Segmentation

2025-1 Mobility UR

2025.05.15 소프트웨어학과 김유진

I. Decoder & Batch size

II. U-net

III. Conclusion

Problem Setting

| Model

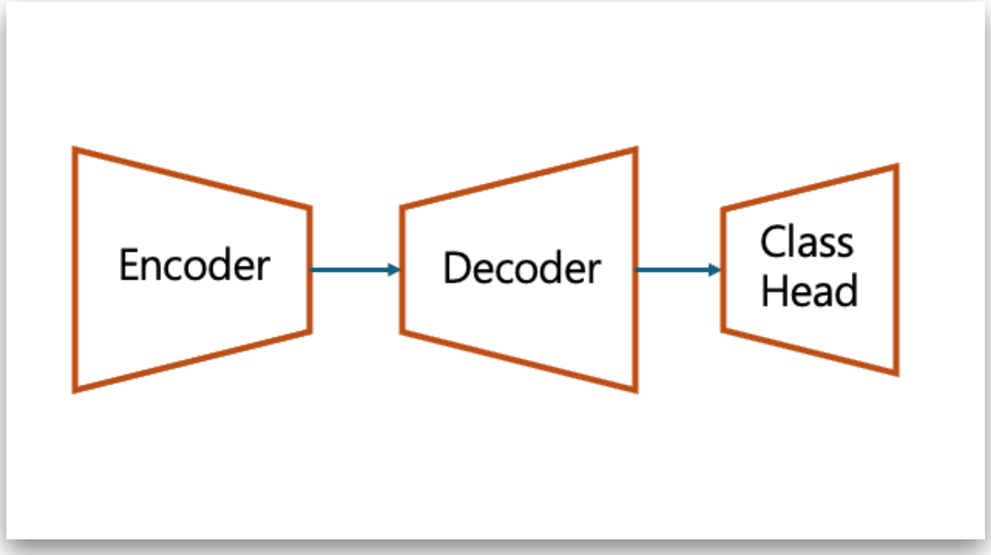
수정 전

- **Decoder** : CNN 기반의 4개의 디코더를 개별 학습
- **Batch size** : 4

수정 후

- **Decoder** : CNN 기반의 1개의 디코더 내 4개의 채널이 각 객체 학습
- **Batch size** : 64

모델 아키텍쳐



Decoder

| Model

```python
class Decoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(32, 128, 7),
            nn.ReLU(),
            nn.ConvTranspose2d(128, 64, 3, stride=1, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, 3, stride=1, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.decoder(x)
```

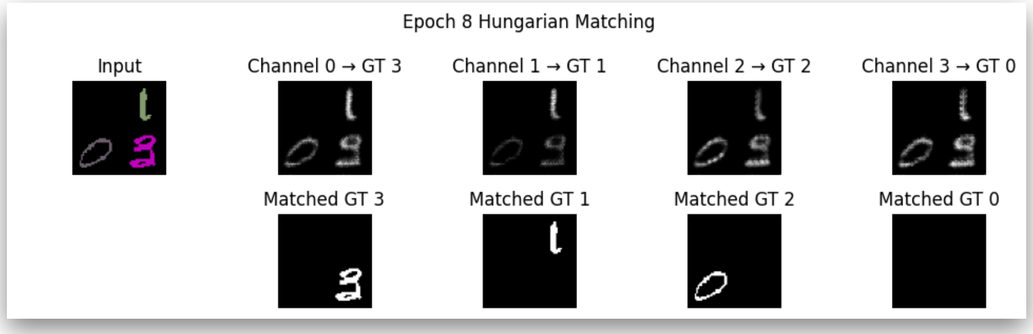
```python
class Decoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(128, 64, 3, stride=1, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(64, 32, 3, stride=1, padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 4, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.decoder(x)
```

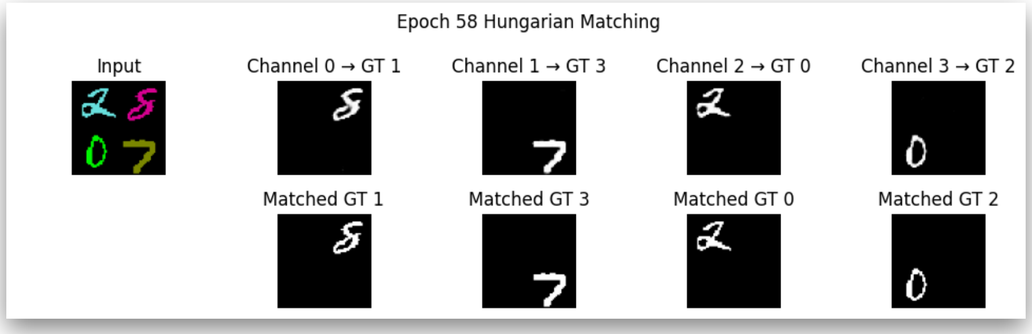
```python
class MultiSegNet(nn.Module):
    def __init__(self, num_instances=4):
        super().__init__()
        self.encoder = Encoder()
        self.decoders = nn.ModuleList([Decoder() for _ in range(num_instances)]) # Decoder 최대 객체 수 만큼 분할
    def forward(self, x):
        f = self.encoder(x)
        return torch.stack([dec(f) for dec in self.decoders], dim=1)  # (B, 4, 1, 64, 64) -> B : batch
```

