한국소음진동공학회 AI혁신위원회 주관
2022년 인공지능 강습회

AI 기계이상진단을 위한 인공지능 학습 기법

제 5강 분류 태스크를 이용한 이상진단 (실습)

한국과학기술원 전기및전자공학부

최정우
jwoo@kaist.ac.kr

KAIST EE

목차

- 분류 태스크를 통한 이상진단 실습

- 실습 1 : 분류기 구현 및 기법 실습
  - Framework
  - Label smoothing
  - Temperature scaling
  - t-SNE visualization

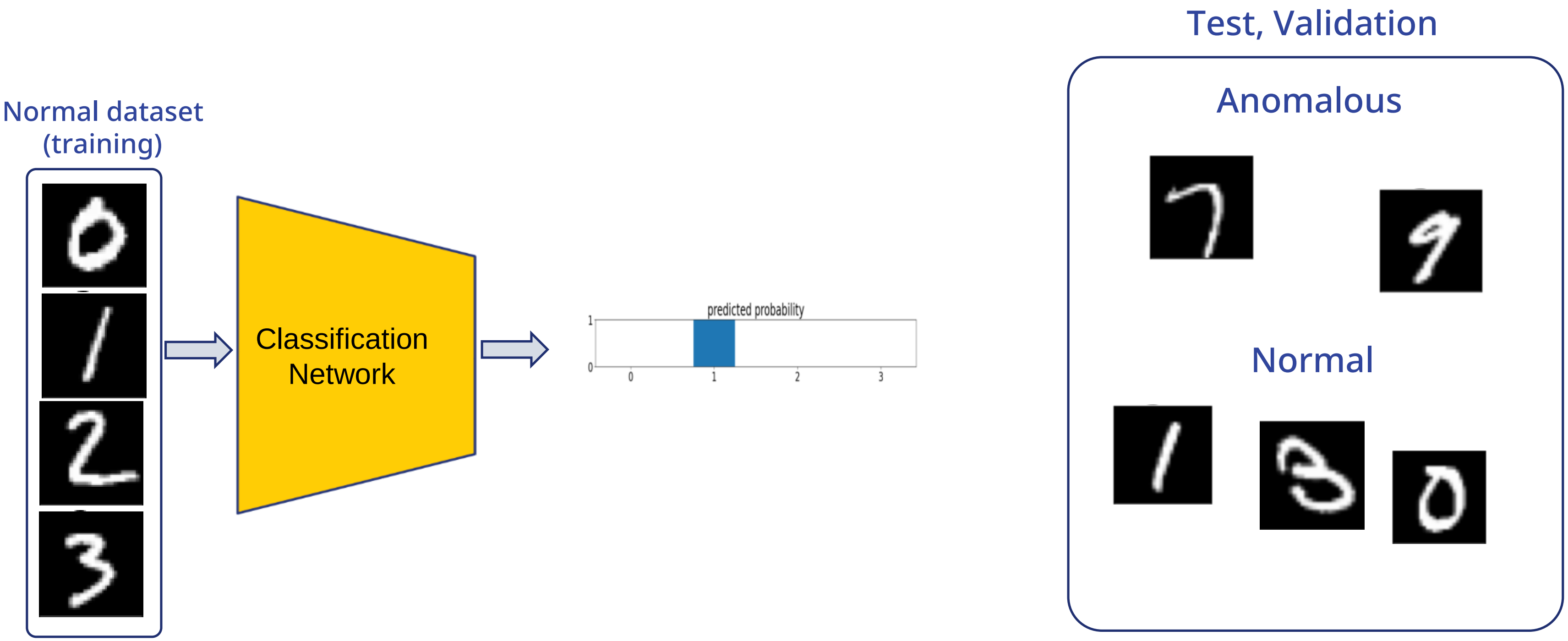
- 실습 2 : 레이블이 없는 경우의 분류기 구현
  - RotNet

Exercise with Classification Tasks

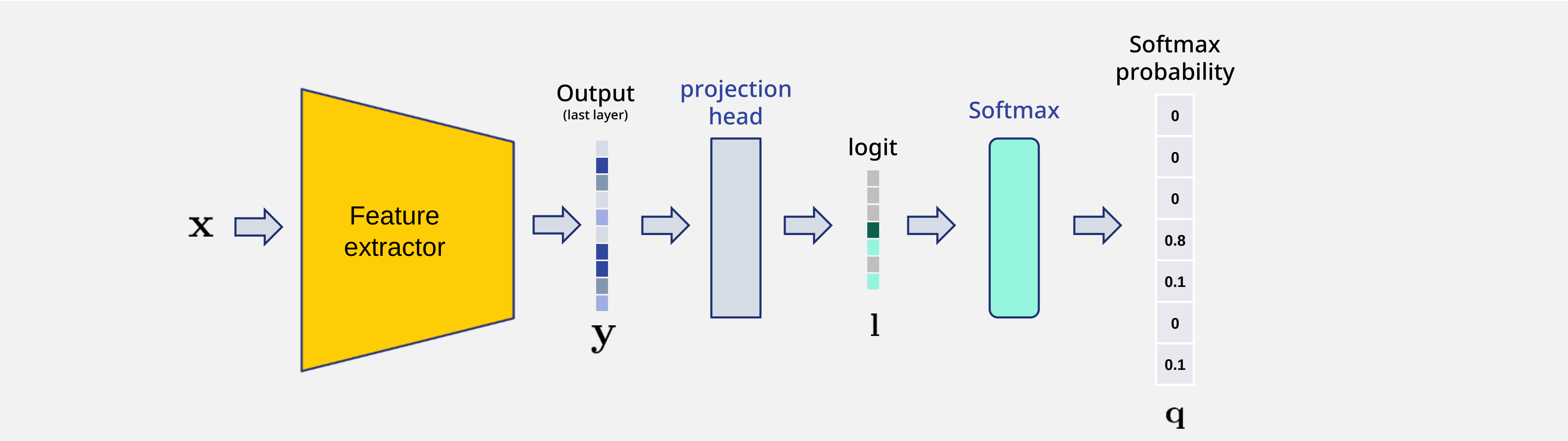
1. Classification of internal labels

실습 데이터 설명

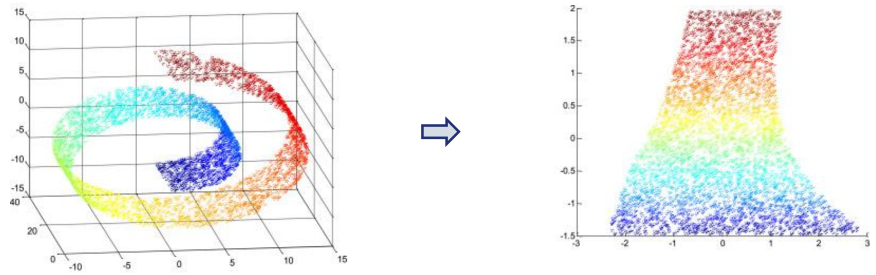
- MNIST dataset
  - Normal data: 0, 1, 2, 3
  - Anomalies: 7, 8, 9
  - Train a CNN model to classify normal data
  - Inspect the model's behavior for anomalous samples



