23-1 딥러닝 및 응용 과제

<AutoEncoder with CNN>

2020095178 최윤선

➢ 코드 설명

✓ AutoEncoder 모델 코드

<Encoder>

```
class Encoder(nn.Module):
  def __init__(self,):
    super(Encoder, self).__init__()
    self.encode = nn.Sequential(nn.Conv2d(3, 16, kernel_size=3, stride=2, padding=1),
                                nn.ReLU(),
                                nn.Conv2d(16, 32, kernel_size=3, stride=2, padding=1),
                                nn.ReLU())
  def forward(self, input):
    return self.encode(input)
```

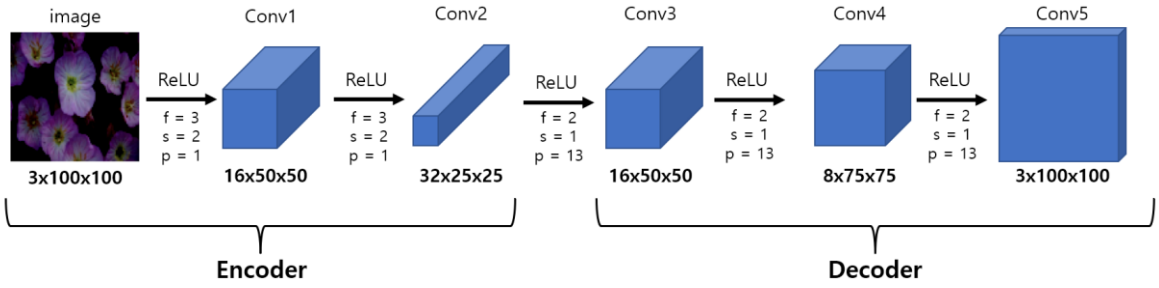
Convolution layer 두 개를 쌓아 Encoder 를 구성하였다. Activation function 은 ReLU 함수를 사용하였다.

<Decoder>

```
class Decoder(nn.Module):
  def __init__(self, ):
    super(Decoder, self).__init__()
    self.decode = nn.Sequential(nn.Conv2d(32, 16, kernel_size=2, stride=1, padding=13),
                                nn.ReLU(),
                                nn.Conv2d(16, 8, kernel_size=2, stride=1, padding=13),
                                nn.ReLU(),
                                nn.Conv2d(8, 3, kernel_size=2, stride=1, padding=13),
                                nn.ReLU())
  def forward(self, input):
    return self.decode(input)
```

Convolution layer 세 개를 쌓아 Decoder 를 구성하였다. Activation function 은 ReLU 함수를 사용하였다. Padding 을 사용하여 Transposed Convolution 과 유사한 기능을 할 수 있도록 구현하여 원래 이미지 사이즈인 3x100x100 으로 복원하였다.

Feature 의 차원을 이미지로 표현하면 다음과 같다.



✓ AutoEncoder 학습 코드

```
batch_size = 64

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size)
```

