MNIST

Presentation for Data Classification

소프트웨어학부
20192490 강준규

TABLE OF CONTENTS

MNIST Classification with Representation Learning

01 MNIST

02 Hypothesis

03 Experiment

04 Conclusion

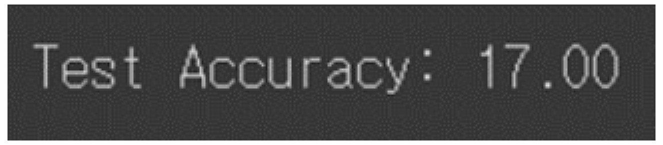
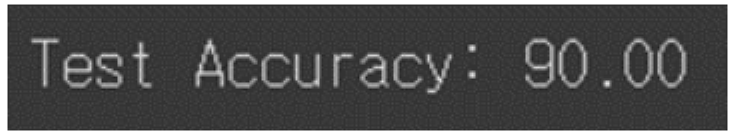
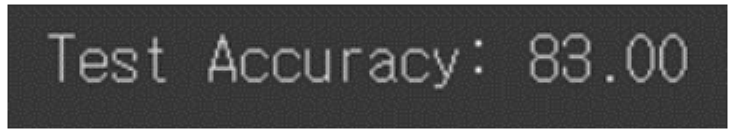
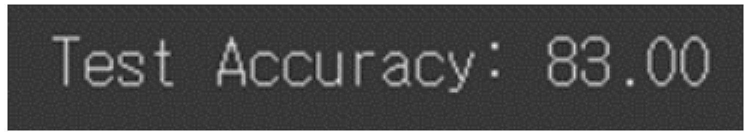
05 Limitation

