MACQUARIE University
SYDNEY·AUSTRALIA

Riemannian SPD Networks for BCI

A NOVEL NETWORK STRUCTURE WHICH IS MORE SUITABLE

DATE: 6 June 2024

A Riemannian Network for SPD matrix Learning

THE BASIC OF RIEMANNIAN NETWORK

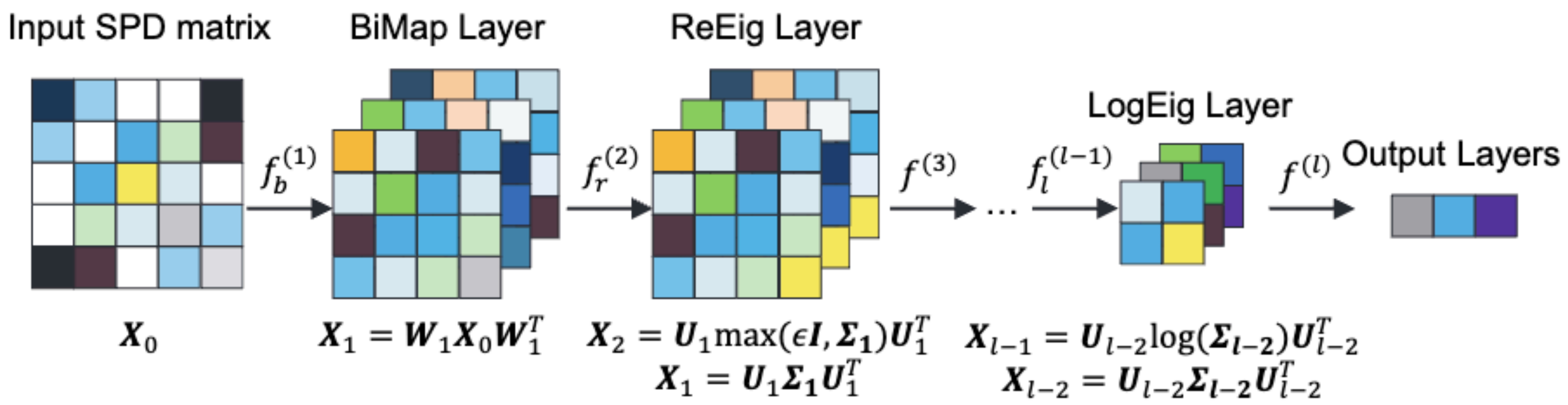
Riemannian Network for SPD Matrix Learning

1ST PUBLICATION, FROM ETH ZURICH, SWITZERLAND

1. Structure for SPDNet
2. Experiments

Structure of SPDNet

SIMILAR TO CONVENTIONAL NETWORK




```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.trans1 = SPDTransform(400, 200)
        self.trans2 = SPDTransform(200, 100)
        self.trans3 = SPDTransform(100, 50)
        self.rect1  = SPDRectified()
        self.rect2  = SPDRectified()
        self.rect3  = SPDRectified()
        self.tangent = SPDTangentSpace(50)
        self.linear = nn.Linear(1275, 7, bias=True)
        # self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.trans1(x)
        x = self.rect1(x)
        x = self.trans2(x)
        x = self.rect2(x)
        x = self.trans3(x)
        x = self.rect3(x)
        x = self.tangent(x)
        # x = self.dropout(x)
        x = self.linear(x)
        return x
```

Code source: https://github.com/adavoudi/spdnet/blob/master/examples/demo.py

Structure of SPDNet

SIMILAR TO CONVENTIONAL NETWORK

- BiMap Layer (similar to FC layer for conventional neural network)
  - $\boldsymbol{X}_k = f_b^{(k)}\left(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k\right) = \boldsymbol{W}_k \boldsymbol{X}_{k-1} \boldsymbol{W}_k^T$
  - $\boldsymbol{W}_k$ is the transformation matrix (similar to weight) and $\boldsymbol{W}_k^T$ is its transpose