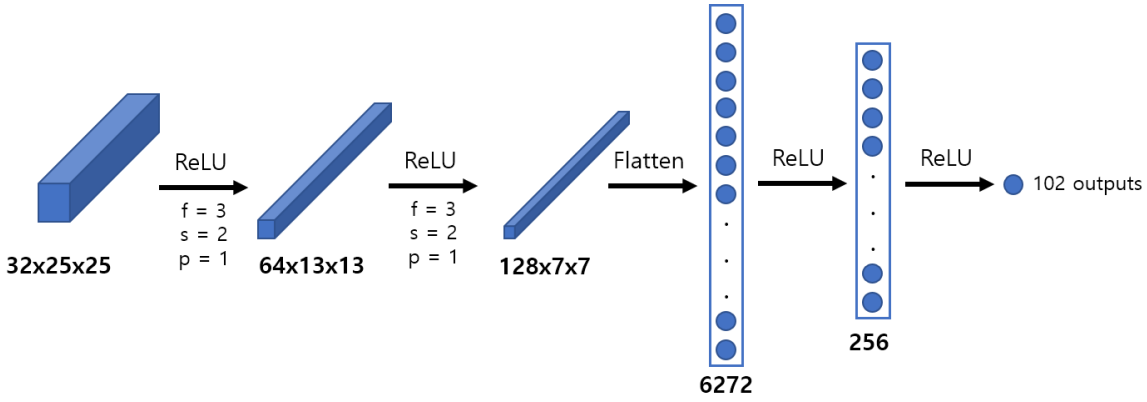
```
AE_model = AutoEncoder().to(device)
AE_optimizer = optim.Adam(AE_model.parameters(), lr=0.1)
AE_criterion = nn.MSELoss()

AE_model.train()
epochs = 10

for epoch in range(epochs):
  AE_model.train()
  avg_cost = 0
  total_batch_num = len(train_loader)

  for b_x, b_y in train_loader:
    b_x = b_x.to(device)
    b_y = b_y.to(device)

    z, b_x_hat = AE_model(b_x)
    AE_loss = AE_criterion(b_x_hat, b_x)

    avg_cost += AE_loss/total_batch_num
    AE_optimizer.zero_grad()
    AE_loss.backward()
    AE_optimizer.step()

  print('Epoch: {}/{}\t Cost: {:04f}'.format(epoch+1, epochs, avg_cost))
```

데이터를 학습시키기 위해 DataLoader 를 사용하였고, Optimizer 는 Adam 으로, loss function 은 MSELoss 로 설정하였다. Forward propagation 에서는 train_loader 에서 feature 인 b_x 와 label 인 b_y 를 가져와 AutoEncoder 모델에 b_x 를 넣어 학습시킨 후, 학습된 feature 인 b_x_hat 과 b_x 의 loss 값을 구한다. 이후 Backpropagation 을 통해 weight 값을 업데이트 한다.

✓ Classifier 모델 코드

```
class Classifier(nn.Module):
  def __init__(self):
    super(Classifier, self).__init__()
    self.classify = nn.Sequential(nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
                                  nn.ReLU(),
                                  nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
                                  nn.ReLU(), nn.Flatten(),
                                  nn.Linear(128*7*7, 256), nn.ReLU(),
                                  nn.Linear(256, 102))
  def forward(self, input):
    return self.classify(input)
```

AutoEncoder 에서 encoder 를 통해 학습한 feature 를 classifier 에 넣어 학습하므로, classifier 의 첫 input channel 은 encoder 의 output channel 과 맞게 설정한다. 따라서 classifier 의 첫 convolution layer 의 input channel 을 32 로 두었다. 내가 구현한 classifier 는 두 개의 convolution layer 를 지난 후, 1 차원 벡터로 flatten 시켜 두 개의 linear layer 를 지나도록 구현하였다. 또한, 활성화 함수로 ReLU 함수를 사용하였다. 이를 이미지로 표현하면 다음과 같다.



✓ Classifier 학습 코드

```
classifier = Classifier().to(device)
optimizer = optim.Adam([{'params':AE_model.parameters(), 'lr':0.1},
                        {'params':classifier.parameters(), 'lr':0.1}])
criterion = nn.CrossEntropyLoss()

AE_model.train()
classifier.train()
total_batch_num = len(train_loader)
epochs = 10

for epoch in range(epochs):
  avg_cost = 0

  for b_x, b_y in train_loader:
    b_x = b_x.to(device)
    b_y = b_y.to(device)

    z, b_x_hat = AE_model(b_x)
    logits = classifier(z)
    loss = criterion(logits, b_y)

    avg_cost += loss/total_batch_num
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

  print('Epoch: {}/{}\t Cost: {:04f}'.format(epoch+1, epochs, avg_cost))
```

