Learning of spatial-temporal EEG representation in the Stiefel manifold using Hilbert transformation

December 30, 2024

Abstract

This method maps EEG signal data to the Stiefel manifold using QR decomposition, without relying on the covariance matrix, and utilizes the cross-entropy loss function for classification. Cross-entropy is commonly used in classification problems, and it measures the difference between the true class labels and the predicted probabilities, optimizing the classification accuracy.

1 Introduction

2 The Stiefel Manifold as a Riemannian Manifold

The Stiefel manifold, denoted as $V_{k,n}$, is the set of all orthonormal k-frames in $\mathbb{R}^n$, where $k \leq n$, that is denoted by

$$V_{k,n} = \{Y \in \mathbb{R}^{n\times k} \mid Y^\top Y = I_k\},$$

The condition $Y^\top Y = I_k$ ensures that the columns of Y are orthonormal. It is a smooth manifold with a natural Riemannian structure. The tangent space of the Stiefel manifold at a point at Y, denoted by $T_Y V_{k,n}$, is given by

$$T_Y V_{k,n} = \{Z \in \mathbb{R}^{n\times k} \mid Y^\top Z + Z^\top Y = 0\}.$$

This condition ensures that the variations of Y preserve the orthonormality constraint.

The canonical metric on $T_Y V_{k,n}$ is defined by the restriction of the Frobenius inner product from $\mathbb{R}^{n\times k}$ as follows

$$\langle Z_1, Z_2\rangle_Y = \mathrm{tr}(Z_1^\top Z_2), \tag{1}$$

where $Z_1, Z_2 \in T_Y V_{k,n}$. The Frobenius inner product (1) measures the alignment between the entries of Z_1 and Z_2 within the tangent space. This restriction ensures that only variations that preserve

*Corresponding Author
†Corresponding Author

the orthonormality of Y are considered. The canonical metric reflects the intrinsic geometry of $V_{k,n}$, defining distances and angles in terms of these orthogonality-preserving variations.

The canonical metric enables the computation of inner products, geodesic paths, and curvature measures on $V_{k,n}$, facilitating its use in optimization problems and manifold learning tasks. Its compatibility with the Euclidean structure simplifies numerical computations and geometric reasoning.

The canonical metric is foundational in manifold-based optimization techniques, including principal component analysis on manifolds, learning representations on the Stiefel manifold for signal and image processing, and trajectory generation and control in robotics.

The geodesics on $V_{k,n}$ are the curves $Y(t)$ that satisfy the geodesic equation derived from the Levi-Civita connection. For the canonical metric, geodesics can be expressed in terms of matrix exponentials. For an initial point $Y(0) = Y$ and an initial tangent vector $Z \in T_Y V_{k,n}$, the geodesic is given by

$$Y(t) = \exp(tA)Y,$$

where $A = ZY^\top - YZ^\top$ is a skew-symmetric matrix in $\mathbb{R}^{n\times n}$, and $\exp(tA)$ is the exponential matrix of tA. Alternatively, the geodesic can be represented in terms of the polar decomposition or QR decomposition.

2.1 Parallel transport

The Levi-Civita connection on the Stiefel manifold $V_{k,n}$ guarantees that the connection is torsion-free and compatible with the metric structure of the manifold. This connection can be derived through the use of the projection operator Π_Y, which projects any matrix $H \in \mathbb{R}^{n\times k}$ onto the tangent space $T_Y V_{k,n}$ at a point Y,

$$\Pi_Y(H) = H - Y \ \mathrm{sym}(Y^\top H),$$

where $\mathrm{sym}(A) = \frac{1}{2}(A + A^\top)$ denotes the symmetric part of a matrix A. This projection ensures that any component of H violating the orthonormality conditions of the Stiefel manifold is removed. As a result, Π_Y guarantees that vectors remain confined to the tangent space after transformations, which is essential for the proper definition of derivatives and vector transport on the manifold.

Parallel transport, which allows the movement of a tangent vector along a curve $Y(t)$ while preserving its position in the tangent space and maintaining the inner product structure, is another important concept. The evolution of the transported vector $P(t)$ along the curve $Y(t)$ is governed by the equation:

$$\frac{d}{dt}P(t) + P(t) \ \mathrm{sym}(Y(t)^\top \dot{Y}(t)) = 0, \tag{2}$$

where $\dot{Y}(t) = \frac{d}{dt}Y(t)$ represents the derivative of the curve. The term $\mathrm{sym}(Y(t)^\top \dot{Y}(t))$ ensures that the parallel transport is consistent with the geometry of the Stiefel manifold.