- Here is the FC layer from conventional neural network for comparison
  - $\boldsymbol{X}_k = f^{(k)}\left(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k, \boldsymbol{b}_k\right) = \boldsymbol{W}_k \boldsymbol{X}_{k-1} + \boldsymbol{b}_k$

Code source: https://github.com/adavoudi/spdnet/blob/master/examples/demo.py

Structure of SPDNet

SIMILAR TO CONVENTIONAL NETWORK

- Eigenvalue decomposition (EIG): $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1}\boldsymbol{\Sigma}_{k-1}\boldsymbol{U}_{k-1}^T$, where Σ_{k-1} is a diagonal matrix

- ReEig Layer (similar to ReLU layer for conventional neural network)
  - $\boldsymbol{X}_k = f_r^{(k)}\left(\boldsymbol{X}_{k-1}\right) = \boldsymbol{U}_{k-1} \max\left(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_{k-1}\right) \boldsymbol{U}_{k-1}^T$
  - Where $\max\left(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_{k-1}\right)$ is a diagonal matrix $\boldsymbol{A}$, $\boldsymbol{A}(i,i) = \begin{cases} \boldsymbol{\Sigma}_{k-1}(i,i), & \boldsymbol{\Sigma}_{k-1}(i,i) > \epsilon \\ \epsilon, & \boldsymbol{\Sigma}_{k-1}(i,i) \le \epsilon \end{cases}$

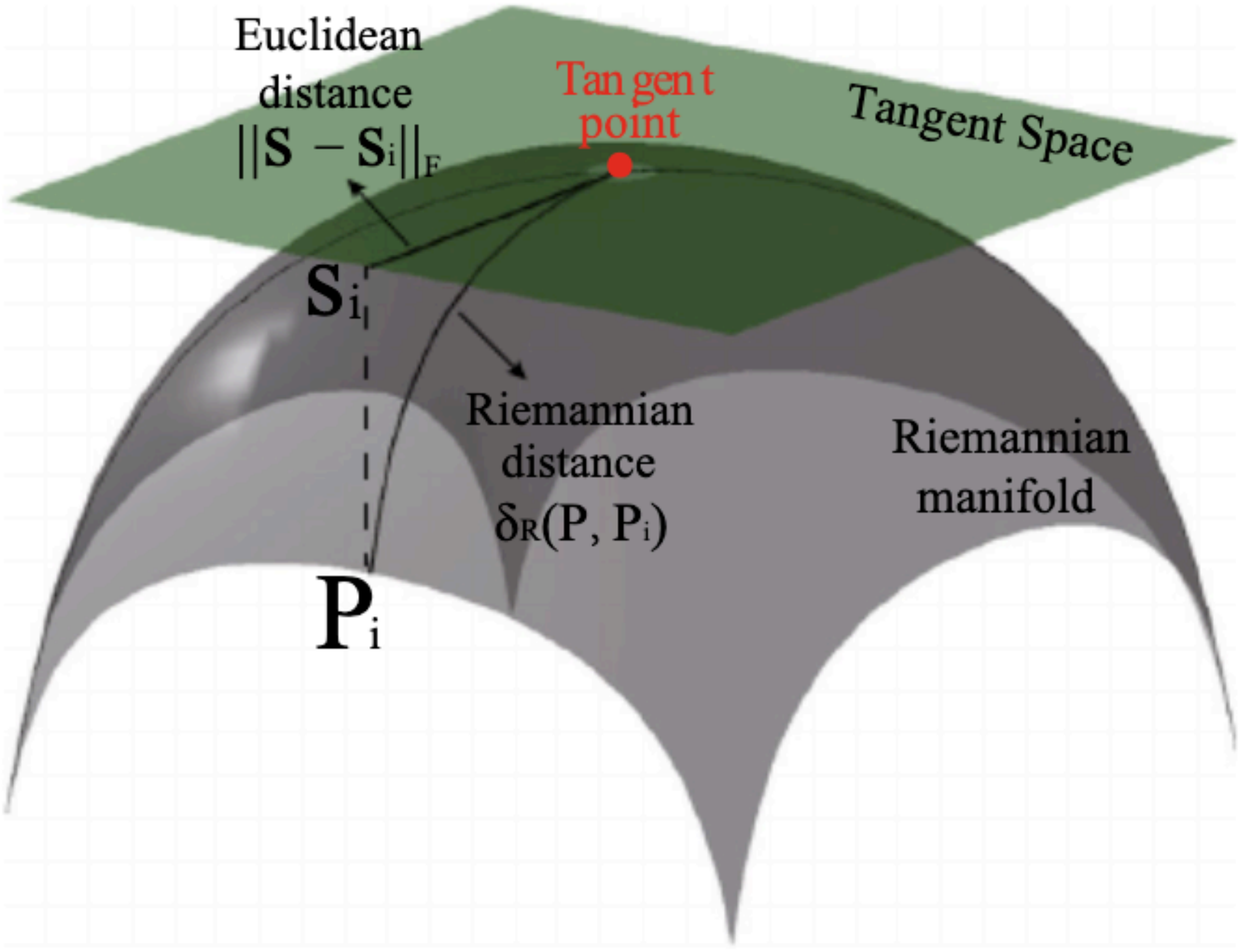
- Here is the ReLU layer from conventional neural network for comparison
  - $f(x) = x^+ = \max(0,x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

