Appendix

Algorithm 1: Pseudo code of attention layer in a PyTorch-like style.

```
/* B: batch size, H: height, W:
   width, C: channel number, M: head
   number, ak: Attention kernel
   size, s: stride                    */
Input : query(Q), key (K), and value (V)
// ######## Initialization ########
Unfold = nn.Unfold(ak, s);
Fold = nn.Fold(output_size = (H, W), ak, s);
AvgPooling = nn.AdaptiveAvgPool3d((M, H//2,
  W//2))
// ######## Forward pass ########
Q = Q.view(B*M, C, H, W);
K = K.view(B*M, C, H, W);
V = V.view(B*M, C, H, W);
/* T(•) in Eq.(3) :
   (K, ∏(Patch Size), B)              */
unfold_q = Unfold(Q).view(B*M, C, ak*ak, -1);
unfold_k = Unfold(K).view(B*M, C, ak*ak, -1);
unfold_v = Unfold(V).view(B*M, C, ak*ak, -1);
// Q⋆K: A(q_i^p, K_j^p)^b in Eq.(4)
attention = unfold_q*unfold_k;
attention = Softmax(attention);
// Attention(Q^p, K^p, V^p)^b in Eq.(5)
out_att = (attention*unfold_v);
out_att = out_att.view(B*M, C*ak*ak, -1);
// T(•)^{-1} in Eq.(6)
out_att = Fold(out_att);
// Adaptive average pooling in
   Eq.(7)
out_att = AvgPooling(out_att)
Return out_att
```

A. Setting of Spectrogram Parameters

In Table I, we show the parameters used to create spectrograms from SHL 2018, WISDM, and SLEEP-EDF-20 in our paper.

TABLE I
THE SETTINGS OF SPECTROGRAM PARAMETERS.

Dataset	Sampling Frequency	Size of FFT	Length of hop	Padding
SHL 2018	100Hz	500	10	Yes
WSIDM	20Hz	40	10	Yes
SLEEP-EDF-20	100Hz	500	10	Yes

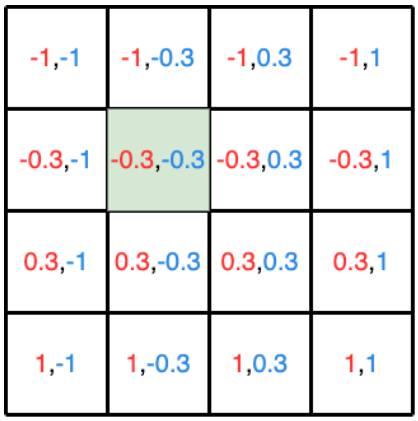

Fig. 1. An example of relative distance computation (a 4×4 spectrogram). The relative distances are computed with respect to the position of the highlighted pixel. The format of distances is row offset, column offset ($\mathbb{R}^{2\times4\times4}$).

B. Positional Encoding

Positional encoding is widely adopted in self-attention modules. In this paper, we adopt positional encoding in the adaptive convolution module. Specifically, the popular relative positional encoding [1] is adopted when computing the attention weights, where the position information is added to the value of **K** (Algorithm 1, line 8). The row and column offsets are associated with an pixel embedding ($pixel\_embedding \in [-1, 1]$), in which a one-dimensional tensor is created whose values are evenly spaced from -1 to 1 for each of the row and column, respectively. Eventually, a position matrix is generated with the size of $\mathbb{R}^{2\times H\times W}$, as shown in Fig. 1. The relative distances are computed using a 1×1 convolution, where the embedding is convert from $\mathbb{R}^{2\times H\times W}$ to $\mathbb{R}^{1\times H\times W}$ (positional encoding).

C. Pseudo code of Attention Layer

As the core operations $\mathcal{T}$ and $\mathcal{T}^{-1}$ in the attention layer are implemented using the PyTorch functions, to facilitate understanding, a pseudo code of the attention layer is illustrated in Algorith 1 using a PyTorch-like style.

REFERENCES

[1] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.