recorded signals at the individual electrodes are already a smoothened mixture of multiple signals coming from different underlying sources. This mixing and smoothing make the underlying spatial pattern extremely hard to comprehend and thwarts the classifiication performance.

The proposed method

This main idea of work is to leverage both spatial and temporal context in terms of inter-region interactions and inherent temporal patterns, respectively. They propose a novel EEG-ConvTransformer Network, a deep learning network that employs a series of 'ConvTransformer' modules, each consisting of *multi-head attention* and *convolutional feature expansion*, to learn the inter-region representational similarities together with the inherent temporal activities.

Implementation Details

1 Azimuthal Equidistant Projection

The AEP allows the relative distances between the neighboring electrodes of EEG to be preserved while projecting them from their location in 3-dimensional space to a 2-dimensional surface.

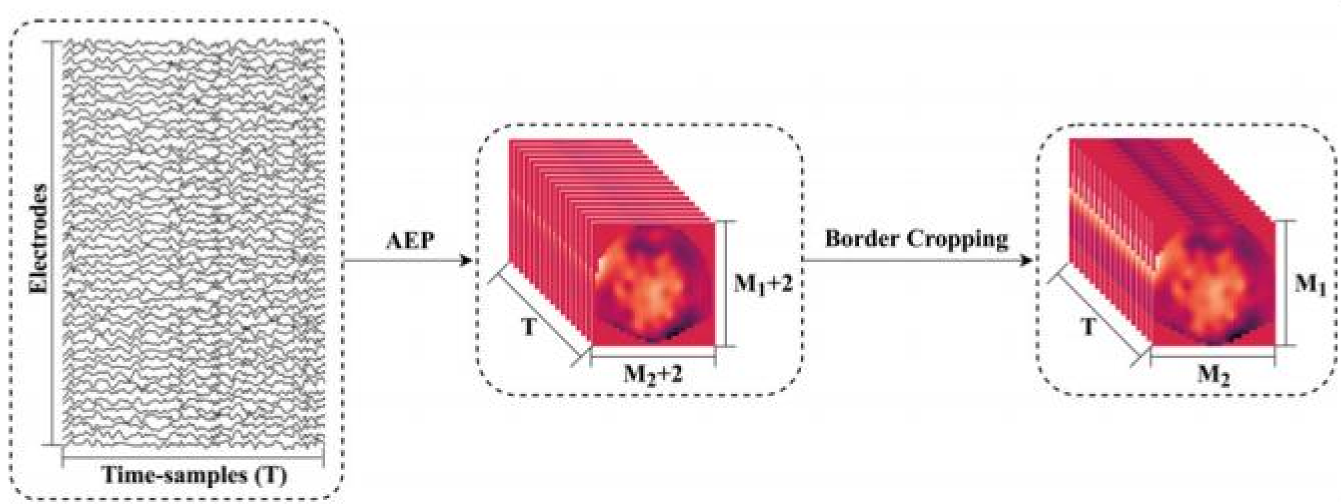


Fig. 1. Visualization of the transformation of multi-channel EEG data to time-frames of activity maps. Firstly, topology preserving Azimuthal Equidistant Projection (AEP) is used to convert the 3-D electrode

signals at each time point into a 2-D EEG-image, creating time-frames of activity maps. Next, border cropping is applied to prepare the maps for network training.