AutoEncoder 학습 코드와 마찬가지로, 데이터를 학습시키기 위해 DataLoader 를 사용하였고, Optimizer 는 Adam 으로, loss function 은 CrossEntropyLoss()로 설정하였다. train_loader 에서 받은 b_x(feature) 데이터를 AutoEncoder 에 넣은 후, encoder 를 거쳐 나온 z 를 다시 classifier 모델에 넣어 예측 결과(logits)를 추출한다. 이 logits 와 b_y(label)를 가지고 loss 값을 측정한다. 이후 Backpropagation 을 통해 weight 값을 업데이트 한다.

✓ Classifier 정확도 측정 코드

```
correct = 0
total = 0

AE_model.eval()
classifier.eval()

for b_x, b_y in test_loader:
  b_x = b_x.to(device)
  b_y = b_y.to(device)
  with torch.no_grad():
    z, b_x_hat = AE_model(b_x)
    logits = classifier(z)

  predicts = torch.argmax(logits, dim=1)

  total += len(b_y)
  correct += (predicts == b_y.to(device)).sum().item()

accuracy = 100*correct/total

print('Accuracy: {:.2f} %'.format(accuracy))
```

Test DataLoader 를 사용하여 예측한 결과와 실제 class 를 비교한다. 맞게 예측한 data 의 개수를 count 하여 총 데이터 개수로 나누어 준 후 백분위를 계산하여 정확도를 측정하였다.

➢ **실험결과**

Epoch size, Learning rate, Optimizer 를 변화시켜 가며 실험을 진행하였다.

✓ **Epoch size 조절** (batch_size=64, lr=0.1 로 설정 후 실험 진행)

<epochs = 10>

-AutoEncoder

```
Epoch: 1/10     Cost: 0.363752
Epoch: 2/10     Cost: 0.361106
Epoch: 3/10     Cost: 0.361219
Epoch: 4/10     Cost: 0.361240
Epoch: 5/10     Cost: 0.361170
Epoch: 6/10     Cost: 0.361152
Epoch: 7/10     Cost: 0.361130
Epoch: 8/10     Cost: 0.361202
Epoch: 9/10     Cost: 0.361315
Epoch: 10/10    Cost: 0.361256
```

-Classifier

```
Epoch: 1/10     Cost: 56336.921875
Epoch: 2/10     Cost: 5.152885
Epoch: 3/10     Cost: 4.669889
Epoch: 4/10     Cost: 4.663661
Epoch: 5/10     Cost: 4.652588
Epoch: 6/10     Cost: 4.651872
Epoch: 7/10     Cost: 4.647525
Epoch: 8/10     Cost: 4.648973
Epoch: 9/10     Cost: 4.649838
Epoch: 10/10    Cost: 4.647523
```

-Accuracy

```
Accuracy: 0.46 %
```

➔ Epoch 을 10 으로 설정했을 때, 정확도가 1%도 되지 않아 성능이 매우 좋지 않음을 확인할 수 있고, Classifier 의 cost 값 또한 매우 큰 것을 확인할 수 있었다. 이에 epoch size 를 늘려 보기로 하였다.