Code source: https://github.com/adavoudi/spdnet/blob/master/examples/demo.py

Structure of SPDNet

SIMILAR TO CONVENTIONAL NETWORK

- Eigenvalue decomposition (EIG): $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1}\boldsymbol{\Sigma}_{k-1}\boldsymbol{U}_{k-1}^T$
- LogEig Layer
  - $\boldsymbol{X}_k = f_l^{(k)}\left(\boldsymbol{X}_{k-1}\right) = \log\left(\boldsymbol{X}_{k-1}\right) = \boldsymbol{U}_{k-1}\log\left(\boldsymbol{\Sigma}_{k-1}\right)\boldsymbol{U}_{k-1}^T$
  - Where $\log\left(\boldsymbol{\Sigma}_{k-1}\right)$ is the diagonal matrix of eigenvalue logarithms
- This layer projects the SPD matrices on a tangent space of Riemannian manifold
- Within this flat space, classical Euclidean computations can be used



Code source: https://github.com/adavoudi/spdnet/blob/master/examples/demo.py

Experiment

BETTER PERFORMANCE

Dataset: Acted Facial Expression in Wild (AFEW)

Method	AFEW	HDM05	PaSC1	PaSC2
STM-ExpLet	31.73%	–	–	–
RSR-SPDML	30.12%	48.01%±3.38	–	–
DeepO2P	28.54%	–	68.76%	60.14%
HERML-DeLF	–	–	58.0%	59.0%
VGGDeepFace	–	–	78.82%	68.24%
CDL	31.81%	41.74%±1.92	78.29%	70.41%
LEML	25.13%	46.87%±2.19	66.53%	58.34%
SPDML-AIM	26.72%	47.25%±2.78	65.47%	59.03%
SPDML-Stein	24.55%	46.21%±2.65	61.63%	56.67%
RSR	27.49%	41.12%±2.53	–	–
SPDNet-0BiRe	26.32%	48.12%±3.15	68.52%	63.92%
SPDNet-1BiRe	29.12%	55.26%±2.37	71.75%	65.81%
SPDNet-2BiRe	31.54%	59.13%±1.78	76.23%	69.64%
SPDNet-3BiRe	**34.23%**	**61.45%±1.12**	**80.12%**	**72.83%**

Table 1: The results for the AFEW, HDM05 and PaSC datasets. PaSC1/PaSC2 are the control/handheld testings.