2 TheLocalFeatureExtractor

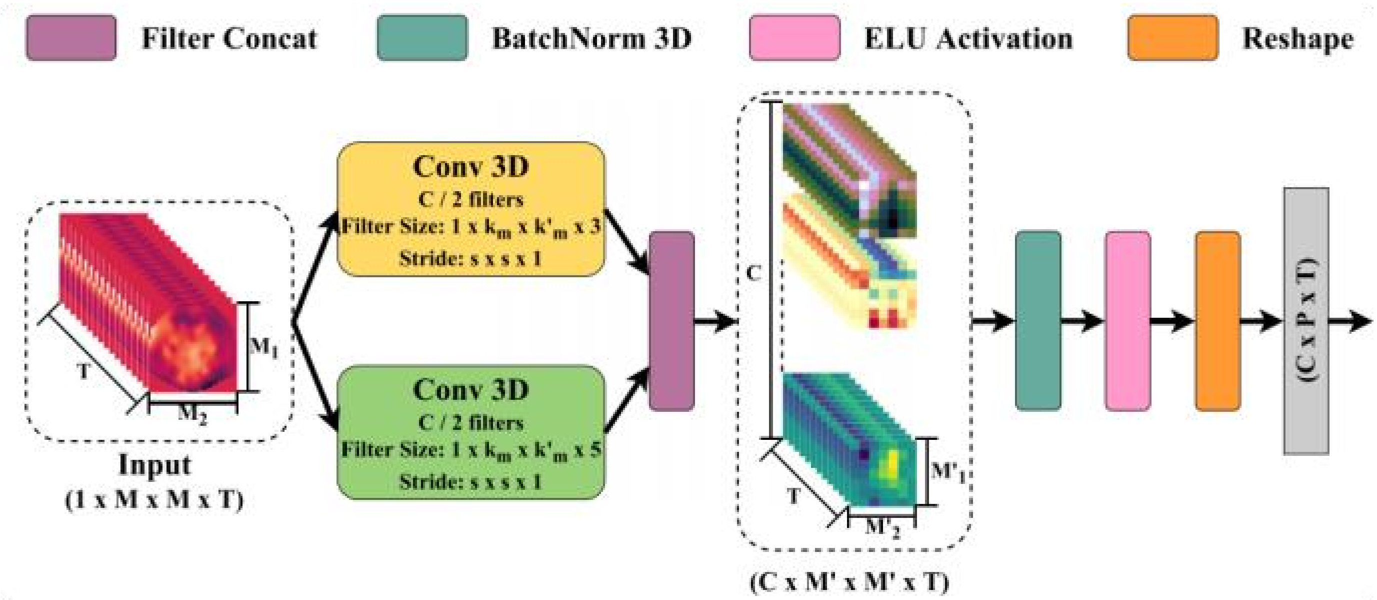


Fig. 2. Visualization of the LFE module. EEG activity maps are fifiltered using 3-D spatio-temporal convolution kernels to generate C feature maps for each time-frame. To capture the inherent temporal patterns, size of the kernels is varied along the temporal axis (kt ∈ [3, 5]). P=M'*M' indicates patches that represent brain regions instead of eletrodes, since they are high-dimentional features come from Convolution layers.

3 Multi-Head Attention

Each self-attention head, $h \in [1, 2, . . ., H]$, with H being the total number of heads, relies on q_P (queries), k_P (keys) and v_P (values) vectors for patch assessment. The patch representations are projected to latent representations of $q_P, k_P, v_P \in \mathrm{R}^{D\times P\times T}$, through point-wise convolution operations [34], such that $D << C$. The point-wise convolution operation refers to convolution with a kernel size of 1, which, without any bias, encodes individual patches to smaller channel spaces. This encoding reduces the computational complexity of the matrix-multiplications invoked right after these latent representations permute axis-wise and reshape such that $q_P, k_P, v_P \in \mathrm{R}^{P\times D.T}$ *Thencalculate themlike the previous transformers.*