The solution to this equation, $P(t)$, represents the parallel transport of the vector $P(t)$ along the curve. This process is crucial for various manifold-based computations, such as constructing geodesics, computing gradients, and performing optimization, all while preserving the manifold's intrinsic geometric properties.

2.2 Exponential and Logarithm Maps and Retraction

The exponential map $\exp_Y(Z)$ on the Stiefel manifold maps a tangent vector $Z \in T_Y V_{k,n}$ to a point on the manifold, representing the endpoint of the geodesic originating at Y in the direction of Z. It is given by

$$\exp_Y(Z) = \mathrm{QR}(Y + Z),$$

where $\mathrm{QR}(Y+Z)$ ensures that the result lies on the manifold by orthonormalizing $Y+Z$. Here, $Y+Z$ represents the raw displacement in the ambient space and the QR decomposition ensures the result maintains the orthonormality constraint of the Stiefel manifold.

This operation is used in optimization algorithms and geometric flows, where accurate geodesic computations are needed. The exponential map provides a natural way to move from the tangent space back to the manifold, enabling manifold-aware optimization, geodesic interpolation, and understanding curvature.

Retraction is a computationally simpler alternative to the exponential map, approximating the geodesic trajectory for small tangent vectors. It is defined as:

$$R_Y(Z) = \mathrm{QR}(Y+Z),$$

where $Z \in T_Y V_{k,n}$. The retraction uses the same QR decomposition as the exponential map but does not capture the full geodesic structure.

Retraction avoids the computational complexity of exact geodesic calculations, making it suitable for iterative optimization methods like gradient descent on manifolds. For small tangent vectors Z, the retraction closely approximates the exponential map. In applications such as Riemannian optimization, retractions allow iterative algorithms to update points on the manifold efficiently, balancing computational cost and accuracy.

The logarithmic map, denoted as $\log_Y(Y_2)$, maps a point $Y_2 \in V_{k,n}$ back to the tangent space $T_Y V_{k,n}$ at a base point Y. It provides the tangent vector Z that represents the geodesic connecting Y to Y_2. For points $Y_1, Y_2 \in V_{k,n}$, the logarithmic map can be approximated using the matrix logarithm and QR decomposition:

$$Z = \log_Y(Y_2) = \mathrm{QR}(Y_2 - Y_1 M) + Y_1 \log(M),$$

where $M = Y_1^\top Y_2$ represents the relative orientation, $\log(M)$ is the matrix logarithm, capturing rotational differences, and QR decomposition ensures Z respects the orthonormality constraints.

The logarithmic map is particularly useful for computing tangent directions and understanding the local structure of the manifold.

The geodesic distance $d(Y_1, Y_2)$ measures the shortest path between two points $Y_1, Y_2 \in V_{k,n}$ along the manifold. This is a natural distance metric that respects the manifold's curved geometry.

The geodesic distance is defined as

$$d(Y_1, Y_2) = \| \log(Y_1^\top Y_2) \|_F, \tag{3}$$

where $\| \cdot \|_F$ is the Frobenius norm, using (1)and $\log(Y_1^\top Y_2)$ is the matrix logarithm of the relative transformation $M = Y_1^\top Y_2$.

This distance accounts for both the rotational alignment and the orthogonal displacement between the two points, ensuring that the computed distance reflects the geometry of $V_{k,n}$.

The exponential map $\exp_Y(Z)$ maps a tangent vector $Z \in T_Y V_{k,n}$ to a point on the manifold. It defines the endpoint of the geodesic originating at Y in the direction Z. The logarithmic map $\log_Y(Y_2)$ performs the inverse operation, retrieving the tangent vector Z such that $\exp_Y(Z) = Y_2$.

2.3 Gradient Computation of a Function on the Stiefel Manifold

The function $F(Y)$ is defined as a scalar-valued function, $F : \mathbb{V}_k(\mathbb{R}^n) \to \mathbb{R}$, that operates on the matrix Y where its columns are orthonormal vectors. The goal is to compute the gradient of this function with respect to Y, while maintaining the constraint that the columns of Y remain orthonormal.