Recap: Structure of DNN classifier

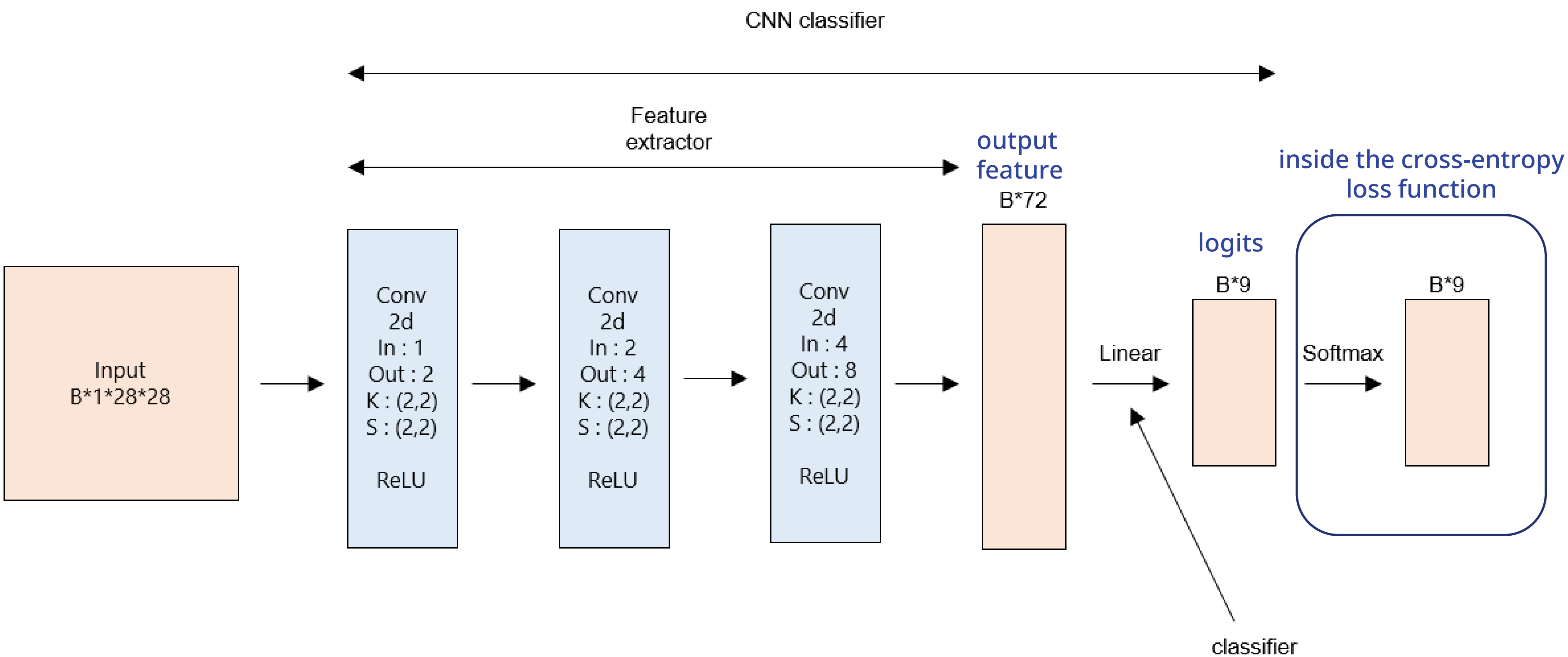


- Feature extractor
  - Transform input data **x** to feature space data **y**
  - Trained to find a good mapping for classification



분류를 위한 신경망 모델 (initial)

- CNN 를 사용한 분류 모델



Code outline

2. Hyperparameters

You can change the hyperparameter below. Parameter `ANOMALY_NUM` means which number to set as an anomaly.

```
EPOCHS = 5          # Number of epochs to train
BATCH = 32          # Minibatch size
ORGCLASS_NUM = 10   # Num of original classes (10: 0 to 9)
ANOMALY_NUM = [7,8,9]   # (list) Digits will be used as anomalous data
NORMAL_NUM =  [0,1,2,3]   # (list) Digits used as normal data

TEMPSC = 1.0    # temperature parameter (for temperature scaling)
LBSMOOTH = 0.0  # label smoothing parameter
```

Code outline

- CNN model

```
class ClassificationCNNModel(nn.Module):
    def __init__(self, nclass=len(NORMAL_NUM)):
        super(ClassificationCNNModel, self).__init__()
        self.feature_extractor = nn.Sequential(
            nn.Conv2d(1,2,kernel_size = (2,2), stride = (2,2)),
            nn.ReLU(),
            nn.Conv2d(2,4,kernel_size = (2,2), stride = (2,2)),
            nn.ReLU(),
            nn.Conv2d(4,8,kernel_size = (2,2), stride = (2,2)),
            nn.ReLU()
        )
        self.classifier = nn.Sequential(
            nn.Linear(72,nclass),
        )

    def forward(self, x):
        feature = self.feature_extractor(x)
        feature = feature.reshape(feature.shape[0],-1)
        out = self.classifier(feature)
        return out, feature
```

return the feature for feature embedding visualization (t-SNE)

Code outline

- Loss function

$$l_n = -\sum_{c=1}^{C} w_c \log \frac{\exp(x_{n,c})}{\sum_{i=1}^{C} \exp(x_{n,i})} y_{n,c}$$

```python
loss_fn = nn.CrossEntropyLoss(label_smoothing=LBSMOOTH) # CCE

def anomaly_score(logits): # Anomaly score = 1 - MSP
    softmaxprob = torch.softmax(logits, dim=1)
    MSP = torch.max(softmaxprob, dim=1).values
    return torch.tensor(1) - MSP

optimizer = torch.optim.Adam(model.parameters(), lr=1e-3) # Adam as optimizer
```

maximum softmax prob.

- (!) Pythorch CrossEntropyLoss includes Softmax function
- Input to the loss function is logits (not Softmax probability)

- Anomaly score = 1 – Maximum softmax probability

Code outline (train)

```
def train(dataloader, model, loss_fn, optimizer):
    model.train()  switch to training mode
    size = len(dataloader.dataset)
    losses = []
    for batch, dat in enumerate(dataloader):                    # dat is tuple of (img, label)
        logits, _ = model(dat[0].to(device))                    # logit (batch, class)

        p = F.one_hot(dat[1],ORGCLASS_NUM)[:,NORMAL_NUM]
        p = p.float().to(device)                  Generate one-hot vector from the label.
                                                  Only use normal class labels from the one-hot vector

        loss = loss_fn(logits, p)  temperature scaling is not applied for training
        losses.append(loss.cpu().detach())

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

        if batch % 300 == 0:
            loss, current = loss.item(), batch * logits.shape[0]
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")

    return np.mean(losses)
```

Code outline (test)

```python
def test(dataloader, model, loss_fn, anomaly_score, valid_mode = True,  draw_mode = False):
    model.eval()

    y_true, y_pred = [], []

    normal_loss = []
    normal_scores = []
    anomaly_scores  =[]
    outfeature = []
    total_labels = []

    with torch.no_grad():

        for Imgs, labels in dataloader:

            Imgs = Imgs.to(device)                    # upload to GPU
            logits, features = model(Imgs)            # logits out (idx,class), features out (idx, num_feature )
            logits = logits/TEMPSC                    # temperature scaling
            if not valid_mode:
                outfeature.append(features)           # append features to tensor list for t-SNE plot
                total_labels.append(labels)

            score = anomaly_score(logits)             # anomaly score (idx)
            y_pred.extend(score.cpu().tolist())           # stack prediction score
```

for test mode (not validation mode), return feature outputs for visualization

test mode: do not update gradients

Code outline (test)

```
with torch.no_grad():

    for Imgs, labels in dataloader:
```

```
        # We do nothing if label does not belong to NORMAL_NUM or ANOMALY_NUM
        for idata in range(logits.shape[0]):
            score_ = score[idata].item()
```

anomaly score of each image

```
            if labels[idata] in NORMAL_NUM:  # for normal data
                # calc CE loss for normal data
                p = F.one_hot(labels[idata],10)[NORMAL_NUM].reshape(1,-1)
                logit = logits[idata,:].reshape(1,-1)
                loss = loss_fn(logit.to(device), p.float().to(device)) # Cross-Entropy loss for normal
                normal_loss.append(loss.cpu())

                # record scores
                y_true.append(0.)
                normal_scores.append(score_)
```