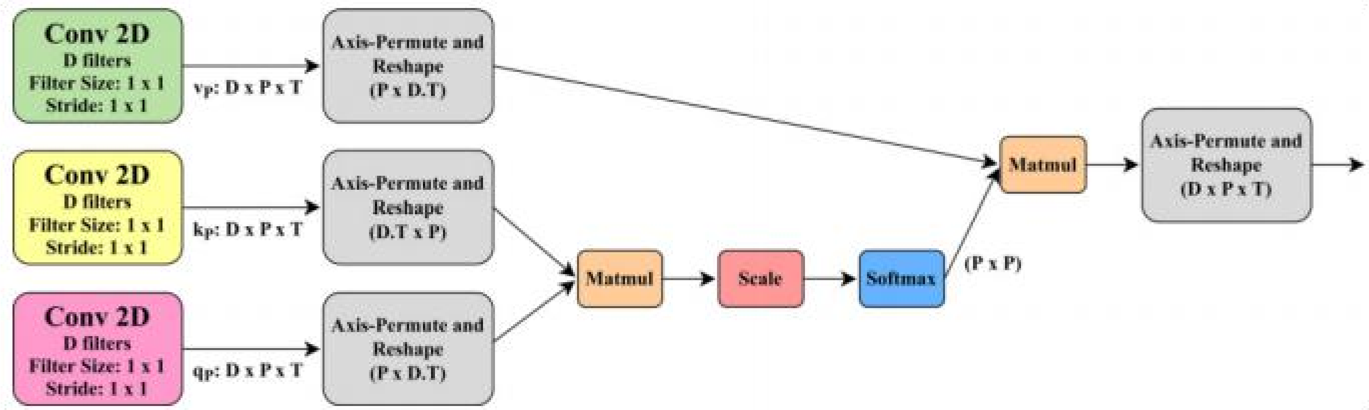


Fig. 3. Structure of a single self-attention head. For a given set of patches, the module fifirst calculates the queries (q_P), keys (k'_P) and values (v_P). Inter-patch similarities are identified by applying Softmax to the scaled dot product of q_P and k'_P The resulting attention vector is multiplied with v_P to generate the output of a single head. The *D* is calculated as *C/H.*

4 Convolutional Feature Expansion

The output of the MHA module, $x^{[MHA]} \in R^{C\times P\times T}$, gets forwarded to a temporal convolutional sub-module to extract patch-wise temporal features. The convolution operations performed by every kernel $\hat{k}^{e} \in R^{C\times 