<epochs = 30>

-AutoEncoder

```
Epoch: 1/30     Cost: 0.363752
Epoch: 2/30     Cost: 0.361106
Epoch: 3/30     Cost: 0.361219
Epoch: 4/30     Cost: 0.361240
Epoch: 5/30     Cost: 0.361170
Epoch: 6/30     Cost: 0.361152
Epoch: 7/30     Cost: 0.361130
Epoch: 8/30     Cost: 0.361202
Epoch: 9/30     Cost: 0.361315
Epoch: 10/30    Cost: 0.361256
Epoch: 11/30    Cost: 0.361253
Epoch: 12/30    Cost: 0.361207
Epoch: 13/30    Cost: 0.361277
Epoch: 14/30    Cost: 0.361221
Epoch: 15/30    Cost: 0.361290
Epoch: 16/30    Cost: 0.361257
Epoch: 17/30    Cost: 0.361184
Epoch: 18/30    Cost: 0.361235
Epoch: 19/30    Cost: 0.361232
Epoch: 20/30    Cost: 0.361278
Epoch: 21/30    Cost: 0.361168
Epoch: 22/30    Cost: 0.361261
Epoch: 23/30    Cost: 0.361207
Epoch: 24/30    Cost: 0.361286
Epoch: 25/30    Cost: 0.361244
Epoch: 26/30    Cost: 0.361366
Epoch: 27/30    Cost: 0.361275
Epoch: 28/30    Cost: 0.361266
Epoch: 29/30    Cost: 0.361274
Epoch: 30/30    Cost: 0.361262
```

-Classifier

```
Epoch: 1/30     Cost: 35004.878906
Epoch: 2/30     Cost: 6.577455
Epoch: 3/30     Cost: 4.761860
Epoch: 4/30     Cost: 4.653944
Epoch: 5/30     Cost: 4.651192
Epoch: 6/30     Cost: 4.649647
Epoch: 7/30     Cost: 4.650625
Epoch: 8/30     Cost: 4.653595
Epoch: 9/30     Cost: 4.649524
Epoch: 10/30    Cost: 4.650709
Epoch: 11/30    Cost: 4.651116
Epoch: 12/30    Cost: 4.646378
Epoch: 13/30    Cost: 4.650455
Epoch: 14/30    Cost: 4.647914
Epoch: 15/30    Cost: 4.648530
Epoch: 16/30    Cost: 4.650312
Epoch: 17/30    Cost: 4.648475
Epoch: 18/30    Cost: 4.652888
Epoch: 19/30    Cost: 4.650090
Epoch: 20/30    Cost: 4.651399
Epoch: 21/30    Cost: 4.651618
Epoch: 22/30    Cost: 4.652192
Epoch: 23/30    Cost: 4.652690
Epoch: 24/30    Cost: 4.652766
Epoch: 25/30    Cost: 4.652272
Epoch: 26/30    Cost: 4.648038
Epoch: 27/30    Cost: 4.647784
Epoch: 28/30    Cost: 4.650689
Epoch: 29/30    Cost: 4.648839
Epoch: 30/30    Cost: 4.650300
```

-Accuracy

```
Accuracy: 0.68 %
```

➔ Epoch 을 30 으로 늘려도 여전히 accuracy 가 1% 미만으로 큰 성능 차이를 보이지 못했다. 따라서 epoch size 를 조절하는 것이 아니라, learning rate 를 줄여 보기로 하였다.

✓ **Learning rate 조절** (batch_size=64, epochs=30 으로 설정 후 실험 진행)

<lr = 0.01>

-AutoEncoder

```
Epoch: 1/30     Cost: 0.370954
Epoch: 2/30     Cost: 0.361193
Epoch: 3/30     Cost: 0.361219
Epoch: 4/30     Cost: 0.361240
Epoch: 5/30     Cost: 0.361170
Epoch: 6/30     Cost: 0.361152
Epoch: 7/30     Cost: 0.361130
Epoch: 8/30     Cost: 0.361202
Epoch: 9/30     Cost: 0.361315
Epoch: 10/30    Cost: 0.361256
Epoch: 11/30    Cost: 0.361253
Epoch: 12/30    Cost: 0.361207
Epoch: 13/30    Cost: 0.361277
Epoch: 14/30    Cost: 0.361221
Epoch: 15/30    Cost: 0.361290
Epoch: 16/30    Cost: 0.361257
Epoch: 17/30    Cost: 0.361184
Epoch: 18/30    Cost: 0.361235
Epoch: 19/30    Cost: 0.361232
Epoch: 20/30    Cost: 0.361278
Epoch: 21/30    Cost: 0.361168
Epoch: 22/30    Cost: 0.361261
Epoch: 23/30    Cost: 0.361207
Epoch: 24/30    Cost: 0.361286
Epoch: 25/30    Cost: 0.361244
Epoch: 26/30    Cost: 0.361366
Epoch: 27/30    Cost: 0.361275
Epoch: 28/30    Cost: 0.361266
Epoch: 29/30    Cost: 0.361274
Epoch: 30/30    Cost: 0.361262
```