make one-hot vector & reshape to 2D tensor

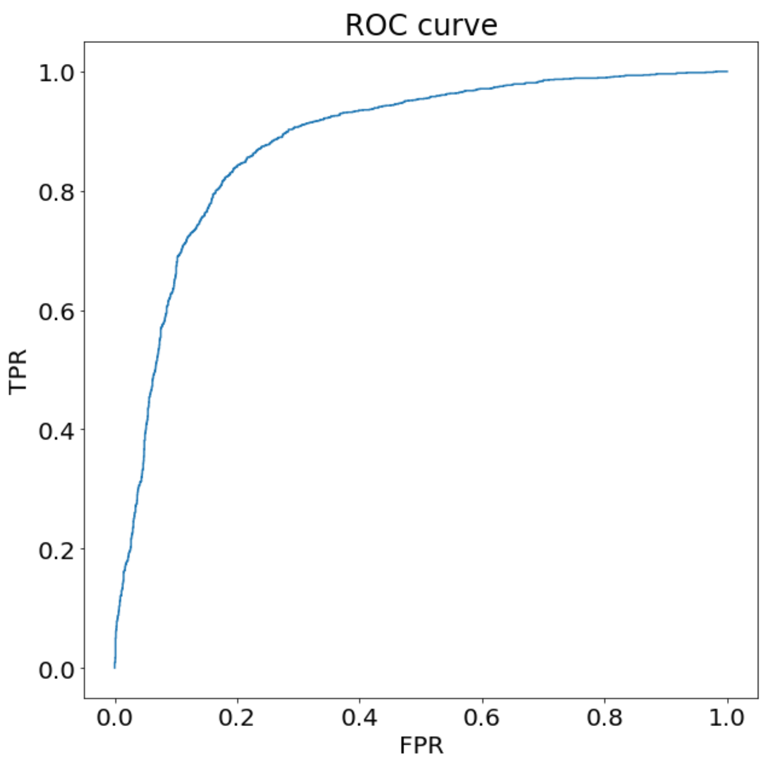
append scores

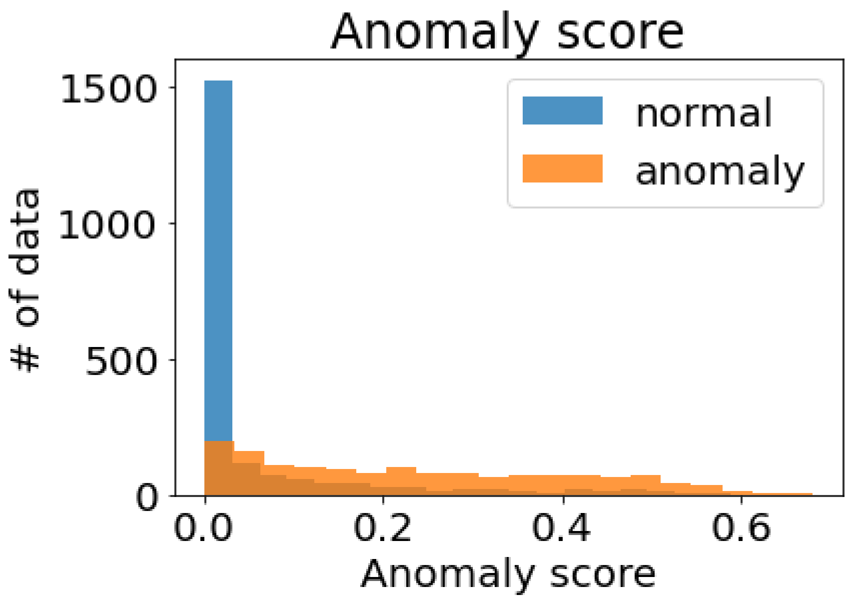
```
            elif labels[idata] in ANOMALY_NUM: # for abnormal data
                y_true.append(1.)
                anomaly_scores.append(score_)

    roc_auc = metrics.roc_auc_score(y_true, y_pred)
```

```
if not valid_mode:
  outfeature = torch.cat(outfeature, dim=0).cpu().numpy()
  total_labels = torch.cat(total_labels, dim=0).cpu().numpy()
```

Classification result (Init)

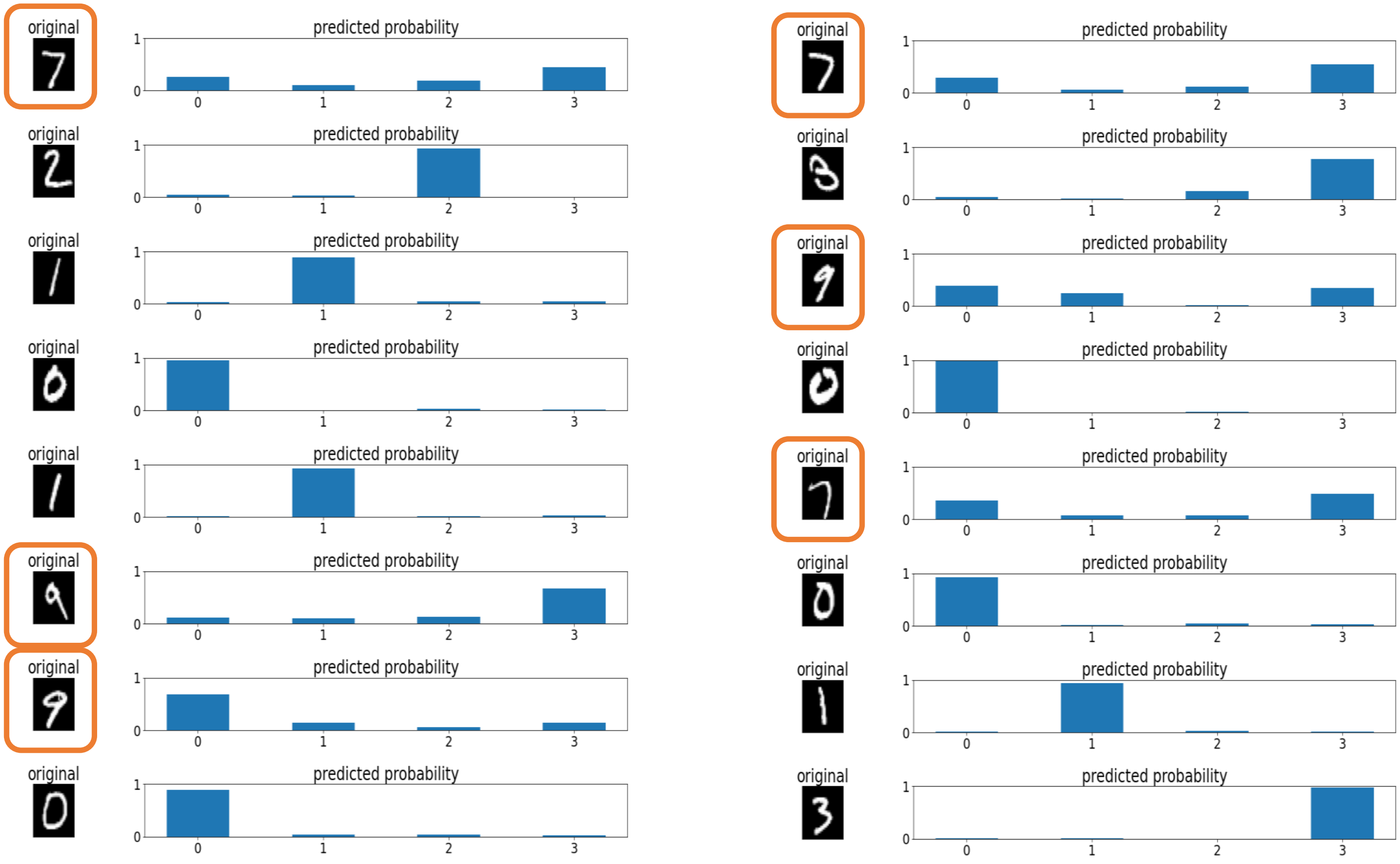
- Without label smoothing & temperature scaling





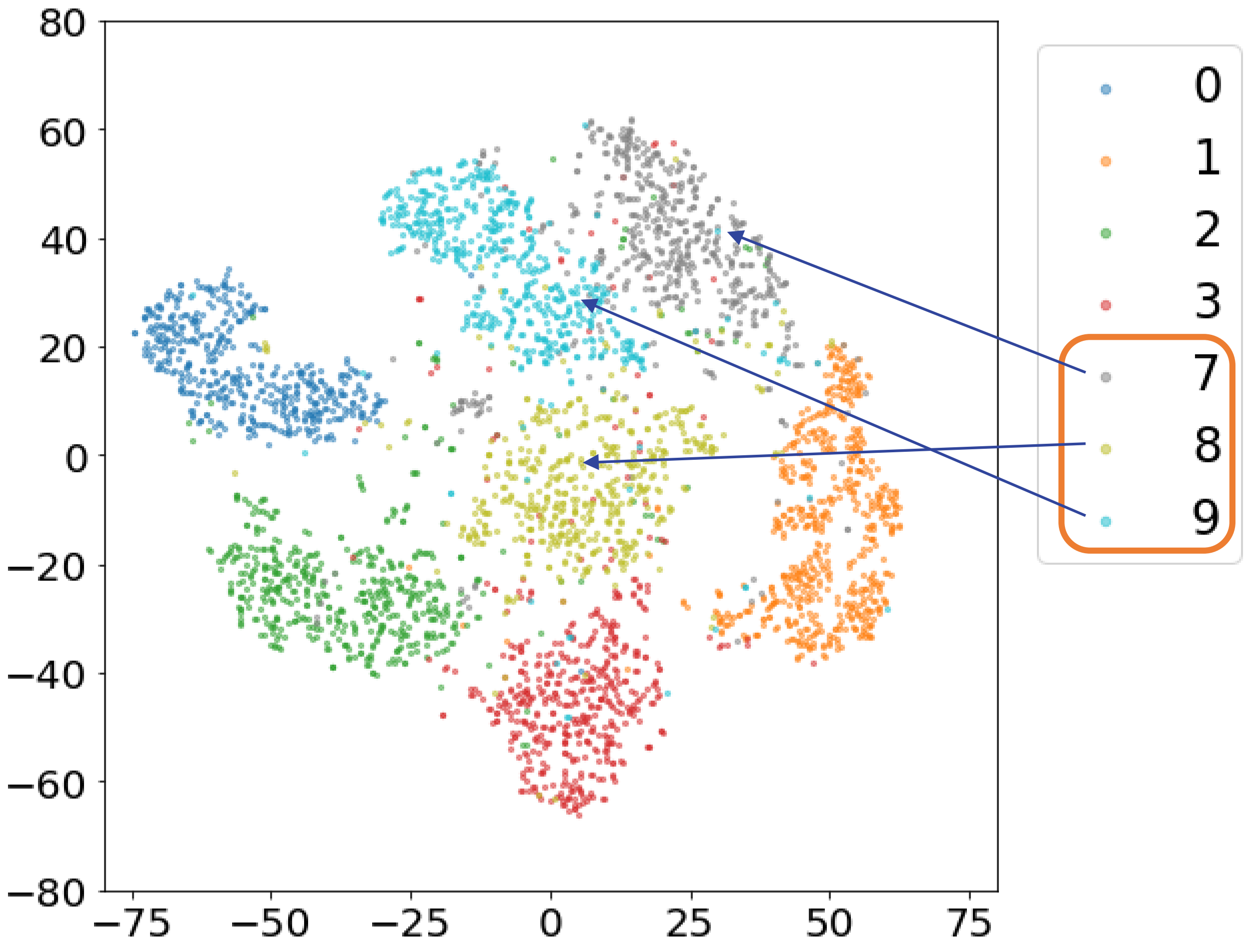
ROC-AUC: 0.868

Classification result (initial)