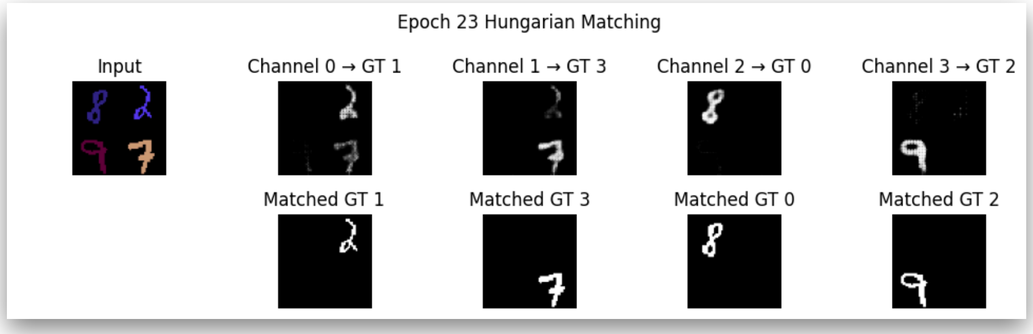
```python
class MultiSegNet(nn.Module):
    def __init__(self, num_instances=4):
        super().__init__()
        self.encoder = Encoder()
        self.decoder = Decoder()
    def forward(self, x):
        f = self.encoder(x)
        out = self.decoder(f)
        return out
```