-Classifier

```
Epoch: 1/30     Cost: 4.777967
Epoch: 2/30     Cost: 4.628533
Epoch: 3/30     Cost: 4.627972
Epoch: 4/30     Cost: 4.627578
Epoch: 5/30     Cost: 4.627705
Epoch: 6/30     Cost: 4.627503
Epoch: 7/30     Cost: 4.627632
Epoch: 8/30     Cost: 4.627857
Epoch: 9/30     Cost: 4.627441
Epoch: 10/30    Cost: 4.627601
Epoch: 11/30    Cost: 4.627706
Epoch: 12/30    Cost: 4.627146
Epoch: 13/30    Cost: 4.627572
Epoch: 14/30    Cost: 4.627286
Epoch: 15/30    Cost: 4.627334
Epoch: 16/30    Cost: 4.627565
Epoch: 17/30    Cost: 4.627307
Epoch: 18/30    Cost: 4.627729
Epoch: 19/30    Cost: 4.627486
Epoch: 20/30    Cost: 4.627579
Epoch: 21/30    Cost: 4.627690
Epoch: 22/30    Cost: 4.627647
Epoch: 23/30    Cost: 4.627635
Epoch: 24/30    Cost: 4.627746
Epoch: 25/30    Cost: 4.627599
Epoch: 26/30    Cost: 4.627240
Epoch: 27/30    Cost: 4.627282
Epoch: 28/30    Cost: 4.627577
Epoch: 29/30    Cost: 4.627286
Epoch: 30/30    Cost: 4.627485
```

-Accuracy

```
Accuracy: 0.46 %
```

➔ Learning rate 를 0.1 에서 0.01 로 바꾸어도 성능에 별 영향을 끼치지 못했다. 따라서 좀 더 줄여서 실험을 진행하였다.

<lr = 0.001>

-AutoEncoder

```
Epoch: 1/30     Cost: 0.387291
Epoch: 2/30     Cost: 0.379139
Epoch: 3/30     Cost: 0.374032
Epoch: 4/30     Cost: 0.370526
Epoch: 5/30     Cost: 0.368865
Epoch: 6/30     Cost: 0.367557
Epoch: 7/30     Cost: 0.366304
Epoch: 8/30     Cost: 0.365216
Epoch: 9/30     Cost: 0.364212
Epoch: 10/30    Cost: 0.363064
Epoch: 11/30    Cost: 0.362001
Epoch: 12/30    Cost: 0.361236
Epoch: 13/30    Cost: 0.361277
Epoch: 14/30    Cost: 0.361221
Epoch: 15/30    Cost: 0.361290
Epoch: 16/30    Cost: 0.361257
Epoch: 17/30    Cost: 0.361184
Epoch: 18/30    Cost: 0.361235
Epoch: 19/30    Cost: 0.361232
Epoch: 20/30    Cost: 0.361278
Epoch: 21/30    Cost: 0.361168
Epoch: 22/30    Cost: 0.361261
Epoch: 23/30    Cost: 0.361207
Epoch: 24/30    Cost: 0.361286
Epoch: 25/30    Cost: 0.361244
Epoch: 26/30    Cost: 0.361366
Epoch: 27/30    Cost: 0.361275
Epoch: 28/30    Cost: 0.361266
Epoch: 29/30    Cost: 0.361274
Epoch: 30/30    Cost: 0.361262
```