To compute the gradient of $F(Y)$ on the Stiefel manifold, we must use the Riemannian gradient, which takes into account the geometry of the manifold. The Riemannian gradient $\nabla_Y F(Y)$ is the

projection of the Euclidean gradient $\nabla_Y F(Y)_{\text{Euclid}}$ onto the tangent space of the Stiefel manifold at Y.

The Euclidean gradient $\nabla_Y F(Y)_{\text{Euclid}}$ is simply the standard gradient of $F(Y)$ with respect to Y, computed without considering the manifold constraint. This can typically be derived by taking the derivative of the function $F(Y)$ with respect to the entries of Y. Given the Euclidean gradient $\nabla_Y F(Y)_{\text{Euclid}}$, the Riemannian gradient is obtained by projecting this Euclidean gradient onto the tangent space of the Stiefel manifold. The projection operator is given by

$$\nabla_Y F(Y) = (I_n - YY^\top)\nabla_Y F(Y)_{\text{Euclid}}, \tag{4}$$

where I_n is the identity matrix in $\mathbb{R}^n$, and $YY^\top$ is the orthogonal projection onto the column space of Y.

2.4 Gradient Descent on the Stiefel Manifold

For optimization, the Riemannian gradient descent method is used to minimize $F(Y)$ over the Stiefel manifold. The update rule for gradient descent is:

$$Y^{\text{new}} = \text{prox}_{\mathbb{V}_k(\mathbb{R}^n)}\left(Y - \eta\nabla_Y F(Y)\right), \tag{5}$$

where η is the learning rate, and the operator $\text{prox}_{\mathbb{V}_k(\mathbb{R}^n)}$ projects the updated matrix $Y - \eta\nabla_Y F(Y)$ back onto the Stiefel manifold, ensuring that the columns of Y remain orthonormal.

In EEG signal processing and similar applications, the function $F(Y)$ may represent a cost or objective function involving spatial filters, covariance matrices, or other matrix quantities that need to remain orthonormal. The gradient computation on the Stiefel manifold ensures that these quantities are optimized while preserving their manifold constraints.

3 Hilbert Transform (HT) and Hilbert-Huang Transform (HHT) on the Stiefel Manifold

The advantages of applying the Hilbert-Huang Transform (HHT) on the Stiefel manifold are multi-faceted. First, the approach preserves the geometry of EEG data, ensuring that the orthogonality and spatial structure across channels are maintained due to the manifold constraints. The use of Empirical Mode Decomposition (EMD) enables adaptive decomposition, which allows for a detailed analysis of EEG signals by adapting to local time scales. Additionally, the HHT provides accurate instantaneous features such as amplitude, phase, and frequency, offering fine-grained temporal and spectral insights into the EEG signals. By incorporating Riemannian geometry, the method respects the non-linear nature of EEG data, leading to improved representation and classification accuracy. Furthermore, the combination of EMD and the Hilbert Transform enhances noise handling, reducing signal distortion and improving interpretability while adhering to the constraints of the Stiefel manifold.

3.1 Hilbert Transform (HT)

The Hilbert Transform plays a significant role in the analysis of signals residing on the Stiefel manifold, a manifold that enforces orthogonality constraints on the signal. Given a signal $i(t)$, which is mapped to the Stiefel manifold $V_{k,n}$, the Hilbert Transform is defined as:

$$i_h(t) = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{i(\tau)}{t-\tau}\,d\tau,$$

where $i(t)$ represents the signal, and the operation preserves the orthogonality constraints inherent to the Stiefel manifold. In this context, t denotes the time variable, and τ is the integration variable. The Hilbert Transform, in essence, provides a time-domain representation of the analytic signal by utilizing the convolution integral, which maps a given signal to its phase-shifted counterpart, while maintaining the structural integrity of the manifold.

In the Stiefel manifold, the signal $i(t)$ is represented as a matrix $I(t) \in \mathbb{R}^{n \times k}$, where the columns of $I(t)$ are orthogonal to each other. The orthogonality condition is expressed as:

$$I(t)^\top I(t) = \mathbf{I}_k,$$

where $\mathbf{I}_k$ is the identity matrix of size $k \times k$, ensuring that the signal lies within the constraint space of the Stiefel manifold. The Hilbert Transform is extended to operate on each orthogonal column of $I(t)$, preserving the manifold constraints and maintaining the orthogonality of the resulting transformed signal.