01 MNIST

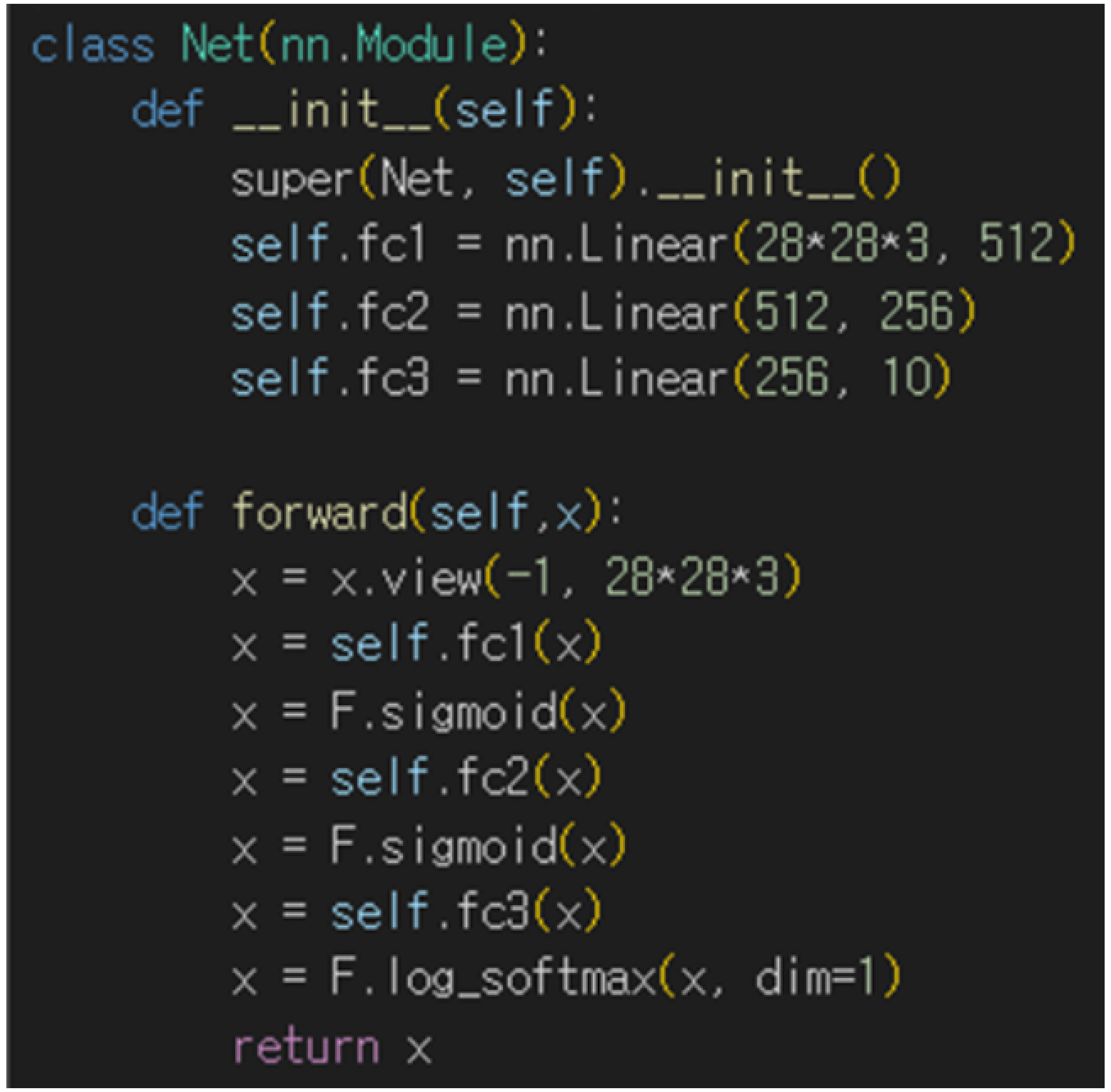
02 Hypothesis

1. Epoch
2. Convolutional Layer
3. Focal Loss

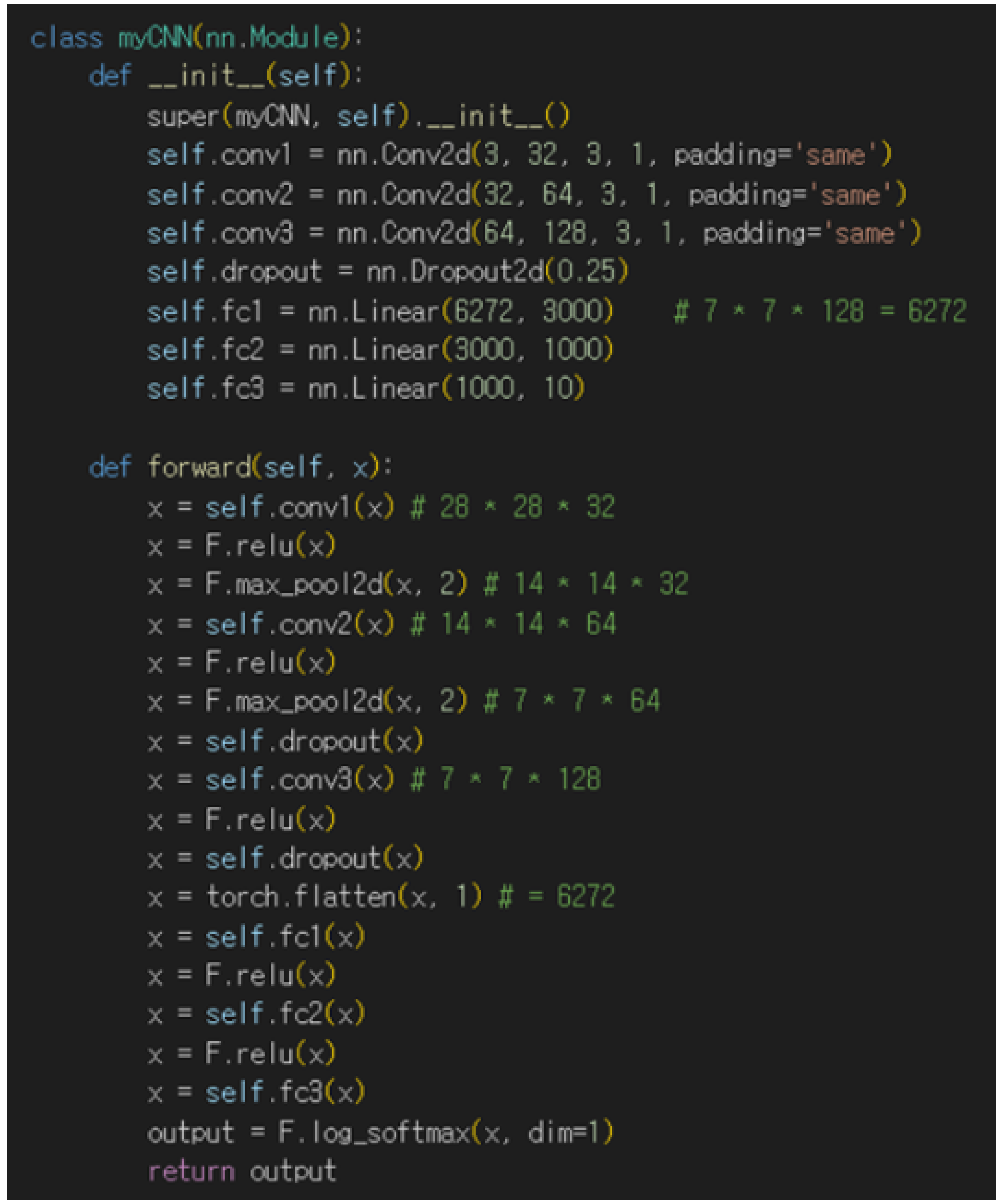
03 Experiment - Epoch

Epoch = 1	 Test Accuracy: 17.00
Epoch = 20	 Test Accuracy: 90.00