-Classifier

```
Epoch: 1/30     Cost: 4.633357
Epoch: 2/30     Cost: 4.624183
Epoch: 3/30     Cost: 4.588961
Epoch: 4/30     Cost: 4.349771
Epoch: 5/30     Cost: 3.953976
Epoch: 6/30     Cost: 3.555044
Epoch: 7/30     Cost: 2.984323
Epoch: 8/30     Cost: 2.434753
Epoch: 9/30     Cost: 1.777004
Epoch: 10/30    Cost: 1.138965
Epoch: 11/30    Cost: 0.676833
Epoch: 12/30    Cost: 0.385674
Epoch: 13/30    Cost: 0.334122
Epoch: 14/30    Cost: 0.235445
Epoch: 15/30    Cost: 0.149276
Epoch: 16/30    Cost: 0.116870
Epoch: 17/30    Cost: 0.076581
Epoch: 18/30    Cost: 0.050650
Epoch: 19/30    Cost: 0.028287
Epoch: 20/30    Cost: 0.010237
Epoch: 21/30    Cost: 0.005928
Epoch: 22/30    Cost: 0.018648
Epoch: 23/30    Cost: 0.002449
Epoch: 24/30    Cost: 0.001173
Epoch: 25/30    Cost: 0.000740
Epoch: 26/30    Cost: 0.000492
Epoch: 27/30    Cost: 0.000361
Epoch: 28/30    Cost: 0.000277
Epoch: 29/30    Cost: 0.000228
Epoch: 30/30    Cost: 0.000196
```

-Accuracy

```
Accuracy: 13.24 %
```

➔ Learning rate 를 0.001 까지 줄이자, 정확도가 아주 높다고 할 수는 없지만 이 전의 실험 결과에 비하여 눈에 띄게 성능이 좋아진 것을 확인할 수 있다. AutoEncoder 의 loss 값을 전의 실험들과 큰 차이가 없었지만 Classifier 의 loss 값은 마지막 epoch 에서 거의 0 에 수렴하는 것을 확인할 수 있다. 정확도 또한 이전 실험에서는 1% 미만이었지만, learning rate 를 0.001 로 설정하였을 때, 13%까지 오른 것을 볼 수 있다. Learning rate 를 더 줄여도 성능이 더 좋아지는지 확인하기 위하여 한 번 더 learning rate 를 줄여 실험해 보았다.

<lr = 0.0001>

-AutoEncoder

```
Epoch: 1/30     Cost: 0.391998
Epoch: 2/30     Cost: 0.390268
Epoch: 3/30     Cost: 0.389230
Epoch: 4/30     Cost: 0.388198
Epoch: 5/30     Cost: 0.387145
Epoch: 6/30     Cost: 0.386149
Epoch: 7/30     Cost: 0.385194
Epoch: 8/30     Cost: 0.384463
Epoch: 9/30     Cost: 0.383814
Epoch: 10/30    Cost: 0.382979
Epoch: 11/30    Cost: 0.382219
Epoch: 12/30    Cost: 0.381410
Epoch: 13/30    Cost: 0.380701
Epoch: 14/30    Cost: 0.379938
Epoch: 15/30    Cost: 0.379462
Epoch: 16/30    Cost: 0.378892
Epoch: 17/30    Cost: 0.378269
Epoch: 18/30    Cost: 0.377773
Epoch: 19/30    Cost: 0.377220
Epoch: 20/30    Cost: 0.376645
Epoch: 21/30    Cost: 0.375909
Epoch: 22/30    Cost: 0.375403
Epoch: 23/30    Cost: 0.374865
Epoch: 24/30    Cost: 0.374545
Epoch: 25/30    Cost: 0.374111
Epoch: 26/30    Cost: 0.373848
Epoch: 27/30    Cost: 0.373377
Epoch: 28/30    Cost: 0.372991
Epoch: 29/30    Cost: 0.372627
Epoch: 30/30    Cost: 0.372231
```