The analytic signal associated with $i(t)$ on the manifold is constructed by combining the original signal with its Hilbert Transform:

$$N(t) = I(t) + jI_h(t),$$

where j represents the imaginary unit, and $I_h(t)$ is the Hilbert Transform of $I(t)$. This formulation results in a complex-valued signal $N(t)$, which lies on the complexified Stiefel manifold. The real and imaginary parts of $N(t)$ provide instantaneous amplitude and phase information, which are crucial for further spectral analysis, such as instantaneous frequency estimation or time-frequency representation, on the Stiefel manifold. This approach facilitates detailed analysis while preserving the geometric structure inherent to the manifold, thus ensuring accurate and meaningful signal processing for applications such as EEG signal analysis.

3.2 Hilbert-Huang Transform (HHT)

The Hilbert-Huang Transform (HHT) is a powerful method for analyzing nonlinear and non-stationary signals. It is composed of two main components: Empirical Mode Decomposition (EMD) and the Hilbert Transform. EMD decomposes a signal $I(t)$ into a set of Intrinsic Mode Functions (IMFs), denoted as $I_k(t)$, which capture the underlying oscillatory modes of the signal. The decomposition is expressed as:

$$I(t) = \sum_{k=1}^{K} I_k(t),$$

where $I_k(t)$ represents the k-th IMF and the sum over k yields the original signal. Each IMF $I_k(t)$ is designed to be an oscillatory function with well-defined local extrema, thus ensuring that it adheres to the conditions of orthogonality when mapped to the Stiefel manifold. This property allows for a decomposition that captures the intricate temporal and frequency characteristics of the signal, which is crucial for analyzing complex, time-varying phenomena such as EEG data.

Once the IMFs are obtained, the Hilbert Transform is applied to each IMF to generate the analytic signal. The Hilbert Transform of $I_k(t)$, denoted $I_{k,h}(t)$, provides a means to compute the instantaneous frequency and amplitude of the oscillatory modes. The analytic signal corresponding to each IMF is then:

$$N_k(t) = I_k(t) + jI_{k,h}(t),$$

where j is the imaginary unit and $I_{k,h}(t)$ is the Hilbert Transform of $I_k(t)$. The result, $N_k(t)$, is a complex-valued signal that encodes both the instantaneous amplitude and phase, facilitating detailed time-frequency analysis. This method is particularly beneficial for signals with non-stationary or

transient components, as it allows for precise tracking of time-varying frequencies without the need for predefined basis functions.

By utilizing HHT in conjunction with Riemannian geometry, particularly on the Stiefel manifold, we can ensure that the orthogonality of the IMFs is maintained during decomposition and transformation, thus enhancing the accuracy and interpretability of the results. The combined power of EMD and the Hilbert Transform offers a robust framework for exploring complex signals, making it an invaluable tool in fields like EEG signal processing, where dynamic, non-stationary behavior is common.

3.3 Instantaneous Amplitude, Phase, and Frequency

The Hilbert-Huang Transform (HHT) provides a powerful framework for analyzing time-varying signals, particularly in capturing instantaneous characteristics such as amplitude, phase, and frequency. These instantaneous properties allow for a detailed understanding of the signal's behavior at each moment in time, which is essential for analyzing complex, non-stationary signals like EEG data.

For each Intrinsic Mode Function (IMF) $I_k(t)$, we can compute the following instantaneous properties:

- **Instantaneous Amplitude (IA):** The instantaneous amplitude represents the time-varying envelope of the IMF, reflecting the magnitude of the oscillations. It is defined as:

$$A_k(t) = \sqrt{I_k^\top(t) I_k(t) + I_{k,h}^\top(t) I_{k,h}(t)}, \tag{6}$$

  where $I_k(t)$ is the IMF and $I_{k,h}(t)$ is its Hilbert Transform. This formula captures the instantaneous energy of the signal, combining both the IMF and its Hilbert Transform.