Epoch = 50	 Test Accuracy: 83.00
Epoch = 100	 Test Accuracy: 83.00

03 Experiment - Convolutional Layer

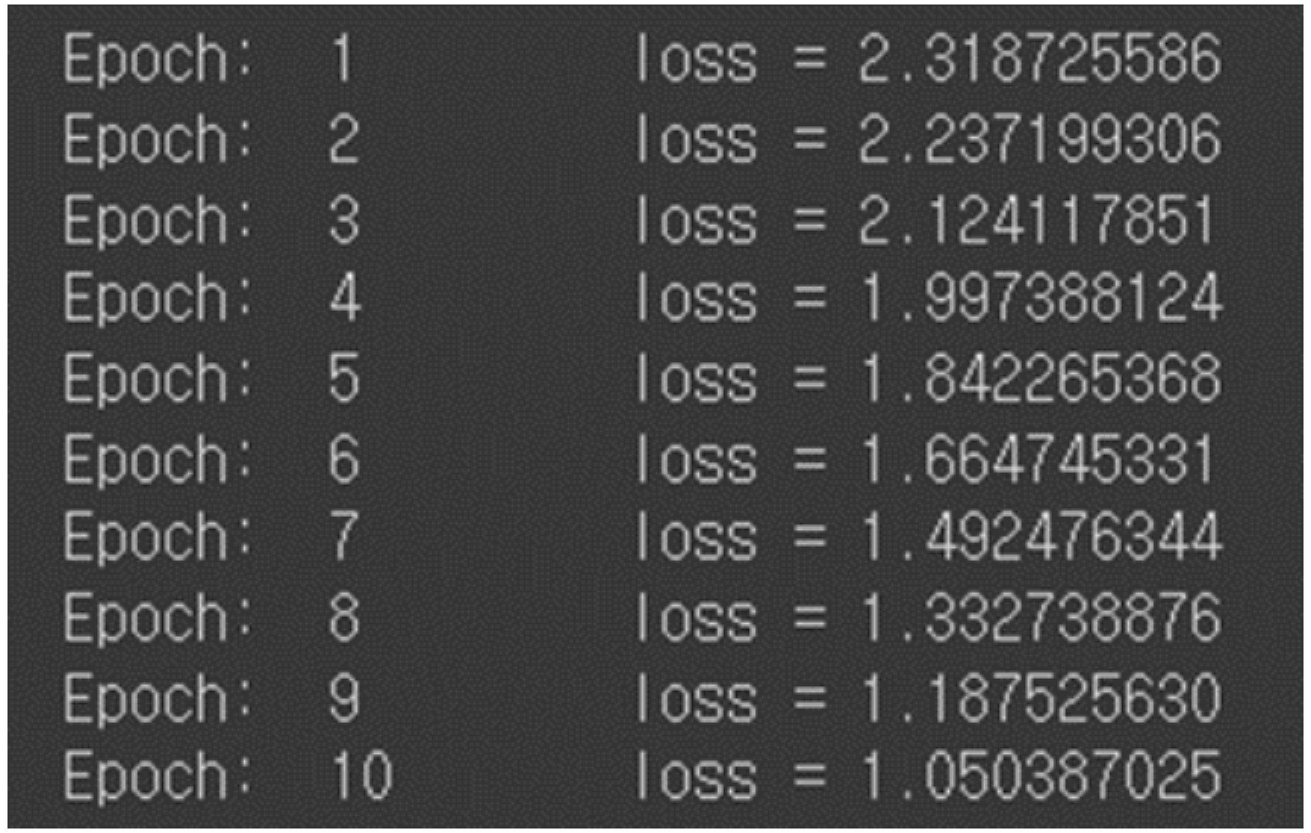


```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(28*28*3, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 10)

    def forward(self,x):
        x = x.view(-1, 28*28*3)
        x = self.fc1(x)
        x = F.sigmoid(x)
        x = self.fc2(x)
        x = F.sigmoid(x)
        x = self.fc3(x)
        x = F.log_softmax(x, dim=1)
        return x
```