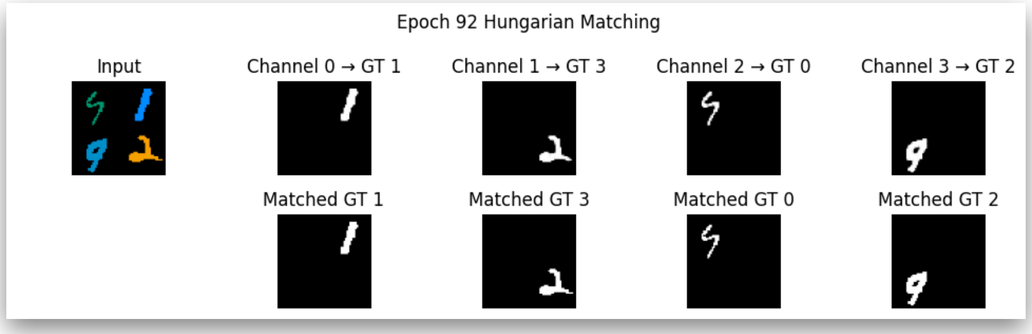
Decoder

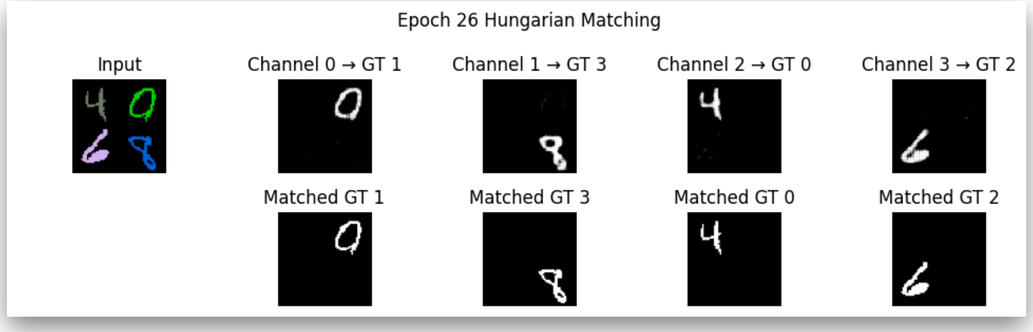
| Train - Batch size = 64, epoch = 100

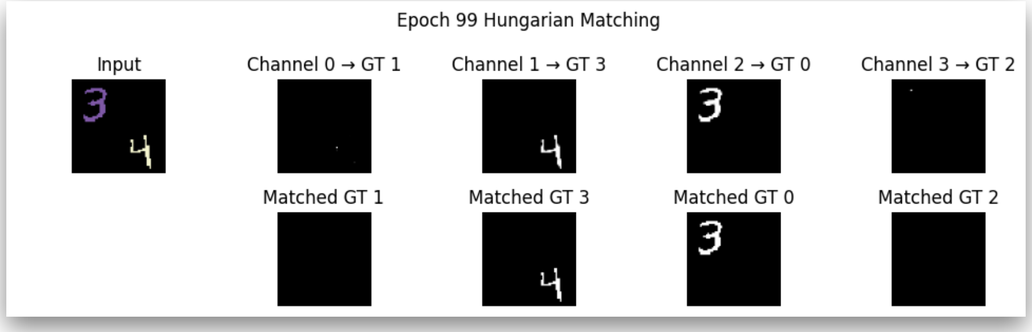






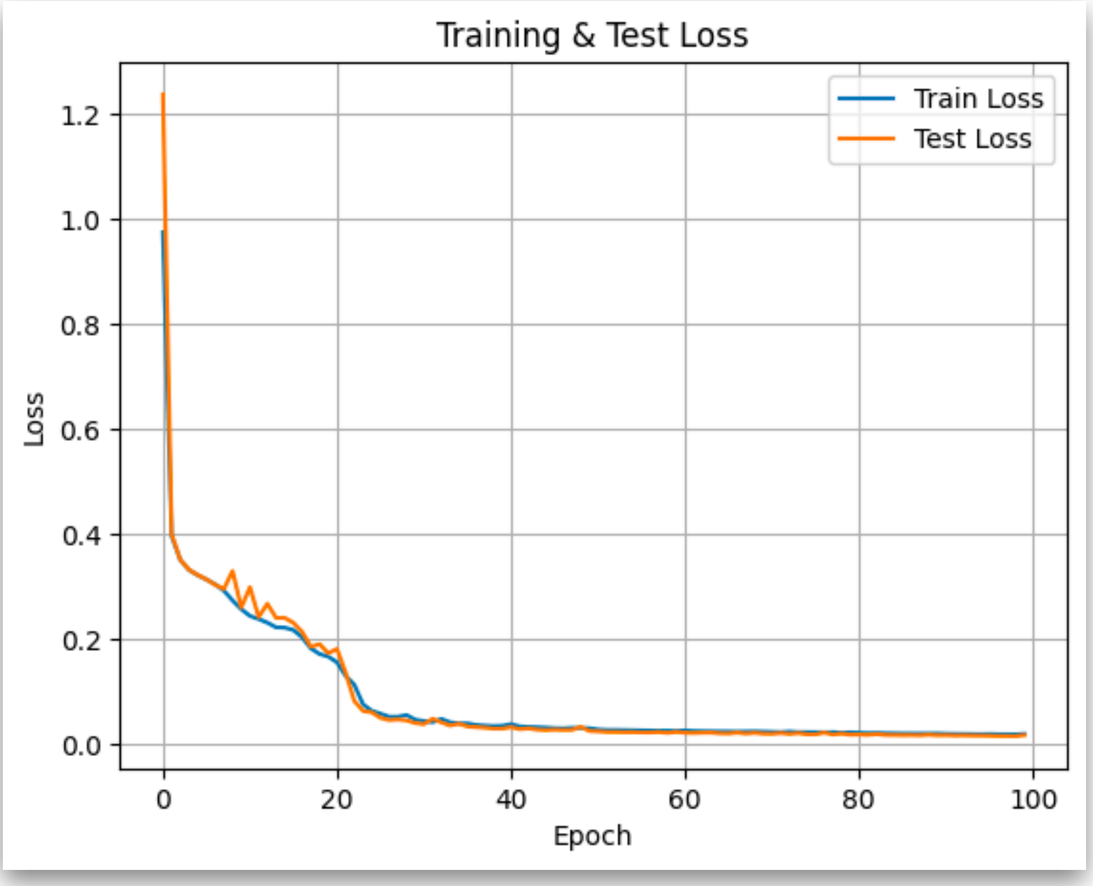






Decoder

| Train - Batch size = 64, epoch = 100



Min Train Loss : 0.0178

Min Test Loss : 0.0146

batch_size = 4

- Min Train Loss : 0.0108
- Min Test Loss : 0.0149

Classification

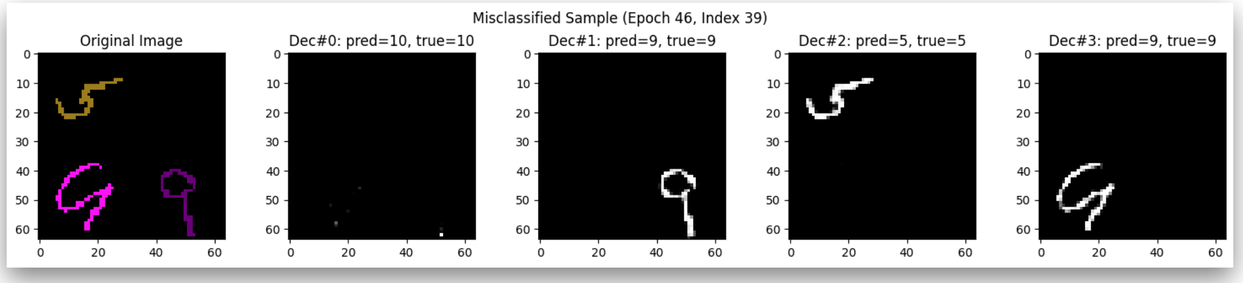
| Train - Batch size = 64, epoch = 50

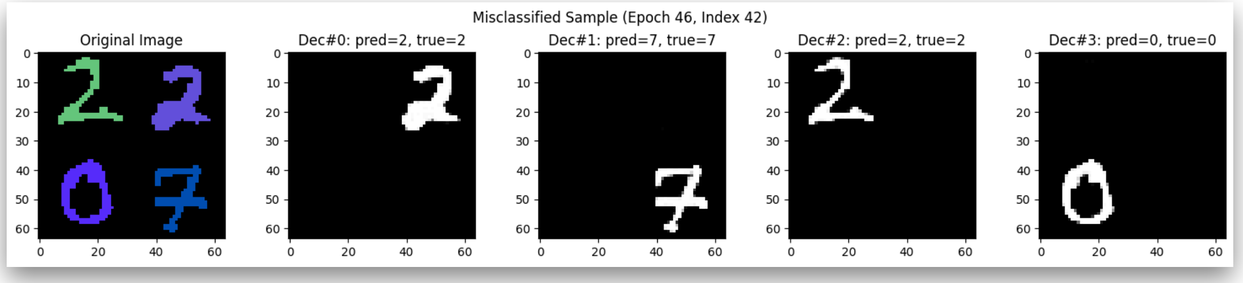
Min Train Loss : 0.0963

Min Test Loss : 0.0964

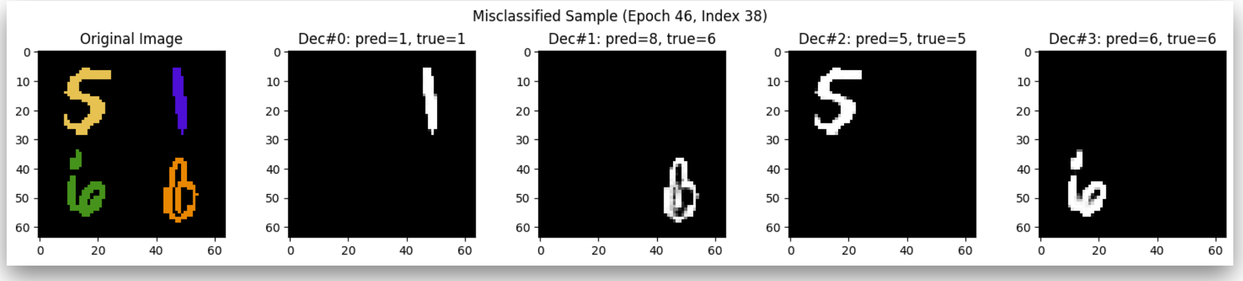
Best Test Acc : 97.26

Right





Wrong



I. Decoder & Batch size

II. U-net

III. Conclusion