- **Instantaneous Phase (IP):** The instantaneous phase provides the phase angle of the oscillatory mode at each point in time, which is crucial for understanding the time-varying behavior of the signal. It is given by:

$$\Phi_k(t) = \arctan\left(\frac{I_{k,h}(t)}{I_k(t)}\right).$$

  The phase function describes how the oscillatory mode progresses through its cycle over time, revealing temporal shifts and periodic behavior.

- **Instantaneous Frequency (IF):** The instantaneous frequency quantifies the rate of change of the phase, providing insight into the frequency dynamics of the IMF at any given time. It is computed as:

$$\Omega_k(t) = \frac{d\Phi_k(t)}{dt}.$$

  The IF captures the local frequency variations of the signal, offering a fine-grained view of its frequency content over time.

Together, these instantaneous properties form a complete description of the oscillatory modes within the signal, enabling a more nuanced understanding of the signal's time-frequency characteristics.

3.4 Hilbert Spectrum

The Hilbert Spectrum offers a time-frequency-energy representation of the signal, effectively combining the instantaneous amplitude, phase, and frequency into a unified framework. It is particularly useful

for signals with non-stationary components, such as EEG data, where the frequency content changes dynamically over time. The Hilbert Spectrum is expressed as

$$H(t,\omega) = \sum_{k=1}^{K} A_k(t)\delta(\omega - \Omega_k(t)), \tag{7}$$

where δ is the Dirac delta function, representing a sharp frequency peak at each instantaneous frequency $\Omega_k(t)$. This spectrum provides a detailed view of the signal's energy distribution in both time and frequency, offering insights into its non-stationary behavior. By utilizing the Hilbert Spectrum on the Stiefel manifold, we can ensure that the orthogonality of the IMFs is preserved, facilitating a more accurate representation of the EEG signals in their intrinsic geometrical structure.

EEG Data Processing

Step 1: Preprocessing the recorded EEG data
Includes 64 channels and 300 trials with 795 samples per channel.
First, relevant filters such as *notch filtering* to remove frequency noise and *band-pass filtering* to eliminate extra noise are applied.

Step 2: Selecting the desired channels from the total number of channels, which is 64.

Step 3: Transforming the data to the surface of the Stiefel manifold using QR matrices.
Changes introduced in the geometric space of the channels generate new features for the spatial and temporal patterns of the recorded system. These changes provide a better description of the data in the *tangent space*.

Step 4: Utilizing channel similarity, EMD, and Hilbert transformation ratio.
The results are transformed to a new space. This transformation is used to train a *CNN classifier*, and the results are analyzed with high accuracy.

Step 5: The final comparison metric for data is accuracy.
A *CNN network* is designed to extract features for each spatial and temporal order and classification. *CNN training* is conducted, and *categorical cross-entropy measurement* is used for error analysis and modeling.

Algorithm 1 EEG Data Processing

0: **Input:** EEG data with 64 channels, 300 trials, and 795 samples per channel.
0: **Output:** Classification results with accuracy as the evaluation metric.
0: **procedure** EEGPROCESSING
0: **Step 1: Preprocessing the recorded EEG data**
Includes 64 channels and 300 trials with 795 samples per channel.
First, relevant filters such as *notch filtering* to remove frequency noise and *band-pass filtering* to eliminate extra noise are applied.

0: **Step 2: Selecting the desired channels from the total number of channels, which is 64.**

0: **Step 3: Transforming the data to the surface of the Stiefel manifold using QR matrices.**
Changes introduced in the geometric space of the channels generate new features for the spatial and temporal patterns of the recorded system. These changes provide a better description of the data in the *tangent space*.

0: **Step 4: Utilizing channel similarity, EMD, and Hilbert transformation ratio.**
The results are transformed to a new space. This transformation is used to train a *CNN classifier*, and the results are analyzed with high accuracy.

