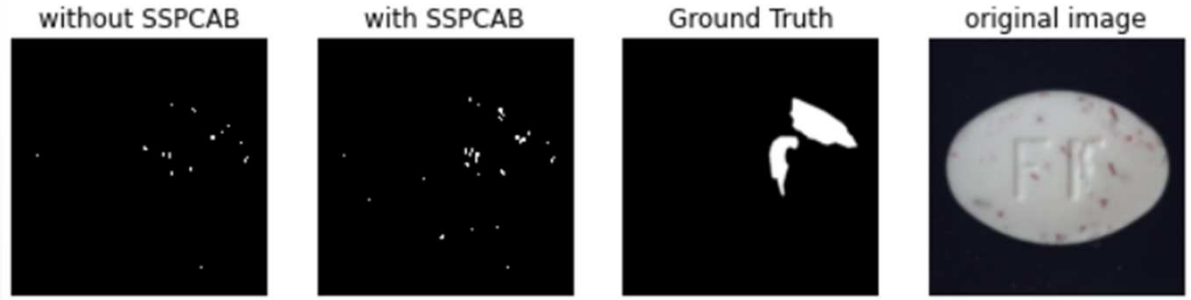
CAE와 CAE+SSPCAB의 reconstruction image 비교

Part 5, 제품 이미지의 재구성 및 이상탐지 실험

원본 이미지와 재구성 이미지간 RGB값의 MSE를 구하여
임계값(0.02)을 초과하는 영역을 이상영역으로 표시



- RGB 값의 차이가 두드러진 결함(변색 등)은 ground truth와 유사함을 보임



- RGB 값의 차이가 적은 결함(작은 깨짐이나 스크래치 등)은 ground truth와 차이가 발생

Part 6, 결론, 한계 및 개선방안

- SSPCAB는 reconstruction 성능을 높이는 데에 도움을 줌
- 재구성 이미지와 원본이미지의 RGB 값의 MSE를 통한 이상영역 탐지 방법은 RGB값의 차이가 적은 영역에 대해 한계를 보임
- 실제 산업현장에 활용하기 위해서는 original과 reconstructed 이미지간의 차이를 잘 확인하는 모델을 추가로 필요함
- 논문상에는 SSPCAB를 결합한 모델의 종류는 표기되어 있지만, 구체적 모델 구조는 설명되어있지 않음. 논문의 설명을 통해 정황상 유추 및 실험을 통한 모델 설계로 탐지를 진행함.

Thank you