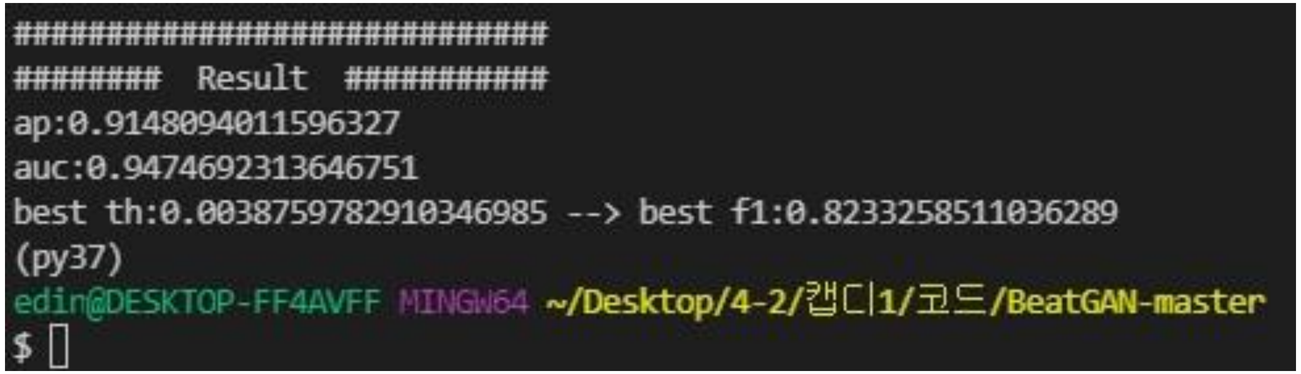
6번째 미팅발표

임예딘

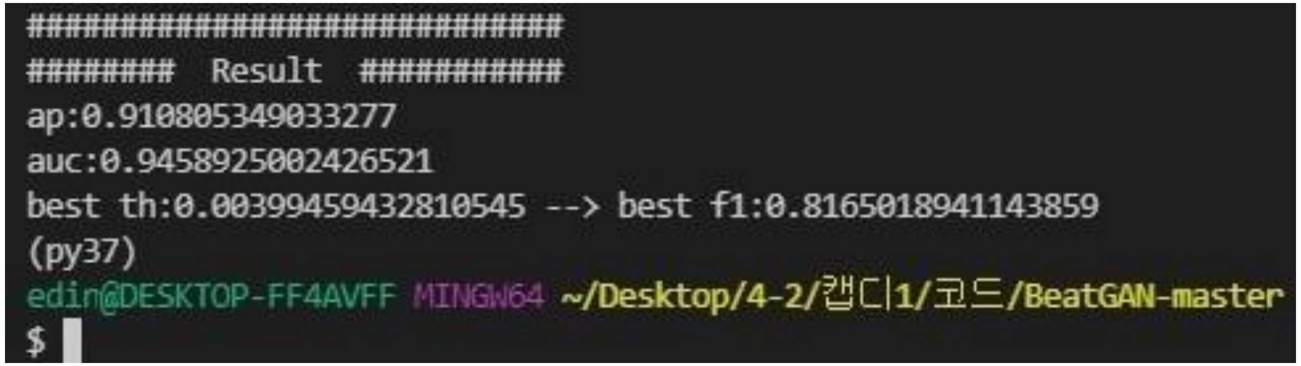
코멘트

성능 향상을 위한 아이디어
새로운 시도, 방법 고안…

아쉬운 성능 향상



##############################
######## Result ###########
ap:0.9148094011596327
auc:0.9474692313646751
best th:0.0038759782910346985 --> best f1:0.8233258511036289
(py37)
edin@DESKTOP-FF4AVFF MINGW64 ~/Desktop/4-2/캡디1/코드/BeatGAN-master
$



##############################
######## Result ###########
ap:0.910805349033277
auc:0.9458925002426521
best th:0.00399459432810545 --> best f1:0.8165018941143859
(py37)
edin@DESKTOP-FF4AVFF MINGW64 ~/Desktop/4-2/캡디1/코드/BeatGAN-master
$

활용하지 못했던 Feature Distillation Loss

$$\mathcal{L}_F(T, F; \theta_c, \theta_t) = \Sigma_{i=1}^{n} ||\phi(T_i) - \phi(F_i)||_2$$

```
class Attention(nn.Module):
    def __init__(self, args):
        super(Attention, self).__init__()
        self.p = 2
        self.kd = DistillKL(args)
        self.alpha = args.alpha
        self.beta = args.beta

    def forward(self, o_s, o_t, g_s, g_t):
        loss = self.alpha * self.kd(o_s, o_t)
        loss += self.beta * sum([self.at_loss(f_s, f_t.detach()) for f_s, f_t in zip(g_s, g_t)])

        return loss

    def at_loss(self, f_s, f_t):
        return (self.at(f_s) - self.at(f_t)).pow(2).mean()

    def at(self, f):
        return F.normalize(f.pow(self.p).mean(1).view(f.size(0), -1))
```