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.trans1 = SPDTransform(400, 200)
        self.trans2 = SPDTransform(200, 100)
        self.trans3 = SPDTransform(100, 50)
        self.rect1  = SPDRectified()
        self.rect2  = SPDRectified()
        self.rect3  = SPDRectified()
        self.tangent = SPDTangentSpace(50)
        self.linear = nn.Linear(1275, 7, bias=True)
        # self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.trans1(x)
        x = self.rect1(x)
        x = self.trans2(x)
        x = self.rect2(x)
        x = self.trans3(x)
        x = self.rect3(x)
        x = self.tangent(x)
        # x = self.dropout(x)
        x = self.linear(x)
        return x
```

Tensor-CSPNet: A Novel Geometric Deep Learning Framework for Motor Imagery Classification

AN IMPROVED RIEMANNIAN NETWORK FOR EEG

Tensor-CSPNet

2ND PUBLICATION, FROM NTU, SINGAPORE

1. Why adopt SPDNet
2. Structure for Tensor-CSPNet
3. Experiments

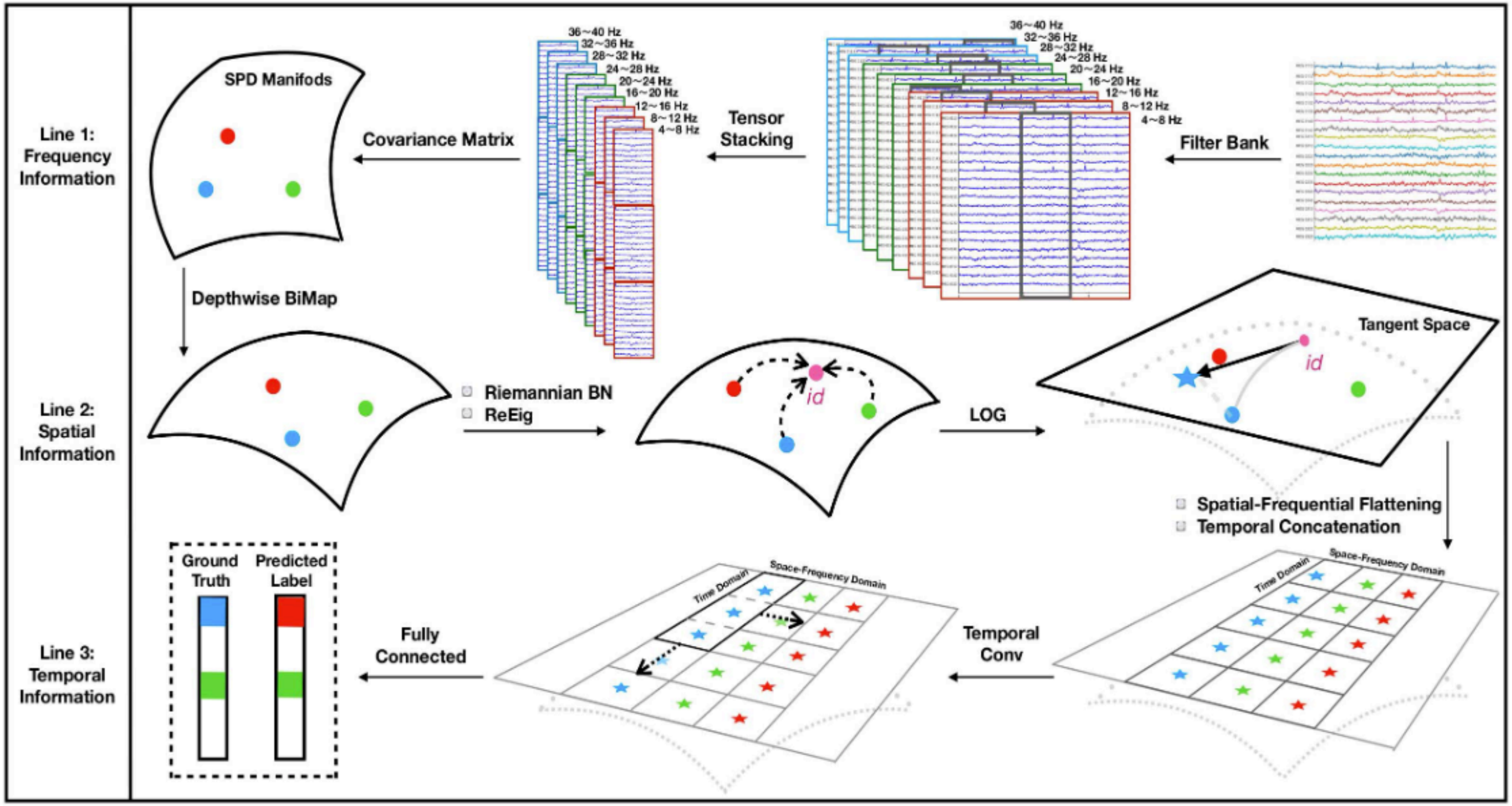
Why Adopt SPDNet (Riemannian manifold)

IT’S MORE SUITABLE

- It could preserve the natural representation of EEG data
  - EEG data contains high complexity comparing with conventional data (image, audio, etc.)
    - Multi channel, Multi Frequency band
    - High sensitivity
  - Use Covariance Matrices to represent EEG signals
    - Can capture spatial relationships
    - Covariance Matrices are SPD by nature
    - SPD matrices are on Riemannian manifold by nature
  - Riemannian manifold
    - Its projected tangent space is euclidean space

Structure of Tensor-CSPNet

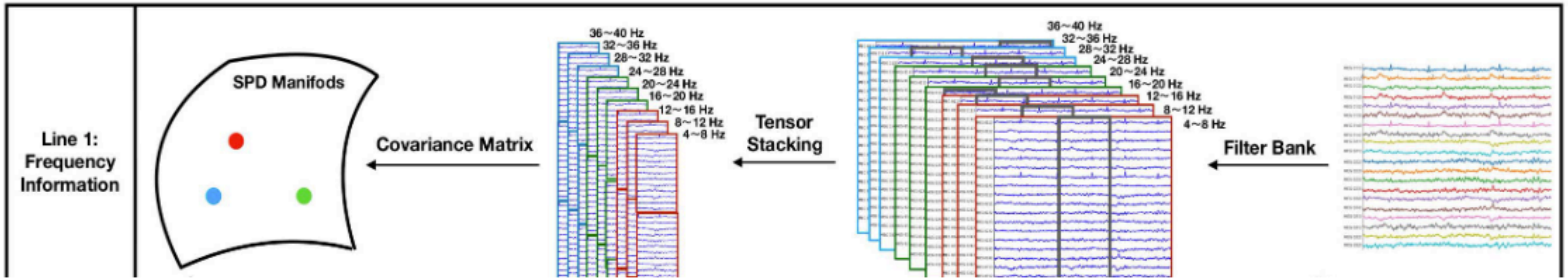
ADOPTED FROM SPDNET



Structure of Tensor-CSPNet

ADOPTED FROM SPDNET