0: **Step 5: The final comparison metric for data is accuracy.**
A *CNN network* is designed to extract features for each spatial and temporal order and classification.
CNN training is conducted, and *categorical cross-entropy measurement* is used for error analysis and modeling.
0: **end procedure**=0

4 EEG Data Processing on the Stiefel Manifold

4.1 Step 1: Preprocessing the Recorded EEG Data

The EEG dataset used in this study consists of 64 channels, 300 trials per subject, and 795 samples per channel, yielding a large-scale multi-dimensional data structure. The preprocessing phase is critical for improving the quality of the EEG signal before further analysis. First, notch filtering is applied to eliminate power-line interference, typically at 50 Hz or 60 Hz, depending on the region of acquisition. The notch filter is designed to attenuate a specific frequency f_n and is defined by the following formula:

$$\text{Notch Filter}(f_n) = 1 - \frac{1}{1 + \left(\frac{f - f_n}{\Delta f}\right)^2}, \tag{8}$$

where Δf represents the width of the notch band. This filtering process effectively removes unwanted noise from the signal, preserving the physiological components of the EEG data. Following the notch filter, a band-pass filter is employed to remove high-frequency noise and low-frequency drift. A typical band-pass filter is designed with cutoffs at 1 Hz and 40 Hz, focusing on the frequencies that are most

relevant for EEG signal analysis. The band-pass filter is applied using the following transfer function:

$$H(f) = \frac{1}{1 + \left(\frac{f}{f_c}\right)^2}, \tag{9}$$

where f_c is the cutoff frequency of the filter. Together, these preprocessing steps—notch and band-pass filtering—help to refine the EEG signal by retaining only the relevant frequency components, thus improving the signal-to-noise ratio and enhancing the subsequent analysis of spatial-temporal EEG representations.

4.2 Step 2: Selecting Desired Channels

In EEG signal processing, selecting the relevant channels is a crucial step to improve classification performance and focus on the areas of the brain most relevant to the task at hand. One approach is to manually select channels based on prior domain knowledge, such as choosing frontal and occipital regions when analyzing motor imagery tasks. These regions are typically known to be involved in motor planning and execution processes, making them relevant for motor imagery-related EEG analysis. Alternatively, automated methods for channel selection can be employed, such as Principal Component Analysis (PCA) or feature selection based on correlation. PCA is commonly used to reduce dimensionality by extracting the principal components that capture the most variance in the data, thus identifying the most informative channels. The PCA process can be formulated as

$$X' = XW$$

where X is the matrix of EEG signals (with dimensions $N \times M$, where N is the number of samples and M is the number of channels), W is the matrix of eigenvectors corresponding to the main components, and X' is the transformed data in the reduced feature space. Another method involves feature selection based on the correlation between channels, where channels that are highly correlated with others may be removed to avoid redundancy, leaving only the most informative ones for further analysis. Once the channels are selected using either manual or automated methods, the data is reduced to those specific channels, retaining the most relevant features for the task, thus improving computational efficiency and the quality of subsequent analysis.

4.3 Step 3: Transforming Data to the Surface of the Stiefel Manifold Using QR Matrices

The Stiefel manifold is a fundamental concept in differential geometry and is defined as the set of all orthonormal frames in a given vector space. In the context of EEG signal processing, the Stiefel manifold corresponds to the 64 channels of EEG data, where each point on the manifold represents a set of orthonormal vectors. To map the EEG data to the Stiefel manifold, QR decomposition is performed on each trial of the data. For a given trial with an EEG data matrix X of size 64×795, the QR decomposition is applied as

$$X = QR$$

Here, X represents the EEG data matrix, Q is the orthonormal matrix (which represents the projection of the data onto the Stiefel manifold), and R is the upper triangular matrix. The matrix Q captures the structure of the data in terms of orthonormal frames, ensuring that the data lies on the Stiefel manifold. This transformation not only maps the data but also generates new features that are well-suited for further analysis.

After the data is transformed onto the Stiefel manifold, the next step is to compute the tangent space, which represents the local variations of the data around each point on the manifold. The tangent vectors are calculated using the following differential relation:

$$T_Q = \left\{ \dot{Q} \in \mathbb{R}^{64\times 795} \mid Q \cdot \dot{Q}^T + \dot{Q} \cdot Q^T = 0 \right\}$$

These tangent vectors, $\dot{Q}$, represent the directions of change or variation of the data on the manifold and are critical for understanding how the data behaves locally on the Stiefel manifold. By calculating these tangent vectors, we gain insights into the local geometry of the data, which is essential for tasks such as classification and clustering on Riemannian manifolds.