Reminder: Various distillation methods

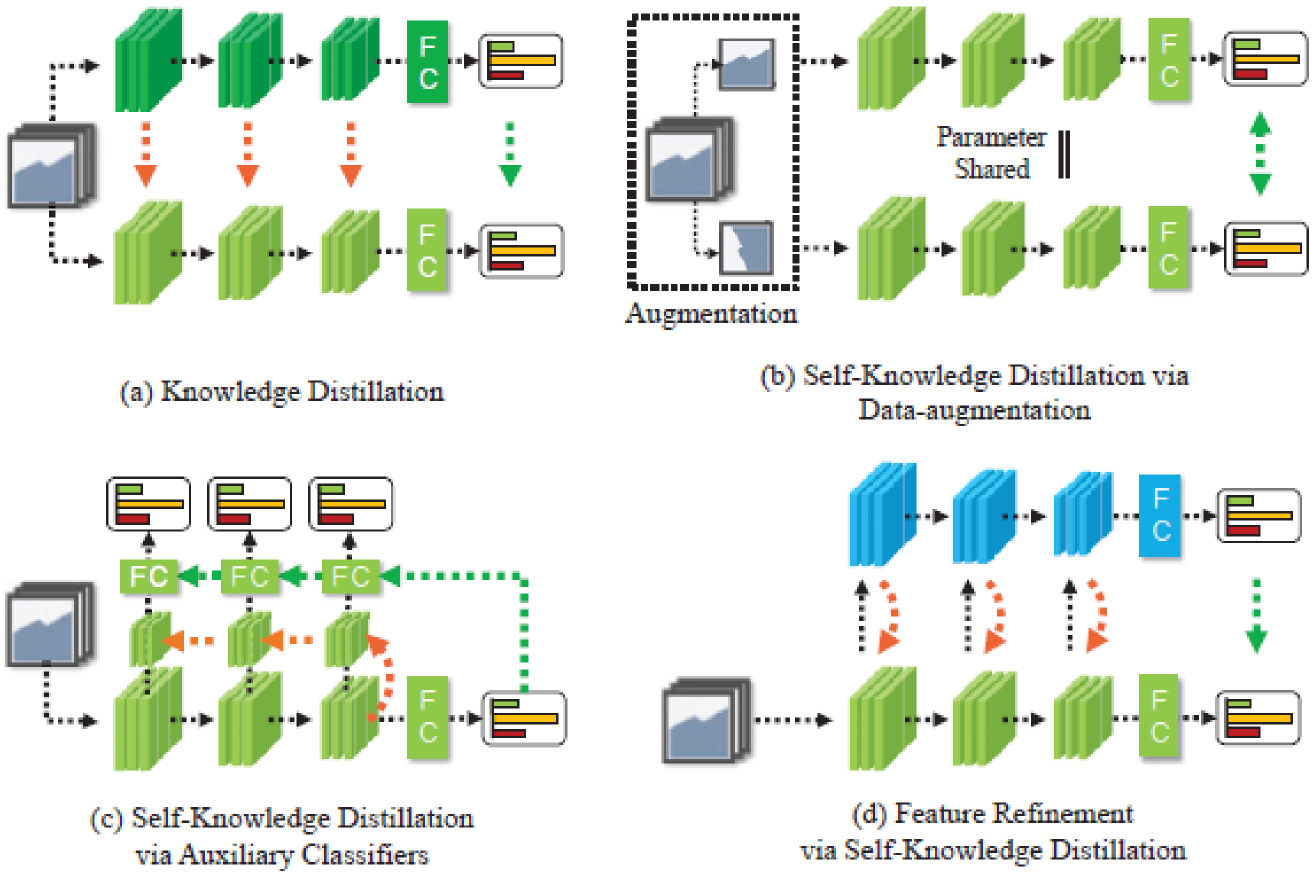


Figure 1: Comparison of various distillation methods. The black line is the forward path; the green line is the soft label distillation; and the orange line is the feature distillation.

개선 아이디어

1. Pretrained (teacher) model 없이 self-knowledge distillation하는 코드로 구성
    - Utilize an auxiliary network
2. Attention을 제대로 활용 → 1D data에서는 how?