1\times kt}$, such that $e \in [1, 2, \ldots, E]$ for the expanded number of channels *E*, should extract suffiffifficient temporal features to aid the classifification. Like the LFE module, for each value of $kt \in [3, 5]$, *E/*2 number of channels are generated, followed by channel concatenation. This Conv layer is followed by BatchNorm and ELU non-linearity. Next, a point-wise convolution maps the temporally fifiltered features to the initial channel space *C*, as required for residual mapping, followed by a BatchNorm. Together, these layers form the Convolutional Feature Expansion (CFE) submodule, visualized in Fig. 4.

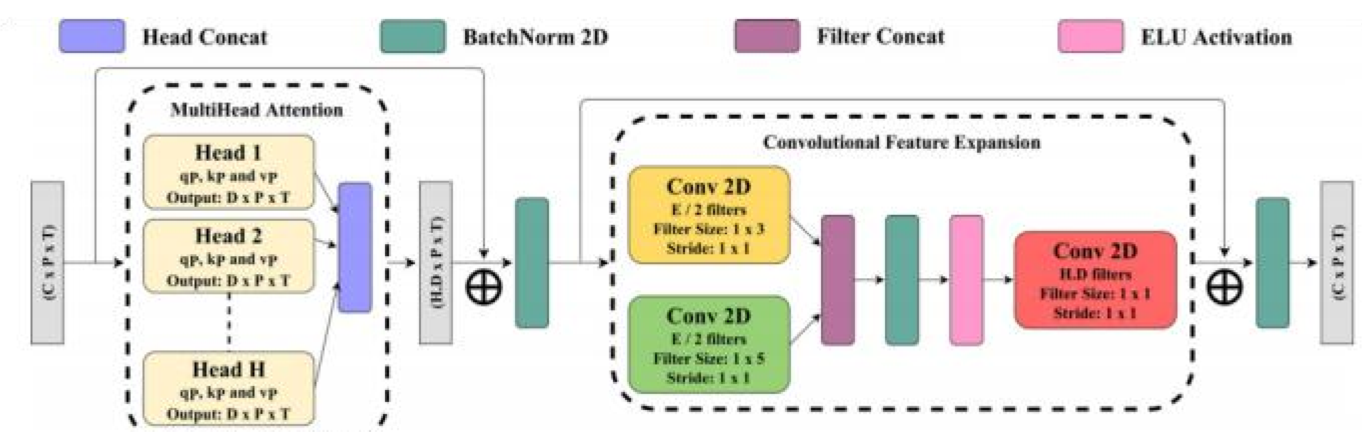
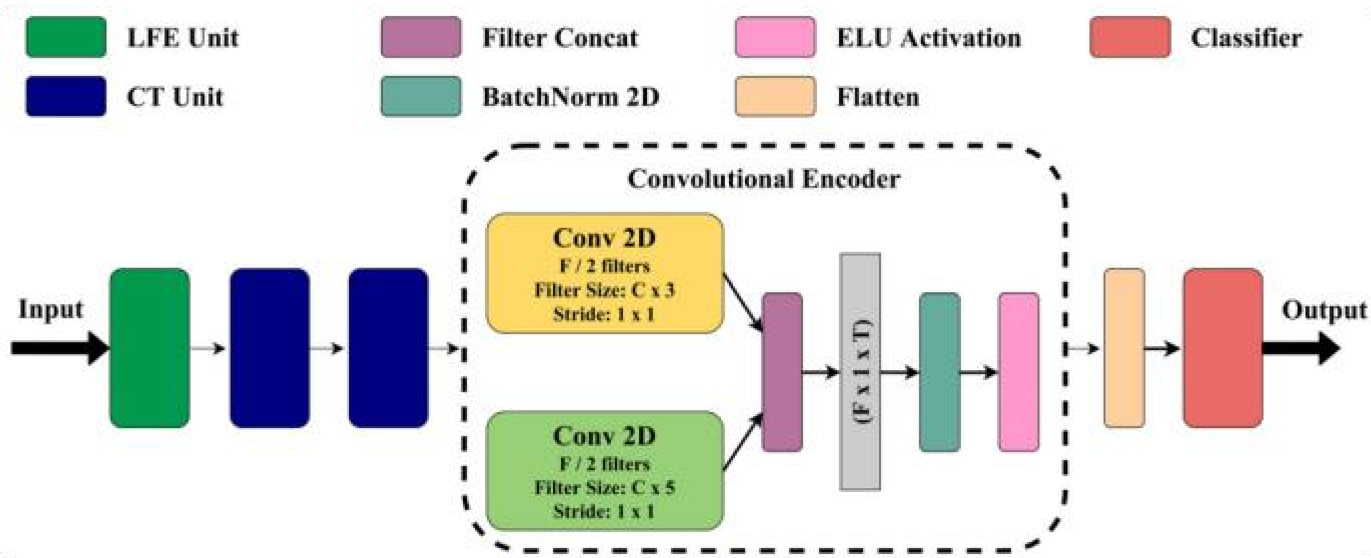


