Learning with Covariance Matrices: Foundations and Applications to Network Neuroscience

Saurabh Sihag[1], Gonzalo Mateos[2], Elvin Isufi[3], and Alejandro Ribeiro[4]

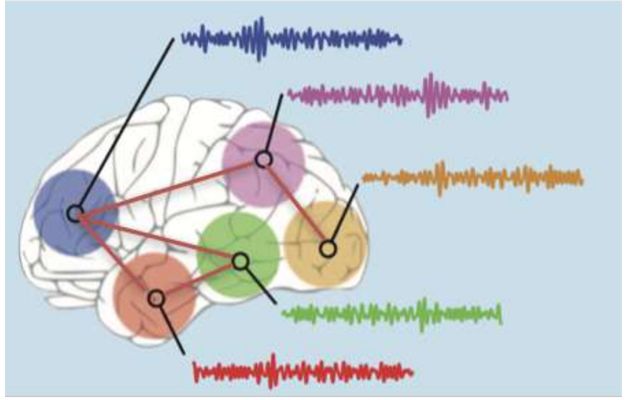
[1]University at Albany – ssihag@albany.edu
[2]University of Rochester – gmateosb@ece.rochester.edu
[3]Delft University of Technology – e.isufi-1@tudelft.nl
[4]University of Pennsylvania – aribeiro@seas.upenn.edu

European Signal Processing Conference (EUSIPCO), 2025

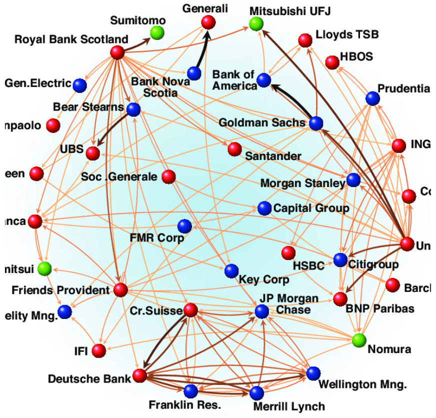
Covariance Matrix

➢ Covariance matrix captures the **redundancies** between data points (features)

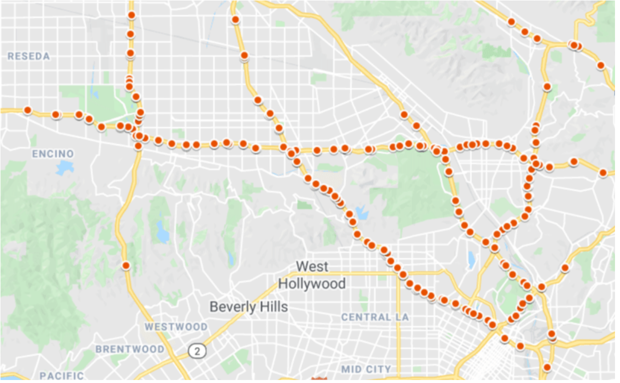
- Brain datasets: some areas of the brain activate together
- Financial datasets: stock prices fluctuate in tandem
- Traffic datasets: traffic volume is correlated across intersections



Brain

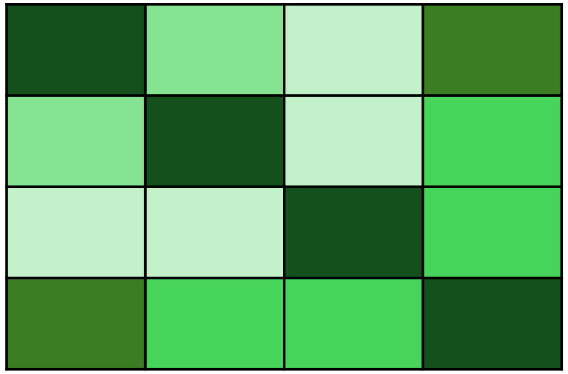


Finance



Traffic

Covariance Matrix



- Evaluating a covariance matrix
  - Consider a random variable $\mathbf{x} \in \mathbb{R}^m$
  - The covariance is

$$\mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T}], \text{ where } \boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$$

  - In practice, we have **sample** covariance matrix (an estimate)

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\mathsf{T}, \text{ where } \hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

n: number of samples (size of a dataset)

Covariance Matrix

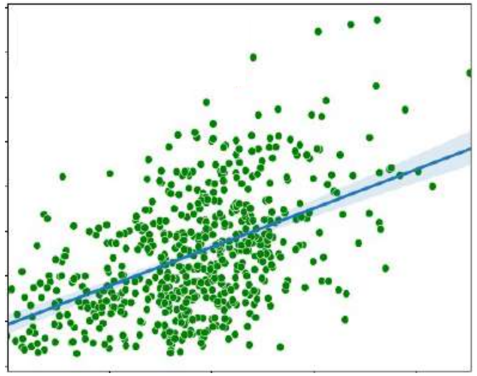
- Covariance matrix encodes **redundancies** between different features in data

Covariance matrix
(2-feature dataset)

$\sigma^2(r_1)$	$\sigma\,(r_1, r_2)$
$\sigma\,(r_1, r_2)$	$\sigma^2(r_2)$

Low redundancy
(smaller $\sigma\,(r_1, r_2)$)

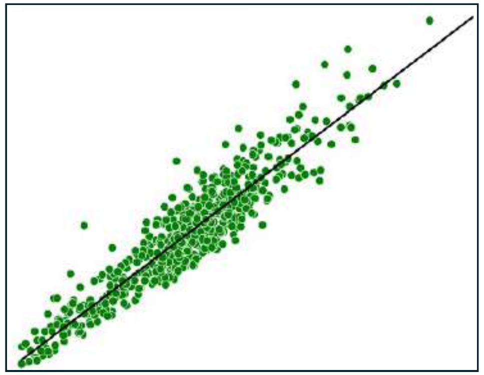
Feature r_1



Feature r_2

High redundancy
(higher $\sigma\,(r_1, r_2)$)

Feature r_1



Feature r_2

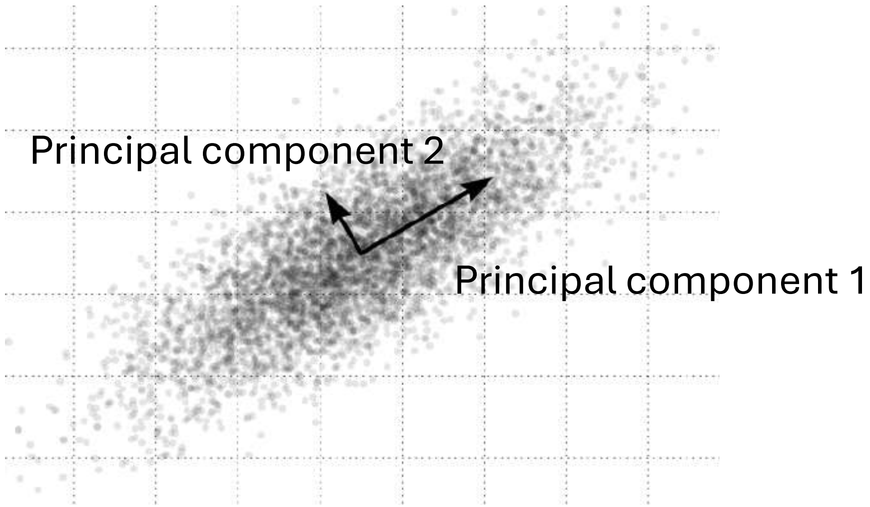
$\sigma\,(r_1, r_2)$ = how features r_1 and r_2 vary with respect to each other

Covariance matrices are widespread in signal processing and machine learning

- Principal component analysis **(PCA)**
  - Eigenvectors of the covariance matrix form principal components (PCs)
  - PCs inform the shape of a dataset (directions of variance)

Given sample $\mathbf{x}$ and eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^{\mathsf{T}}$,

PCA transform: $\tilde{\mathbf{x}} = \hat{\mathbf{V}}^{\mathsf{T}}\mathbf{x}$

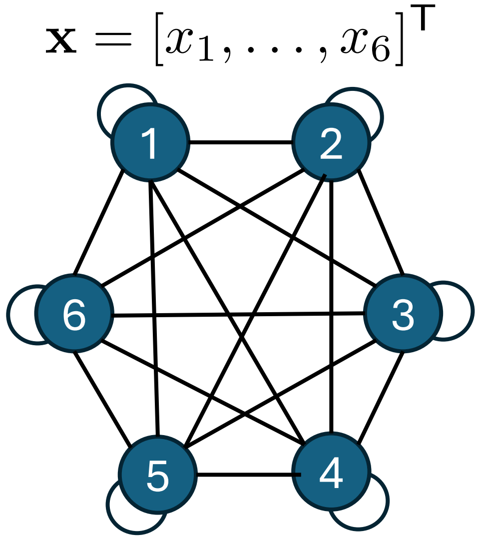


PCA transform in ML

- Unsupervised learning (dim. reduction)
- Supervised learning (regression, classification)

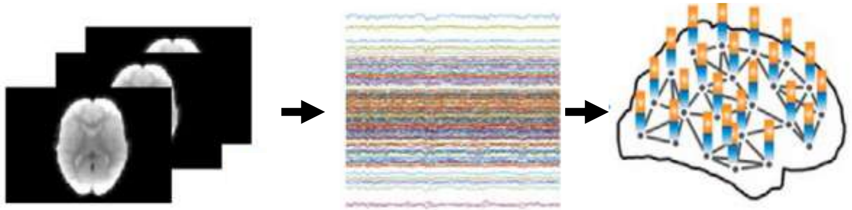
Covariance matrices are widespread in signal processing and machine learning

- Covariance matrices are leveraged as **graphical** representations of data
  - A graph $G = (V, E, W)$
    - Set of nodes V
    - A weight function W
    - Set of edges E
  - Covariance matrix is a **fully connected graph,**
    - nodes are the features
    - edges associated with pairwise covariance values

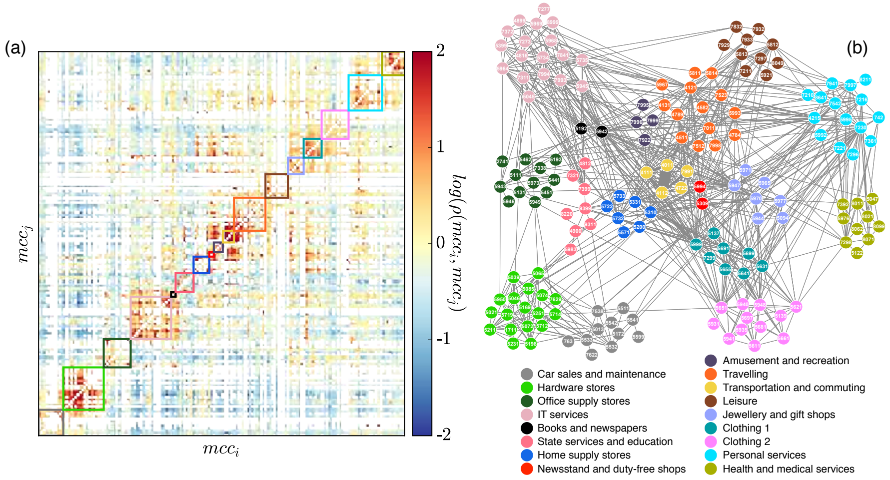


Covariance matrices are widespread in signal processing and machine learning

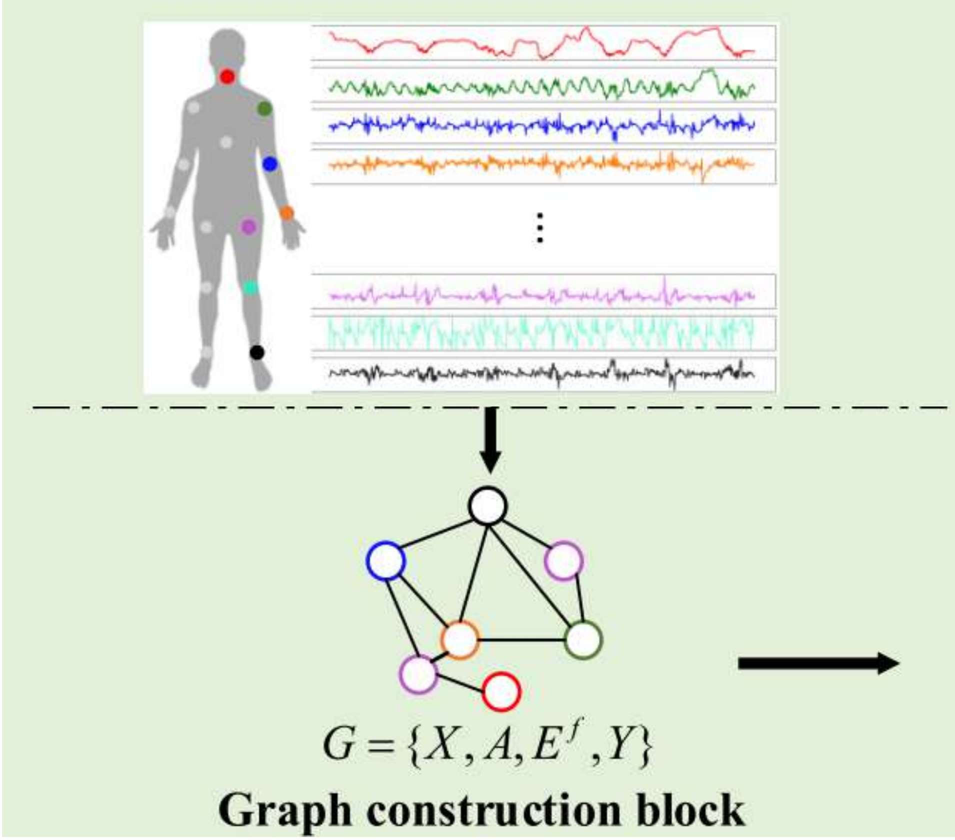
- Covariance matrices as **graphical** representations; used in graph neural nets



Brain connectome [Li, et al. 2021]



Socio-economic networks [Leo, et al. 2016]



Wearable devices [Wang, et al. 2023]

Learning with covariance matrices: Challenges

- Sample covariance matrix is estimate from **finite** data

- ML model is trained on training dataset, deployed on test dataset

- Statistical spaces defined by training and test data may not align perfectly

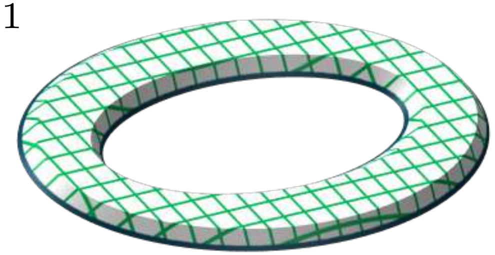
Input data → ML model → Output

Covariance matrix → ML model

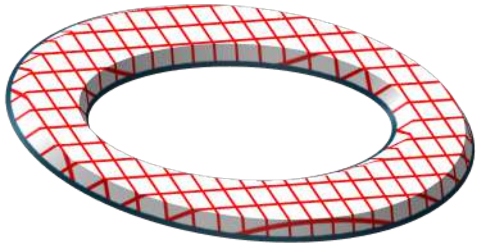
Covariance matrix

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathsf{T}}$$

Representation of training dataset



Representation of test dataset



Learning with covariance matrices: Challenges

- Sample covariance matrix is estimate from **finite** data

- ML model is trained on training dataset, deployed on test dataset

- Statistical spaces defined by training and test data may not align perfectly

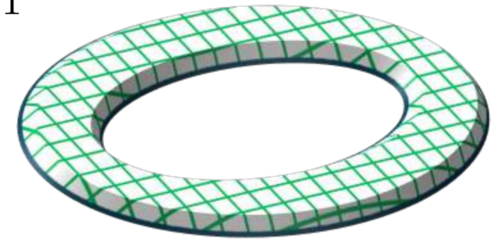
Challenge 1 (stability)
Are inference outcomes **stable** to perturbations in covariance matrix (finite sample effect)?

Input data → ML model → Output

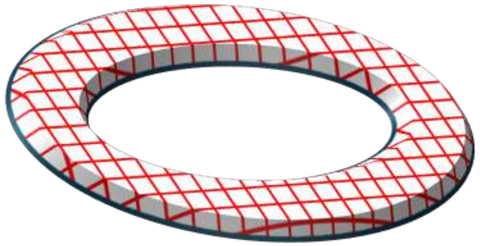
Covariance matrix

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\mathsf{T}$$

Representation of training dataset



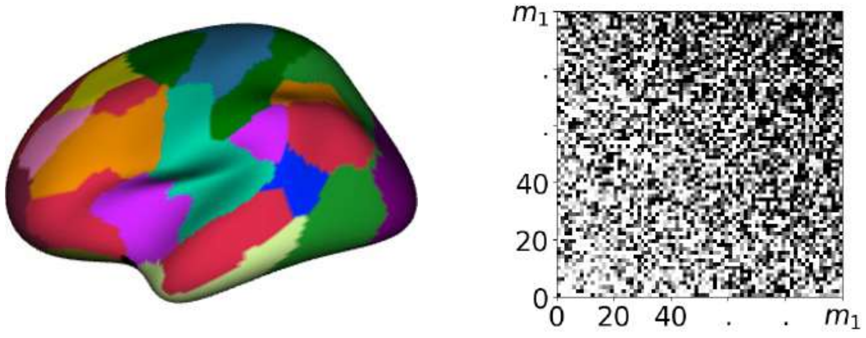
Representation of test dataset



Learning with covariance matrices: Challenges

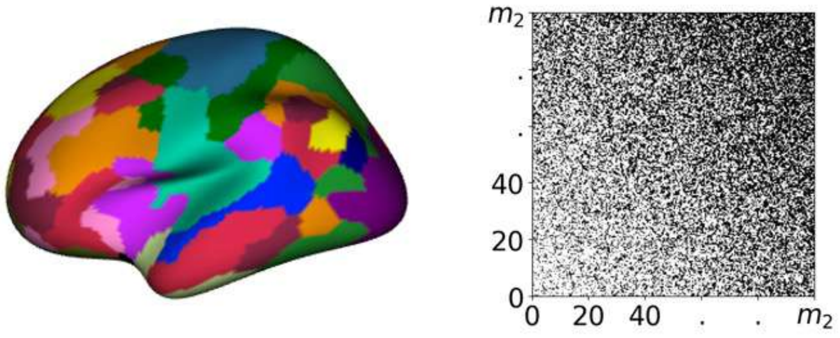
- Datasets capture information about same phenomenon at **different scales**

Dataset with m_1 features



Covariance matrix $\mathbf{C}_{m_1}$
(size $m_1 \times m_1$)

Dataset with m_2 features

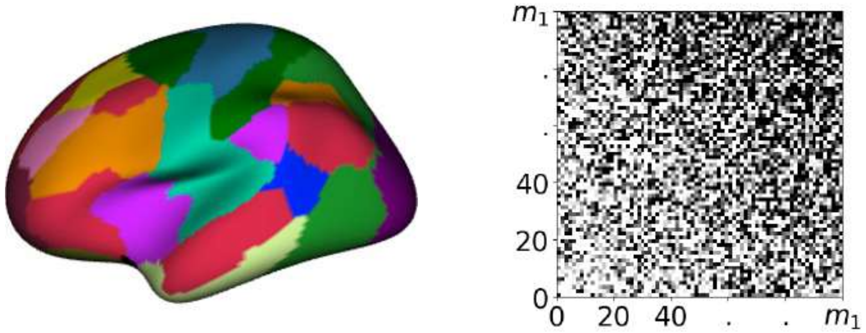


Covariance matrix $\mathbf{C}_{m_2}$
(size $m_2 \times m_2$)

Learning with covariance matrices: Challenges

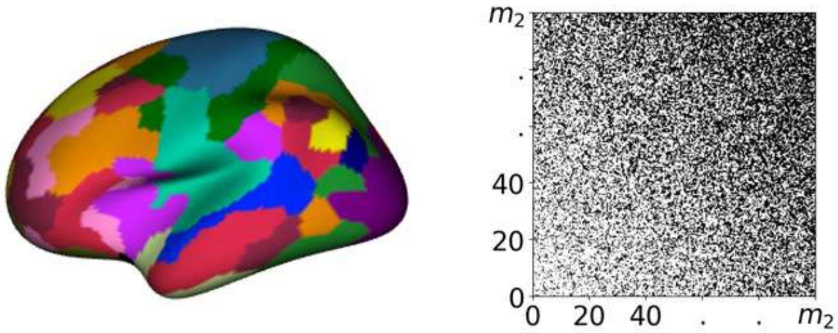
- Datasets capture information about same phenomenon at **different scales**

Dataset with m_1 features



Covariance matrix $\mathbf{C}_{m_1}$
(size $m_1 \times m_1$)

Dataset with m_2 features



Covariance matrix $\mathbf{C}_{m_2}$
(size $m_2 \times m_2$)

Input data (m_1-dimensional) → ML model $\mathbf{C}_{m_1}$ → Output

Input data (m_2-dimensional) → ML model $\mathbf{C}_{m_2}$ → Output

Challenge 2 (transferability)
Can the redundancy in covariance matrices of datasets of different sizes be exploited?

Learning with covariance matrices: A GSP approach

- Signal and information processing is about exploiting **signal structure**
- **Graph signal processing (GSP):** broaden classical signal processing to graphs

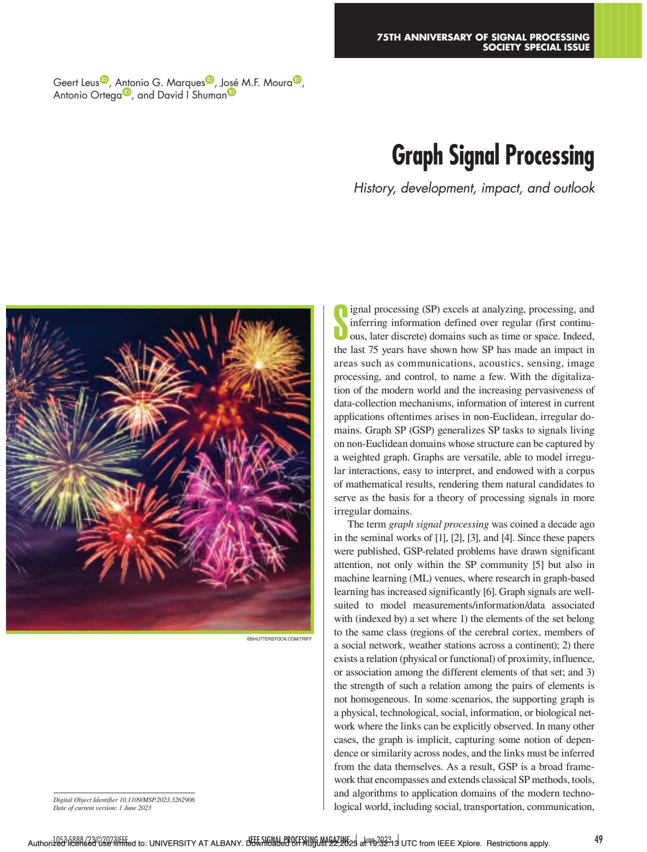
INVITED PAPER

Graph Signal Processing: Overview, Challenges, and Applications

This article presents methods to process data associated to graphs (graph signals) extending techniques (transforms, sampling, and others) that are used for conventional signals.

By Antonio Ortega, *Fellow IEEE*, Pascal Frossard, *Fellow IEEE*, Jelena Kovačević, *Fellow IEEE*, José M. F. Moura, *Fellow IEEE*, and Pierre Vandergheynst

808 Proceedings of the IEEE | Vol. 106, No. 5, May 2018

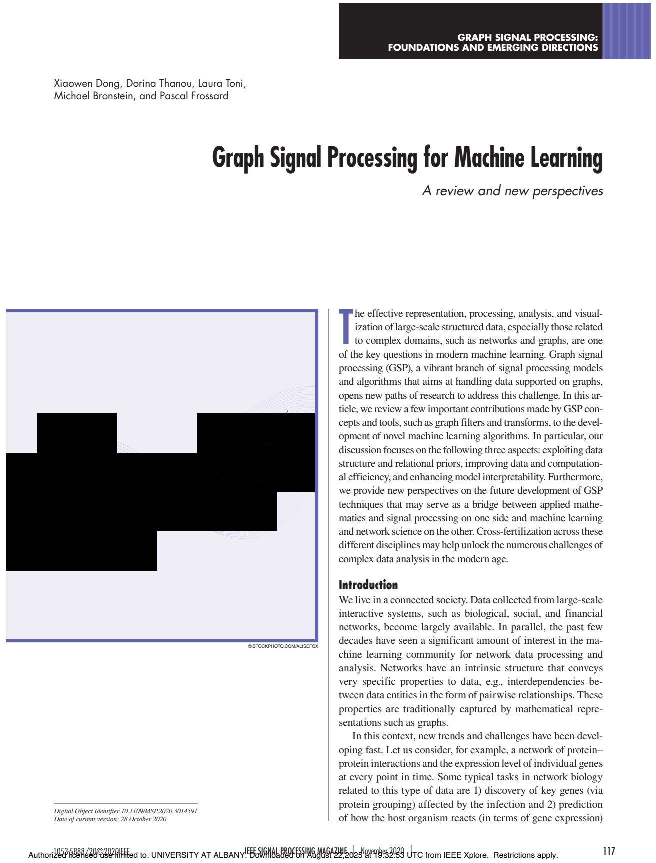


75TH ANNIVERSARY OF SIGNAL PROCESSING SOCIETY SPECIAL ISSUE

Geert Leus, Antonio G. Marques, José M.F. Moura, Antonio Ortega, and David I Shuman

Graph Signal Processing

History, development, impact, and outlook



GRAPH SIGNAL PROCESSING: FOUNDATIONS AND EMERGING DIRECTIONS

Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard

Graph Signal Processing for Machine Learning

A review and new perspectives

Learning with covariance matrices: A GSP approach

- Signal and information processing is about exploiting **signal structure**
- **Graph signal processing (GSP):** broaden classical signal processing to graphs

INVITED PAPER

Graph Signal Processing: Overview, Challenges, and Applications

This article presents methods to process data associated to graphs (graph signals) extending techniques (transforms, sampling, and others) that are used for conventional signals.