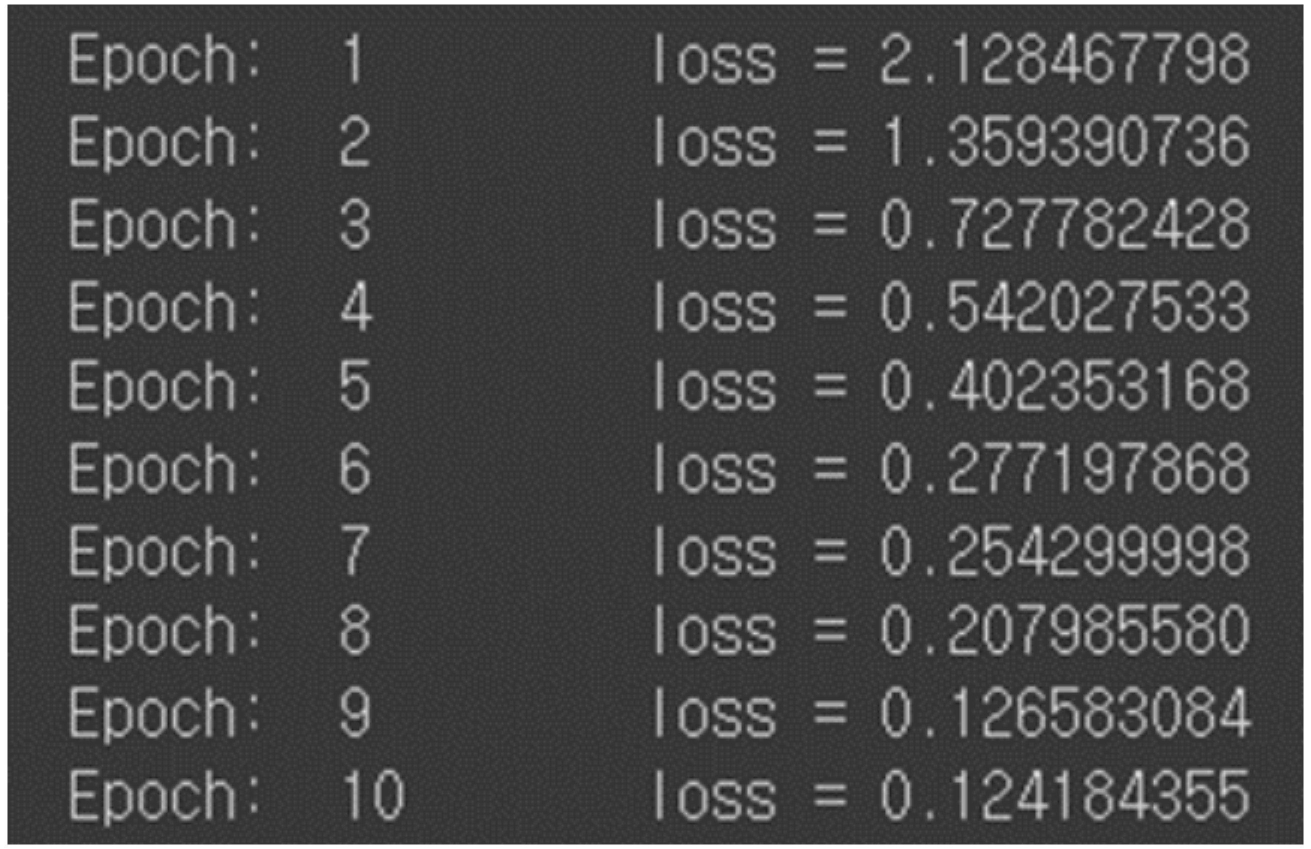


```python
class myCNN(nn.Module):
    def __init__(self):
        super(myCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, 1, padding='same')
        self.conv2 = nn.Conv2d(32, 64, 3, 1, padding='same')
        self.conv3 = nn.Conv2d(64, 128, 3, 1, padding='same')
        self.dropout = nn.Dropout2d(0.25)
        self.fc1 = nn.Linear(6272, 3000)    # 7 * 7 * 128 = 6272
        self.fc2 = nn.Linear(3000, 1000)
        self.fc3 = nn.Linear(1000, 10)

    def forward(self, x):
        x = self.conv1(x) # 28 * 28 * 32
        x = F.relu(x)
        x = F.max_pool2d(x, 2) # 14 * 14 * 32
        x = self.conv2(x) # 14 * 14 * 64
        x = F.relu(x)
        x = F.max_pool2d(x, 2) # 7 * 7 * 64
        x = self.dropout(x)
        x = self.conv3(x) # 7 * 7 * 128
        x = F.relu(x)
        x = self.dropout(x)
        x = torch.flatten(x, 1) # = 6272
        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.fc3(x)
        output = F.log_softmax(x, dim=1)
        return output
```