4.4 Step 4: Utilizing Channel Similarity, EMD, and Hilbert Transformation

In EEG signal analysis, measuring the similarity between channels is crucial for understanding spatial relationships within the brain's electrical activity. Channel similarity can be assessed using various distance metrics such as correlation coefficients, the Frobenius norm, or the geodesic distance on a manifold. Using the Frobenius inner product (1) on Stiefel manifold, the Frobenius norm, commonly used to compare matrices representing EEG signals from different channels, is defined as

$$\|A - B\|_F = \sqrt{\sum_{i,j} (A_{ij} - B_{ij})^2}, \tag{10}$$

which calculates the element-wise distance between matrices A and B. Alternatively, for data lying on a manifold such as the Stiefel manifold, the geodesic distance is more suitable. Using the geodesic distance formula (3), he geodesic distance between two matrices Q_1 and Q_2 on the Stiefel manifold is computed as

$$d_{\text{geo}}(Q_1, Q_2) = \| \log(Q_1^T Q_2) \|_F, \tag{11}$$

where $\log(Q_1^T Q_2)$ is the matrix logarithm that ensures the distance is calculated along the manifold, preserving the orthogonality constraints.

To further analyze the EEG signals, Empirical Mode Decomposition (EMD) is applied to decompose the signal into a set of Intrinsic Mode Functions (IMFs), each representing oscillatory modes at different frequencies. The EMD can be expressed as

$$x(t) = \sum_{i=1}^{n} \text{IMF}_i(t) + r_n(t), \tag{12}$$

where $\text{IMF}_i(t)$ are the intrinsic modes, and $r_n(t)$ is the residual signal. The IMFs capture the signal's local oscillatory components, which are essential for understanding the non-stationary nature of EEG data. This decomposition allows for the identification of oscillations at various frequency bands, facilitating a deeper understanding of the signal's temporal dynamics.

The Hilbert transform is then used to analyze the instantaneous frequency and phase of each IMF. The Hilbert transform is defined as

$$\hat{x}(t) = H(x(t)) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau,$$

where $H(x(t))$ represents the Hilbert transform of the signal $x(t)$, providing a complex-valued signal with instantaneous amplitude and phase. The combination of EMD and the Hilbert transform allows for the extraction of temporal features such as instantaneous frequency, phase, and amplitude, which are crucial for understanding the signal's time-varying behavior. These features, when mapped onto the Stiefel manifold, retain the geometrical structure of the EEG data, leading to more accurate and meaningful analyses.

4.5 Step 5: CNN Classifier Training

In this section, we describe the architecture of the Convolutional Neural Network (CNN) employed for the classification of spatio-temporal EEG signals mapped onto the Stiefel manifold. The network is designed to efficiently extract both spatial and temporal features from the EEG data, which has the shape $300 \times 64 \times 795$, where 300 represents the number of trials, 64 is the number of channels, and 795 corresponds to the time steps per trial. The architecture consists of multiple convolutional layers, each designed to apply several filters for feature extraction. The filters convolve across the input data, allowing the network to learn spatial patterns within the EEG signals. Specifically, at each convolutional layer, the output feature map $\mathbf{F}_{out}$ is given by the equation

$$\mathbf{F}_{out} = \sum_{i=1}^{K} \mathbf{W}_i * \mathbf{F}_{in} + b_i \tag{13}$$

where $\mathbf{W}_i$ is the i-th convolution filter, $*$ denotes the convolution operation, $\mathbf{F}_{in}$ is the input feature map, and b_i is the bias term. Pooling layers are applied after each convolutional layer to reduce the spatial dimensions of the feature maps, typically using max pooling. Max pooling selects the maximum value from a set of values within a window, helping to reduce computational complexity and maintain important features. The output of the pooling operation can be represented as

$$\mathbf{P}_{out} = \max(\mathbf{F}_{out})$$

Following the convolutional and pooling layers, fully connected layers are employed to aggregate the extracted spatial-temporal features. These layers map the high-dimensional feature maps to a lower-dimensional space, allowing the network to learn the complex relationships between the features. Finally, the output layer consists of a softmax function for classification, producing the probability distribution across C classes, where C is the total number of categories for classification. The softmax function is defined as

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$$

where z_i is the score for the i-th class, and C is the number of classes. The model is trained using categorical cross-entropy loss to minimize the classification error.