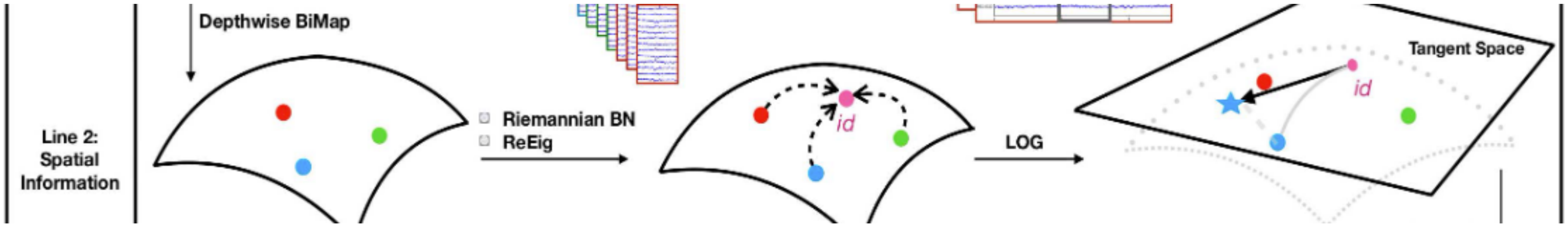
- Stage 1: Tensor Stacking
  - Frequency segmentation
  - Temporal segmentation
  - Tensor staking



Structure of Tensor-CSPNet

ADOPTED FROM SPDNET

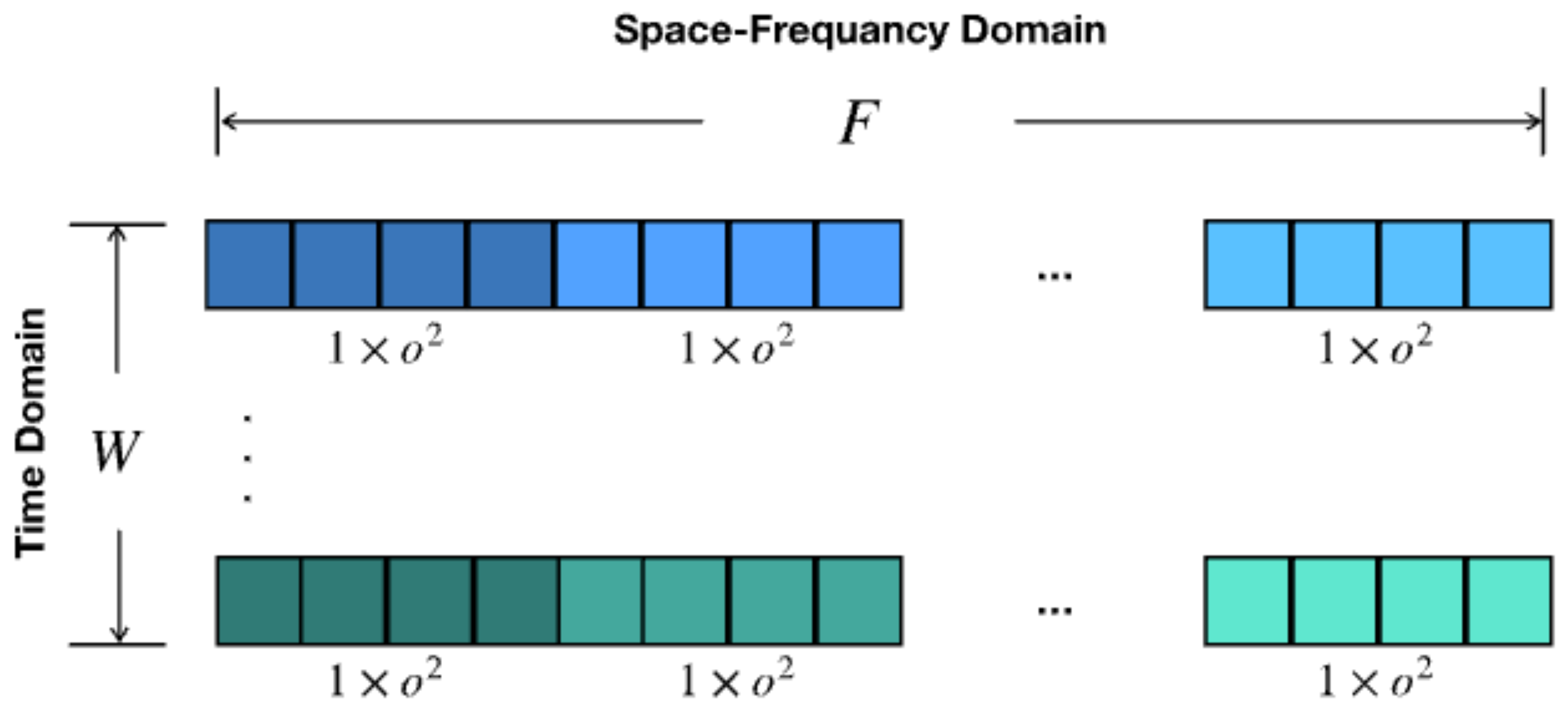
- Stage 2: Extract Common Spatial Pattern (This is where the SPDNet been adopted)
  - Depthwise BiMap
  - Riemannian BN
  - ReEig
  - LogEig

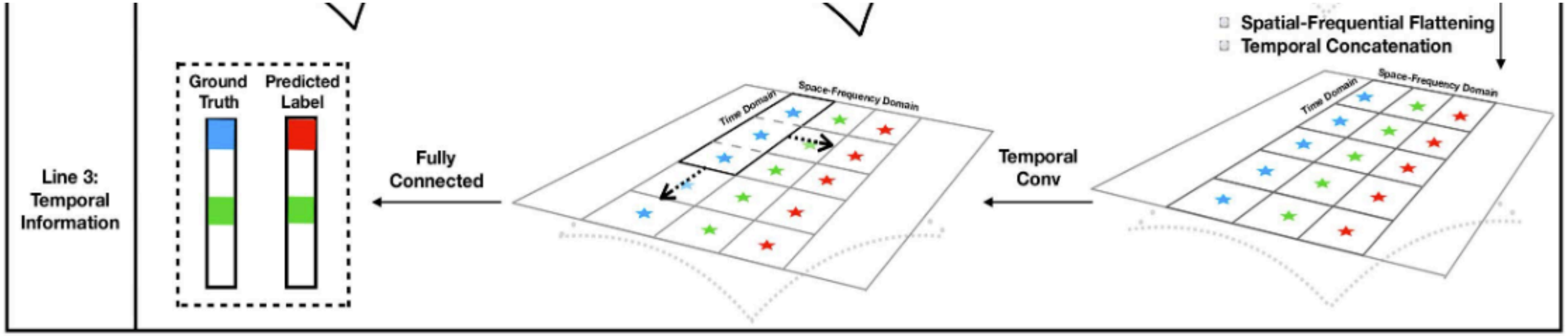


Structure of Tensor-CSPNet

ADOPTED FROM SPDNET

- Stage 3: Temporal Convolutional Stage
  - Core is po^2-width ($p = 1$ or F) and q-hight ($1 \leq q \leq W$)
- Stage 4: FC for final classification





Experiment

BETTER PERFORMANCE

Dataset: Korea University Dataset (MI-KU) and BCI Competition IV 2a (BCIC-IV-2a)

	MI-KU (20 channels, 2 classes)			BCIC-IV-2a (22 channels, 4 classes)		