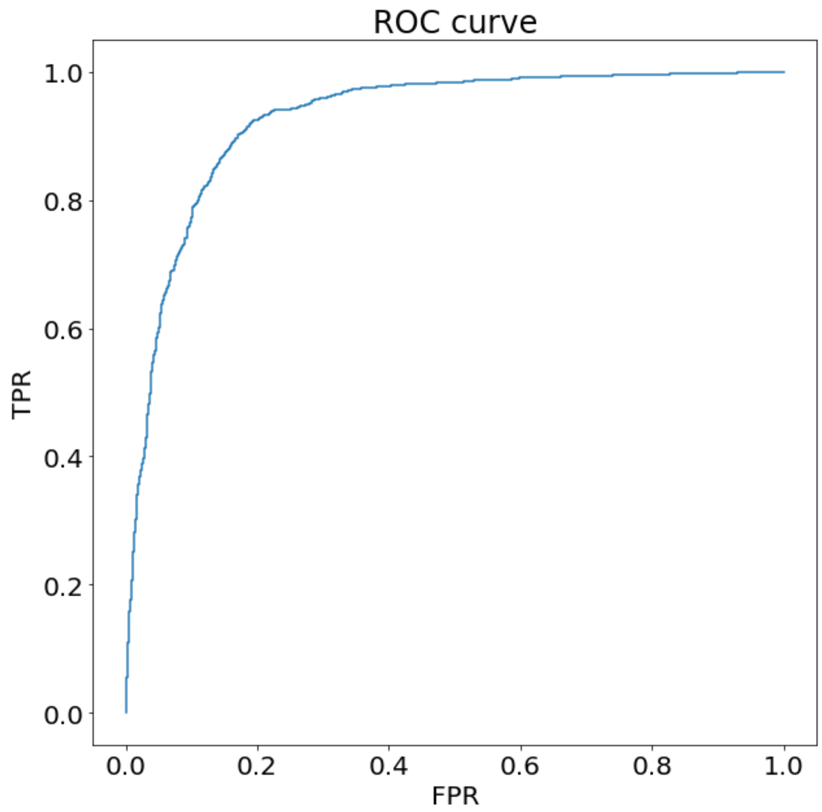
Feature embedding visualization

- t-SNE visualization (Without label smoothing & temperature scaling)

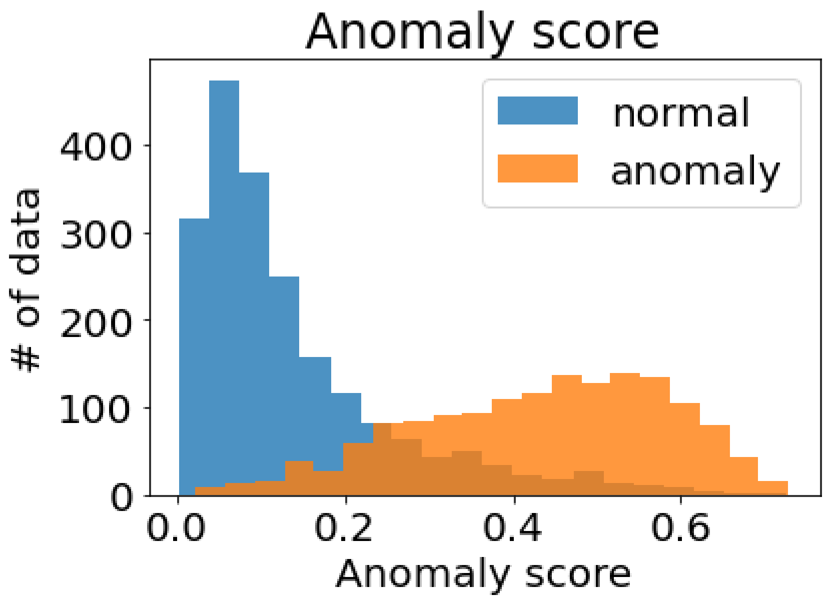


Classification result (Label Smoothing =0.1)

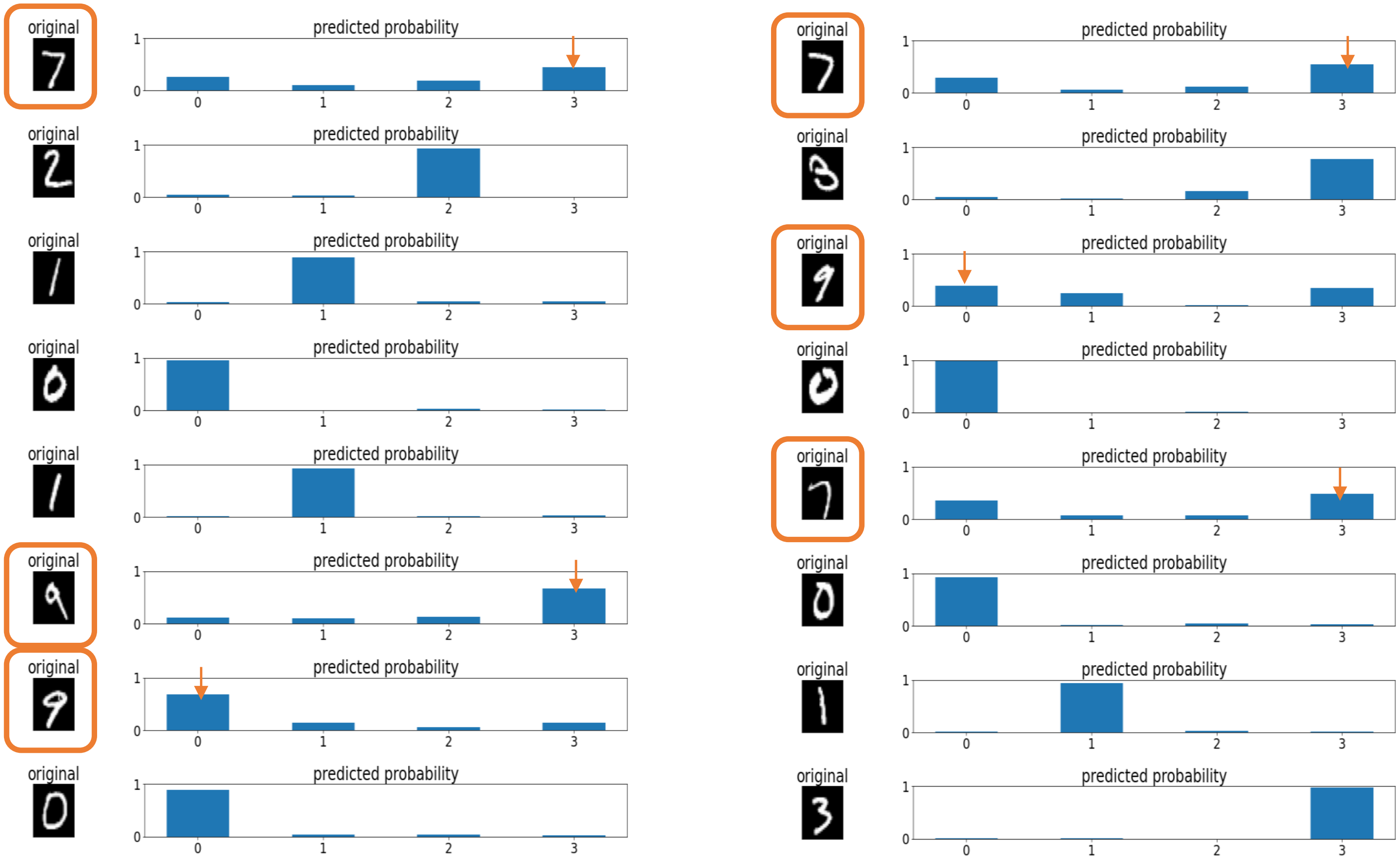
- With label smoothing (smoothing = 0.1)