By Antonio Ortega, *Fellow IEEE*, Pascal Frossard, *Fellow IEEE*, Jelena Kovačević, *Fellow IEEE*, José M. F. Moura, *Fellow IEEE*, and Pierre Vandergheynst

75TH ANNIVERSARY OF SIGNAL PROCESSING SOCIETY SPECIAL ISSUE

Geert Leus, Antonio G. Marques, José M.F. Moura, Antonio Ortega, and David I Shuman

Graph Signal Processing

History, development, impact, and outlook

GRAPH SIGNAL PROCESSING: FOUNDATIONS AND EMERGING DIRECTIONS

Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard

Graph Signal Processing for Machine Learning

A review and new perspectives

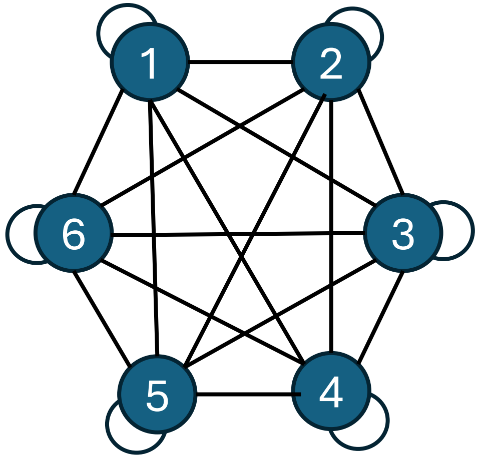
Our view: GSP perspective well-equipped to address challenges to learning with covariance matrices

Learning with covariance matrices: A GSP approach

- Graph neural networks (GNNs) have been shown to be [Ruiz et al., 2023]
  - stable to (abstract) perturbations in graph structure
  - generalizable to graph structures of different sizes

  (similar to convolutional neural nets for images)

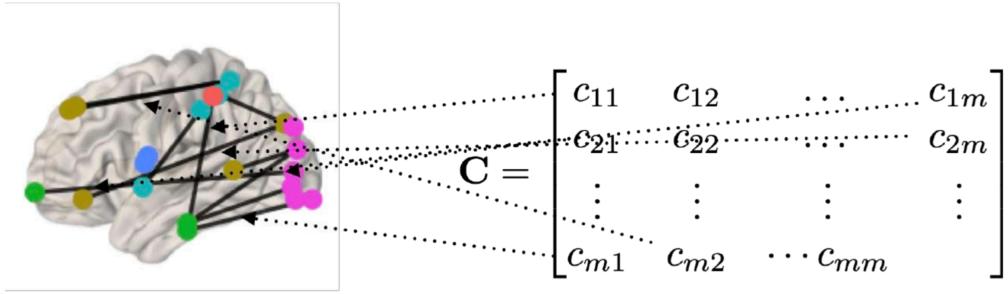
$\mathbf{x} = [x_1, \ldots, x_6]^\mathsf{T}$



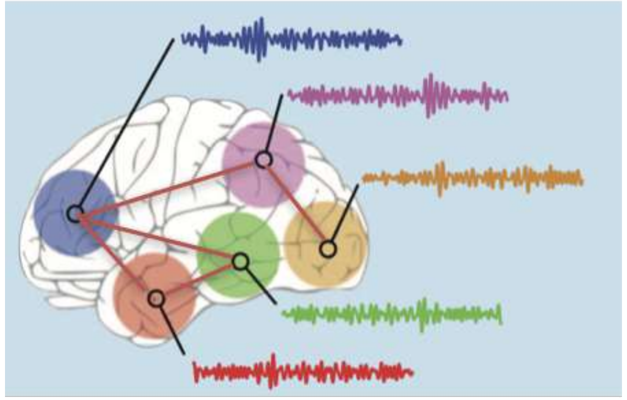
- Covariance matrix is a **data-driven** graph
  - interplay between perturbation theory of covariances and ML over them

Applications to network neuroscience

- Covariance matrices appear commonly in network neuroscience



Anatomical covariance matrix

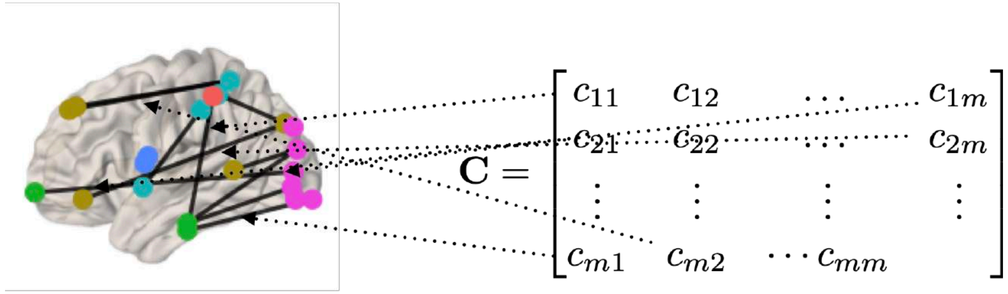


Functional connectome

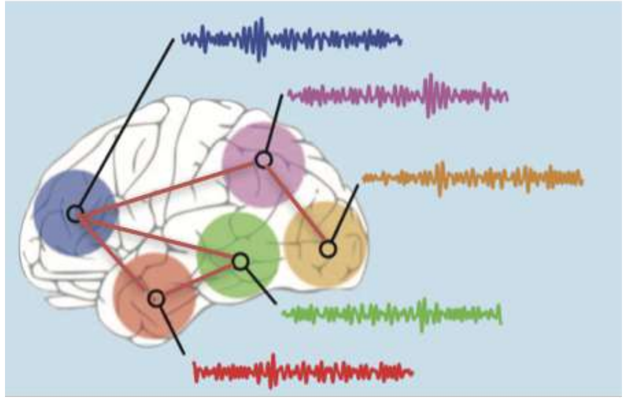
- Principled ML approaches for reproducible, transparent, generalizable findings

Applications to network neuroscience

- Covariance matrices appear commonly in network neuroscience



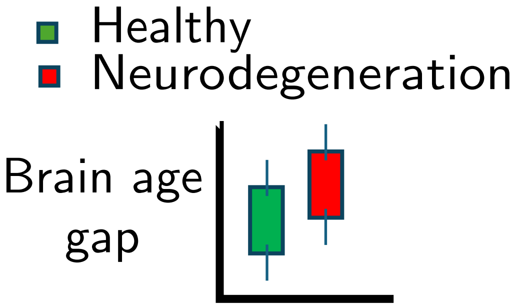
Anatomical covariance matrix



Functional connectome

- Principled ML approaches for reproducible, transparent, generalizable findings

- **Brain age gap** is a biomarker that reflects neurodegeneration
    - How VNN theoretical advances provide principled **brain age gap prediction?**



coVariance neural networks

- **coVariance neural networks (VNNs)**:

  GNNs operating on covariance matrices

- Two tutorial articles in IEEE SPM

  - Tutorial article on ‘*Disentangling neurodegeneration with brain age gap prediction models*’ (to appear in 2025)

  - Tutorial article on ‘*CoVariance Neural Networks: Principal Component Analysis Meets Learning with Graphs’* (under preparation)

SPECIAL ISSUE ON ACCELERATING BRAIN DISCOVERY THROUGH DATA SCIENCE AND NEUROTECHNOLOGY

Saurabh Sihag, Gonzalo Mateos, and Alejandro Ribeiro

Disentangling Neurodegeneration With Brain Age Gap Prediction Models

A graph signal processing perspective

Neurodegeneration is the progressive loss of structure or function of neurons in the brain. Reduction in cortical thickness or volume over time has been a workhorse metric used to assess neurodegeneration in clinical settings; see case study 1 in “Case Study 1: Cortical Atrophy Characterizes Neurodegeneration in Alzheimer’s Disease” for a demonstration of cortical atrophy assessment in the context of Alzheimer’s disease (AD) relative to healthy individuals [healthy cohort (HC) group]. Naturally, visual inspection of T1-weighted brain magnetic resonance imaging (MRI) images and associated MRI quantification products are used along with other biological measurements to make a “subjective” assessment about the brain health of an individual. These assessments tend to be subjective because they lack a deterministic relationship between an individual’s health status and the absolute values of the metrics observed within MRI scans [1]. Moreover, such methods cannot adequately account for the statistical complexities inherent within neuroimaging datasets that capture neurodegeneration. In particular, neurodegeneration is a characteristic of the healthy aging process and various neurological disorders [2], exhibiting correlated patterns across brain regions. Such statistical factors motivate well the use of data-driven methods to characterize neurodegeneration.

Automating or improving the analyses of brain MRI images is appealing for several reasons: MRI is a noninvasive proce-

Outline

- PCA and the graph Fourier transform
- CoVariance neural networks (VNNs)
- Theory of VNNs: Stability and transferability
- Application: Principled brain age gap prediction with VNNs
- Variants of VNNs

Key takeaways

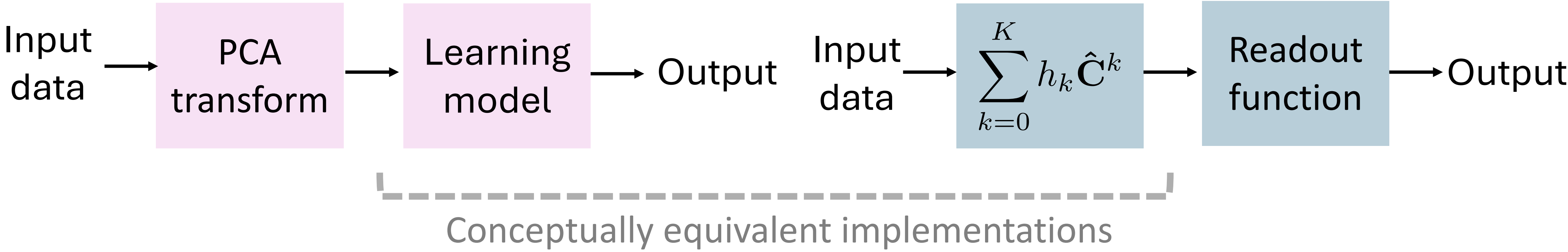
- VNNs offer a novel GSP-inspired perspective to PCA

  addressing challenges in modern data analysis
- Principled deep learning solution for finite-data regimes
  - Stability and transferability
- VNNs address methodological/conceptual obscurities in brain age gap prediction

PCA and Graph Fourier Transform

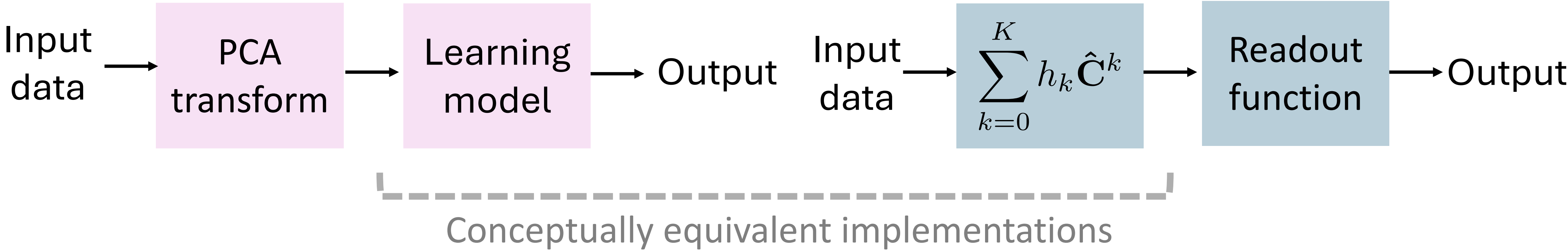
A graph filter implementation of PCA inference

- **To show:** PCA-based inference can be implemented with a polynomial over $\hat{\mathbf{C}}$



A graph filter implementation of PCA inference

- **To show:** PCA-based inference can be implemented with a polynomial over $\hat{\mathbf{C}}$

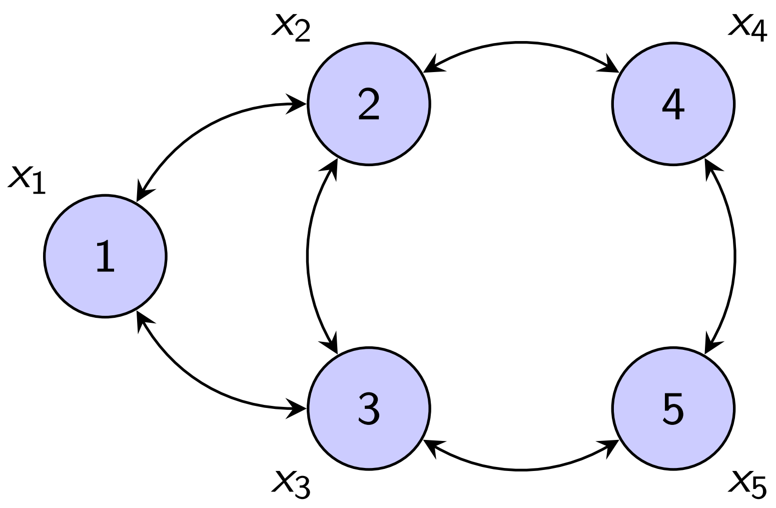


- **How:** Follows from the graph Fourier transform analysis of $\sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k$
- **Implications:**
  - Alternative implementation of PCA-based inference using polynomial over $\hat{\mathbf{C}}$
  - But more importantly, polynomial implementation is **stable, transferable**

Preliminaries: Graph

- **Graph:** a triplet (V, E, W)
  - A set of **nodes** $V = \{1, \dots, m\}$
  - A set of (undirected) **edges** $E \subseteq V \times V$

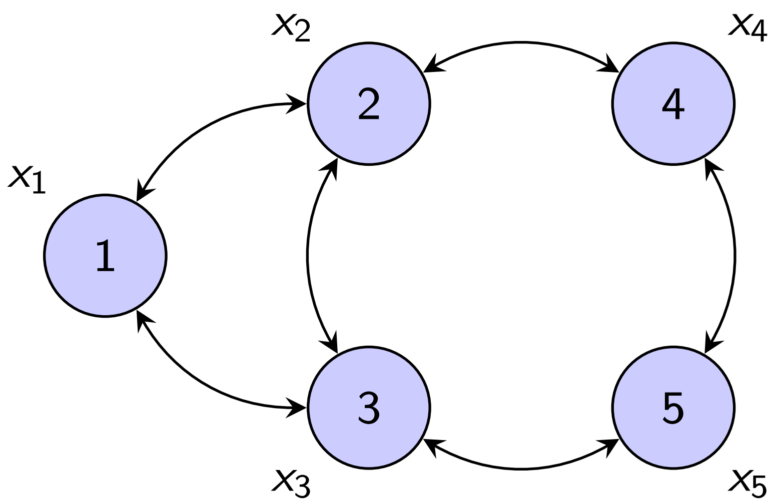
    Edge between node i and j denoted by (i, j)
  - An **edge function** $W: E \mapsto \mathbb{R}$ that maps edge $(i, j)$to weight $w_{ij} \in \mathbb{R}$



Preliminaries: Graph

- **Graph:** a triplet (V, E, W)
  - A set of **nodes** $V = \{1, \ldots, m\}$
  - A set of (undirected) **edges** $E \subseteq V \times V$

    Edge between node i and j denoted by (i, j)
  - An **edge function** $W: E \mapsto \mathbb{R}$ that maps edge $(i, j)$to weight $w_{ij} \in \mathbb{R}$



- **Adjacency matrix** representation of graph

$$[\mathbf{A}]_{ij} = \begin{cases} w_{ij}, & \text{if } (i,j) \in E, \\ 0, & \text{otherwise} \end{cases}$$

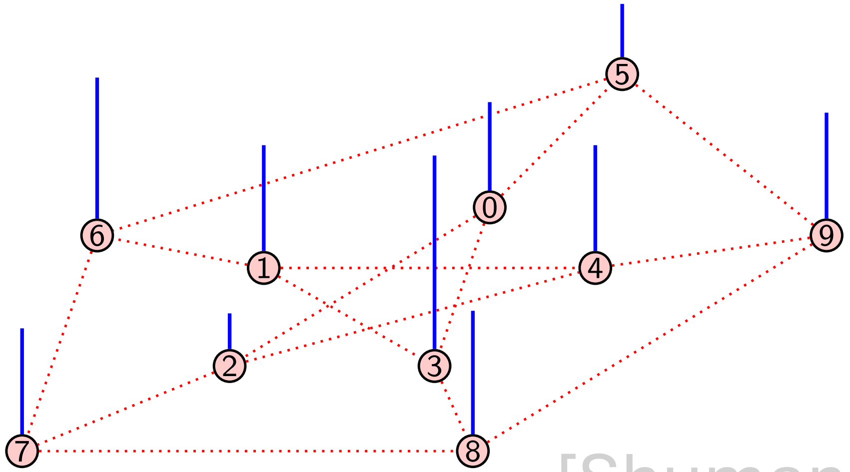
Preliminaries: Graph signal

➢ **Graph signals** are mappings $x: V \mapsto \mathbb{R}$

⇨ graph signal is defined on the vertices of the graph

➢ **Graph signal** can be represented as a vector $\mathbf{x} \in \mathbb{R}^m$

⇨ x_i denotes the graph signal at i-th vertex in V



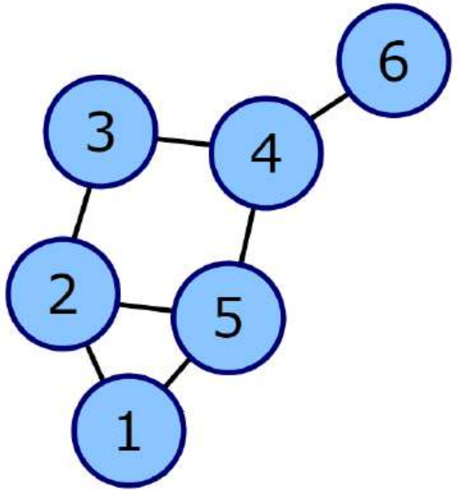
$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_9 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.7 \\ 0.3 \\ \vdots \\ 0.7 \end{bmatrix}$$

[Shuman, 2013]

Preliminaries: Graph shift operator (GSO)

- To understand and analyze graph signal $\mathbf{x}$, GSP accounts for the graph structure
- Graph structure is encoded in a **graph shift operator** $\mathbf{S} \in \mathbb{R}^{m \times m}$

$\Longrightarrow$ $[\mathbf{S}]_{\mathrm{ij}} = 0$ for $i \neq j$ and $(i, j) \notin E$ ($\mathbf{S}$ captures local graph structure)



$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} & 0 & 0 & S_{15} & 0 \\ S_{21} & S_{22} & S_{23} & 0 & S_{25} & 0 \\ 0 & S_{23} & S_{33} & S_{34} & 0 & 0 \\ 0 & 0 & S_{43} & S_{44} & S_{45} & S_{46} \\ S_{51} & S_{52} & 0 & S_{54} & S_{55} & 0 \\ 0 & 0 & 0 & S_{64} & 0 & S_{66} \end{pmatrix}$$

- **Examples**: adjacency matrix, Laplacian

Covariance matrix is a **data-driven** adjacency matrix

Preliminaries: Graph Fourier Transform (GFT)

- Generically, eigendecomposition of GSO $\mathbf{S} = \mathbf{U\Phi U^{-1}}$
- **GFT** is the projection of graph signal on the eigenvector space $\mathbf{U}$

$$\tilde{\mathbf{x}} = \mathbf{U^{-1}x}$$

- **Inverse GFT** is defined as

$$\mathbf{x} = \mathbf{U}\,\tilde{\mathbf{x}}$$

$\Rightarrow$ Eigenvectors $\mathbf{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_m\,]$ are the frequency basis

When GSO is covariance matrix...

- GFT over covariance matrix

Given eigendecomposition

$$\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^{\mathsf{T}}$$

GFT of $\mathbf{x}$ is

$$\tilde{\mathbf{x}} = \hat{\mathbf{V}}^{\mathsf{T}}\mathbf{x}$$

When GSO is covariance matrix…

- GFT over covariance matrix

  Given eigendecomposition

  $$\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\boldsymbol{\Lambda}}\hat{\mathbf{V}}^{\mathsf{T}}$$

  GFT of $\mathbf{x}$ is

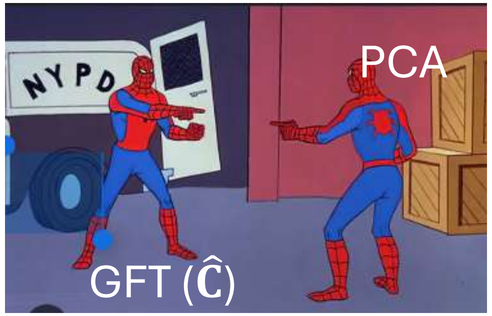
  $$\tilde{\mathbf{x}} = \hat{\mathbf{V}}^{\mathsf{T}}\mathbf{x}$$

- PCA transform

  Projection of sample $\mathbf{x}$ on principal components of $\hat{\mathbf{C}}$

  PCA transform: $\tilde{\mathbf{x}} = \hat{\mathbf{V}}^{\mathsf{T}}\mathbf{x}$

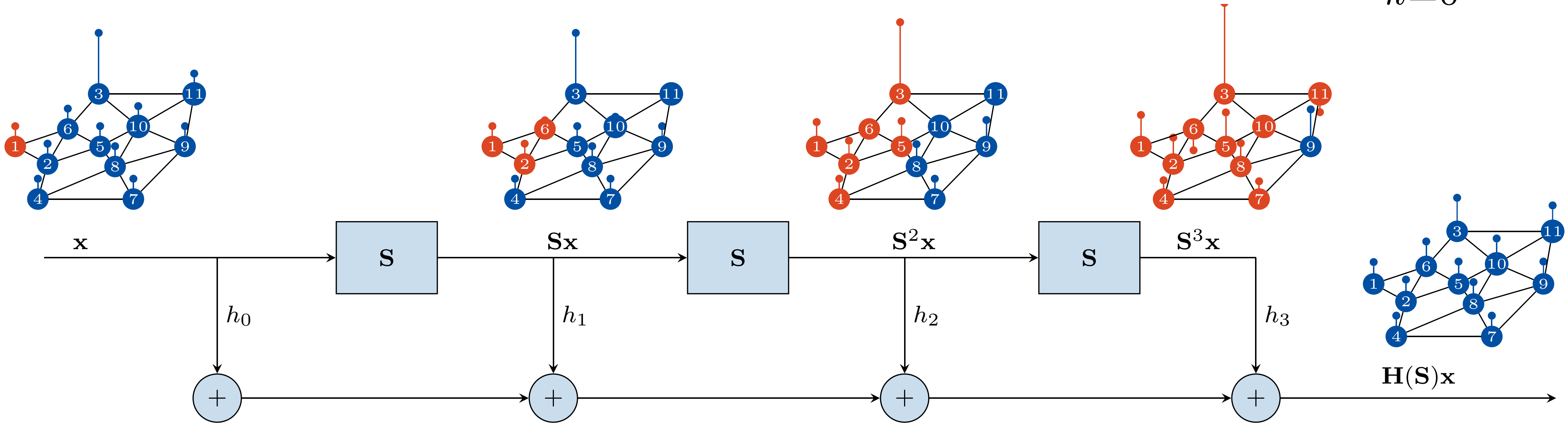
PCA transform is GFT with respect to the covariance graph!