	CV (S1) %	CV (S2) %	Holdout (S1$\rightarrow$ S2) %	CV (T) %	CV (E) %	Holdout (T $\rightarrow$ E) %
FBCSP	64.41 (16.28)	66.47 (16.53)	59.67 (14.32)	73.57 (15.13)	72.46 (16.02)	65.79 (14.21)
MDM	50.47 (8.63)	51.93 (9.79)	52.33 (6.74)	62.96 (14.01)	59.49 (16.63)	50.74 (13.80)
TSM	54.59 (8.94)	54.97 (9.93)	51.65 (6.11)	68.71 (14.32)	63.32 (12.68)	49.72 (12.39)
SPDNet	57.88 (8.68)	58.88 (8.68)	60.41 (12.13)	65.91 (10.31)	61.16 (10.50)	55.67 (9.54)
EEGNet	63.35 (13.20)	64.86 (13.05)	63.28 (11.56)	69.26 (11.59)	66.93 (11.31)	60.31 (10.52)
ConvNet	64.21 (12.61)	62.84 (11.74)	61.47 (11.22)	70.42 (10.43)	65.89 (12.13)	57.61 (11.09)
FBCNet	74.16 (12.60)	73.81(13.99)	67.83 (14.34)	**77.26** (14.82)	**76.58** (13.09)	72.71 (14.67)
Tensor-CSPNet	**74.95** (15.27)	**75.92** (14.63)	**69.65** (14.97)	75.98 (14.26)	74.92 (14.63)	**72.96** (14.98)

Thank you for listening