Fig. 4. Visualization of the proposed EEG-ConvTransformer module. Multi-Head Attention (MHA) module, consisting of multiple heads in parallel, is succeeded by a Convolutional Feature Expansion (CFE) module. Temporal kernels in CFE are designed to generate wide feature maps which are then combined using point-wise convolution to regain the original dimensions.

5 classifier



Success

- ✅ All modules have been built including the five modules described above.
- ✅ Training and test script

Problems

The Azimuthal Equidistant Projection is come from:

[1] Bashivan, et al. "Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks." International conference on learning representations (2016).

It's must be noted that there is a different way to deal with EEG raw data than citation [1] which described as "*However, contrary to the three frequency power bands from the earlier work, the AEP and interpolation are applied to the preprocessed signal to form a singlechannel mesh of G1 × G2 per time-frame.*" The difference can be summarized as:

- Method in [1]: One EEG object one image (multi-channels)

- The proposed method: one EEG object T images (T indicates time patches)

Based on the short describe, I guess the author may divide the EEG signal into many fragments in time axis, and then do FFT at each fragment to get the 'time-frame'. As Shown in following.

Python | 复制代码

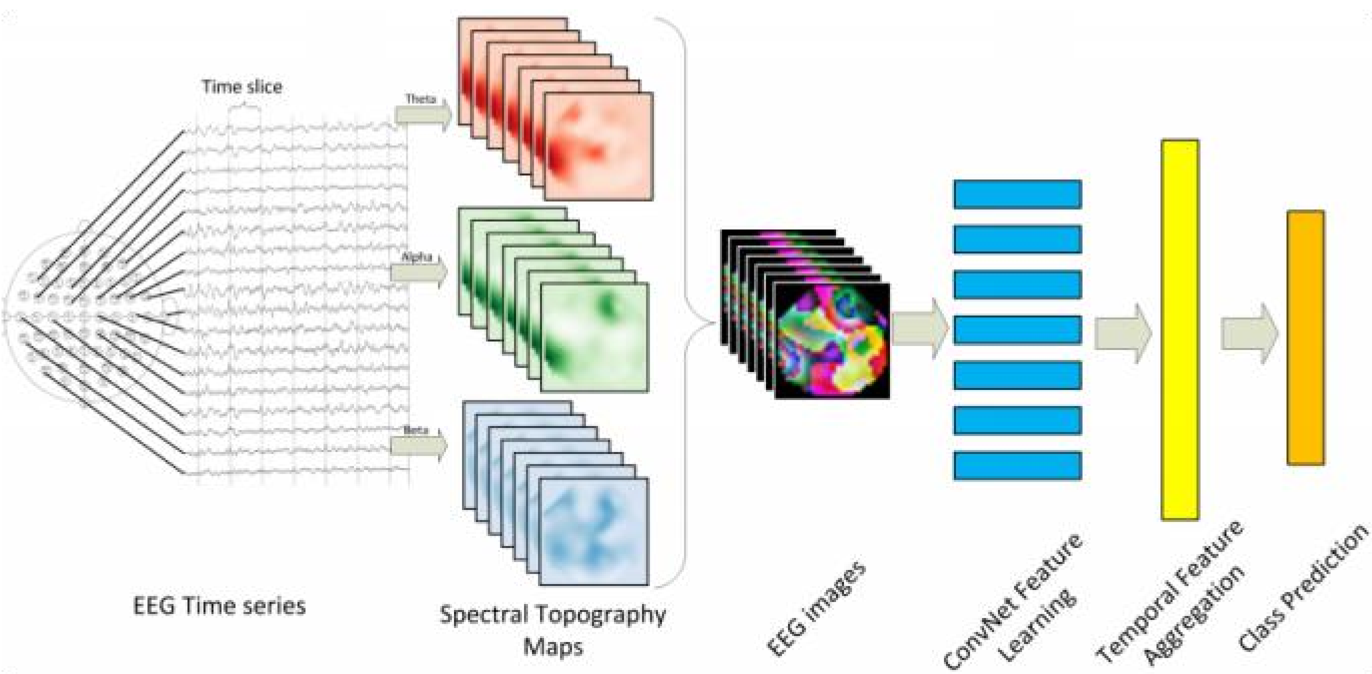
```python
time_win = 128 # length of each fragment
sample_rate = 1000
eeg_ = einops.rearrange(eeg, 'n (f tw) c -> n f c tw ', n=num_samples, tw=time_win, c=channels)
print(eeg_.shape)
power = torch.abs(torch.fft.fft(eeg_, n=time_win, dim=-1, norm= 'forward'))
freqs = torch.fft.fftfreq(n=time_win, d=1/sample_rate)
theta_pass = torch.where((4 < freqs) & (freqs <= 7), True, False)
alpha_pass = torch.where((8 < freqs) & (freqs <= 13), True, False)
beta_pass = torch.where((13 < freqs) & (freqs <= 30), True, False)

theta = power[:, :, :, theta_pass]
alpha = power[:, :, :, alpha_pass]
beta = power[:, :, :, beta_pass]

theta = torch.norm(theta, p=2, dim=-1, keepdim=False)
alpha = torch.norm(alpha, p=2, dim=-1, keepdim=False)
beta = torch.norm(beta, p=2, dim=-1, keepdim=False)

features = torch.cat( [theta, alpha, beta], dim=-1)
```

The way of citation [1], for comparison.



Results

This project has a huge workload due to 1)The unique structure of the propoesd method especially the combination of the self-attention and convolution make it impossible to use the existing framework. 2) there are no shared codes from no mater origin paper or online-communities.

I shared my codes in GitHub, they are availiable now in my GitHub repository https://github.com/MeetXinZhang/EEG-ConvTransformer . As far as I know, this is the first implementation among online-communities.

I got the same results as the original paper.

```
loss=1.32556 acc=0.578
loss=1.25107 acc=0.531
loss=1.26205 acc=0.469
loss=1.25333 acc=0.562
loss=1.26056 acc=0.531
loss=1.26126 acc=0.469
loss=1.24627 acc=0.484
loss=1.31128 acc=0.469
loss=1.43702 acc=0.406
loss=1.23372 acc=0.484
loss=1.23358 acc=0.609
```

Table 2
Comparison of the proposed method with existi

Method	Accuracy (%)
	6-Category
LDA [4]	40.68 ± 5.54
ICA-ERP [18]	43.50
Shallow [11]	49.04 ± 6.99
LSTM [11,23]	44.77 ± 6.30
LSTM + CNN [11]	46.18 ± 6.79
CNN [11,23]	50.00 ± 6.61
Attention CNN [11]	50.37 ± 6.56
CNN-ResNet101 [23]	–
1-D Wide-Res CNN [21]	51.29 ± 7.57
CT-Slim	51.96 ± 8.63
CT-Fit	52.17 ± 8.15
CT-Wide	$\mathbf{52.33 \pm 8.28}$