- **Graph filter $\mathbf{H}$** maps graph signal $\mathbf{x}$ to another graph signal $\mathbf{z}$ via linear-shift-and-sum operation

$$\mathbf{z} = \mathbf{H}(\mathbf{S})\mathbf{x},$$

$$\text{where } \mathbf{H} \coloneqq h_0\mathbf{S}^0 + h_1\mathbf{S}^1 + h_2\mathbf{S}^2 + \cdots + h_K\mathbf{S}^K = \sum_{k=0}^{K} h_k\mathbf{S}^k$$

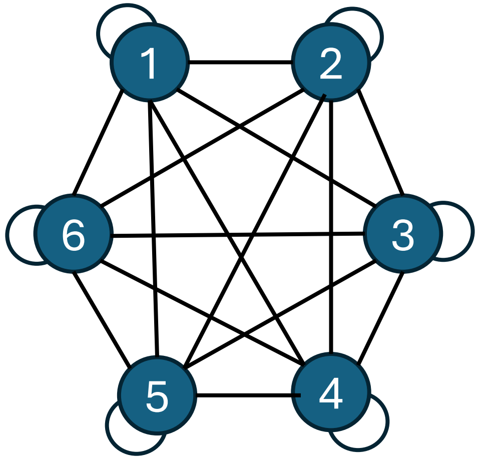


[Isufi et. al, IEEE TSP, 2024]

Graph filter on covariance matrix

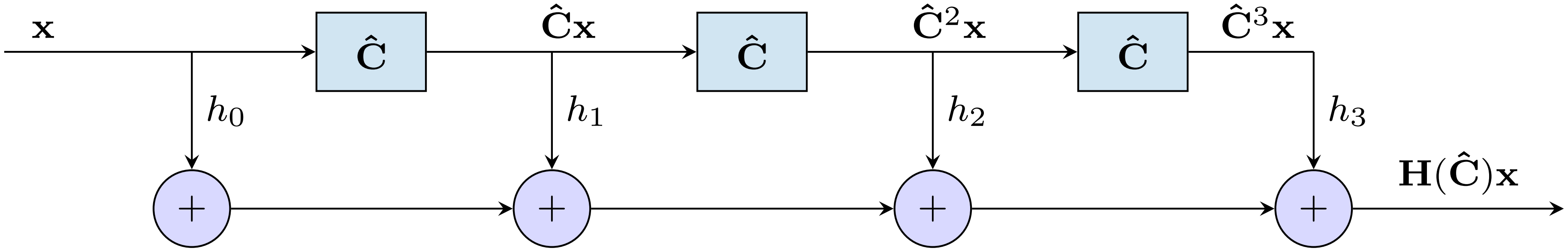
- Covariance matrix forms a fully-connected graph where
  - nodes are features
  - edges are covariance values

$\mathbf{x} = [x_1, \ldots, x_6]^\mathsf{T}$



- Graph filter on covariance matrix $\hat{\mathbf{C}}$ is defined as

$$\mathbf{H}(\hat{\mathbf{C}}) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$



CoVariance filter

- Analogy between $\mathbf{H}(\hat{\mathbf{C}})$ and PCA
  - Using eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^{\mathsf{T}}$, it follows that

$$\mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^k \hat{\mathbf{V}}^{\mathsf{T}} \mathbf{x} = \hat{\mathbf{V}} \underbrace{\left(\sum_{k=0}^{K} h_k \hat{\mathbf{\Lambda}}^k\right)}_{\text{Frequency response}} \underbrace{\hat{\mathbf{V}}^{\mathsf{T}} \mathbf{x}}_{\text{PCA}}$$

CoVariance filter

- Analogy between $\mathbf{H}(\hat{\mathbf{C}})$ and PCA
  - Using eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^\mathsf{T}$, it follows that

$$\mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x} = \sum_{k=0}^{K} h_k \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^k \hat{\mathbf{V}}^\mathsf{T}\mathbf{x} = \hat{\mathbf{V}}\underbrace{\Big(\sum_{k=0}^{K} h_k \hat{\mathbf{\Lambda}}^k\Big)}_{\text{Frequency response}}\underbrace{\hat{\mathbf{V}}^\mathsf{T}\mathbf{x}}_{\text{PCA}}$$

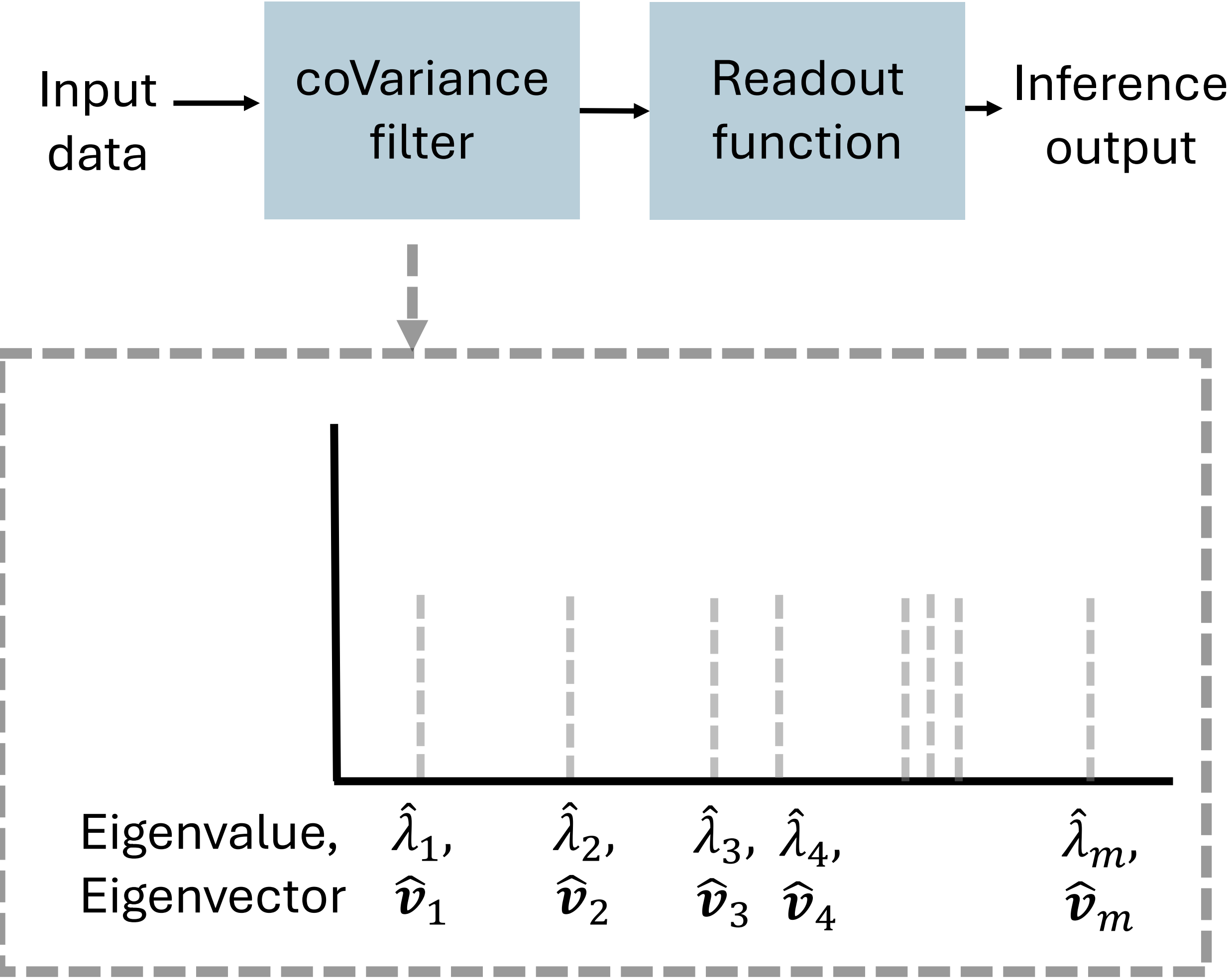
  - GFT of coVariance filter output $\mathbf{z}$ and PCA are **equivalent**

$$\tilde{\mathbf{z}} = \Big(\sum_{k=0}^{K} h_k \hat{\mathbf{\Lambda}}^k\Big)\hat{\mathbf{V}}^\mathsf{T}\mathbf{x}$$

i-th component of $\tilde{\mathbf{z}}$ is modulated by $h(\lambda_i) = \sum_{k=0}^{K} h_k \lambda_i^k$

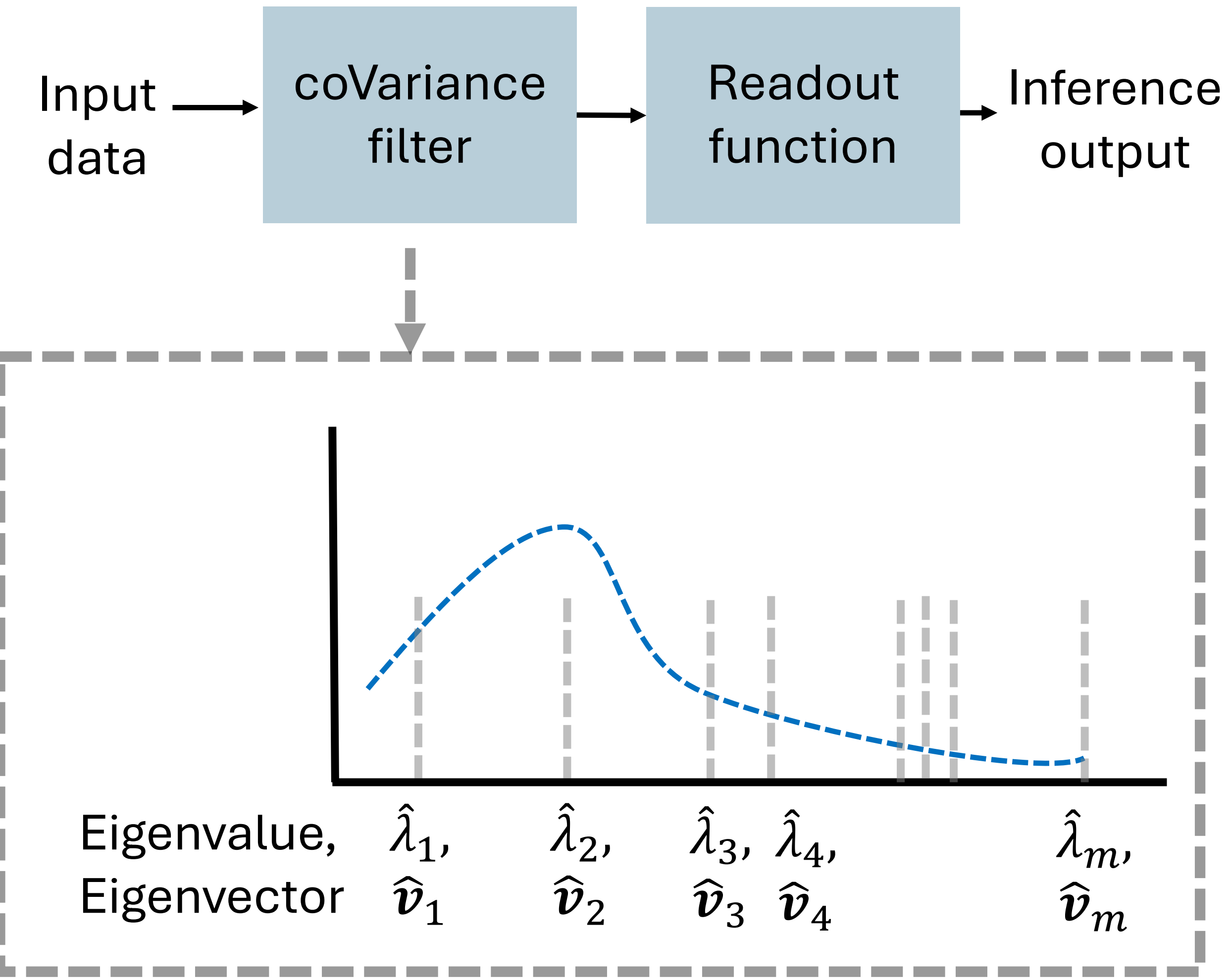
Learning with coVariance filter versus PCA-based learning

➢ Learning with a coVariance filter



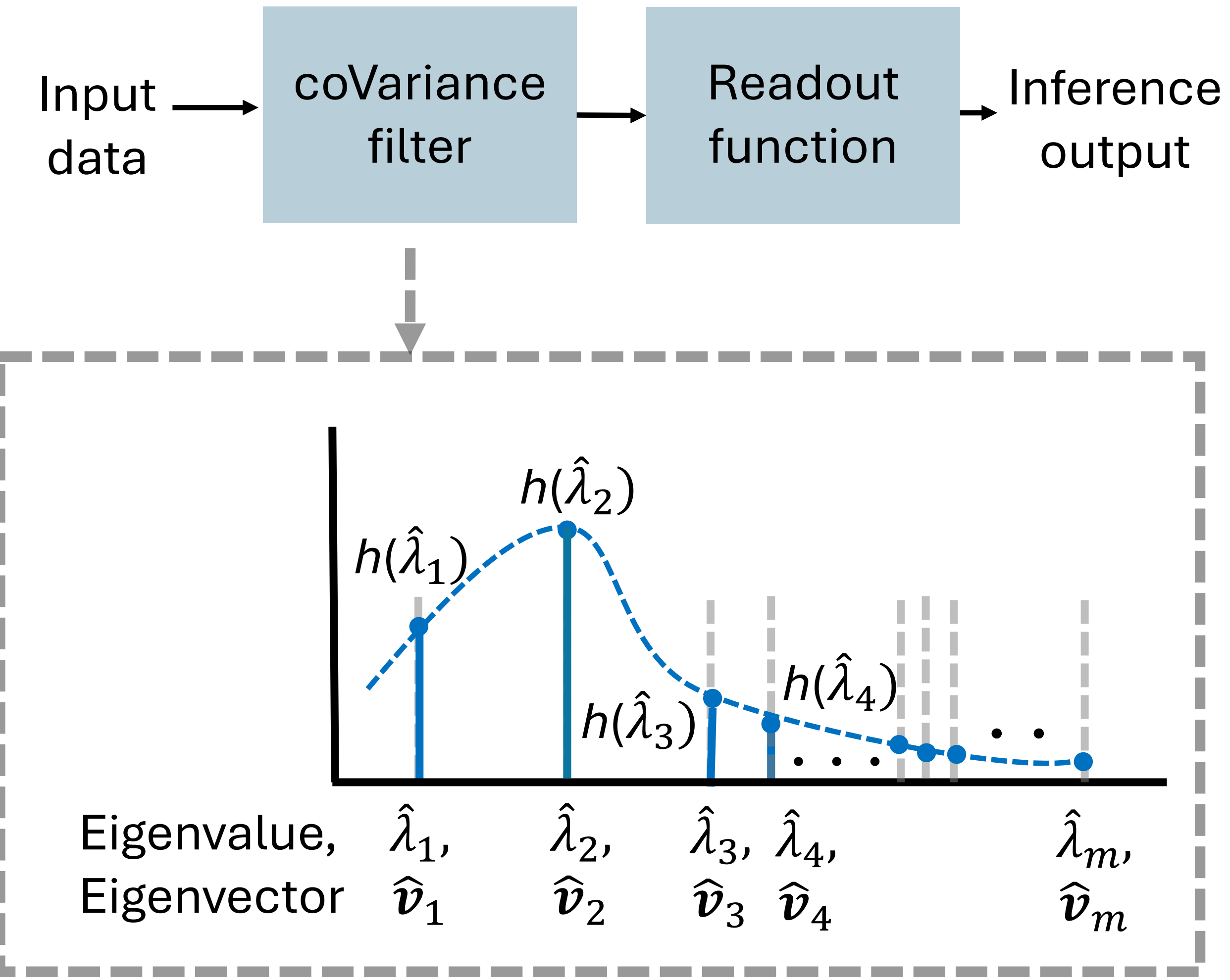
Learning with coVariance filter versus PCA-based learning

- Learning with a coVariance filter



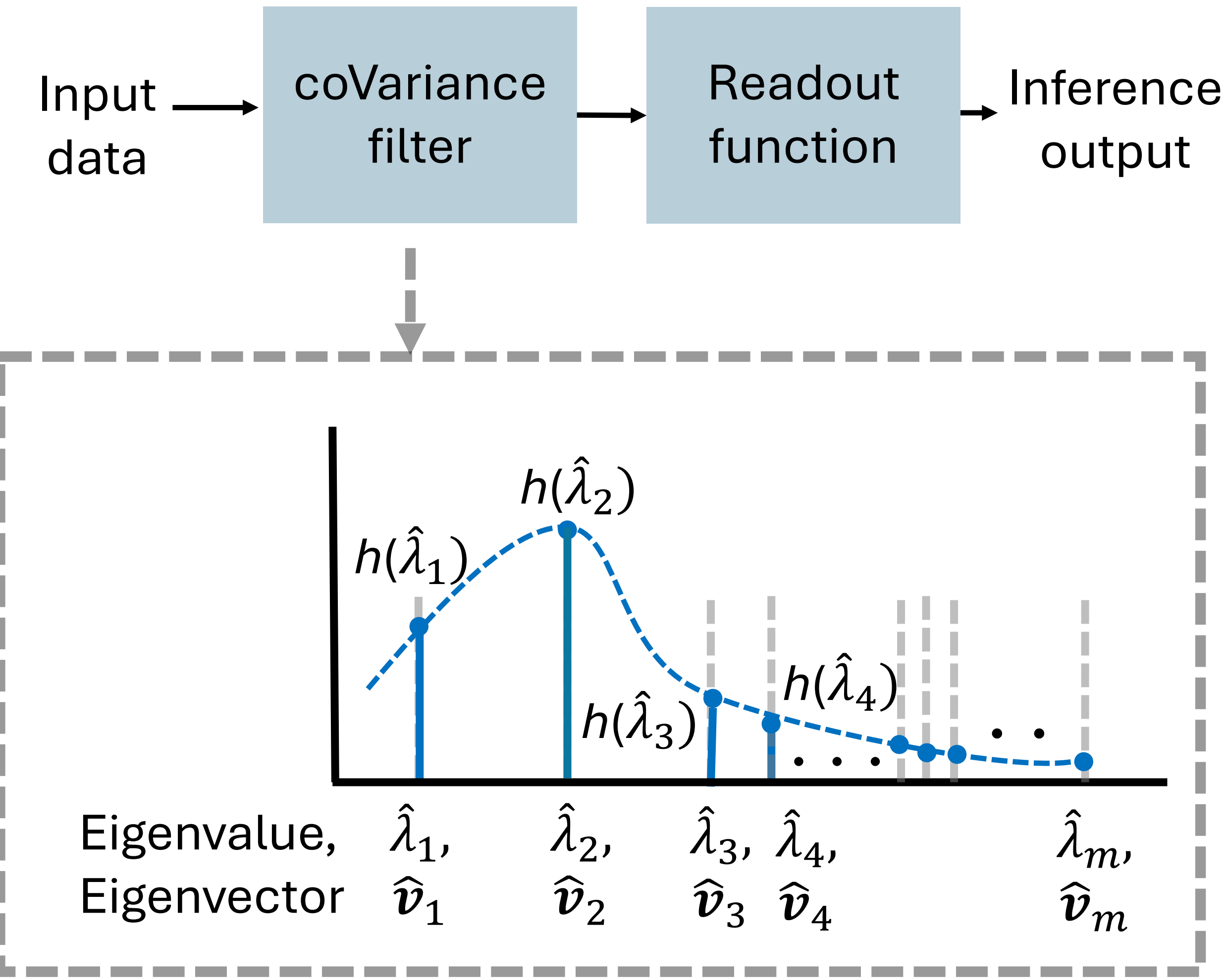
Learning with coVariance filter versus PCA-based learning

