ML2021Spring HW4 Report

<Mechanical Engineering> <陳志臻>
<B07502071>

Public Score	Private Score
0.96714	0.96611

The methods I used to pass the strong baselines include:

1. Conformer import:

```
! git clone https://github.com/sooftware/conformer.git
! cd conformer && pip install -e . && cd ..
! mv conformer shit && mv shit/conformer .
from conformer.encoder import ConformerEncoder

class Classifier(nn.Module):
  def __init__(self, d_model=80, n_spks=600, dropout=0.1):
    super().__init__()
   self.encoder=ConformerEncoder(input_dim=d_model,encoder_dim=256,num_layers=2,
num_attention_heads=2)
    self.pred_layer = nn.Sequential(
      # nn.Linear(d_model, d_model),
      nn.ReLU(),
      nn.Linear(256, n_spks),
    )

  def forward(self, mels):

    out = self.prenet(mels)
    out, _ = self.encoder(out,out.size(1))
    stats = out.mean(dim=1)
    out = self.pred_layer(stats)
    return out
```

2. Conformer from transformerEncoderLayers:

```
class ConformerEncoderLayer(nn.Module):

    def __init__(self, d_model, nhead, dim_feedforward=2048, dropout=0.1, activation="rel
u"):
        super(ConformerEncoderLayer, self).__init__()
        self.self_attn = nn.MultiheadAttention(d_model, nhead, dropout=dropout)

        self.conv1 = nn.Conv1d(d_model, dim_feedforward, kernel_size=1)
        self.dropout = nn.Dropout(dropout)
        self.conv2 = nn.Conv1d(dim_feedforward, d_model, kernel_size=1)
        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.dropout1 = nn.Dropout(dropout)
        self.dropout2 = nn.Dropout(dropout)
        self.activation = nn.ReLU()

    def __setstate__(self, state):
        if 'activation' not in state:
            state['activation'] = F.relu
        super(ConformerEncoderLayer, self).__setstate__(state)

    def forward(self, src: Tensor, src_mask: Optional[Tensor] = None, src_key_padding_ma
sk: Optional[Tensor] = None) -> Tensor:

        src2 = self.self_attn(src, src, src, attn_mask=src_mask,
                              key_padding_mask=src_key_padding_mask)[0]
        src = src + self.dropout1(src2)
        src = self.norm1(src)
        src = src.permute(1,2,0)
        src2 = self.conv1(src)
        src2 = self.dropout(self.activation(src2))
        src2 = self.conv2(src2)
        src = src + self.dropout2(src2)
        src = src.transpose(1,2)
        src = self.norm2(src)
        src = src.transpose(1,2)
        return src
```

(Your report should be written in English. Do not exceed 100 words describing your methods, but you may add comments to your code to make other students easier to understand.)