U-net

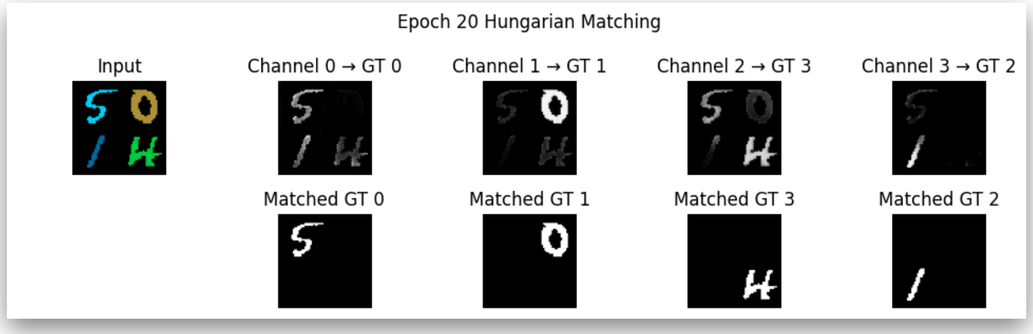
| Model

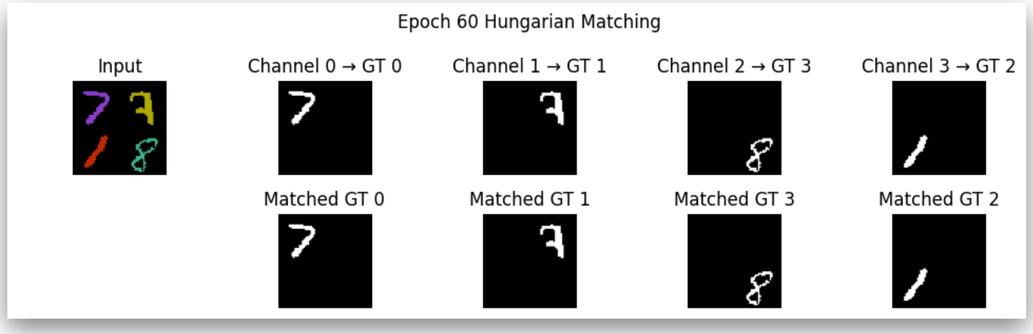
```python
class UNetLight(nn.Module):
    def __init__(self, in_channels=3, out_channels=4):
        super(UNetLight, self).__init__()

        # Encoder
        self.enc1 = self.conv_block(in_channels, 16)
        self.enc2 = self.conv_block(16, 32)
        self.enc3 = self.conv_block(32, 64)
        self.enc4 = self.conv_block(64, 128)
        self.enc5 = self.conv_block(128, 128)

        # Decoder
        self.up4 = nn.ConvTranspose2d(128, 128, kernel_size=2, stride=2)
        self.dec4 = self.conv_block(128 + 128, 64)

        self.up3 = nn.ConvTranspose2d(64, 64, kernel_size=2, stride=2)