-Classifier

```
Epoch: 1/30     Cost: 4.627980
Epoch: 2/30     Cost: 4.621387
Epoch: 3/30     Cost: 4.613539
Epoch: 4/30     Cost: 4.588907
Epoch: 5/30     Cost: 4.529284
Epoch: 6/30     Cost: 4.409223
Epoch: 7/30     Cost: 4.214028
Epoch: 8/30     Cost: 3.962528
Epoch: 9/30     Cost: 3.686557
Epoch: 10/30    Cost: 3.382867
Epoch: 11/30    Cost: 3.177238
Epoch: 12/30    Cost: 2.914681
Epoch: 13/30    Cost: 2.708107
Epoch: 14/30    Cost: 2.452795
Epoch: 15/30    Cost: 2.235297
Epoch: 16/30    Cost: 2.013366
Epoch: 17/30    Cost: 1.819559
Epoch: 18/30    Cost: 1.595311
Epoch: 19/30    Cost: 1.451716
Epoch: 20/30    Cost: 1.256552
Epoch: 21/30    Cost: 1.133675
Epoch: 22/30    Cost: 0.960721
Epoch: 23/30    Cost: 0.837079
Epoch: 24/30    Cost: 0.719887
Epoch: 25/30    Cost: 0.543887
Epoch: 26/30    Cost: 0.481953
Epoch: 27/30    Cost: 0.391703
Epoch: 28/30    Cost: 0.315880
Epoch: 29/30    Cost: 0.257479
Epoch: 30/30    Cost: 0.227653
```

-Accuracy

```
Accuracy: 8.77 %
```

➔ Learning rate 를 0.0001 까지 낮추었더니 0.001 일 때보다 Classifier 의 loss 값은 더 커지고 정확도도 더 낮아진 것을 확인할 수 있었다. 따라서 성능이 가장 좋게 나왔던 epoch size 는 30, learning rate 는 0.001 을 유지한 채 다음 실험을 진행하였다.

✓ **Batch Normalization** (batch_size=64, epochs=30, lr=0.001 로 설정 후 실험 진행)

각 convolution layer 이후에 batch normalization 을 추가하여 실험을 진행하였다.

```
class Encoder(nn.Module):
  def __init__(self,):
    super(Encoder, self).__init__()
    self.encode = nn.Sequential(nn.Conv2d(3, 16, kernel_size=3, stride=2, padding=1),
                                nn.ReLU(), nn.BatchNorm2d(16),
                                nn.Conv2d(16, 32, kernel_size=3, stride=2, padding=1),
                                nn.ReLU(), nn.BatchNorm2d(32))
  def forward(self, input):
    return self.encode(input)

class Decoder(nn.Module):
  def __init__(self, ):
    super(Decoder, self).__init__()
    self.decode = nn.Sequential(nn.Conv2d(32, 16, kernel_size=2, stride=1, padding=13),
                                nn.ReLU(), nn.BatchNorm2d(16),
                                nn.Conv2d(16, 8, kernel_size=2, stride=1, padding=13),
                                nn.ReLU(), nn.BatchNorm2d(8),
                                nn.Conv2d(8, 3, kernel_size=2, stride=1, padding=13),
                                nn.ReLU(), nn.BatchNorm2d(3))
```

```
class Classifier(nn.Module):
  def __init__(self):
    super(Classifier, self).__init__()
    self.classify = nn.Sequential(nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
                                  nn.BatchNorm2d(64), nn.ReLU(),
                                  nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
                                  nn.BatchNorm2d(128), nn.ReLU(), nn.Flatten(),
                                  nn.Linear(128*7*7, 256), nn.ReLU(),
                                  nn.Linear(256, 102))
```