➢ Learning with a coVariance filter

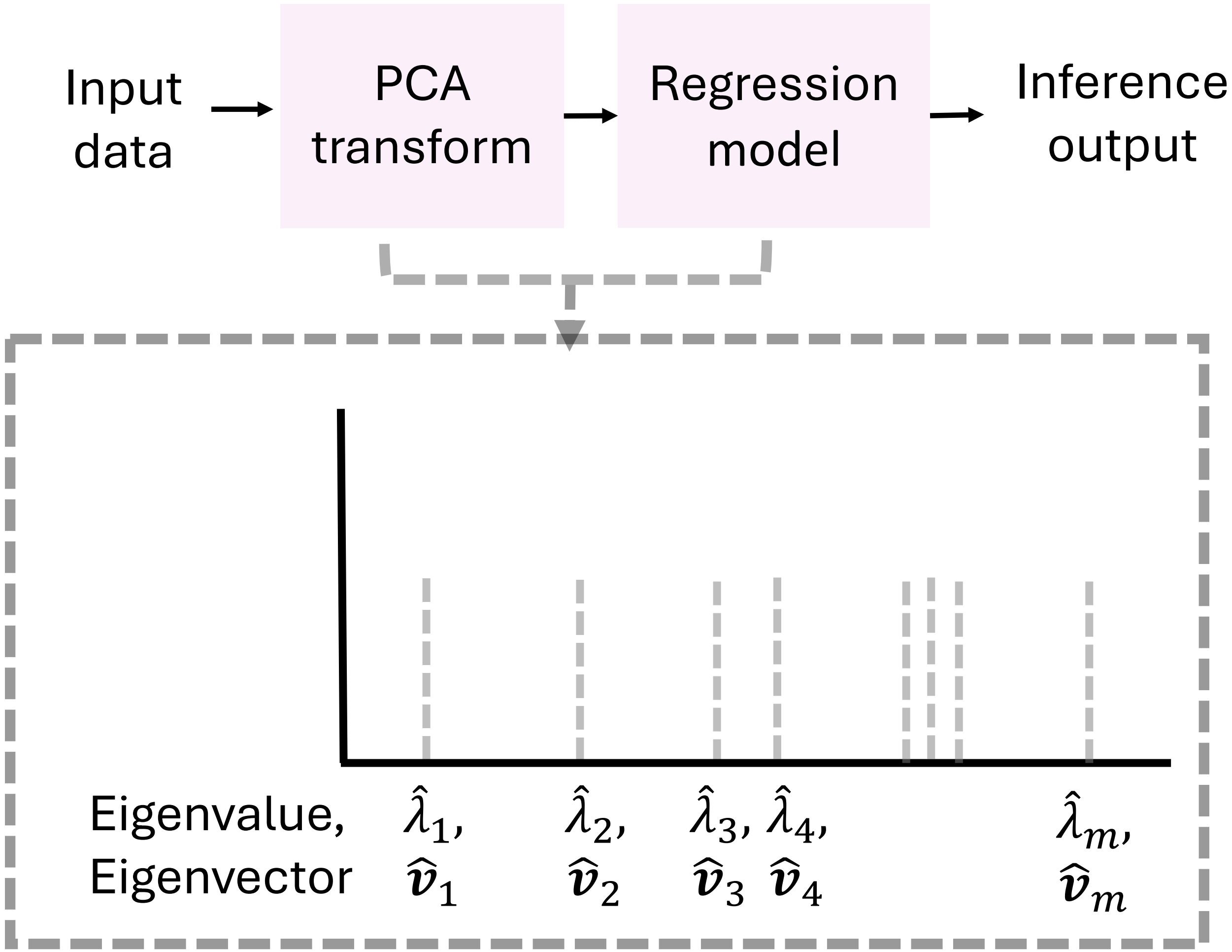


Learning with coVariance filter versus PCA-based learning

- Learning with a coVariance filter

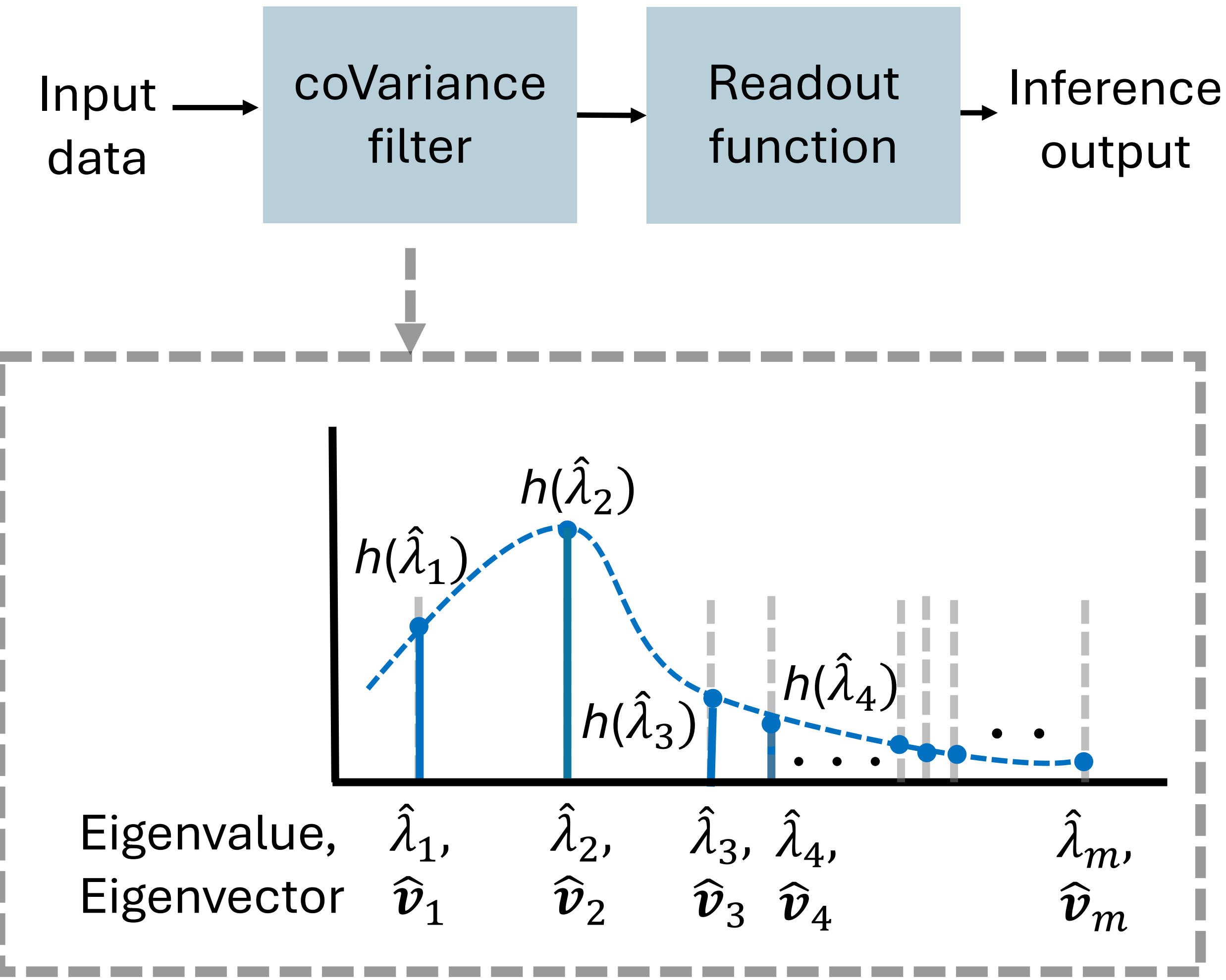


- PCA-based learning

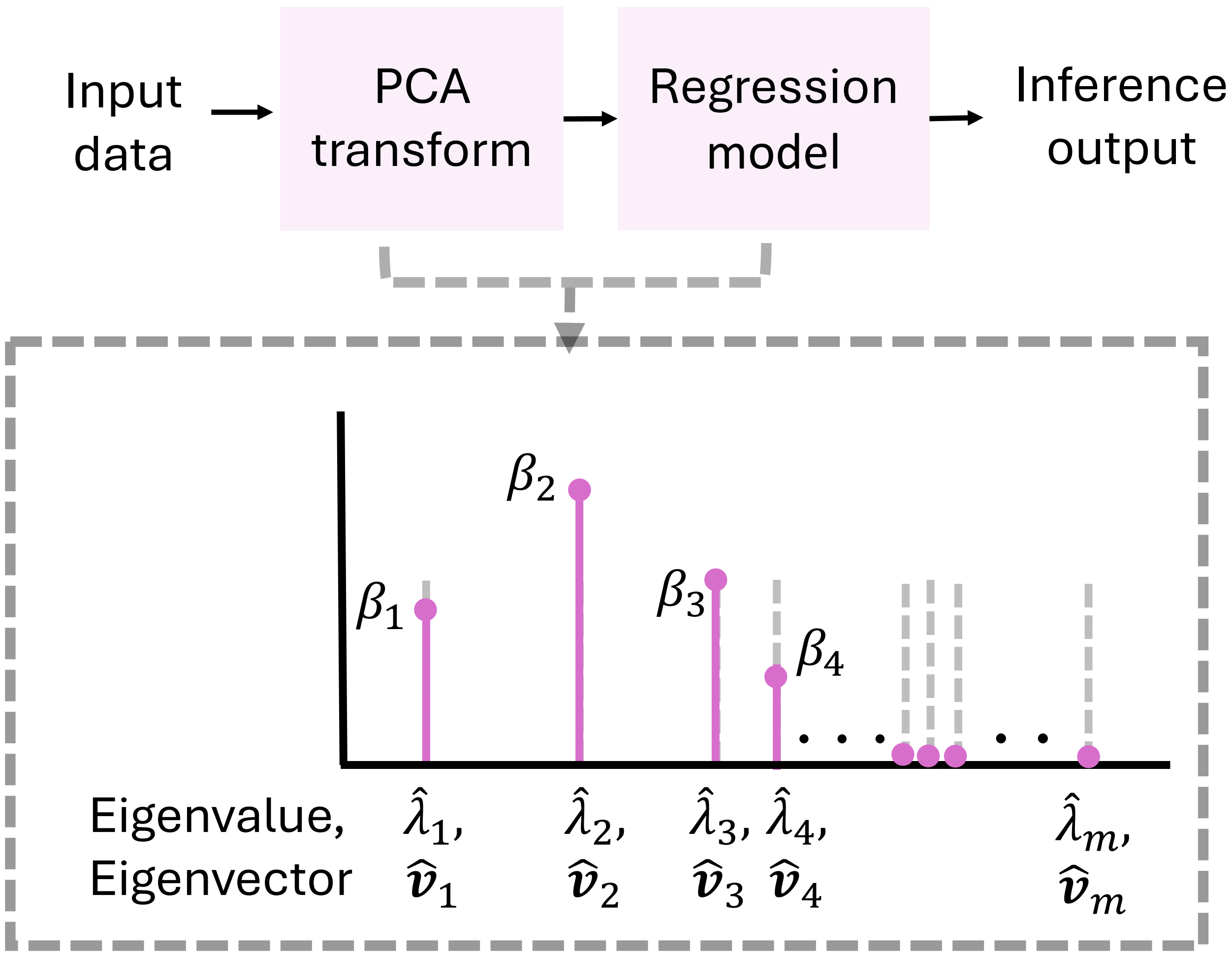


Learning with coVariance filter versus PCA-based learning

- Learning with a coVariance filter



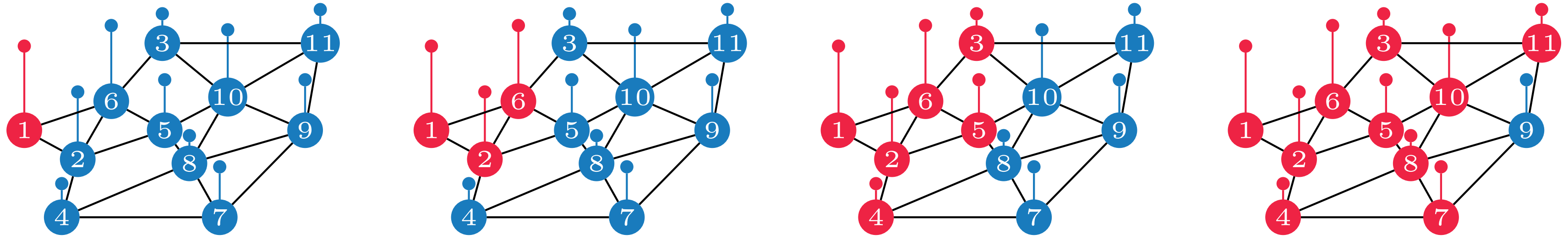
- PCA-based learning

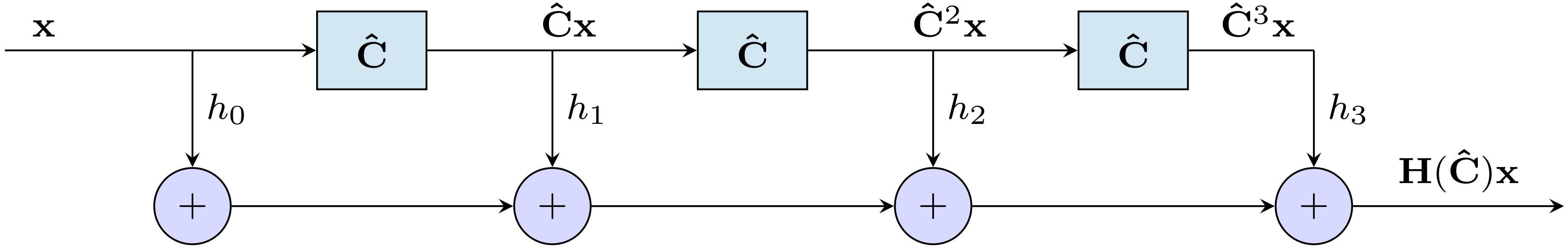


coVariance Neural Networks (VNNs)

coVariance filters as convolutional operators

- Operation $\hat{\mathbf{C}}^k\mathbf{x}$ performs a k-shift of signal $\mathbf{x}$ over graph defined by $\hat{\mathbf{C}}$





- Parameters $\{\boldsymbol{h_k}\}$ are called **filter taps**, are **scalars** and **learnable** parameters

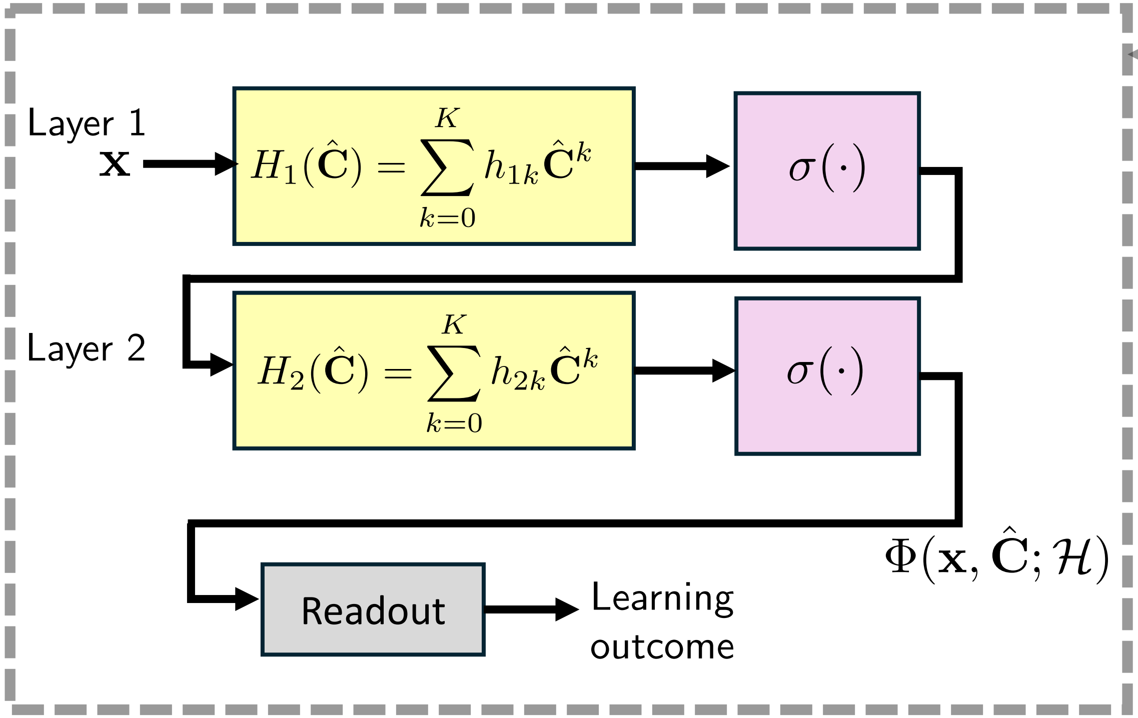
CoVariance Neural Networks (VNNs)

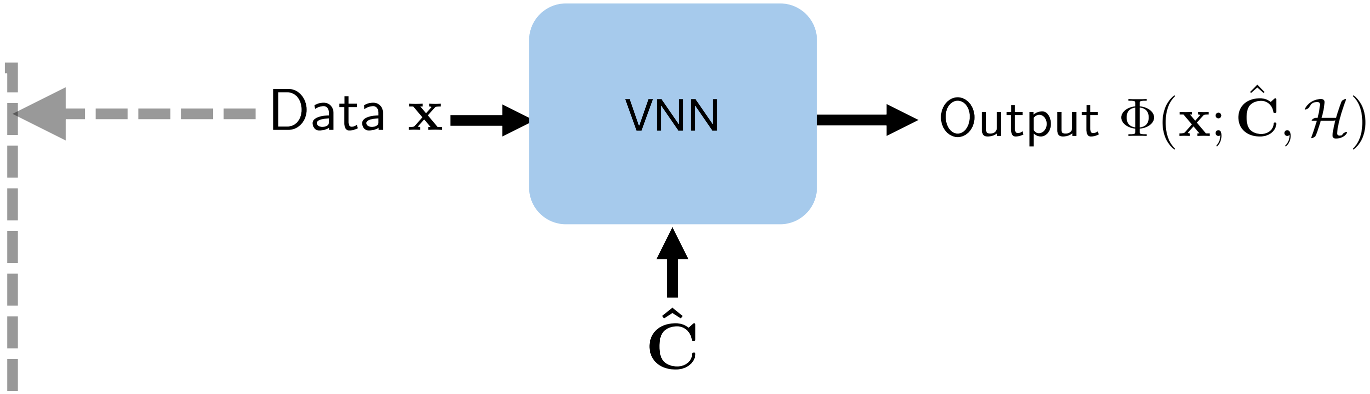
- coVariance filters can learn only **linear** representations
- To accommodate learn **non-linear** representations, concatenate coVariance filter with pointwise non-linearity σ (for e.g., ReLU, sigmoid, etc.)

CoVariance Neural Networks (VNNs)

- coVariance filters can learn only **linear** representations
- To accommodate learn **non-linear** representations, concatenate coVariance filter with pointwise non-linearity σ (for e.g., ReLU, sigmoid, etc.)

Example: A two-layer VNN

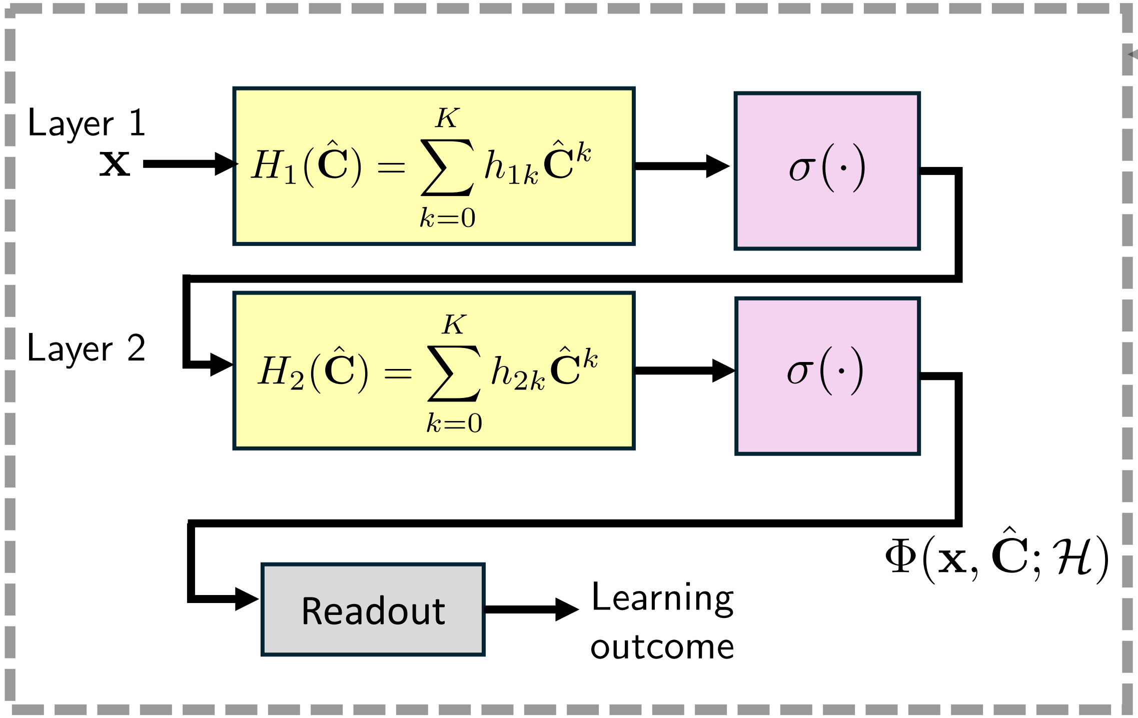


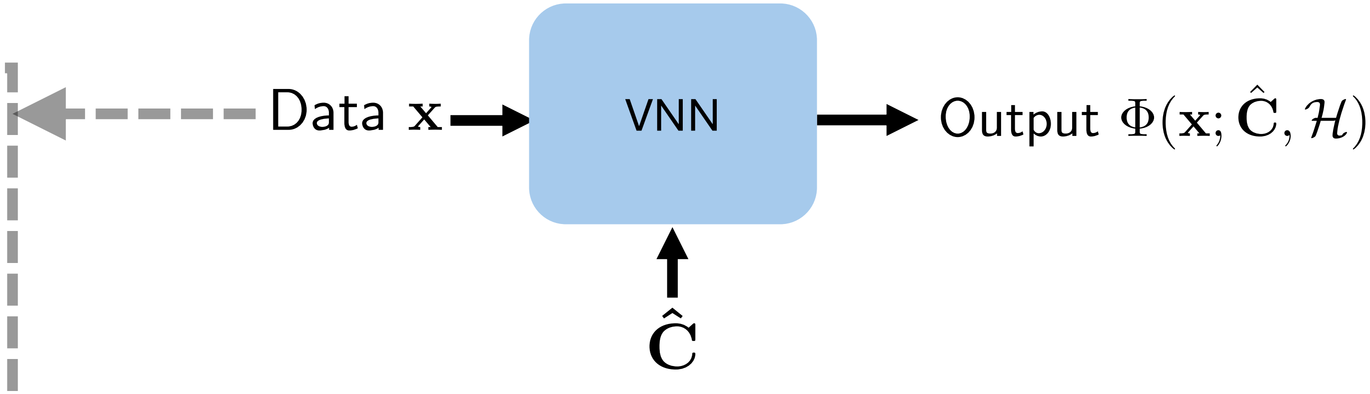


CoVariance Neural Networks (VNNs)

- coVariance filters can learn only **linear** representations
- To accommodate learn **non-linear** representations, concatenate coVariance filter with pointwise non-linearity σ (for e.g., ReLU, sigmoid, etc.)

Example: A two-layer VNN



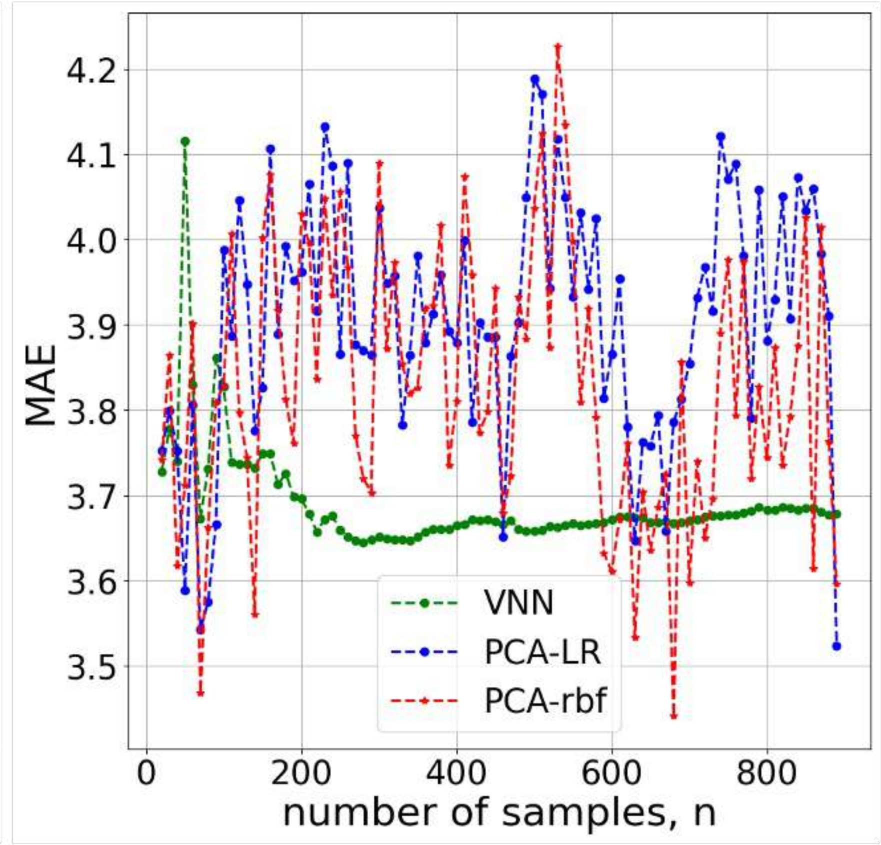


- $\Phi(\mathbf{x}; \hat{\mathbf{C}}, \mathcal{H})$ represents VNN output
- $\mathcal{H}$ is set of all filter taps

VNNs outperform PCA (regression task)

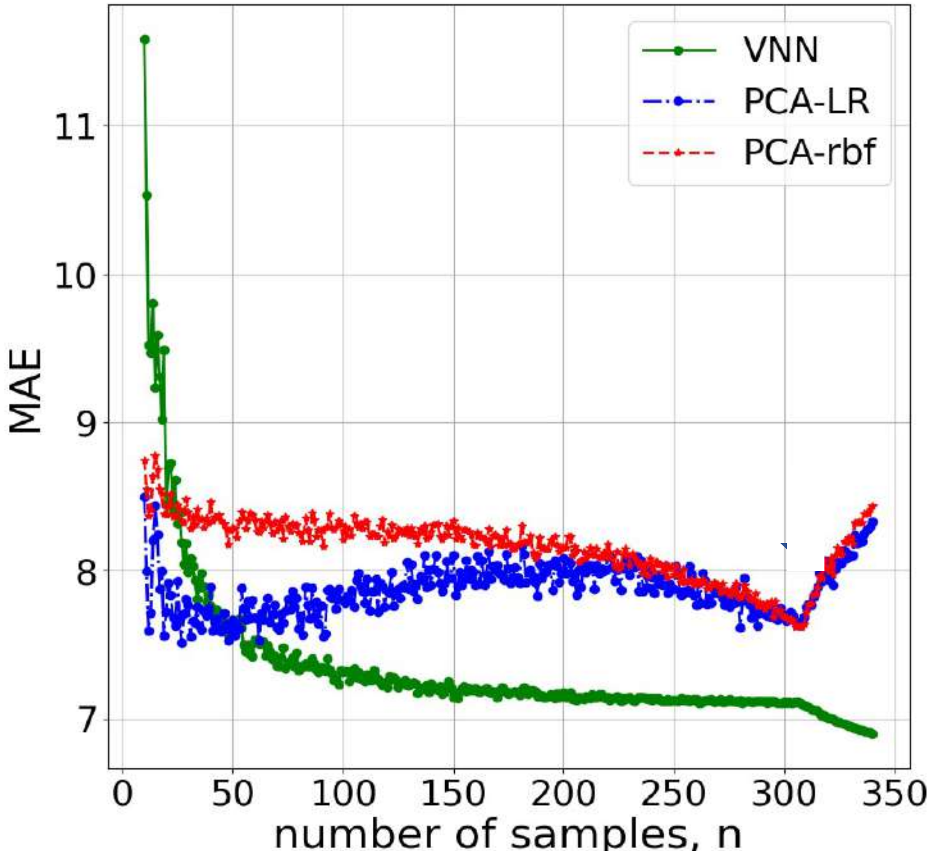
Synthetic data

(Friedman regression problem)



Neuroimaging data

(age prediction task)