4.5.1 Loss Function

In the context of training a Convolutional Neural Network (CNN) for classification tasks, the categorical cross-entropy loss function is used to quantify the difference between the predicted class probabilities and the true labels. The loss function is computed by comparing the predicted probability $\hat{y}_i$ for each class i with the true label y_i, where y_i is typically represented as a one-hot vector. Specifically, for a given sample, the loss function is defined as

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{K} y_i \log(\hat{y}_i), \tag{14}$$

where K is the number of classes, $y_i = 1$ if the true class is i, and $y_i = 0$ for all other classes, and $\hat{y}_i$ is the predicted probability for the i-th class. This loss function penalizes the network when the predicted probability for the correct class (the class corresponding to $y_i = 1$) is far from 1, and conversely, when the predicted probabilities for incorrect classes are high. The goal during training is to minimize this loss by adjusting the weights of the CNN, effectively tuning the model such that the predicted probabilities $\hat{y}_i$ are as close as possible to the true labels y_i, with the correct class's predicted probability nearing 1 and the incorrect ones nearing 0. This optimization process typically employs gradient-based methods like backpropagation and stochastic gradient descent (SGD), ensuring that the model learns to predict more accurately over time.

4.5.2 Backpropagation

During the training of a Convolutional Neural Network (CNN), the gradient of the loss function with respect to the network weights is computed via backpropagation using the chain rule. First, the gradient of the loss with respect to the output predictions $\hat{y}_i$ is computed as follows

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

where $\mathcal{L}$ is the loss function, and y_i and $\hat{y}_i$ are the true and predicted values for the i-th output. This gradient is then propagated backward through the network layers to update the weights at each layer. The gradient with respect to a weight W_l in the l-th layer is given by

$$\frac{\partial \mathcal{L}}{\partial W_l} = \frac{\partial \mathcal{L}}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial W_l}$$

where the term $\frac{\partial \mathcal{L}}{\partial \hat{y}_i}$ represents the gradient from the subsequent layer, and $\frac{\partial \hat{y}_i}{\partial W_l}$ represents the derivative of the output with respect to the weight W_l in layer l. Once the gradients are computed, traditional gradient-based optimization methods, such as stochastic gradient descent (SGD), are used to update the weights.

However, for problems involving weight matrices constrained to a manifold, such as the Stiefel manifold (where weight matrices must remain orthonormal), standard gradient descent is not sufficient. Instead, Riemannian gradient descent is employed, which respects the geometry of the manifold. The weight matrix $W \in \mathbb{R}^{m \times m}$ is updated as follows

$$W \leftarrow W - \alpha \cdot \nabla_{\mathcal{S}} \mathcal{L},$$

where α is the learning rate controlling the size of the weight update, and $\nabla_{\mathcal{S}} \mathcal{L}$ is the Riemannian gradient of the loss function $\mathcal{L}$, which accounts for the orthogonality constraints imposed by the Stiefel manifold. The Riemannian gradient is computed as

$$\nabla_{\mathcal{S}} \mathcal{L} = (I - WW^T)\nabla \mathcal{L},$$

where $\nabla \mathcal{L}$ is the Euclidean gradient (calculated as in standard gradient descent), I is the identity matrix of appropriate dimension, and WW^T is the projection of the gradient onto the tangent space of the Stiefel manifold. This projection ensures that the weight update remains tangential to the manifold, thereby preserving the orthonormality of the weight matrix W. The term $(I - WW^T)$ effectively removes any component of the gradient that would push the weight matrix off the manifold, guaranteeing that after each update, the weight matrix W stays within the Stiefel manifold, maintaining its orthonormal property.

4.6 Final Evaluation

After training the Convolutional Neural Network (CNN) model, the final evaluation metric used to assess the model's performance is accuracy. Accuracy represents the percentage of correct predictions made by the model out of the total number of trials. It is computed by comparing the predicted labels $\hat{y}_i$ to the true labels y_i for each trial. The formula for accuracy is given by

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} 1(\hat{y}_i = y_i)$$

where N is the total number of trials or samples in the dataset, and $1(\hat{y}_i = y_i)$ is the indicator function that outputs 1 if the prediction $\hat{y}_i$ matches the true label y_i, and 0 otherwise. This sum accumulates the number of correct predictions across all trials, and dividing by N normalizes the result, giving the proportion of correct predictions. This metric provides a clear measure of the model's overall performance and helps in comparing different models or training configurations.

5 Implementation