03 Experiment - Convolutional Layer

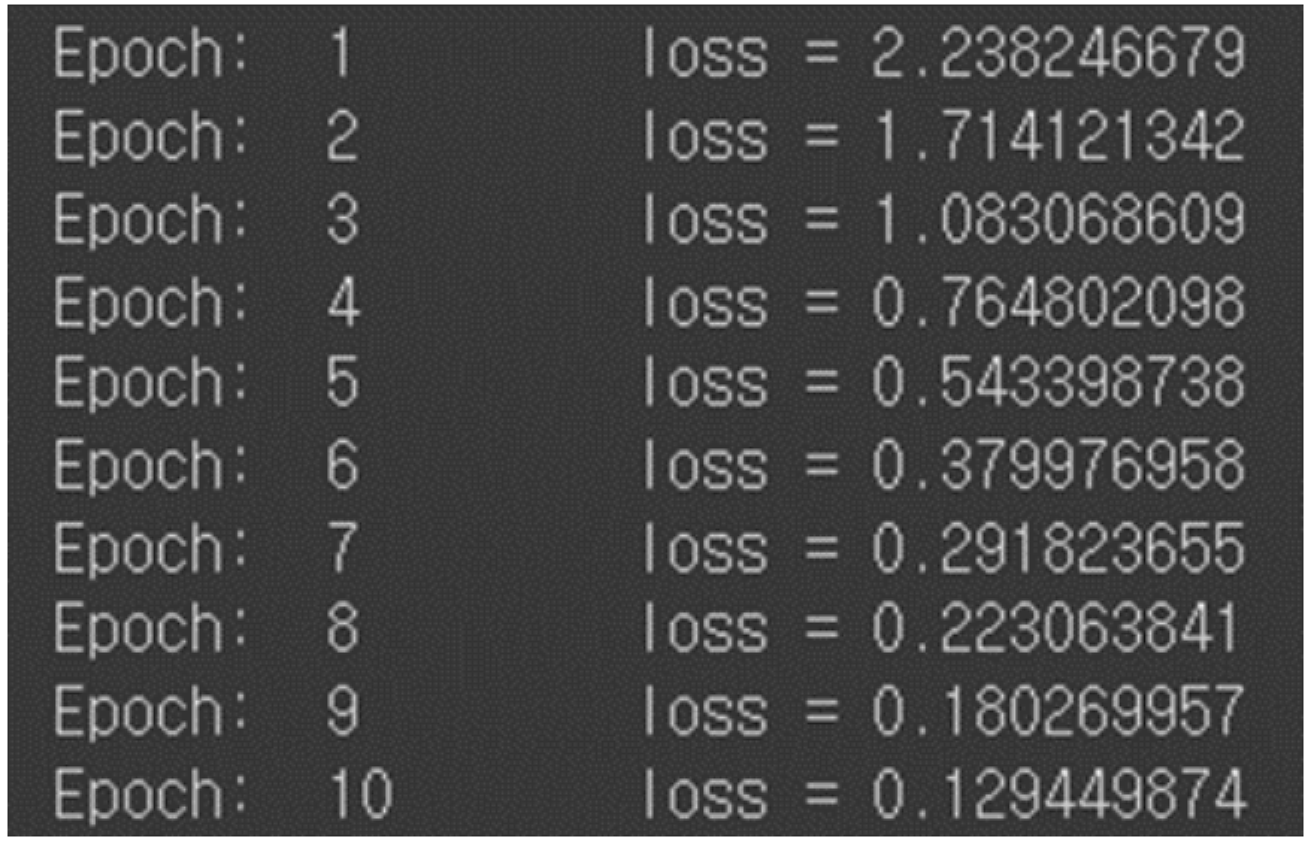


```
Epoch:  1          loss = 2.318725586
Epoch:  2          loss = 2.237199306
Epoch:  3          loss = 2.124117851
Epoch:  4          loss = 1.997388124
Epoch:  5          loss = 1.842265368
Epoch:  6          loss = 1.664745331
Epoch:  7          loss = 1.492476344
Epoch:  8          loss = 1.332738876
Epoch:  9          loss = 1.187525630
Epoch:  10         loss = 1.050387025
```