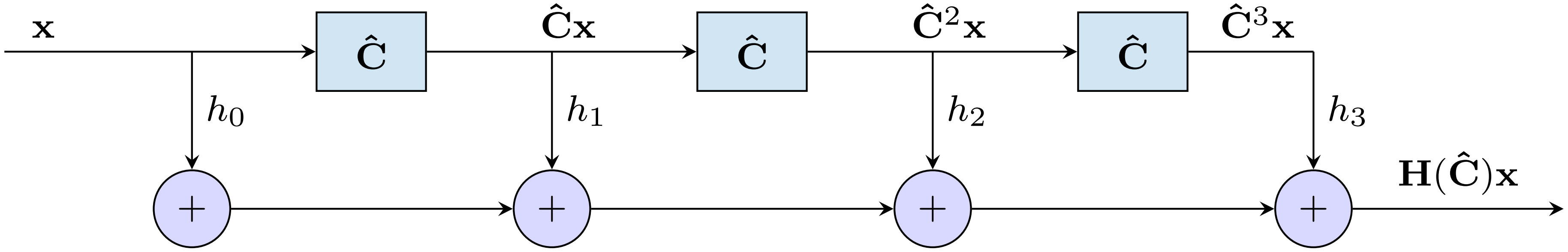


Covariance Filters and Neural Networks

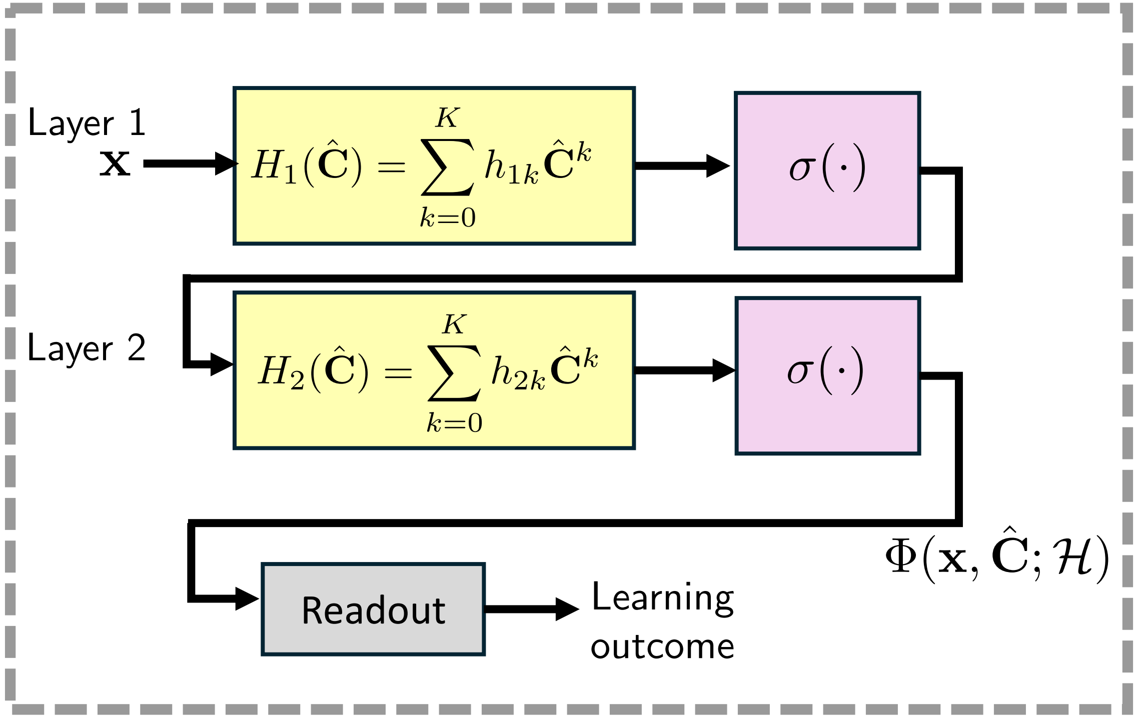
- A covariance filter is a polynomial in the covariance matrix $\hat{\mathbf{C}}$

$$\mathbf{H}(\hat{\mathbf{C}}) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

- We train the filter coefficients h_k to accomplish some task



CoVariance Neural Networks (VNNs)



- A VNN is a composition of layers
- Each of which is a composition of

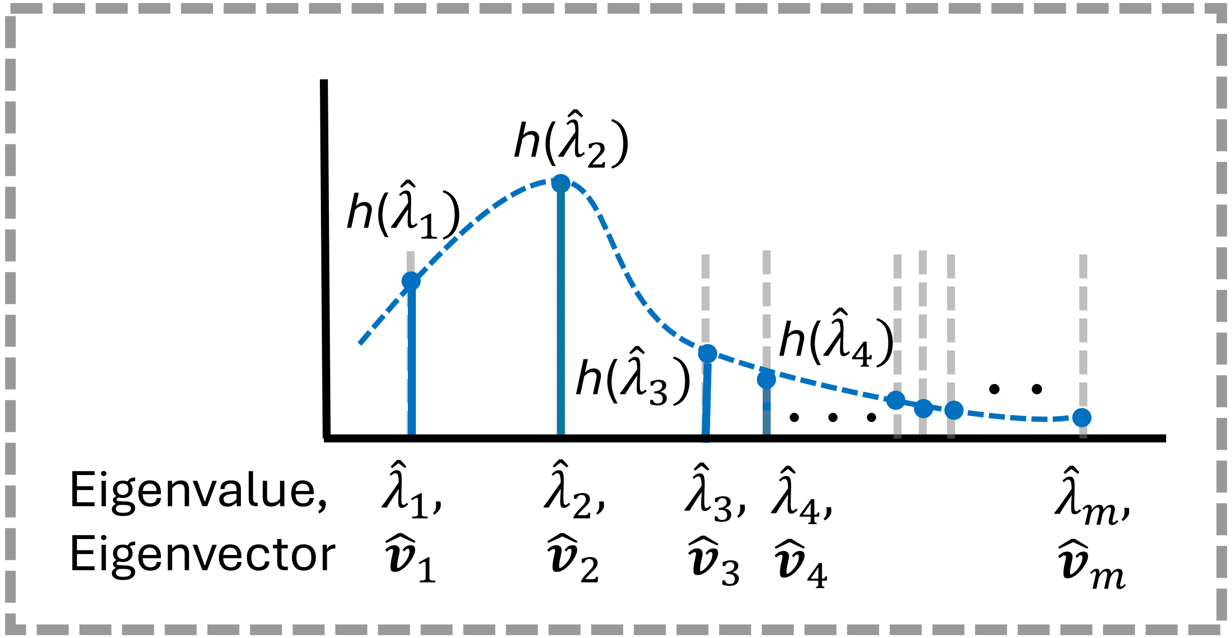
  … a covariance filter

  … with a pointwise nonlinearity

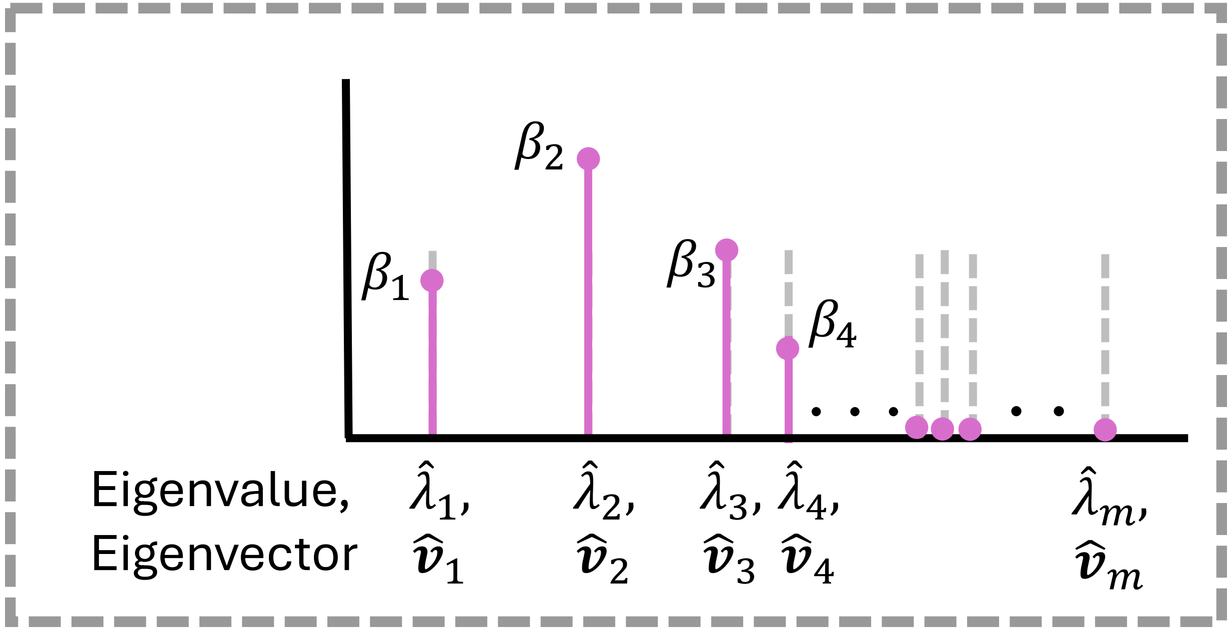
- $\Phi(\mathbf{x}; \hat{\mathbf{C}}, \mathcal{H})$ represents VNN output
- $\mathcal{H}$ is the set of trainable filter taps

Covariance Filters are **Implicitly** Equivalent to PCA

Input data → coVariance filter → Readout function → Inference output



Input data → PCA transform → Regression model → Inference output

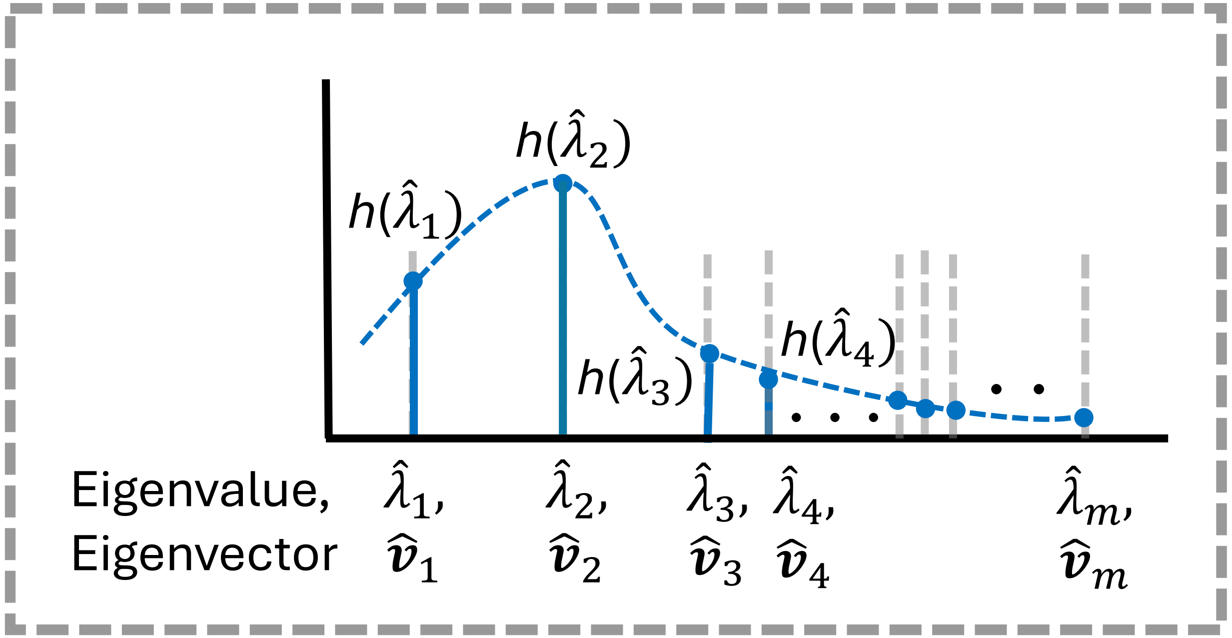


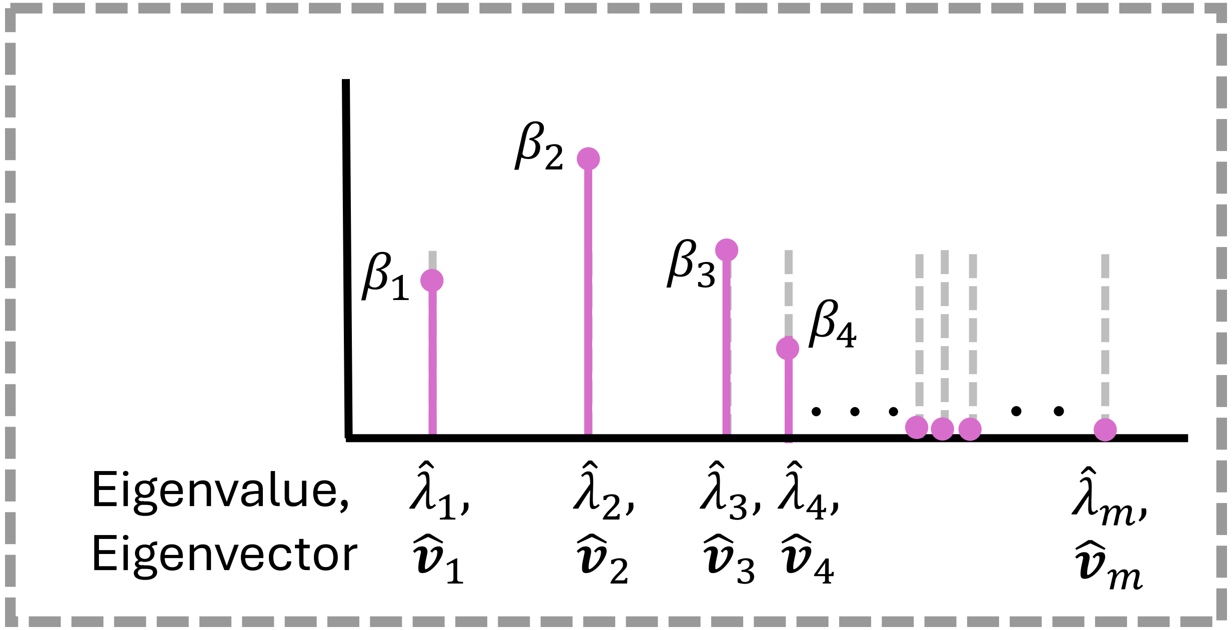
Covariance Filters are **Implicitly** Equivalent to PCA

- The difference is that covariance filters (and VNNs) do not require eigenvectors

  Stability: Leading to more stable signal processing

  Transferability: And the possibility of transferring trained filters across scales



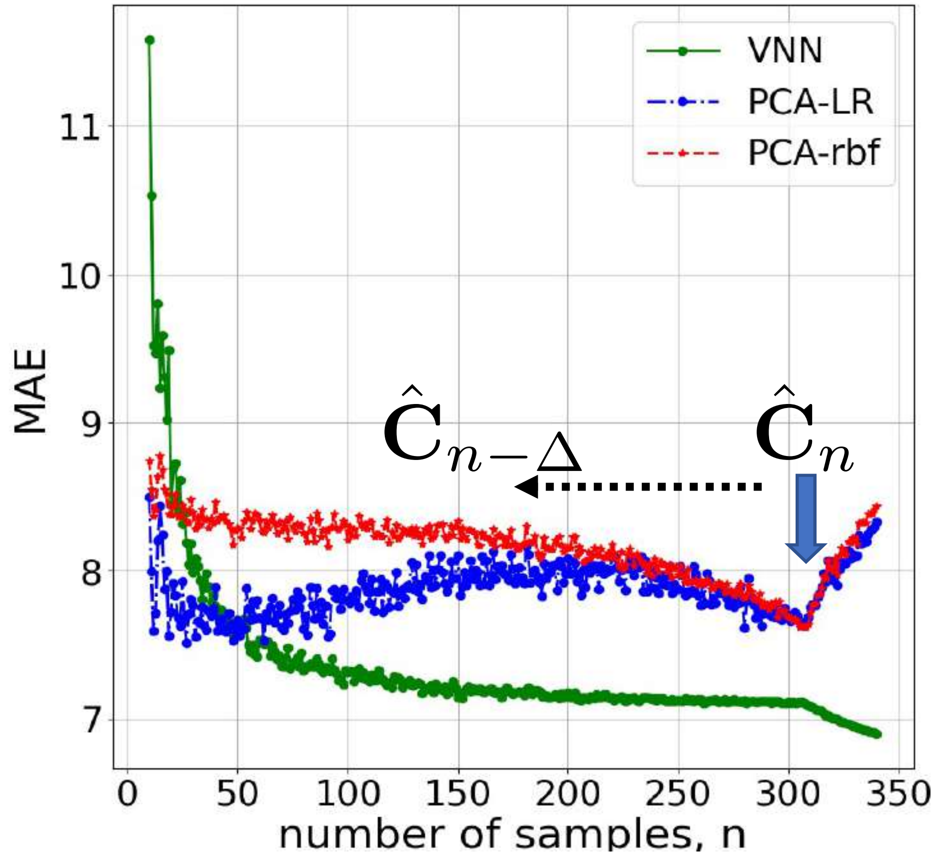


Stable Inference with VNNs

Stability of inference with PCA and VNNs

- PCA-driven inference can be **unstable**
  - stochastic perturbations due to finite sample effect
- VNNs provide **stable** outcomes
  - ⇨ enhanced reproducibility
  - ⇨ avoid overfitting

Performance on regression task



$\hat{\mathbf{C}}_n$: estimated from n samples

Stochastic perturbations in sample covariance matrix

- **Recall**: Sample covariance matrix $\hat{\mathbf{C}}$ is estimate of true covariance matrix $\mathbf{C}$

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathsf{T}} \qquad \mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}}]$$

$\Rightarrow$ eigenvectors/eigenvalues $\hat{\mathbf{V}}, \hat{\boldsymbol{\Lambda}}$ of $\hat{\mathbf{C}}$ are estimates of $\mathbf{V}, \boldsymbol{\Lambda}$ of $\mathbf{C}$

Stochastic perturbations in sample covariance matrix

- **Recall**: Sample covariance matrix $\hat{\mathbf{C}}$ is estimate of true covariance matrix $\mathbf{C}$

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\mathsf{T} \qquad \mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T}]$$

$\Rightarrow$ eigenvectors/eigenvalues $\hat{\mathbf{V}}, \hat{\boldsymbol{\Lambda}}$ of $\hat{\mathbf{C}}$ are estimates of $\mathbf{V}, \boldsymbol{\Lambda}$ of $\mathbf{C}$

- Convergence between $\hat{\mathbf{V}}, \hat{\boldsymbol{\Lambda}}$ and $\mathbf{V}, \boldsymbol{\Lambda}$ [*]

$$\|\hat{\mathbf{V}}\mathbf{x} - \mathbf{V}\mathbf{x}\| = \mathcal{O}\left(\frac{1}{n^{1/2}\min_{i\neq j}|\lambda_i - \lambda_j|}\right)$$

[*] Loukas, Andreas, 2017

Stochastic perturbations in sample covariance matrix

- **Recall**: Sample covariance matrix $\hat{\mathbf{C}}$ is estimate of true covariance matrix $\mathbf{C}$

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathsf{T}} \qquad \mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}}]$$

$\Rightarrow$ eigenvectors/eigenvalues $\hat{\mathbf{V}}, \hat{\boldsymbol{\Lambda}}$ of $\hat{\mathbf{C}}$ are estimates of $\mathbf{V}, \boldsymbol{\Lambda}$ of $\mathbf{C}$

- Convergence between $\hat{\mathbf{V}}, \hat{\boldsymbol{\Lambda}}$ and $\mathbf{V}, \boldsymbol{\Lambda}$ [*]

$$\|\hat{\mathbf{V}}\mathbf{x} - \mathbf{V}\mathbf{x}\| = \mathcal{O}\left(\frac{1}{n^{1/2}\min_{i\neq j}|\lambda_i - \lambda_j|}\right)$$

$\Rightarrow$ **Unstable** PCA transform when eigenvalues of covariance are close

[*] Loukas, Andreas, 2017

Stability of coVariance filter

- How to gauge stability?

$$\mathbf{x} \longrightarrow \boxed{\mathbf{H}(\hat{\mathbf{C}})} \longrightarrow \mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} \qquad \hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathsf{T}}$$

$\Rightarrow$ Output $\mathbf{z}$ must be robust to number of samples n used to estimate $\hat{\mathbf{C}}$

Stability of coVariance filter

- How to gauge stability?

$$\mathbf{x} \longrightarrow \boxed{\mathbf{H}(\hat{\mathbf{C}})} \longrightarrow \mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} \qquad \hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\mathsf{T}}$$

⇨ Output $\mathbf{z}$ must be robust to number of samples n used to estimate $\hat{\mathbf{C}}$

- Compare filter outputs for sample and true covariance matrix

$$\mathbf{x} \longrightarrow \boxed{\mathbf{H}(\hat{\mathbf{C}})} \longrightarrow \mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x} \qquad \mathbf{x} \longrightarrow \boxed{\mathbf{H}(\mathbf{C})} \longrightarrow \mathbf{z} = \mathbf{H}(\mathbf{C})\mathbf{x}$$

⇨ metric of interest: $\|\mathbf{H}(\hat{\mathbf{C}}) - \mathbf{H}(\mathbf{C})\|$

Stability of coVariance filter

$\mathbf{x} \longrightarrow$ $\mathbf{H}(\hat{\mathbf{C}})$ $\longrightarrow \mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x}$

$\mathbf{x} \longrightarrow$ $\mathbf{H}(\mathbf{C})$ $\longrightarrow \mathbf{z} = \mathbf{H}(\mathbf{C})\mathbf{x}$

Stability result [Sihag et al., 2022]

$$\left\| \mathbf{H}(\hat{\mathbf{C}}) - \mathbf{H}(\mathbf{C}) \right\| = \mathcal{O}\left(\frac{1}{n^{1/2-\varepsilon}}\right)$$

coVariance filter output is asymptotically consistent

Stability of coVariance filter

$$\mathbf{x} \longrightarrow \boxed{\mathbf{H}(\hat{\mathbf{C}})} \longrightarrow \mathbf{z} = \mathbf{H}(\hat{\mathbf{C}})\mathbf{x}$$

$$\mathbf{x} \longrightarrow \boxed{\mathbf{H}(\mathbf{C})} \longrightarrow \mathbf{z} = \mathbf{H}(\mathbf{C})\mathbf{x}$$

Stability result [Sihag et al., 2022]

$$\left\| \mathbf{H}(\hat{\mathbf{C}}) - \mathbf{H}(\mathbf{C}) \right\| = \mathcal{O}\left(\frac{1}{n^{1/2-\varepsilon}}\right)$$

coVariance filter output is asymptotically consistent

Assumption.