ROC-AUC: 0.927

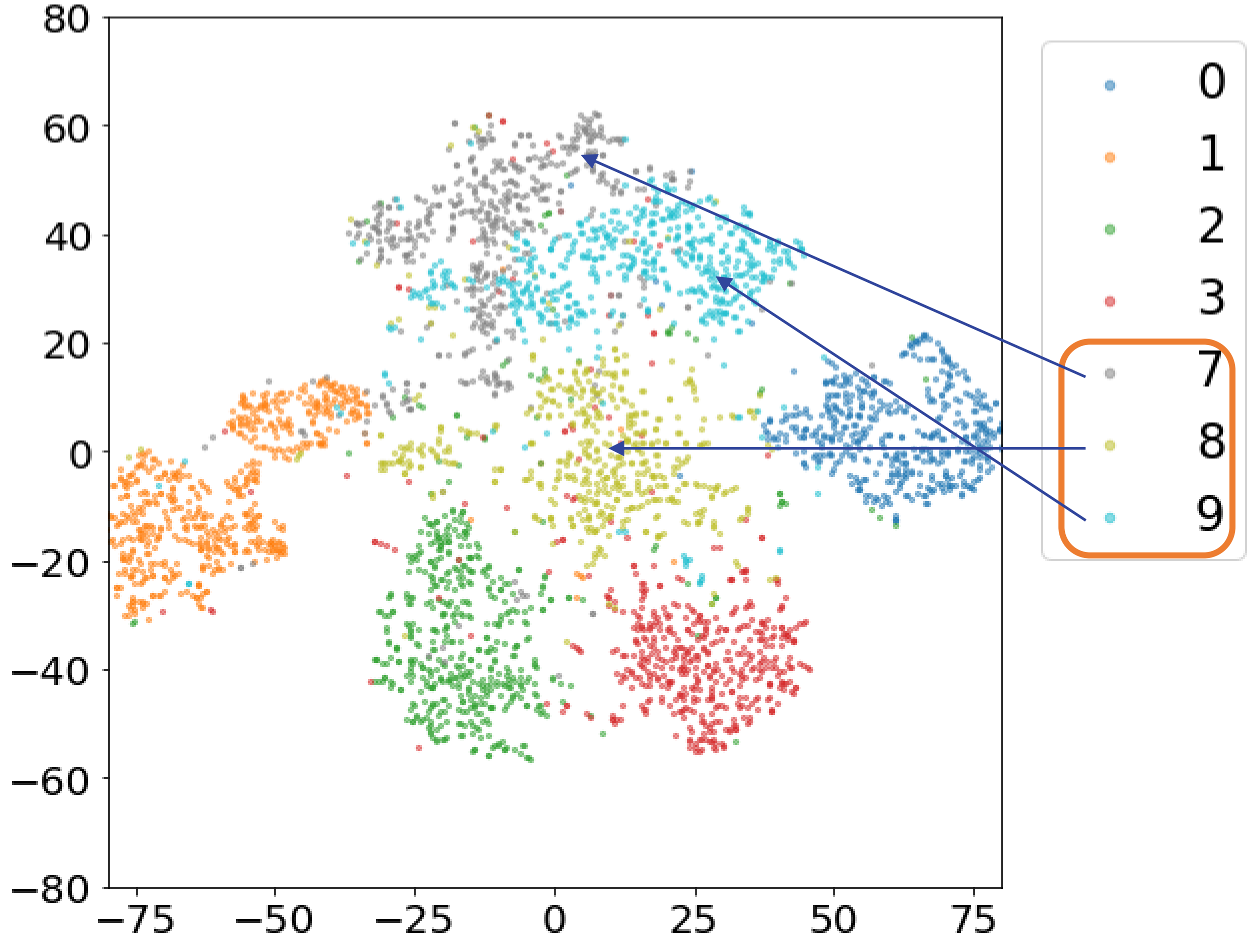


Classification result (Label Smoothing =0.1)



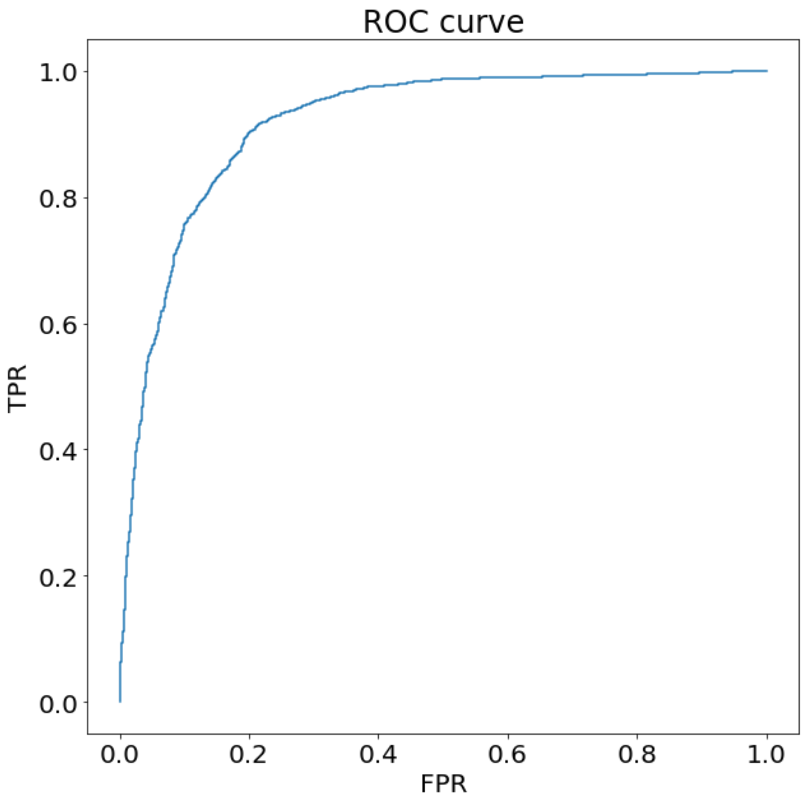
Feature embedding visualization

- t-SNE visualization (Label Smoothing = 0.1)

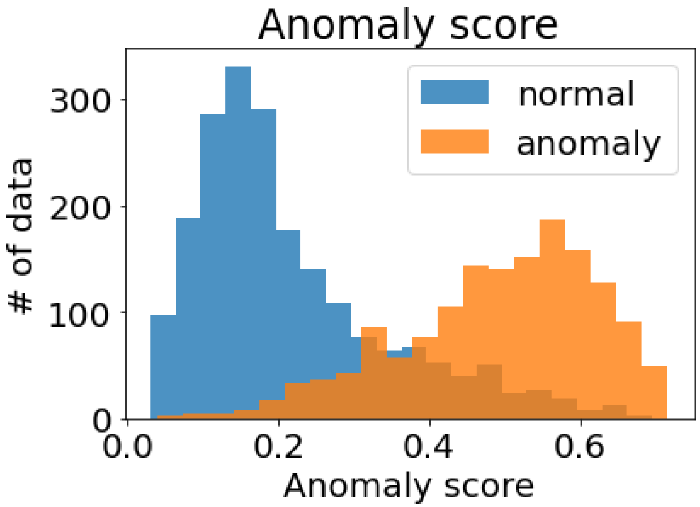


Classification result (Label Smoothing)

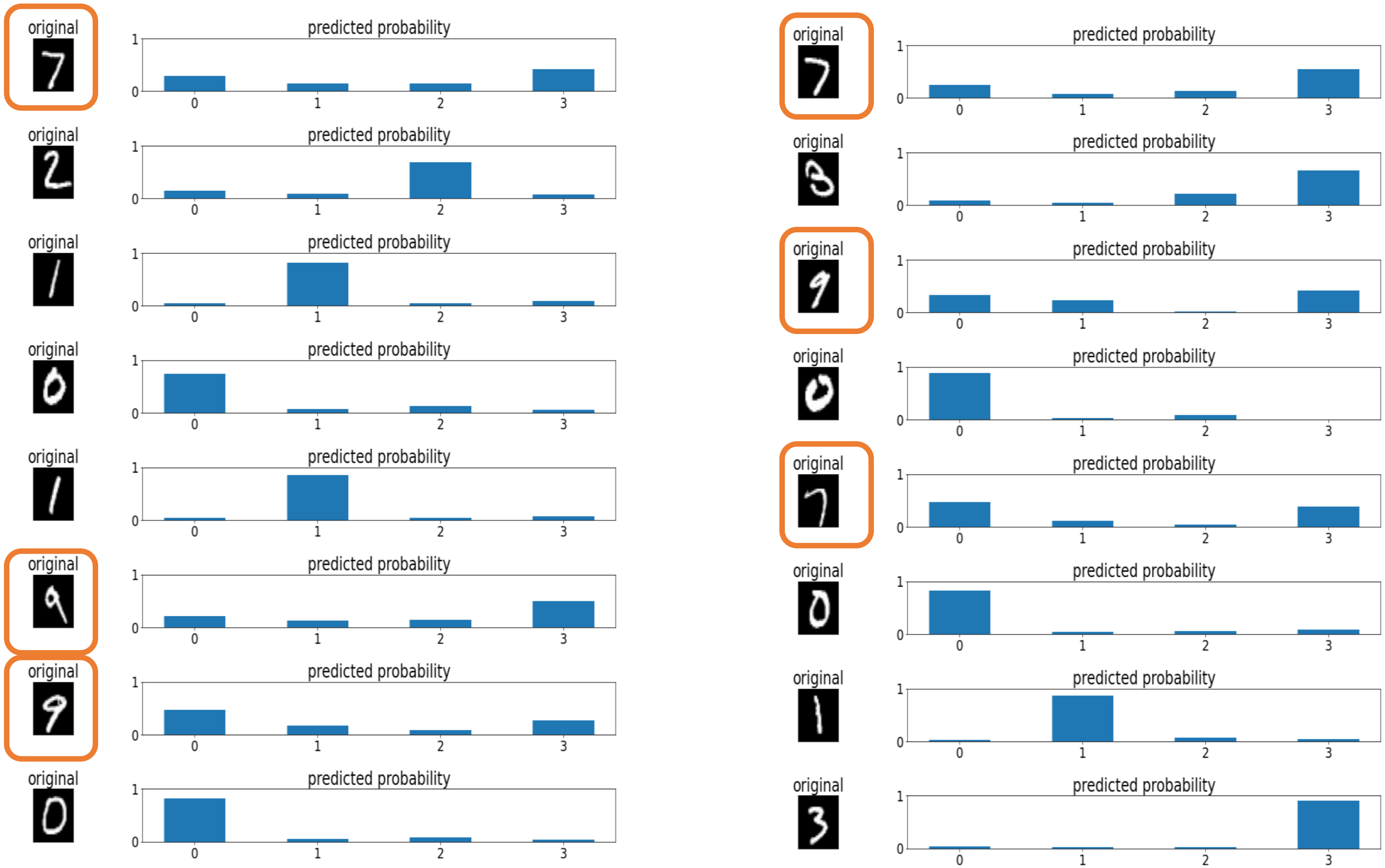
- With label smoothing (smoothing = 0.2)