        self.dec3 = self.conv_block(64 + 64, 32)

        self.up2 = nn.ConvTranspose2d(32, 32, kernel_size=2, stride=2)
        self.dec2 = self.conv_block(32 + 32, 16)

        self.up1 = nn.ConvTranspose2d(16, 16, kernel_size=2, stride=2)
        self.dec1 = self.conv_block(16 + 16, 16)

        # Final conv
        self.final_conv = nn.Conv2d(16, out_channels, kernel_size=1)
        self.sigmoid = nn.Sigmoid()

    def conv_block(self, in_channels, out_channels):
        return nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 3, padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True),
            nn.Conv2d(out_channels, out_channels, 3, padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(inplace=True)
        )
```

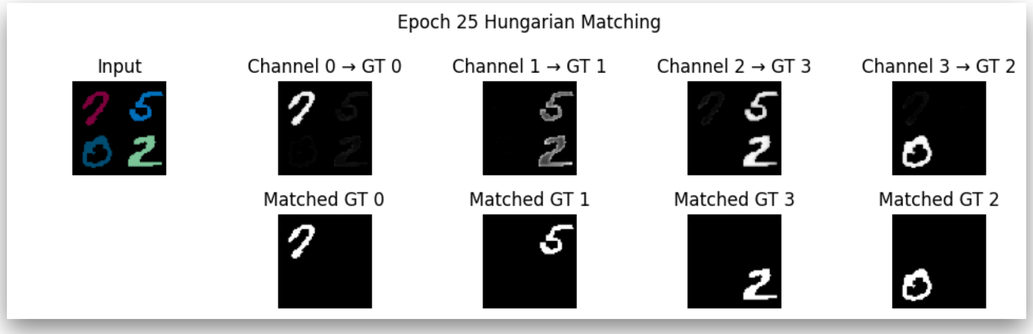
```python
def forward(self, x):
    # Encoder
    e1 = self.enc1(x)
    e2 = self.enc2(F.max_pool2d(e1, 2))
    e3 = self.enc3(F.max_pool2d(e2, 2))
    e4 = self.enc4(F.max_pool2d(e3, 2))
    e5 = self.enc5(F.max_pool2d(e4, 2))

    # Decoder
    d4 = self.up4(e5)
    d4 = torch.cat([d4, e4], dim=1)
    d4 = self.dec4(d4)

    d3 = self.up3(d4)
    d3 = torch.cat([d3, e3], dim=1)
    d3 = self.dec3(d3)

    d2 = self.up2(d3)
    d2 = torch.cat([d2, e2], dim=1)
    d2 = self.dec2(d2)

    d1 = self.up1(d2)
    d1 = torch.cat([d1, e1], dim=1)
    d1 = self.dec1(d1)

    out = self.final_conv(d1)
    out = self.sigmoid(out)

    return out
```