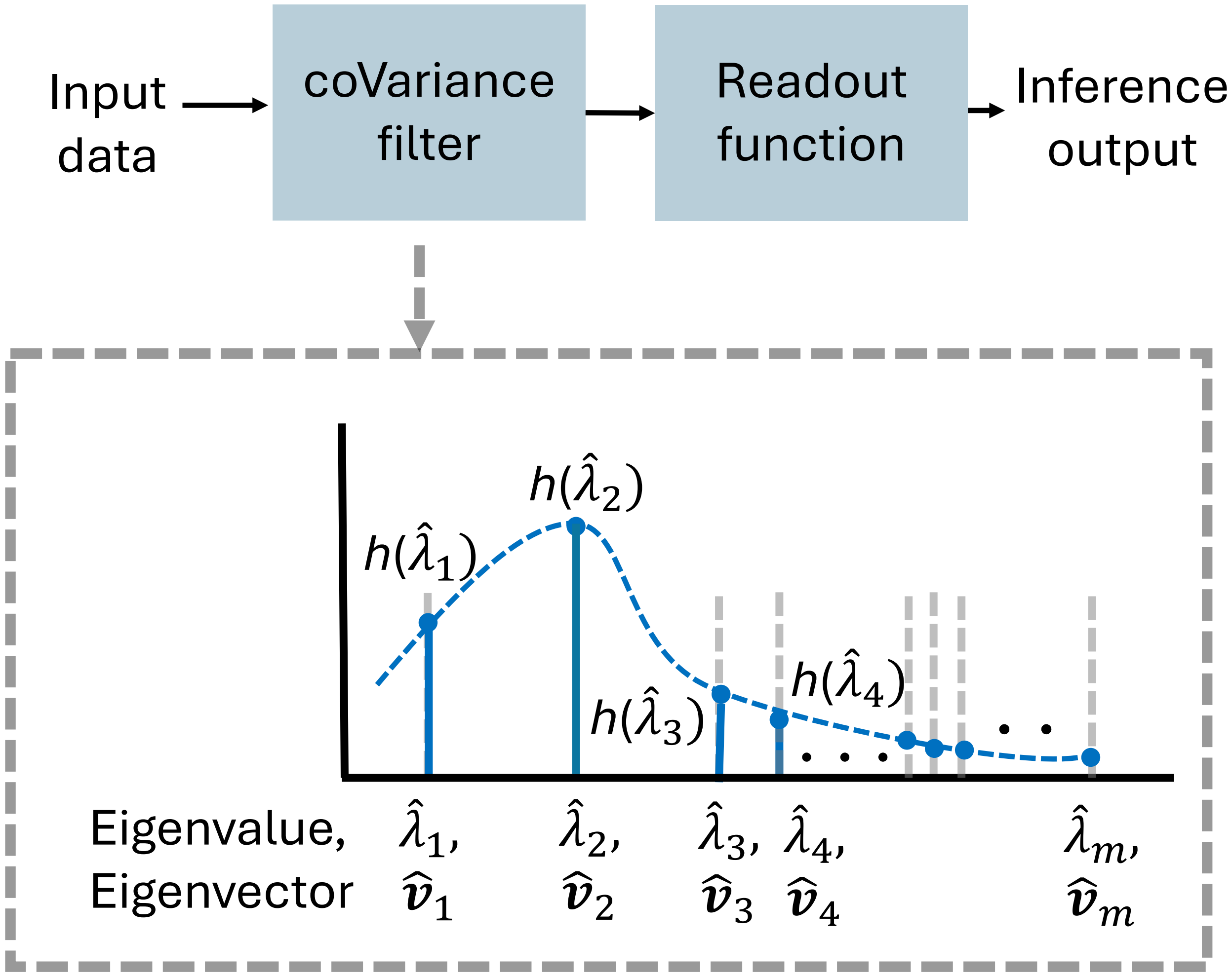
Frequency response of filter $\mathbf{H}(\mathbf{C})$ satisfies

$$|h(\lambda_i) - h(\lambda_j)| \leq Q\frac{|\lambda_i - \lambda_j|}{k_i}$$

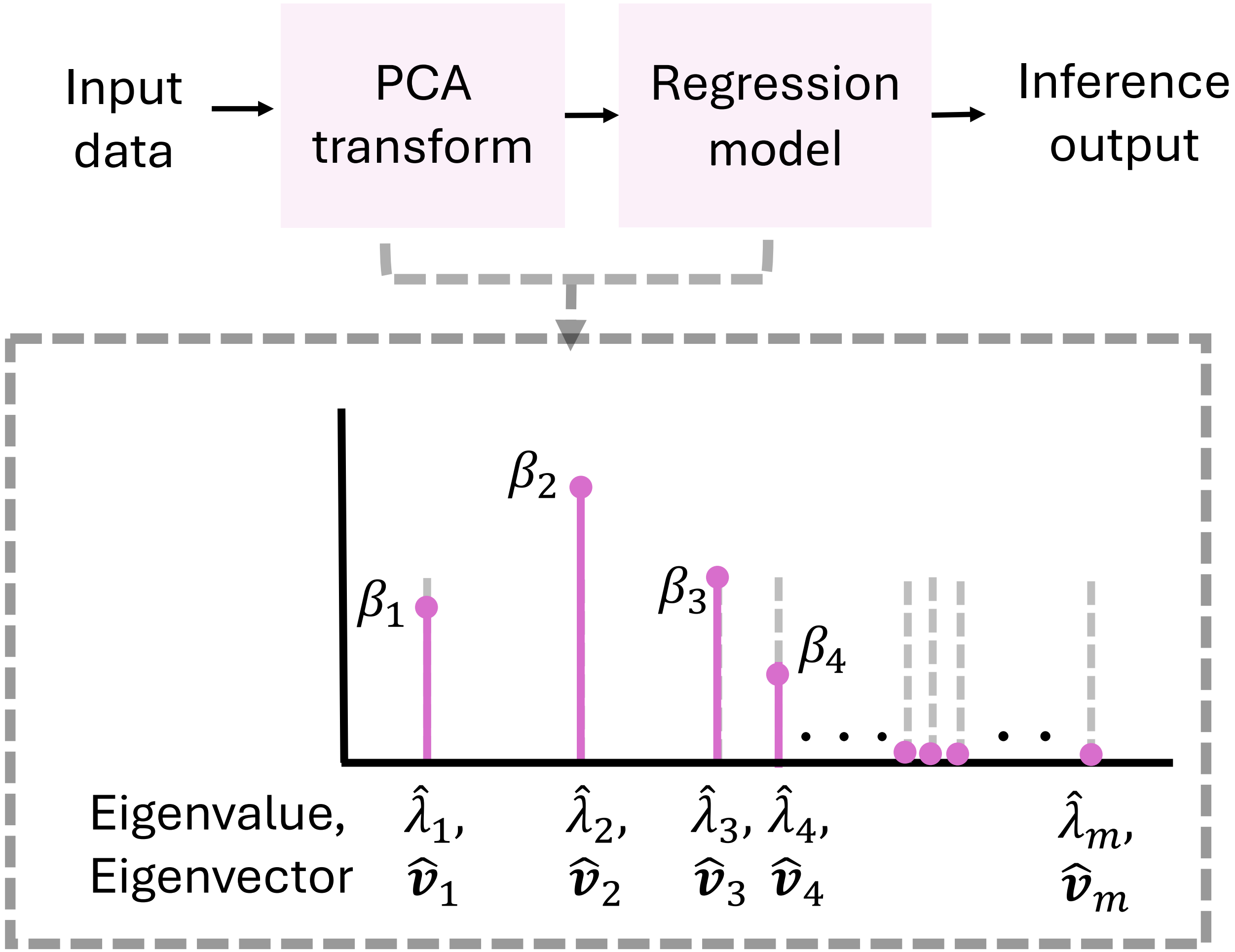
coVariance filter sacrifices discriminability between close eigenvalues for stability

Recall: Learning with coVariance filter versus PCA-based learning

- Learning with a coVariance filter

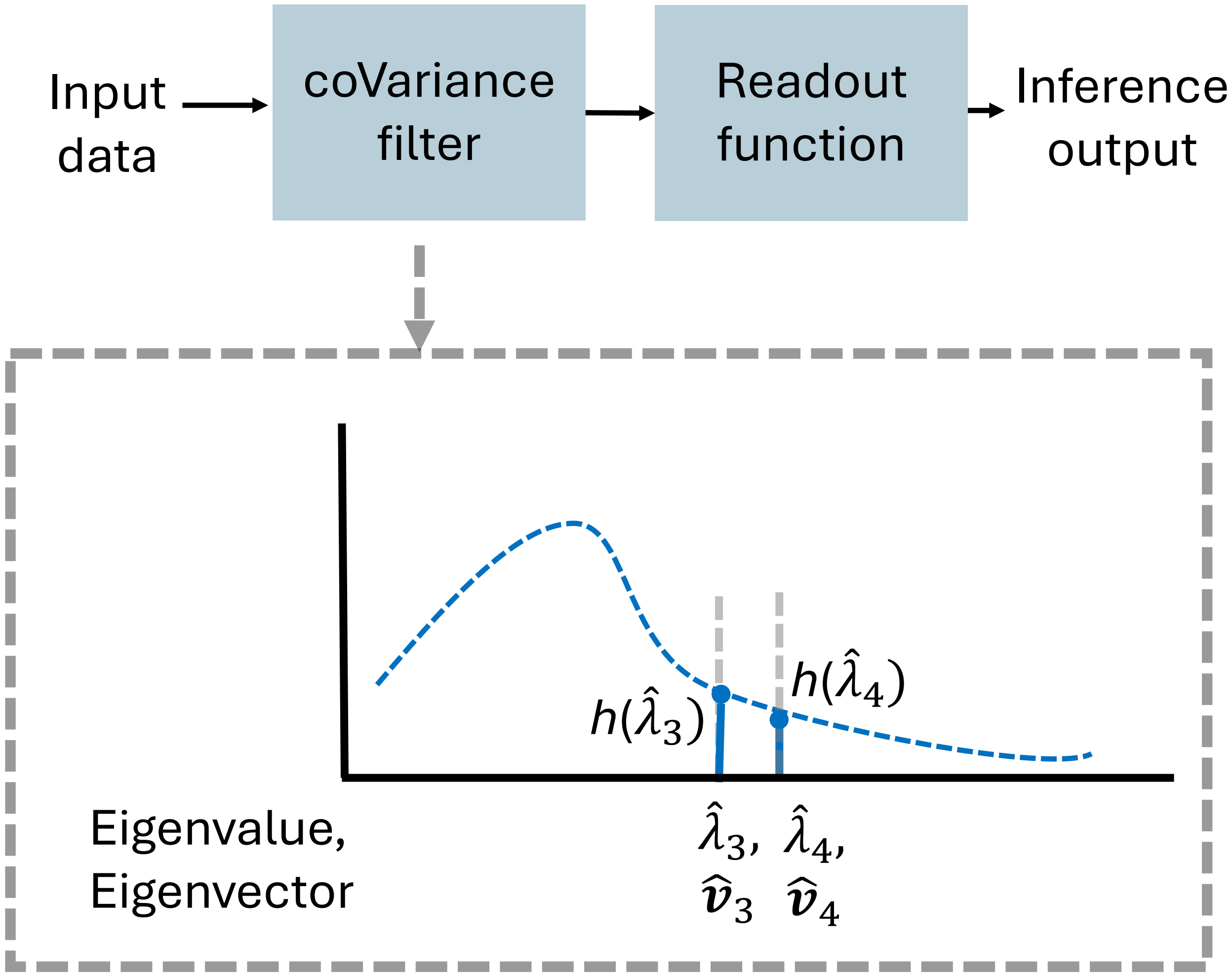


- PCA-based learning

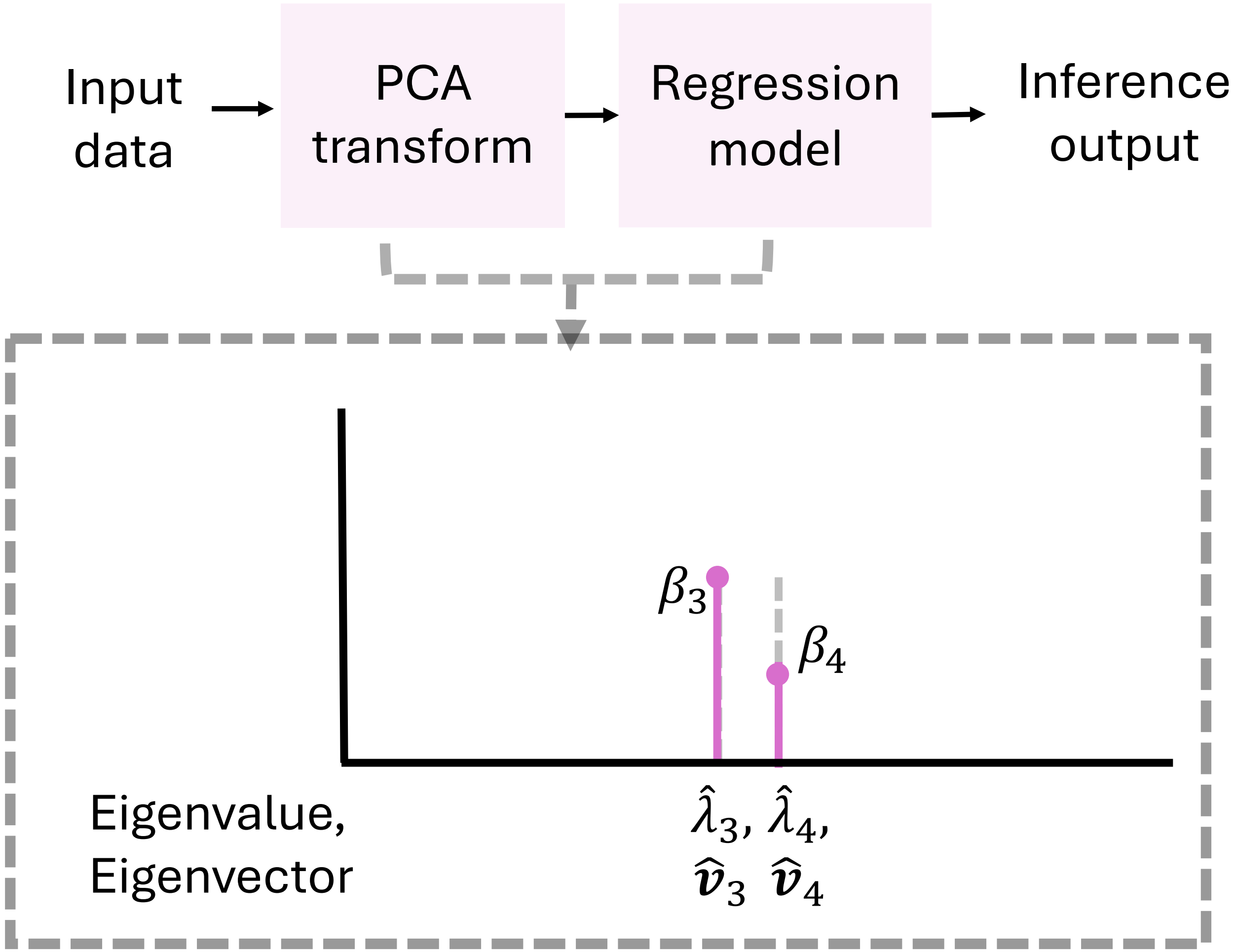


Why is coVariance filter more stable than PCA?

➢ Learning with a coVariance filter

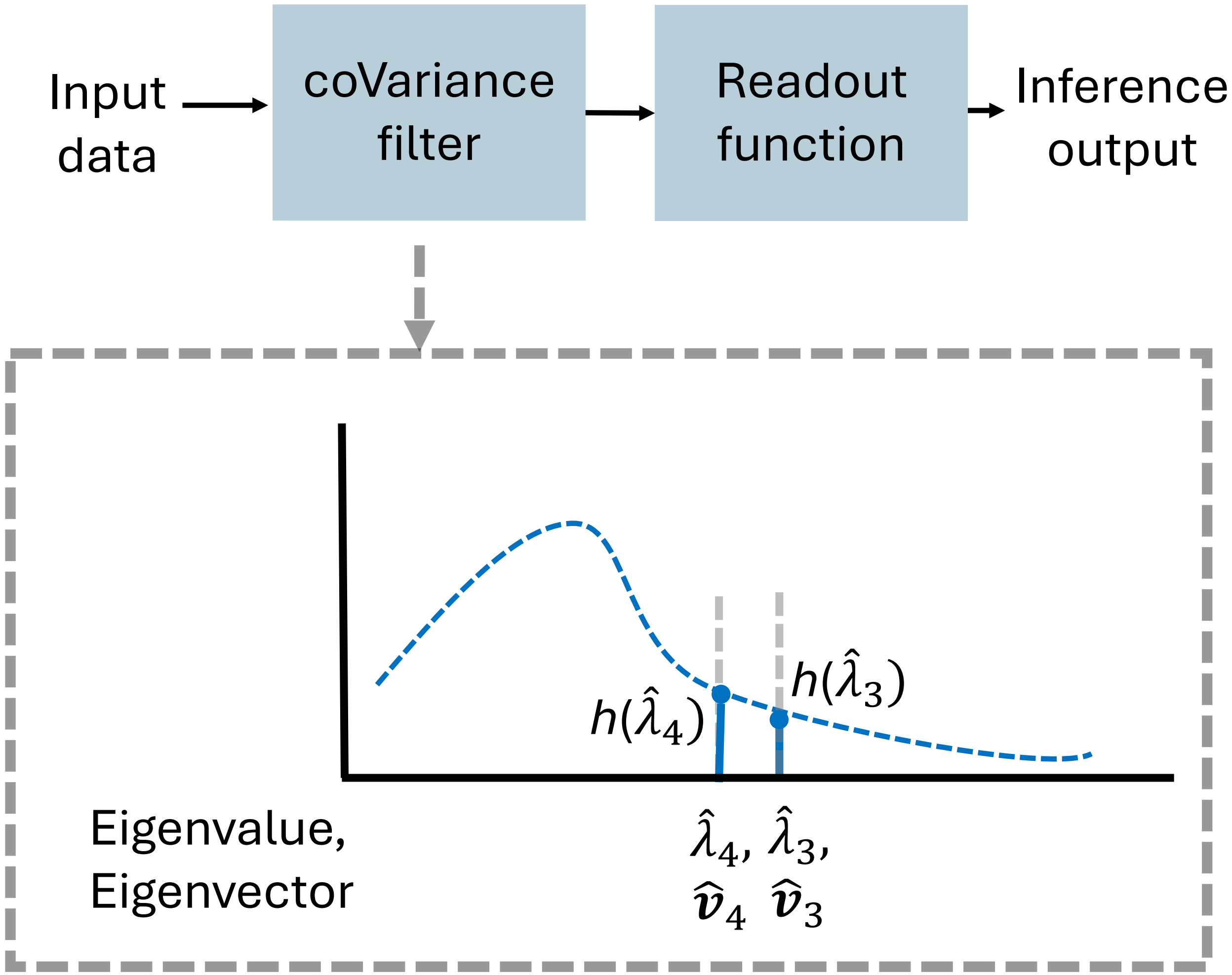


➢ PCA-based learning

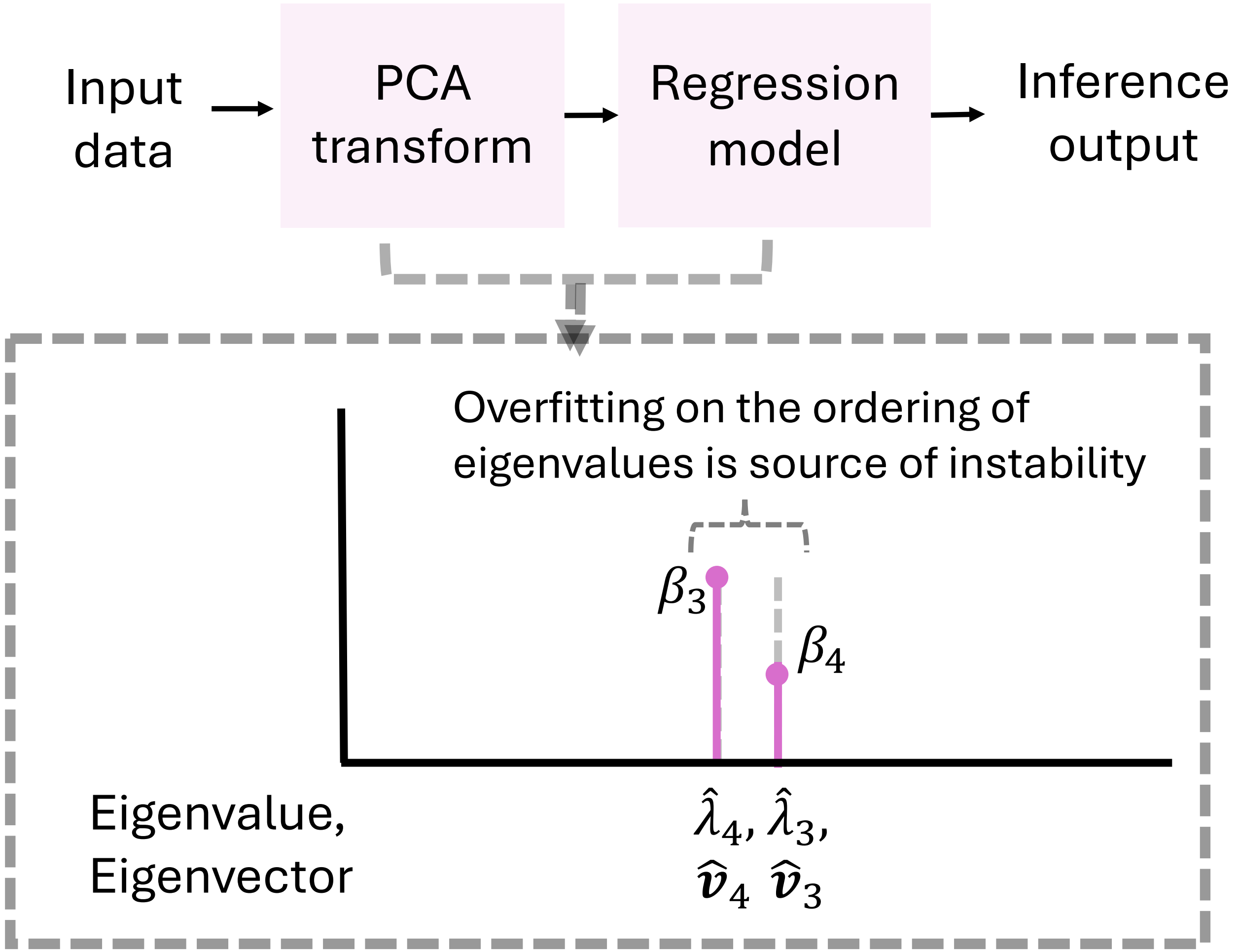


Why is coVariance filter more stable than PCA?

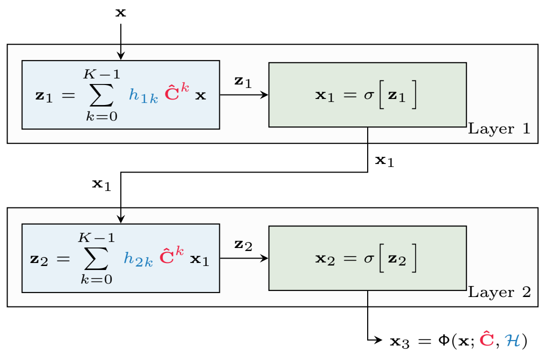
- Learning with a coVariance filter



- PCA-based learning



Stability of VNNs



- VNNs inherit the stability from coVariance filters
  - Stability bound depends on the bound for filters

$$\left\| \mathbf{H}(\hat{\mathbf{C}}) - \mathbf{H}(\mathbf{C}) \right\| = \mathcal{O}\left(\frac{1}{n^{\frac{1}{2}} - \varepsilon} \right) = \alpha_n$$

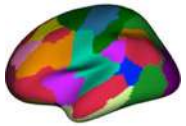
  - For a VNN with L layers and F filters in parallel,

$$\left\| \Phi(\mathbf{x}, \hat{\mathbf{C}}; \mathcal{H}) - \Phi(\mathbf{x}, \mathbf{C}; \mathcal{H}) \right\| \leq LF^{L-1} \alpha_n$$

  - Stability bound increases with number of layers and size of filter banks

Stability of VNNs: Experiments

- Regression task

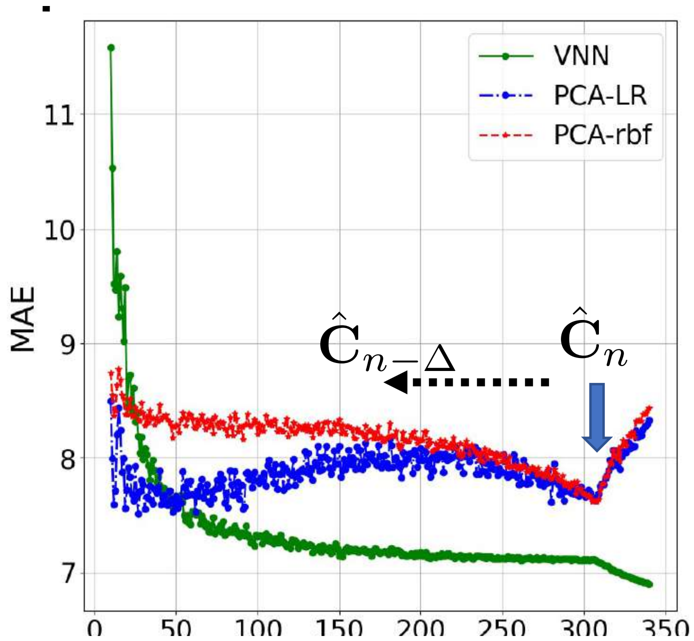


Cortical thickness data → VNN → Estimate of age

- Comparison against PCA-regression

  Data: cortical thickness dataset (m = 104) from (n = 341) human subjects

- **Metric**: MAE (mean absolute error)



VNN: coVariance Neural Network

PCA-LR: PCA-regression with linear kernel

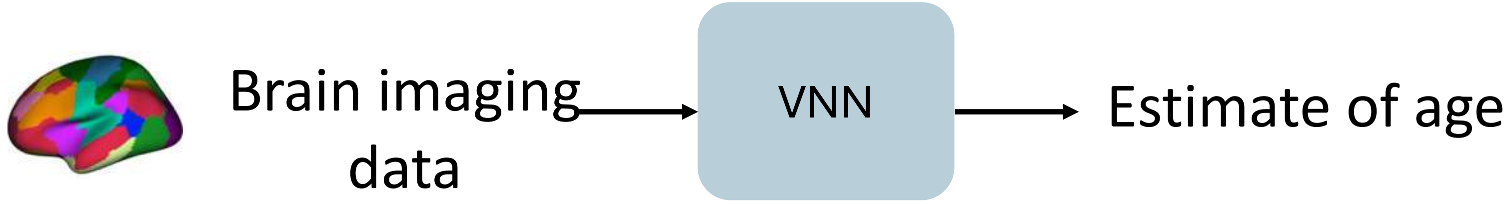
PCA-rbf: PCA regression with rbf kernel

VNN outperforms PCA and is more stable

Transferability of VNNs

Empirical evidence of transferability across multiscale data

- Transferability across multiscale datasets
  - **Multiscale** datasets capture same phenomenon at different scales



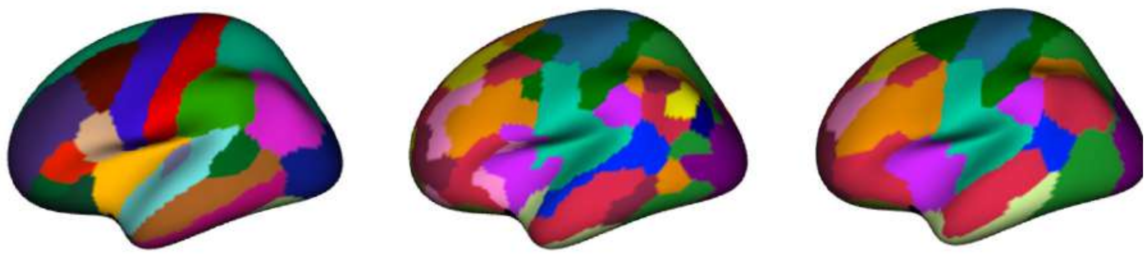
Transferability across datasets with different number of features