ROC-AUC: 0.918

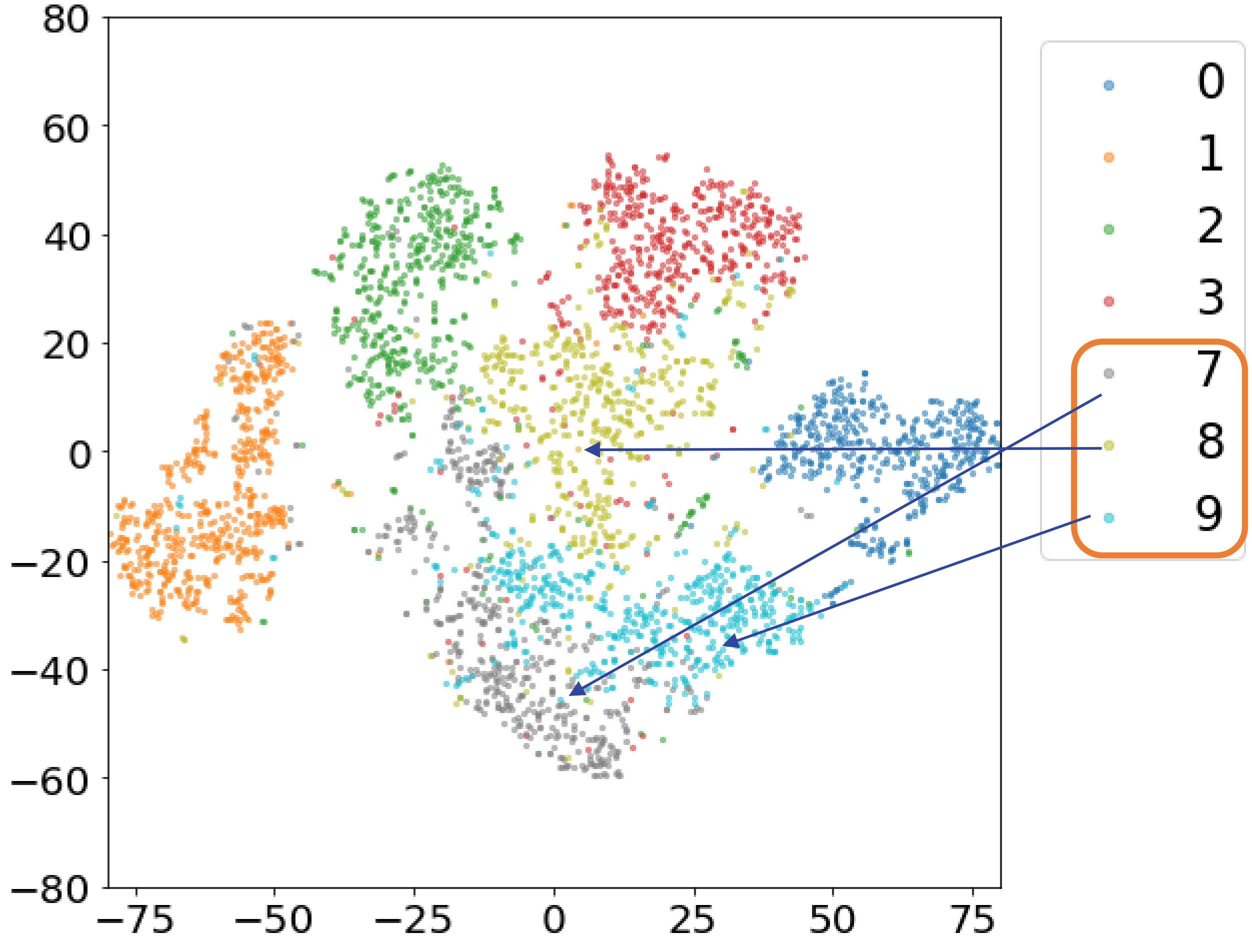


Classification result (Label smoothing = 0.2)