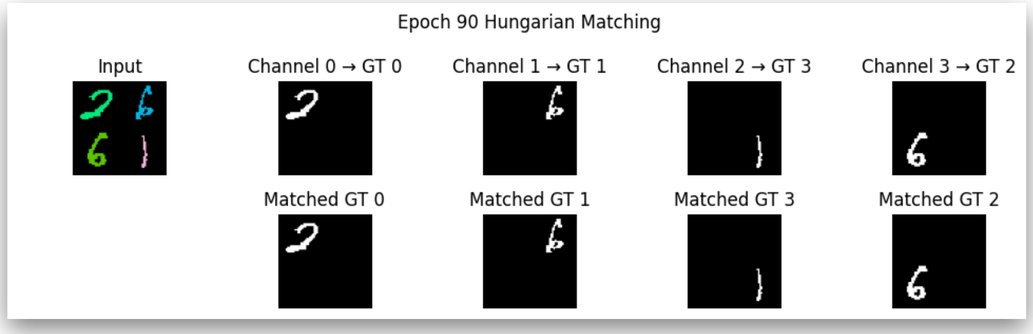
U-net

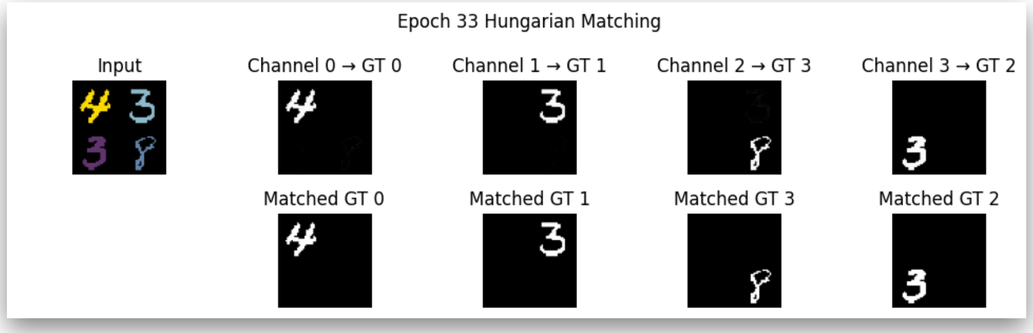
| Train

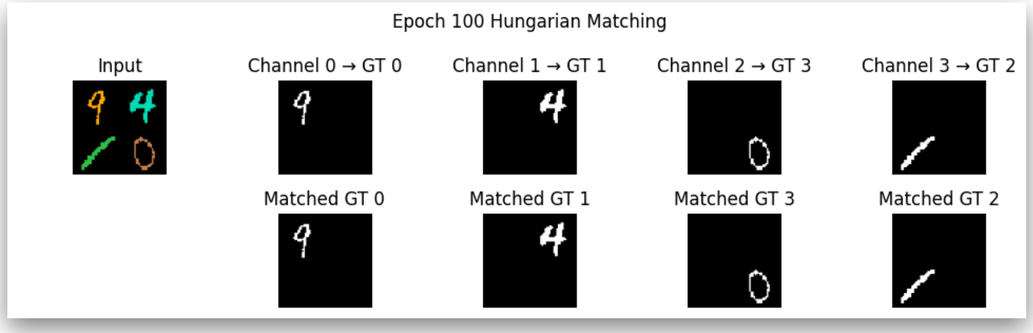

Epoch 20 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2


Epoch 60 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2


Epoch 25 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2

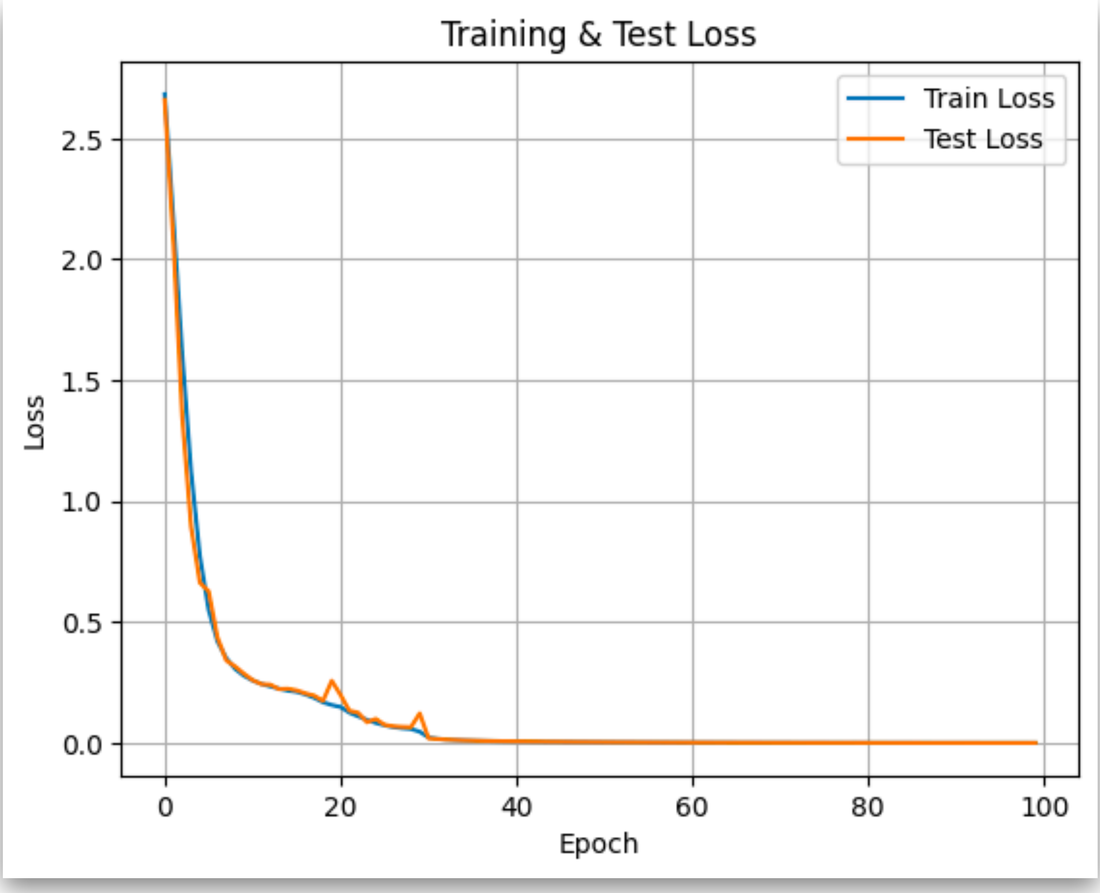

Epoch 90 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2


Epoch 33 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2


Epoch 100 Hungarian Matching
Input
Channel 0 → GT 0
Channel 1 → GT 1
Channel 2 → GT 3
Channel 3 → GT 2
Matched GT 0
Matched GT 1
Matched GT 3
Matched GT 2