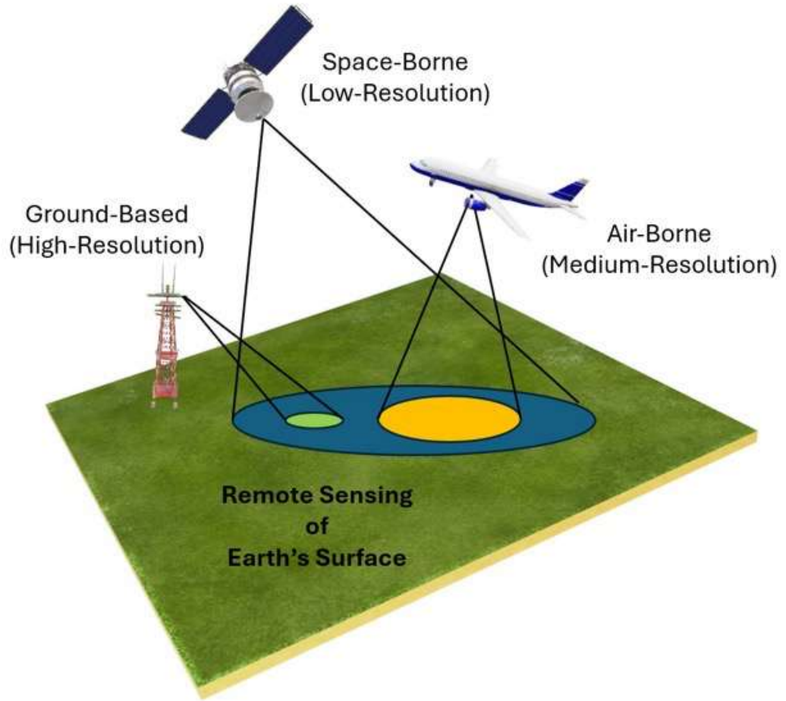
Training \ Testing	 100-feature dataset	 300-feature dataset
 100-feature dataset	5.39 ± 0.084	5.5 ± 0.101

Transferability

- Learning models could generalize to **compatible** datasets
- **compatible**: different dimensionalities and describing the same domain

Remote sensing





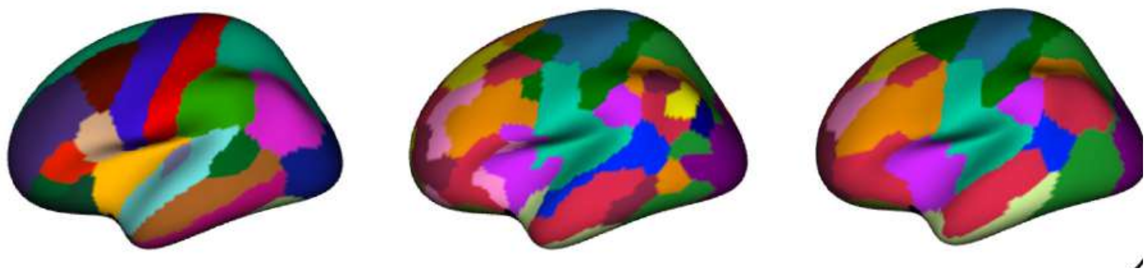
Brain imaging data

Credit: Mustafa Aksoy, UAlbany

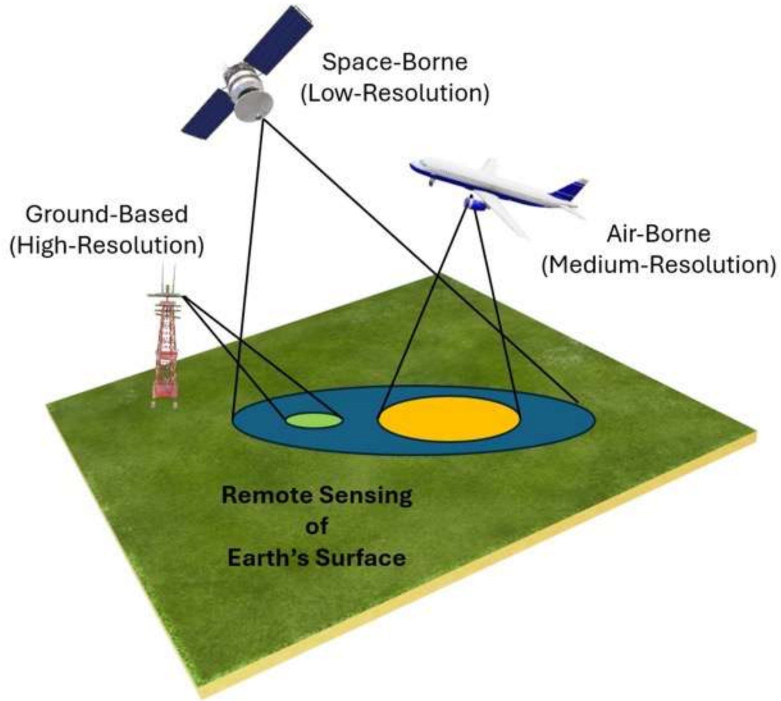
Transferability

- Learning models could generalize to **compatible** datasets
- **compatible**: different dimensionalities and describing the same domain

Remote sensing



Brain imaging data



- **Motivation**: novel metric for generalizability,
  managing high dimensional data…

Credit: Mustafa Aksoy, UAlbany

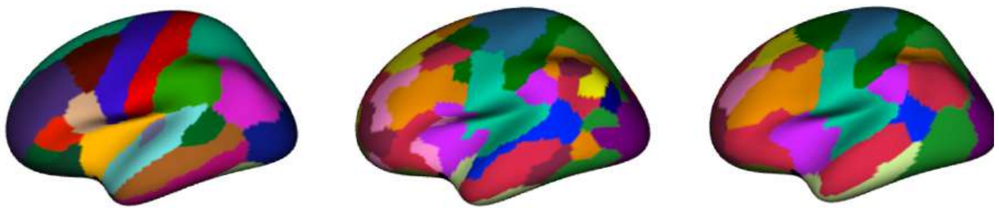
Transferability

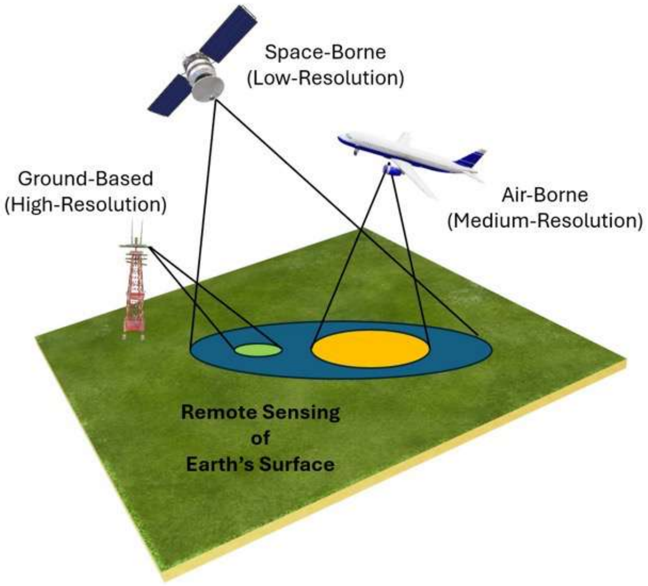
- Most statistical approaches, including PCA, operate within the dimensionality

  ⇨ seamless transference not possible across different dimensionalities

- **This section**: How do VNNs transfer?

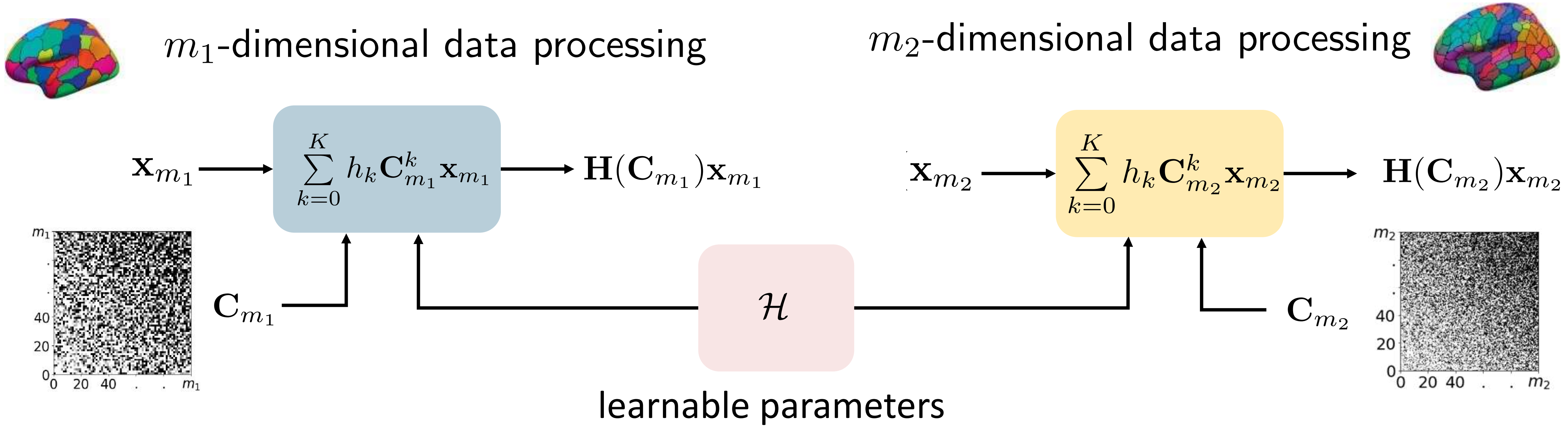
  When is transference successful?





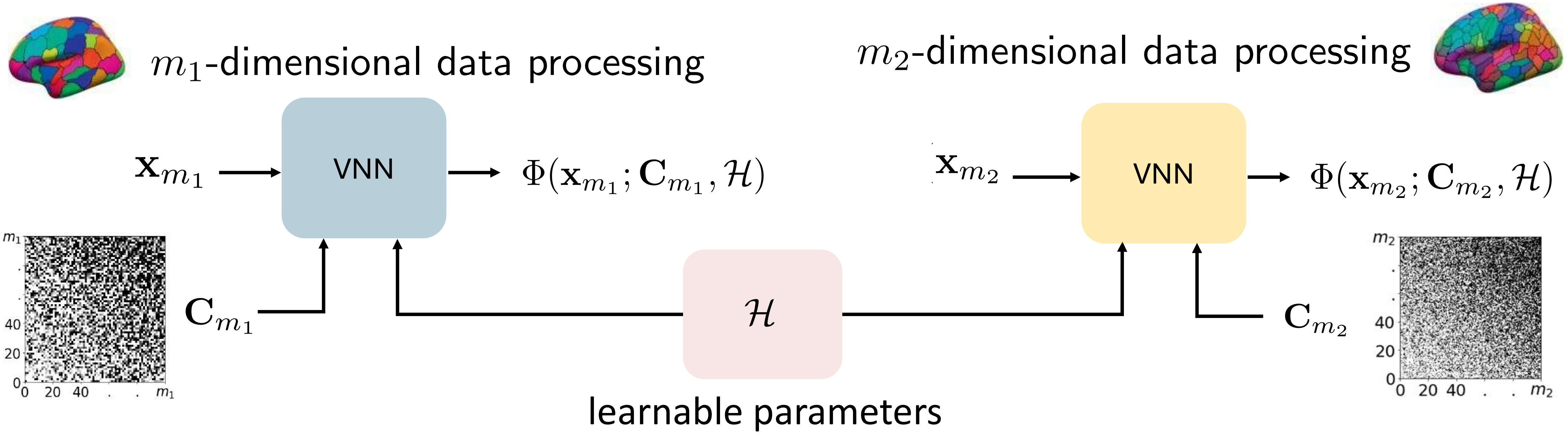
Credit: Mustafa Aksoy, UAlbany

coVariance filters are scale-free models



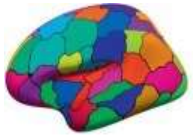
- A coVariance filter $\mathbf{H}(\cdot)$ with scalar filter taps $\{h_k\}$ can process dataset (covariance matrix) of any arbitrary dimensionality: **scale-free model**

VNNs as scale-free models



How to compare $\Phi(\mathbf{x}_{m_1}; \mathbf{C}_{m_1}, \mathcal{H})$ and $\Phi(\mathbf{x}_{m_2}; \mathbf{C}_{m_2}, \mathcal{H})$?

VNNs as scale-free models

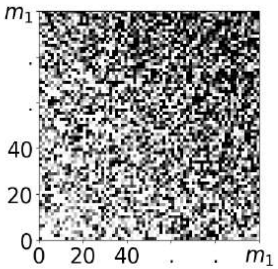


m_1-dimensional data processing

data processing in continuous limit

$\mathbf{x}_{m_1} \longrightarrow$ VNN $\longrightarrow \Phi(\mathbf{x}_{m_1}; \mathbf{C}_{m_1}, \mathcal{H})$

$y_{\mathbf{x}} \longrightarrow$ VNN $\longrightarrow \Phi(y_{\mathbf{x}}; \mathbf{W}, \mathcal{H})$

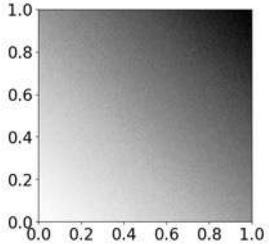


$\mathbf{C}_{m_1}$

$\mathcal{H}$

learnable parameters

$\mathbf{W}$



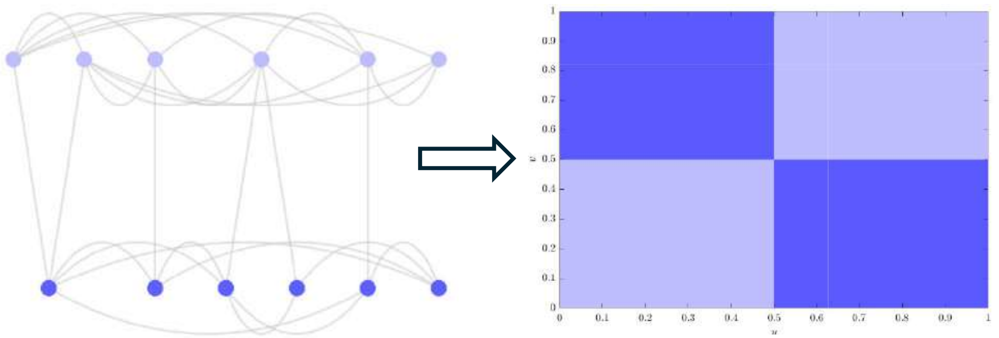
Continuous limit of covariance matrices as $m \to \infty$

How to compare $\Phi(\mathbf{x}_{m_1}; \mathbf{C}_{m_1}, \mathcal{H})$ and $\Phi(y_{\mathbf{x}}; \mathbf{W}, \mathcal{H})$?

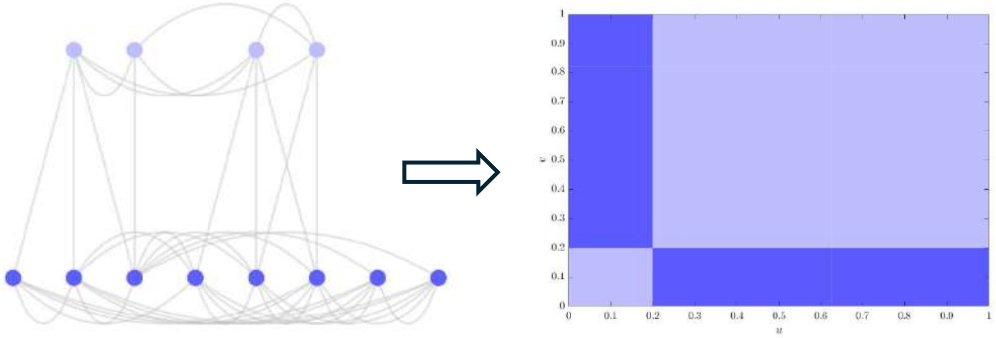
Graphons as continuous limits

- Graphs can have **limit objects** with uncountable number of nodes

- **Example:** Stochastic block models [Ruiz et al., 2021]



Balanced SBM



Unbalanced SBM

Graphons as continuous limits

- **Graphon:** A graphon is a symmetric, bounded measurable function
  - Node labels are graphon arguments $u \in [0,1]$
  - edge weights are graphon values $\mathbf{W}(u, v) = \mathbf{W}(v, u)$

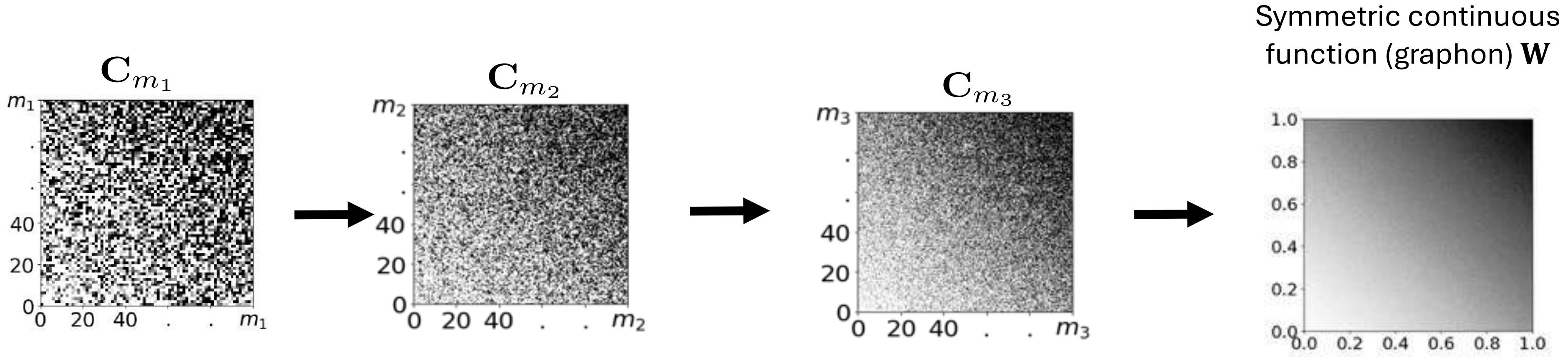
$$\mathbf{W} : [0, 1]^2 \mapsto \mathbb{R}$$

Graphons as continuous limits

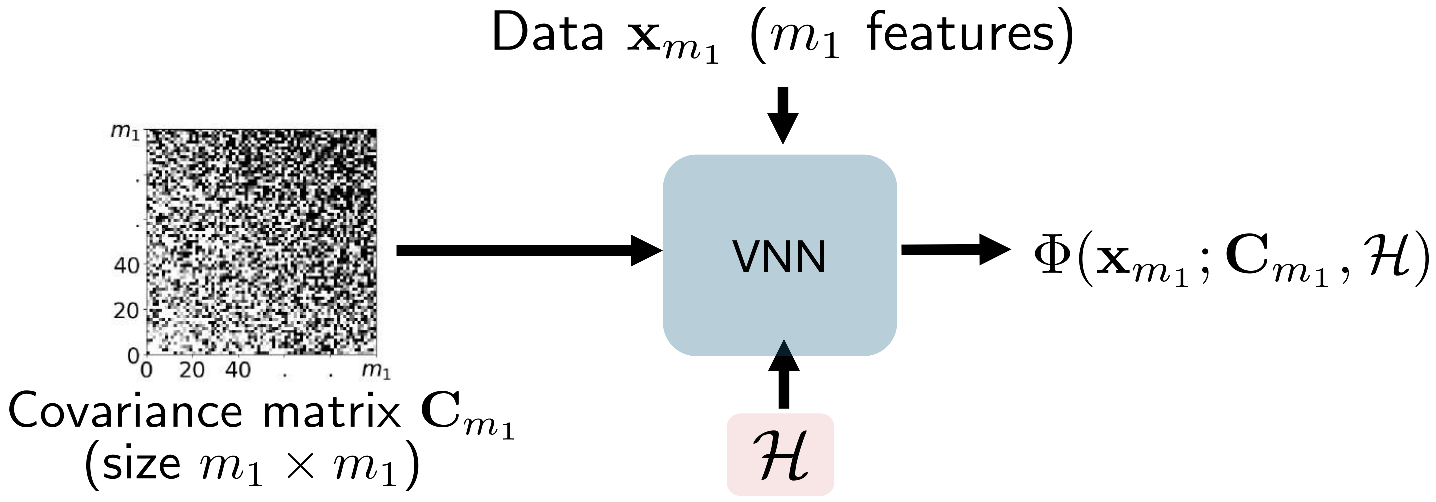
- **Graphon:** A graphon is a symmetric, bounded measurable function
  - Node labels are graphon arguments $u \in [0,1]$
  - edge weights are graphon values $\mathbf{W}(u,v) = \mathbf{W}(v,u)$

$$\mathbf{W} : [0,1]^2 \mapsto \mathbb{R}$$

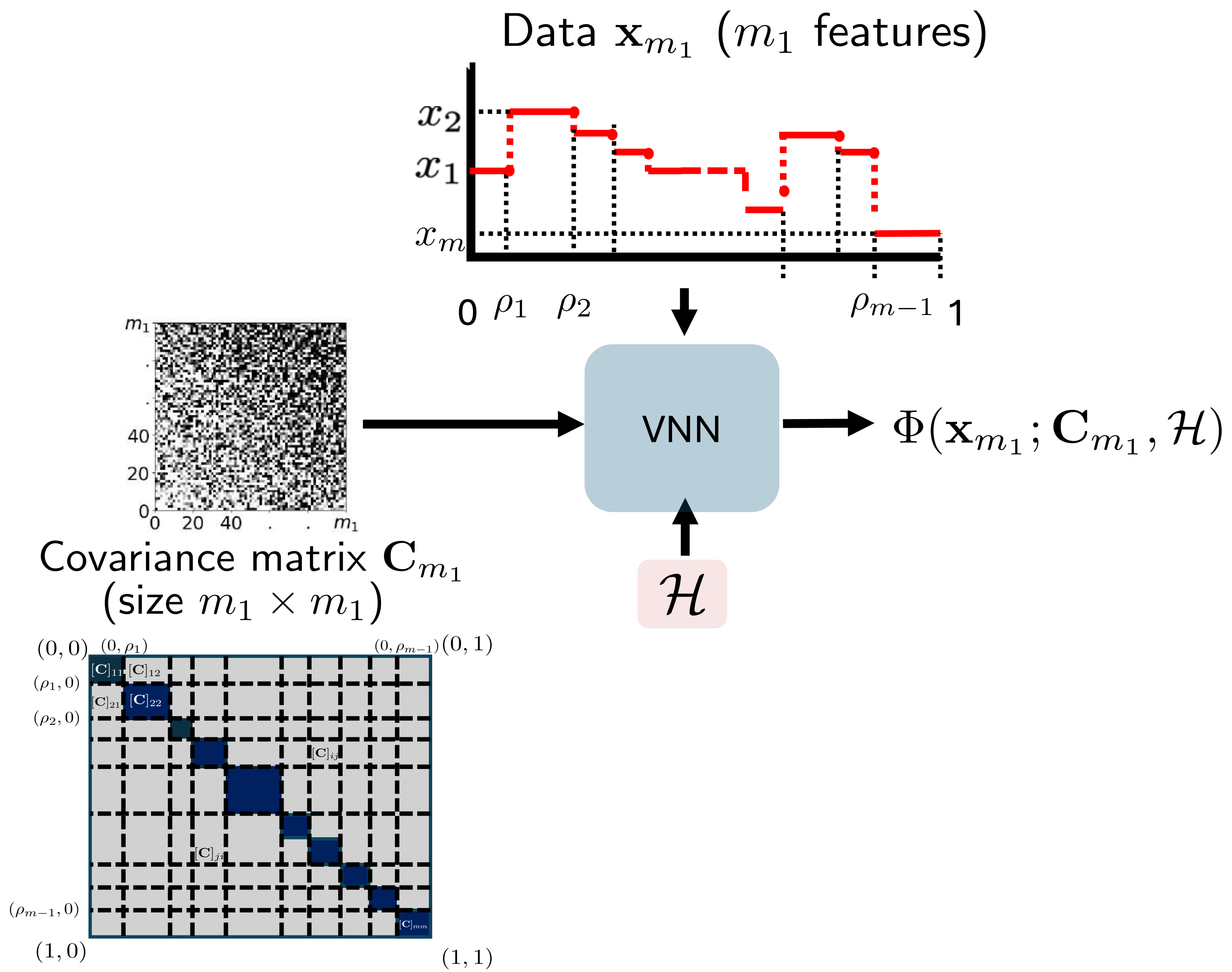
- Transferability when covariance matrix is part of some converging sequence