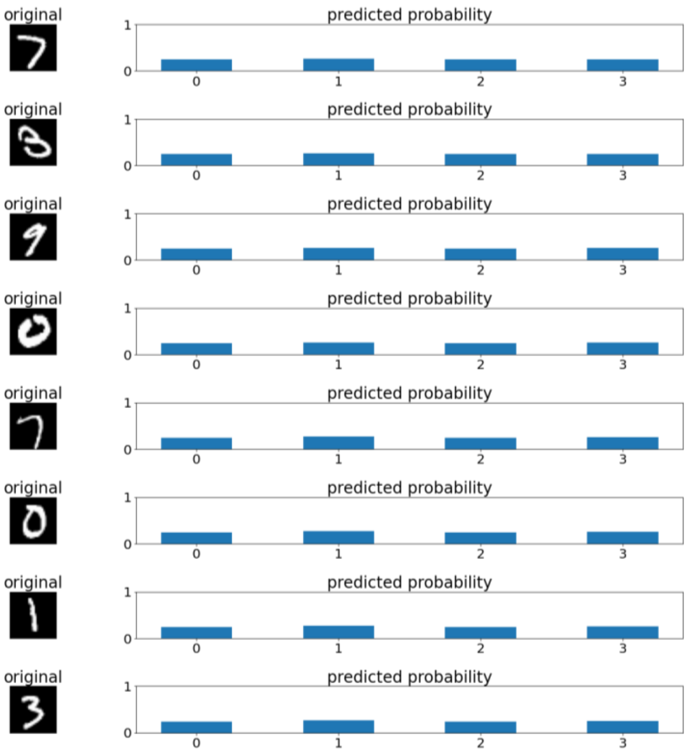
Feature embedding visualization

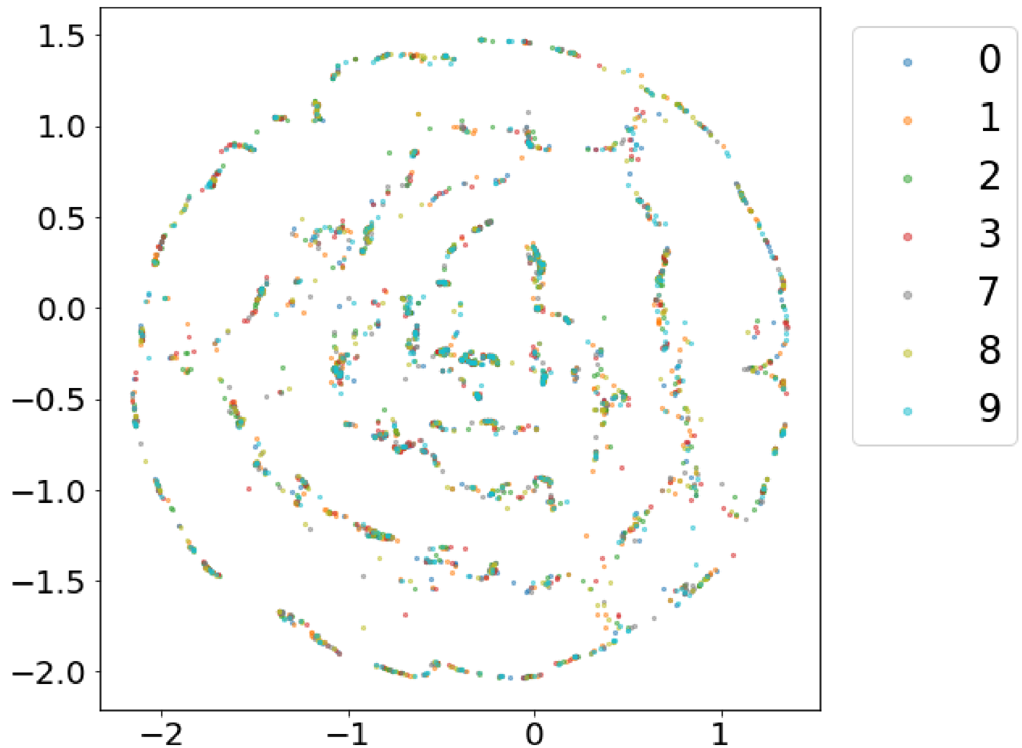
- t-SNE visualization



Excessive label smoothing

- Label Smoothing = 0.3





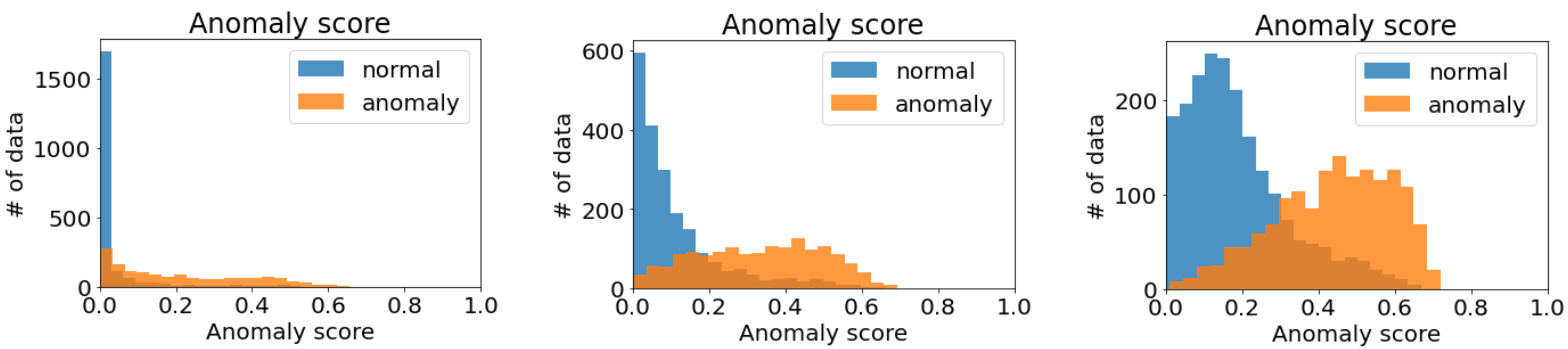
Temperature Scaling

Calibration of trained model

Effect of temperature scaling

- Histograms for different temperatures

Over-confident ⟷ Under-confident



T = 1. , ROC-AUC=0.886

T = 2. , ROC-AUC=0.867

T = 3. , ROC-AUC=0.878

```
TEMPSC = 3.0    # temperature parameter (for temperature scaling)
LBSMOOTH = 0.1  # label smoothing parameter
```

Label smoothing vs. Temp. scaling

- Label smoothing
  - Change the target probability
  - Trained models are different

- Temperature scaling
  - Scale the logit
  - Does not change the model & training
  - Calibration of the trained model

- Effect
  - Prevent an over-confident model
  - Enlarge inter-class feature distances
  - Optimal parameter: by repeating training

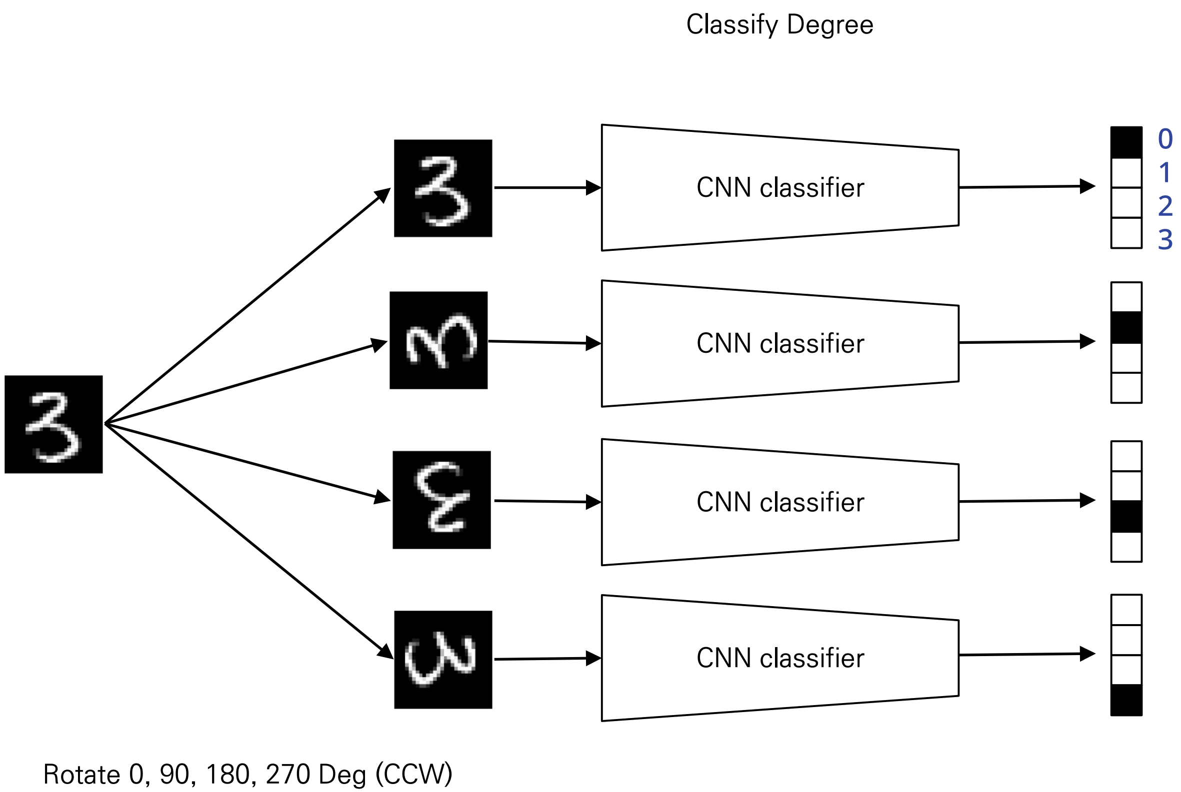
- Effect
  - Prevent an over-confident scoring
  - Features do not change
  - Optimal parameter can be found by tuning with a validation dataset

RotNet

Classifying Rotations of Images

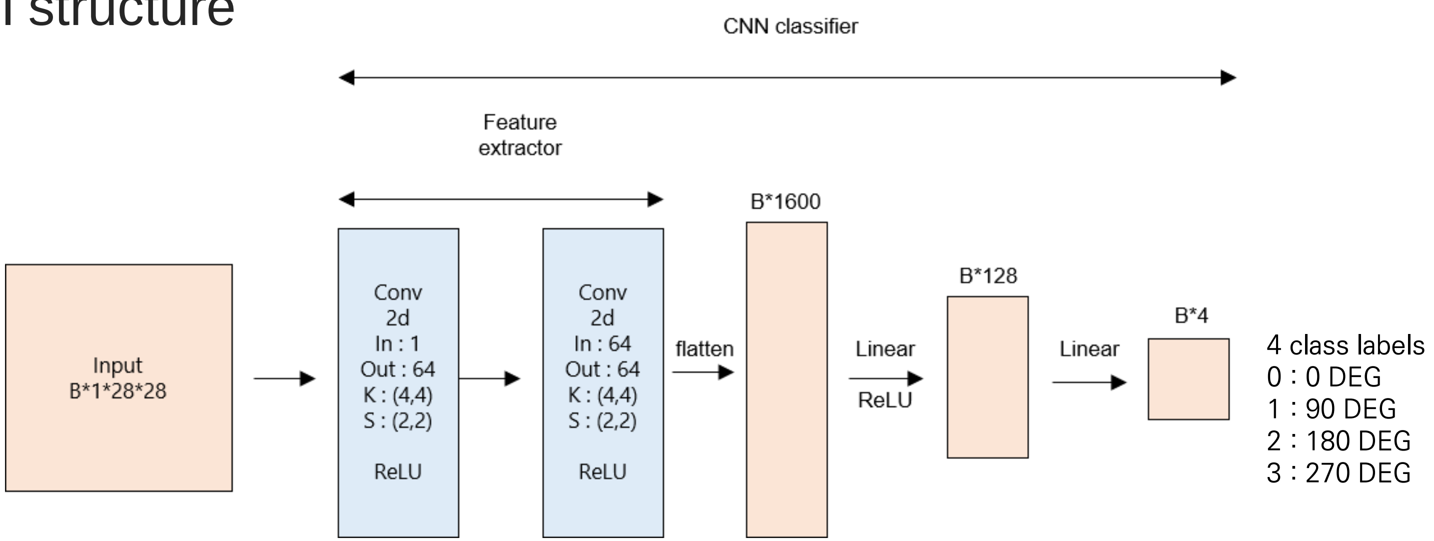
RotNet

- Does not require internal labels
  - Apply rotation transform to each image
  - Model is forced to predict rotation angles (labels)



