模型來源:李宏毅教授深度學習課程

https://colab.research.google.com/drive/1u-610KA-urqfJjDH5O0pecwfP--V9DQs?usp=sharing

模型任務:聲音特徵分類
原本的模型架構:

```python
class Classifier(nn.Module):
    def __init__(self, d_model=80, n_spks=600, dropout=0.1):
        super().__init__()
        # Project the dimension of features from that of input into d_model.
        self.prenet = nn.Linear(40, d_model)
        # TODO:
        #   Change Transformer to Conformer.
        #   https://arxiv.org/abs/2005.08100
        self.encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_model, dim_feedforward=256, nhead=2
        )
        # self.encoder = nn.TransformerEncoder(self.encoder_layer, num_layers=2)

        # Project the the dimension of features from d_model into speaker nums.
        self.pred_layer = nn.Sequential(
            nn.Linear(d_model, d_model),
            nn.Sigmoid(),
            nn.Linear(d_model, n_spks),
        )

    def forward(self, mels):
        """
        args:
            mels: (batch size, length, 40)
        return:
            out: (batch size, n_spks)
        """
        # out: (batch size, length, d_model)
        out = self.prenet(mels)
        # out: (length, batch size, d_model)
        out = out.permute(1, 0, 2)
        # The encoder layer expect features in the shape of (length, batch size, d_model).
        out = self.encoder_layer(out)
        # out: (batch size, length, d_model)
        out = out.transpose(0, 1)
        # mean pooling
        stats = out.mean(dim=1)

        # out: (batch, n_spks)
        out = self.pred_layer(stats)
        return out
```

修改過後的模型架構:

增加 self-attention pooling、head 數改成 8

```
class SelfAttentionPooling(nn.Module):
        def __init__(self, d_model):
                super(SelfAttentionPooling, self).__init__()
                self.attention = nn.Linear(d_model, 1)

        def forward(self, x):
                # x: (length, batch size, d_model)
                attn_weights = torch.softmax(self.attention(x), dim=0)
                attn_output = torch.sum(attn_weights * x, dim=0)
                return attn_output
```

```
class Classifier(nn.Module):
    def __init__(self, d_model=80, n_spks=600, dropout=0.1):
        super().__init__()

        self.prenet = nn.Linear(40, d_model)

        self.encoder_layer = nn.TransformerEncoderLayer(
            d_model=d_model, dim_feedforward=256, nhead=8
        )

        self.self_attention_pooling = SelfAttentionPooling(d_model)
        self.pred_layer = nn.Sequential(
            nn.Linear(d_model, d_model),
            nn.Sigmoid(),
            nn.Linear(d_model, d_model),
            nn.Sigmoid(),
            nn.Linear(d_model, n_spks),
        )


    def forward(self, mels):

        out = self.prenet(mels)
        out = out.permute(1, 0, 2)
        out = self.encoder_layer(out)
        out = self.self_attention_pooling(out)
        out = self.pred_layer(out)
```

修改過後的效果:53.2%

```
Train: 100% 2000/2000 [00:42<00:00, 46.67 step/s, accuracy=0.59, loss=1.87, step=7e+4]
Valid: 100% 5664/5667 [00:02<00:00, 1911.74 uttr/s, accuracy=0.53, loss=2.17]
Train:   0% 0/2000 [00:00<?, ? step/s]
Step 70000, best model saved. (accuracy=0.5323)
```

心得:

一開始加了 self-attention pooling 沒甚麼效果，後來把 head 數改成 8 之後效果才變好。