-Autoencoder

```
Epoch: 1/30     Cost: 1.224686
Epoch: 2/30     Cost: 1.149044
Epoch: 3/30     Cost: 1.103871
Epoch: 4/30     Cost: 1.065303
Epoch: 5/30     Cost: 1.029874
Epoch: 6/30     Cost: 0.995550
Epoch: 7/30     Cost: 0.964321
Epoch: 8/30     Cost: 0.932917
Epoch: 9/30     Cost: 0.902536
Epoch: 10/30    Cost: 0.874551
Epoch: 11/30    Cost: 0.848420
Epoch: 12/30    Cost: 0.823801
Epoch: 13/30    Cost: 0.799003
Epoch: 14/30    Cost: 0.775812
Epoch: 15/30    Cost: 0.754119
Epoch: 16/30    Cost: 0.733225
Epoch: 17/30    Cost: 0.713288
Epoch: 18/30    Cost: 0.693842
Epoch: 19/30    Cost: 0.674943
Epoch: 20/30    Cost: 0.657089
Epoch: 21/30    Cost: 0.640108
Epoch: 22/30    Cost: 0.622920
Epoch: 23/30    Cost: 0.606691
Epoch: 24/30    Cost: 0.588993
Epoch: 25/30    Cost: 0.530418
Epoch: 26/30    Cost: 0.509349
Epoch: 27/30    Cost: 0.495834
Epoch: 28/30    Cost: 0.484120
Epoch: 29/30    Cost: 0.473130
Epoch: 30/30    Cost: 0.463094
```

-Classifier

```
Epoch: 1/30     Cost: 4.579401
Epoch: 2/30     Cost: 3.681389
Epoch: 3/30     Cost: 2.816148
Epoch: 4/30     Cost: 1.987860
Epoch: 5/30     Cost: 1.200297
Epoch: 6/30     Cost: 0.602762
Epoch: 7/30     Cost: 0.262633
Epoch: 8/30     Cost: 0.106519
Epoch: 9/30     Cost: 0.053892
Epoch: 10/30    Cost: 0.029365
Epoch: 11/30    Cost: 0.019942
Epoch: 12/30    Cost: 0.015449
Epoch: 13/30    Cost: 0.012002
Epoch: 14/30    Cost: 0.010194
Epoch: 15/30    Cost: 0.008885
Epoch: 16/30    Cost: 0.007571
Epoch: 17/30    Cost: 0.006624
Epoch: 18/30    Cost: 0.005909
Epoch: 19/30    Cost: 0.005833
Epoch: 20/30    Cost: 0.004890
Epoch: 21/30    Cost: 0.004696
Epoch: 22/30    Cost: 0.004191
Epoch: 23/30    Cost: 0.003920
Epoch: 24/30    Cost: 0.003679
Epoch: 25/30    Cost: 0.003323
Epoch: 26/30    Cost: 0.003178
Epoch: 27/30    Cost: 0.002810
Epoch: 28/30    Cost: 0.002696
Epoch: 29/30    Cost: 0.002441
Epoch: 30/30    Cost: 0.002278
```

-Accuracy

```
Accuracy: 23.45 %
```

➔ AutoEncoder 학습 후에는 loss 값이 약간 늘어나긴 했지만 최종적으로 정확도가 23.45%까지 오른 것을 확인할 수 있었다.

- **결론**

내가 구현한 CNN AutoEncoder 와 Classifier 를 사용하여 실험한 결과, epoch size 를 30, learning rate 를 0.001 로 설정 후 Batch normalization 을 추가했을 때 가장 좋은 성능을 보여주었다. 그러나 성능이 가장 좋을 때 loss 값은 0 에 수렴했을지라도 정확도는 약 23% 정도로 그렇게 높지 않았다. 이는 train dataset 과 test dataset 의 개수를 확인해 본 결과 train data 의 개수보다 test data 의 개수가 훨씬 많았기에 overfitting 을 초래하여 낮은 정확도를 보인 것 같다.

cf)

```
print(len(train_dataset))
print(len(test_dataset))
```

```
1020
6149
```