RotNet Structure

- Model structure

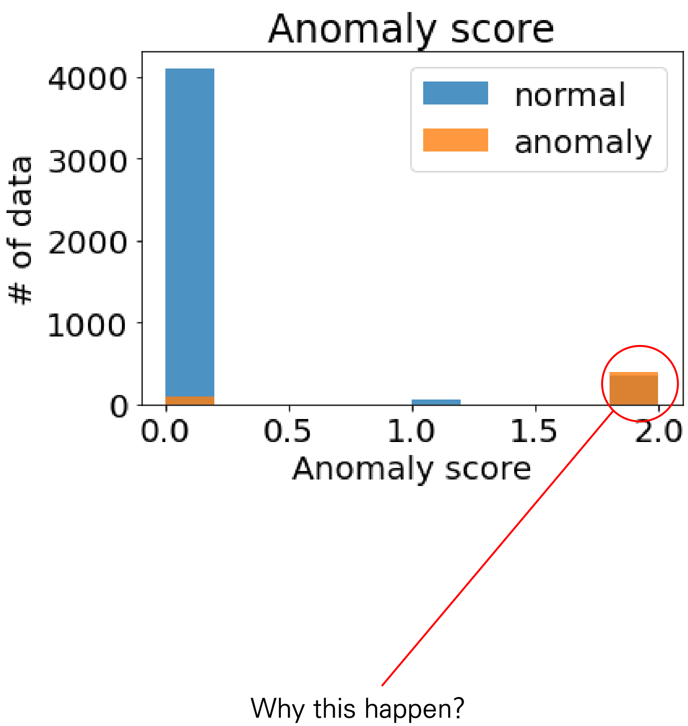


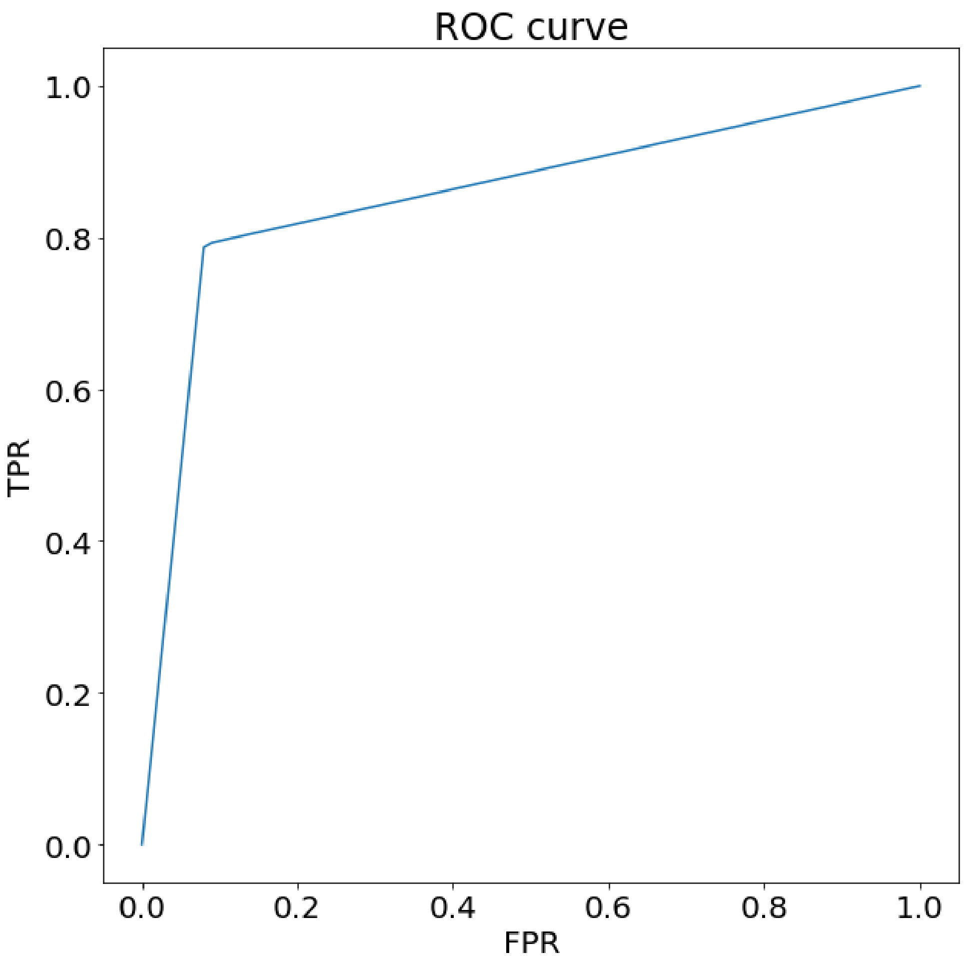
- Loss function & Anomaly score
  - Loss function : Cross Entropy
  - Anomaly score :
    - <ex> : pred : 0 , true : 1 → Anomaly score : 1
    - <ex> : pred : 0 , true : 3 → Anomaly score : 1 (Think that degree circulates [0,360).)

$$min(|pred - true|, 4 - |pred - true|)$$

Anomaly Detection using Self-supervised learning

- Rotnet

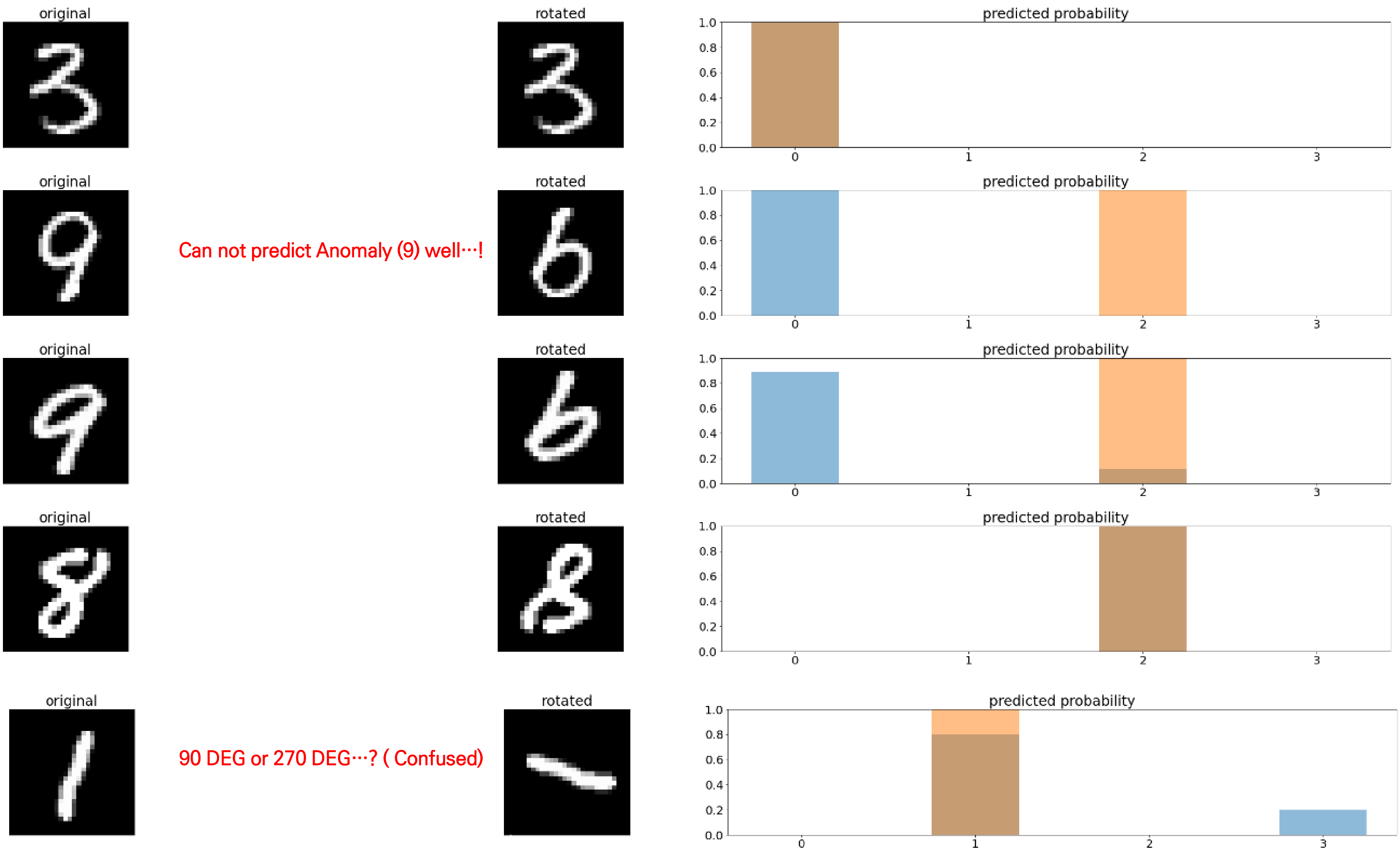




ROC AUC = 0.856

Anomaly Detection using Self-supervised learning

Blue for predicted class label and orange for true class label



Can not predict Anomaly (9) well…!

90 DEG or 270 DEG…? (Confused)