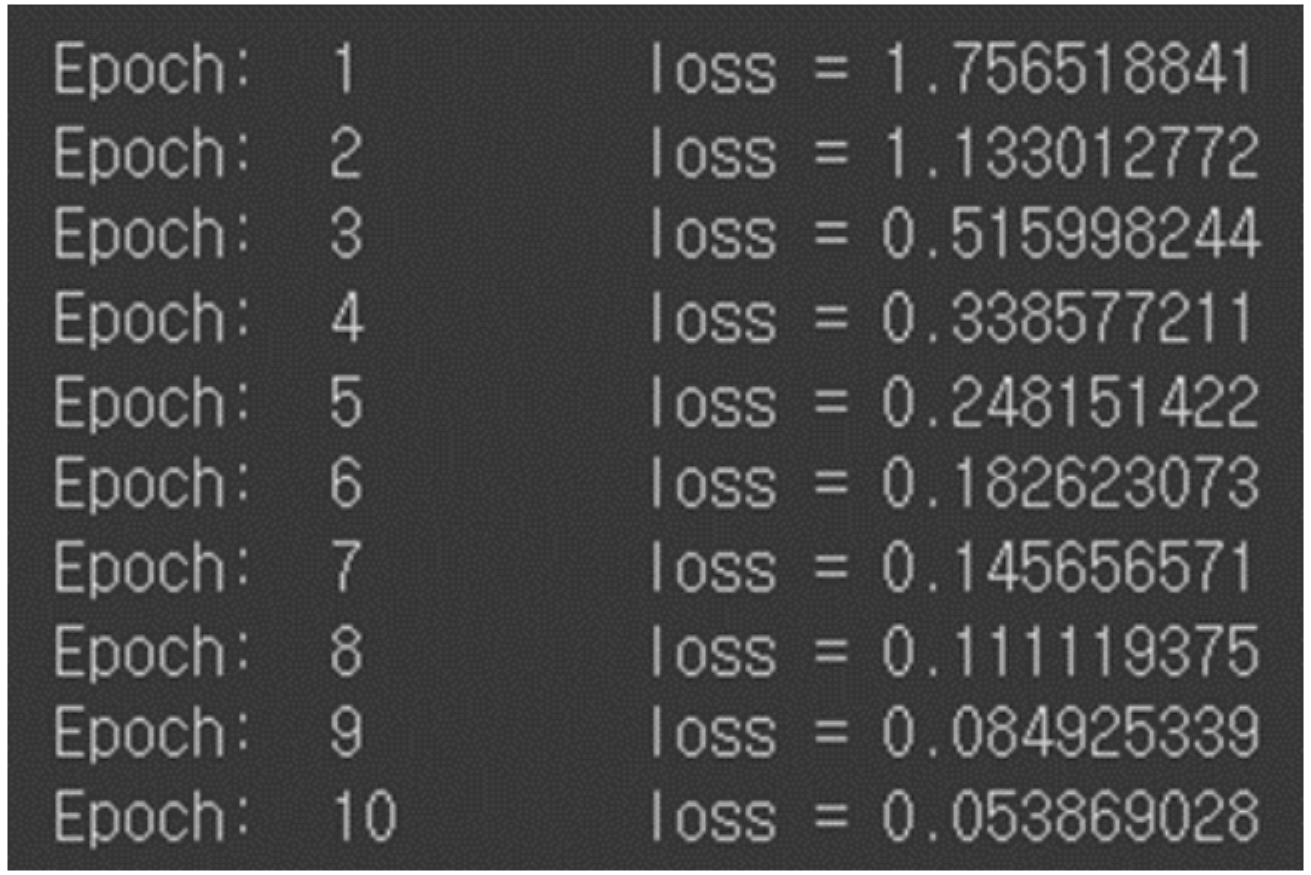


```
Epoch:  1          loss = 2.128467798
Epoch:  2          loss = 1.359390736
Epoch:  3          loss = 0.727782428
Epoch:  4          loss = 0.542027533
Epoch:  5          loss = 0.402353168
Epoch:  6          loss = 0.277197868
Epoch:  7          loss = 0.254299998
Epoch:  8          loss = 0.207985580
Epoch:  9          loss = 0.126583084
Epoch:  10         loss = 0.124184355
```

03 Experiment - Focal Loss



```
Epoch:  1          loss = 2.238246679
Epoch:  2          loss = 1.714121342
Epoch:  3          loss = 1.083068609
Epoch:  4          loss = 0.764802098
Epoch:  5          loss = 0.543398738
Epoch:  6          loss = 0.379976958
Epoch:  7          loss = 0.291823655
Epoch:  8          loss = 0.223063841
Epoch:  9          loss = 0.180269957
Epoch:  10         loss = 0.129449874
```



```
Epoch:  1          loss = 1.756518841
Epoch:  2          loss = 1.133012772
Epoch:  3          loss = 0.515998244
Epoch:  4          loss = 0.338577211
Epoch:  5          loss = 0.248151422
Epoch:  6          loss = 0.182623073
Epoch:  7          loss = 0.145656571
Epoch:  8          loss = 0.111119375
Epoch:  9          loss = 0.084925339
Epoch:  10         loss = 0.053869028
```

04 Conclusion

1. Epoch

2. Convolutional Layer

3. Focal Loss

05 Limitation

1. Visualization

2. Dataset

3. Activation Function