U-net

| Train - Batch size = 64, epoch = 100



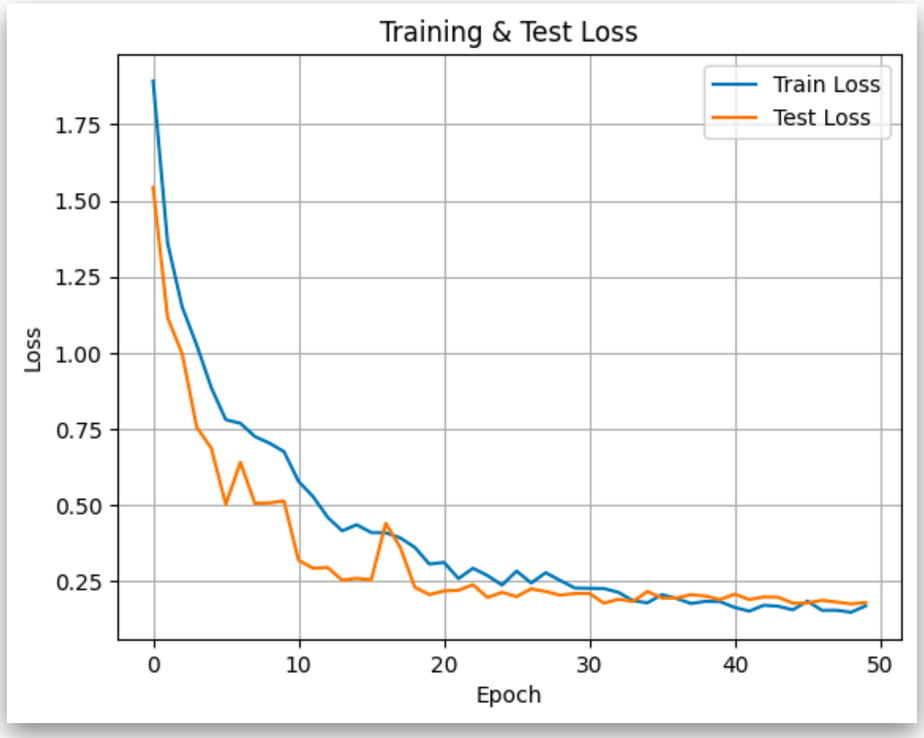
Min Train Loss : 0.0008

Min Test Loss : 0.0009

- Min Train Loss : 0.0178
- Min Test Loss : 0.0146

Classification

| Train - Batch size = 64, epoch = 50

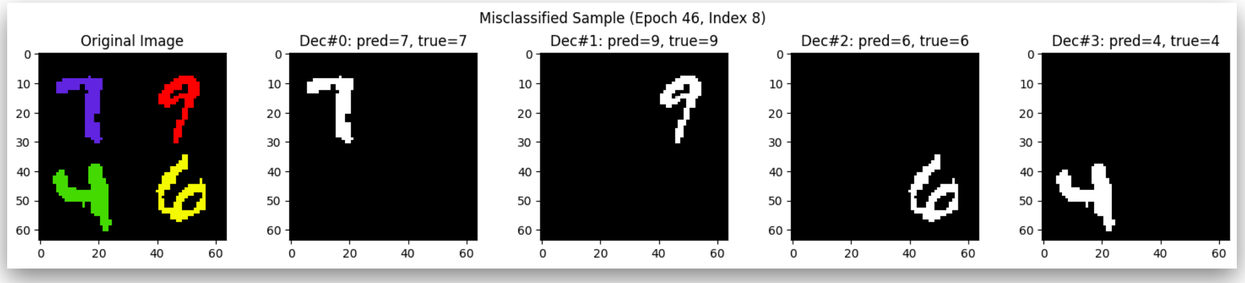


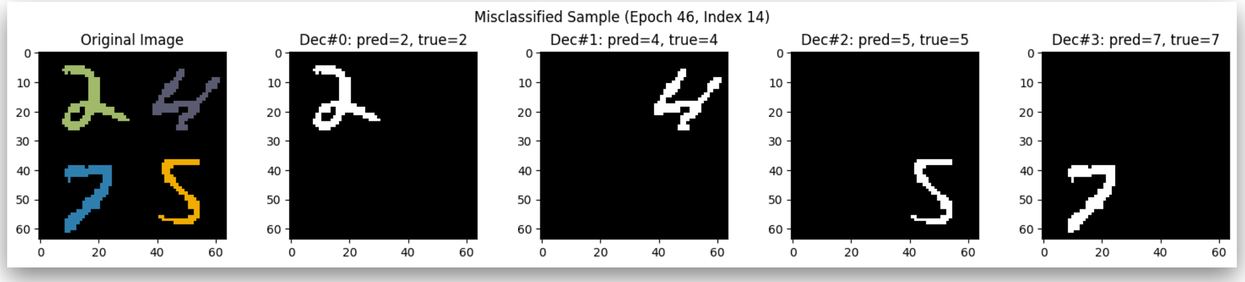
Min Train Loss : 0.1490

Min Test Loss : 0.1763

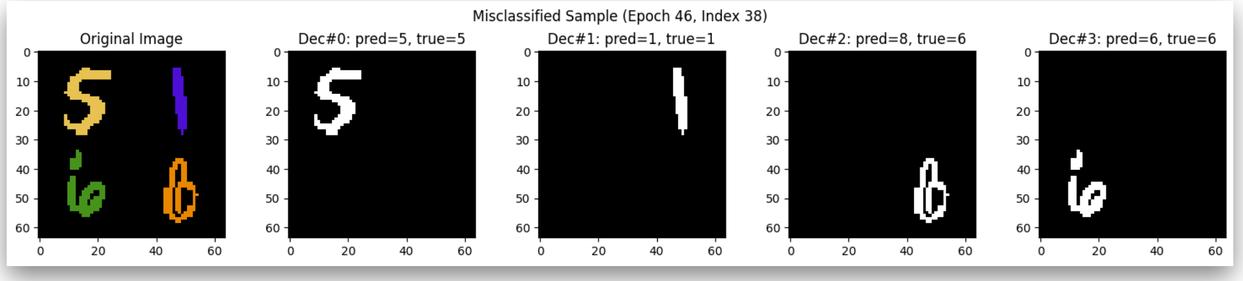
Best Test Acc : 96.66

Right





Wrong



I. Decoder & Batch size

II. U-net

III. Conclusion

결론

- **Decoder & Batch size 수정**

  : decoder 4개를 사용하는 방법에서 decoder 하나를 사용하고 채널을 4개로 지정해 각 채널이 객체를 탐지하는 방식으로 변경

  : Batch size를 증가시키니 초반에는 전체적인 loss가 높았지만 epoch가 증가함에 따라 batch size가 작은 경우보다 조금 더 나은 성능으로 측정됨

- **U-net**

  : skip connection 구조를 포함해, 인코더의 feature를 디코더에 직접 전달하는 효과가 있어 모델의 loss가 효율적으로 감소한 경향이 있음

Thanks for Listening

2025-1 모빌리티 UR 김유진

2025.05.15