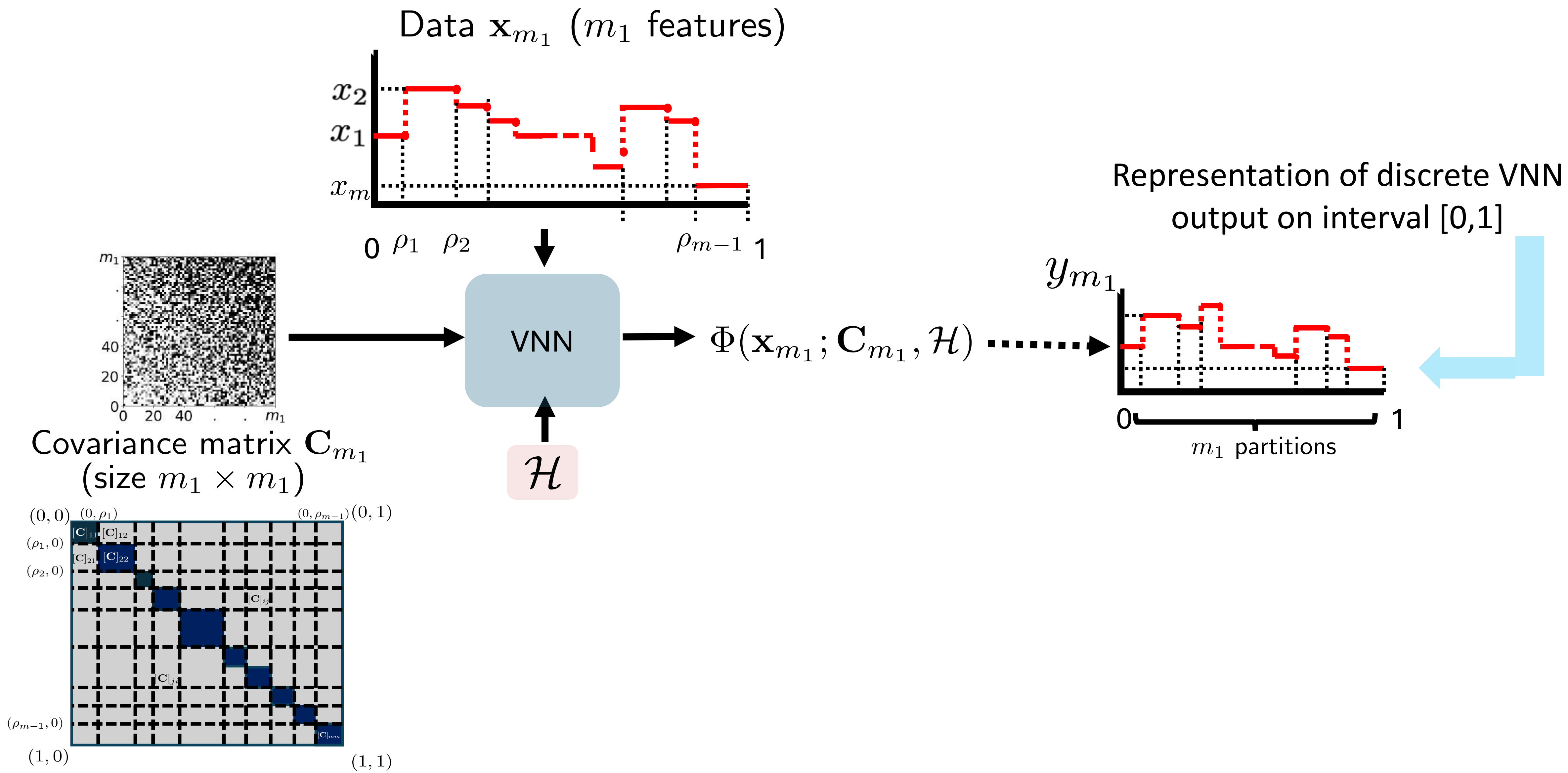
Redefining VNNs in continuous domain



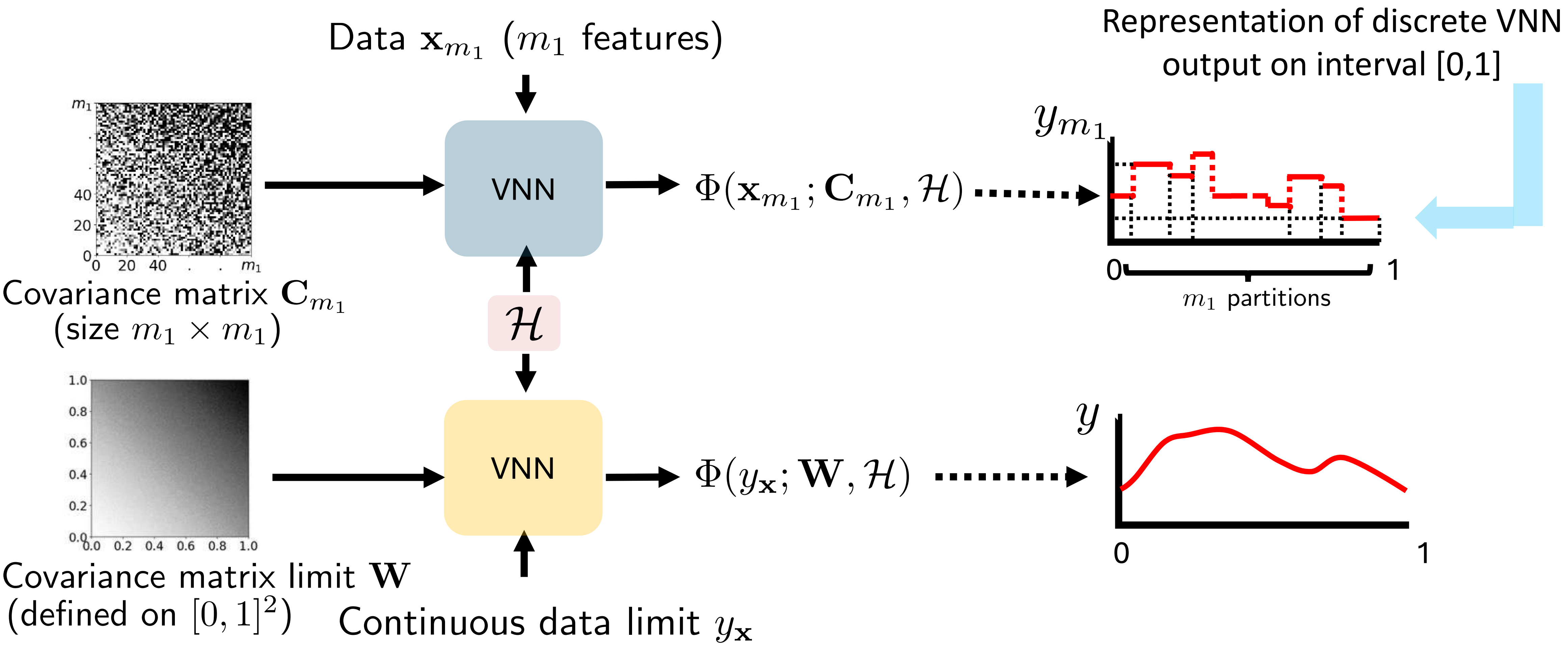
Redefining VNNs in continuous domain



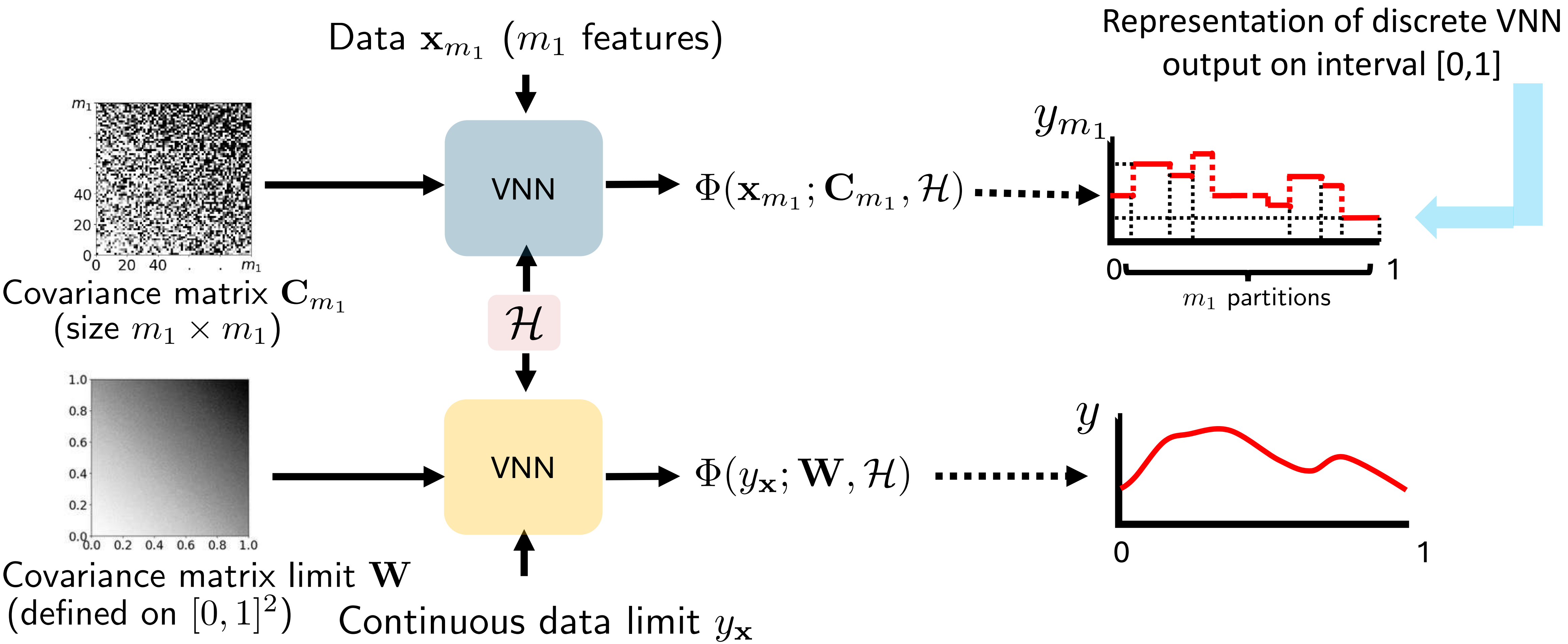
Redefining VNNs in continuous domain



Problem formulation for transferability

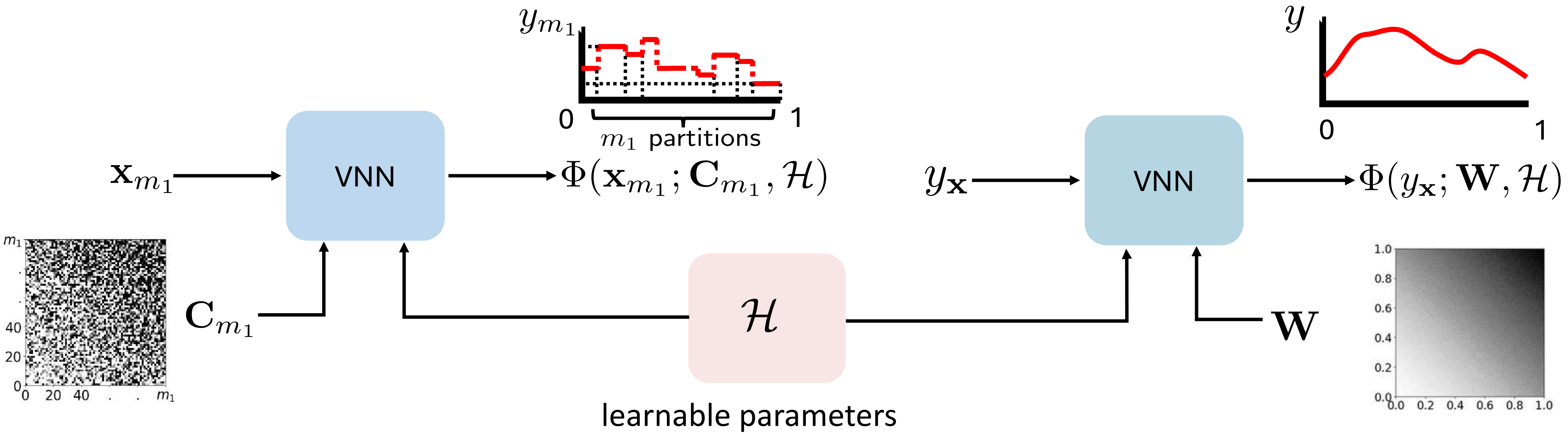


Problem formulation for transferability



Find ϑ, such that, $\|y_{m_1} - y\|_2 \leq \vartheta$

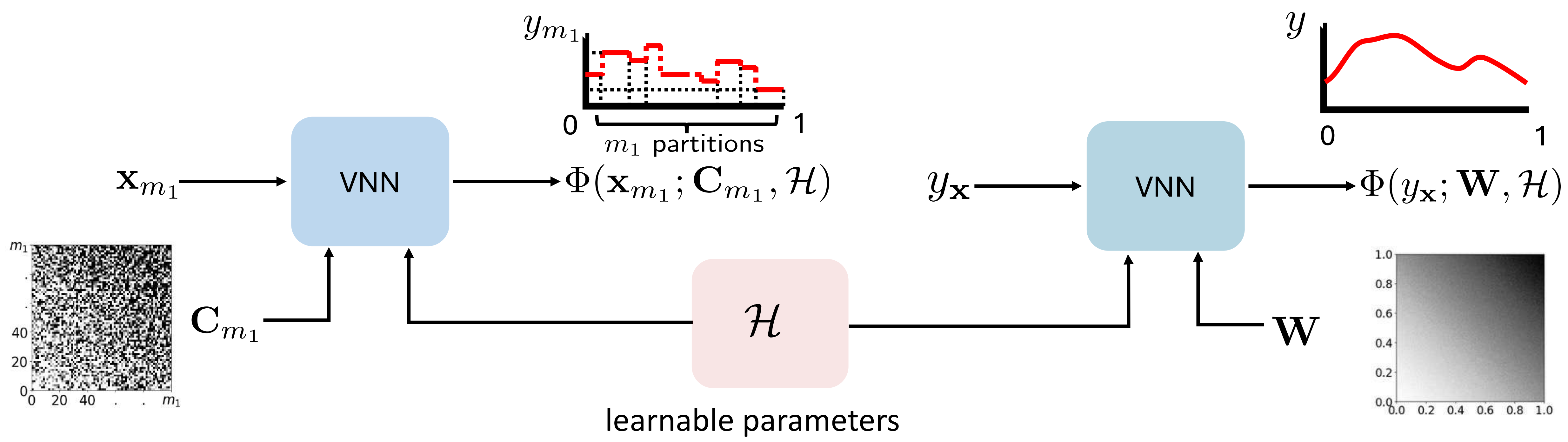
VNNs are provably transferable



Transferability bound* [Sihag et al., 2024]

$$\|y_{m_1} - y\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}}\right) \text{, for } \zeta \in (2/3, 1]$$

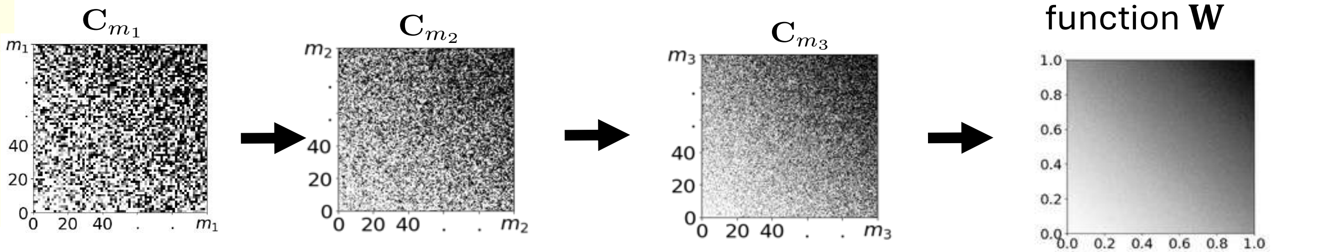
VNNs are provably transferable



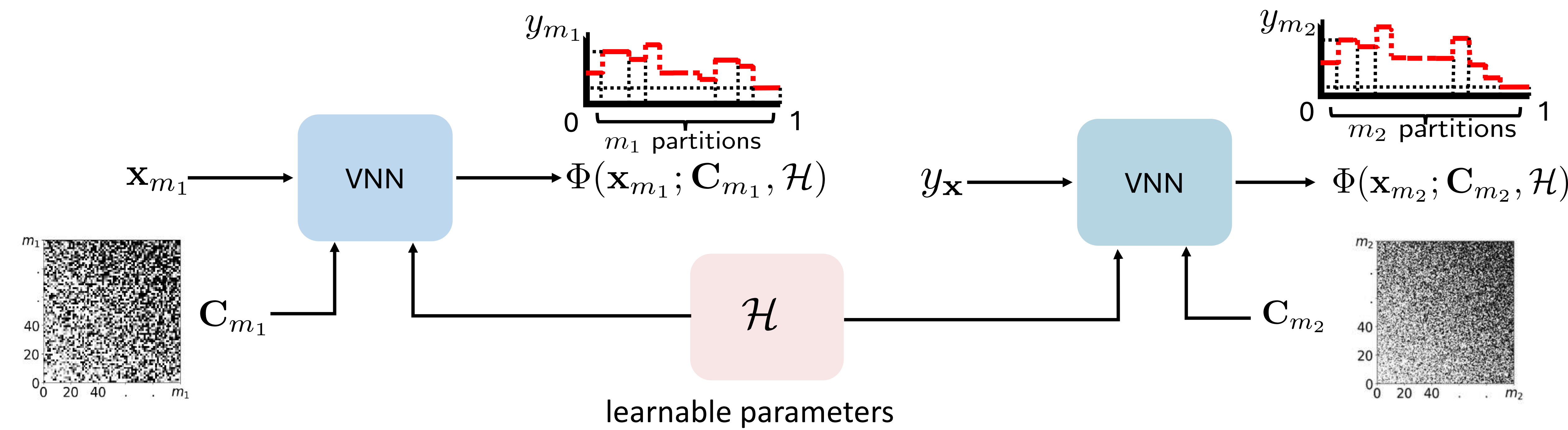
Transferability bound* [Sihag et al., 2024]

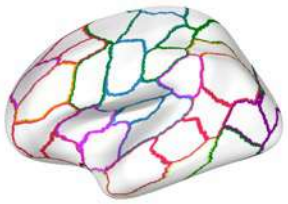
$$\|y_{m_1} - y\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}}\right), \text{ for } \zeta \in (2/3, 1]$$

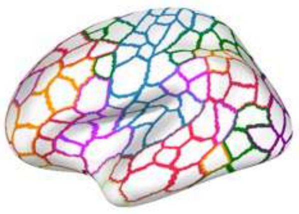
***Assumption**: data is a discretization of a common continuous model



VNNs are provably transferable





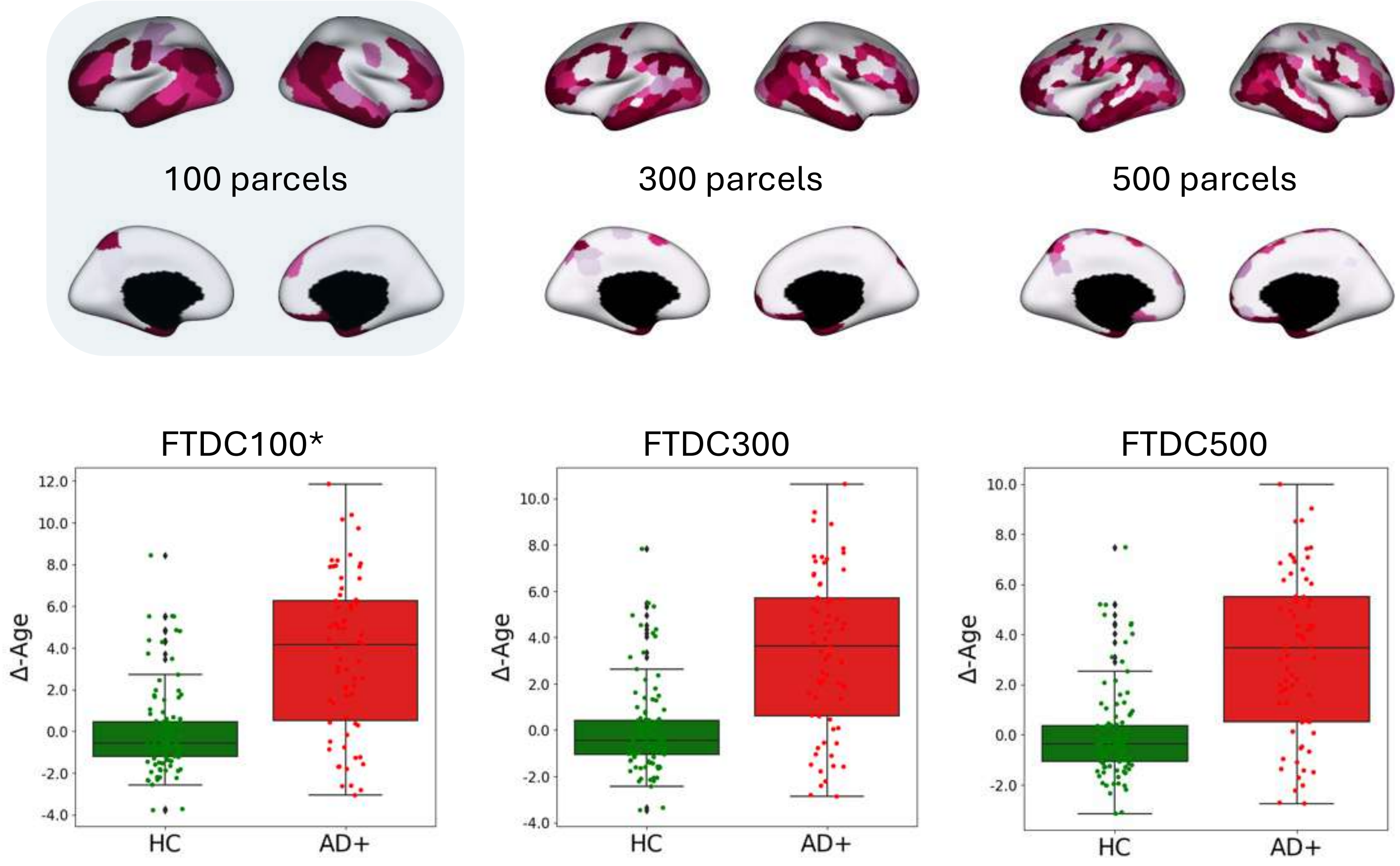


Transferability bound

$$\|y_{m_1} - y_{m_2}\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}} + \frac{1}{m_2^{3\zeta/2-1}}\right), \text{ for } \zeta \in (2/3, 1]$$

Experiments

Objective: Brain age gap prediction in HC (healthy) and AD+ (Alzheimer’s) cohorts from VNNs trained on 100-feature dataset [Sihag et al., NeurIPS, 2024, JSTSP 2024, SPM 2025]



- ROIs contributing to elevated brain age gap in AD+ across different resolutions
- Brain age gap is elevated in AD+ w.r.t HC cohort in 100-feature dataset
- Results on brain age gap retained after transferring VNN to 300 and 500-feature datasets

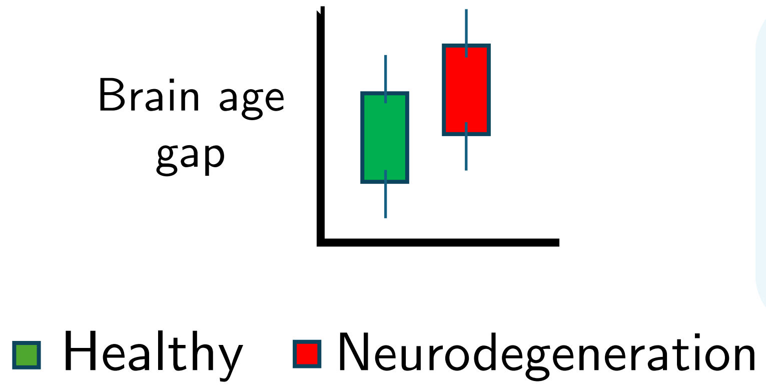
Principled brain age gap prediction with VNNs

Brain age gap

- Individual rate of “aging” is different from chronological rate of aging
  - Driven by environment, genetics, neurodegeneration
- Brain age provides a biological estimate brain age, derived from **neuroimaging**

Brain age gap

- Individual rate of “aging” is different from chronological rate of aging
  - Driven by environment, genetics, neurodegeneration
- Brain age provides a biological estimate brain age, derived from **neuroimaging**
- The brain age gap is the **deviation** between brain age and chronological age

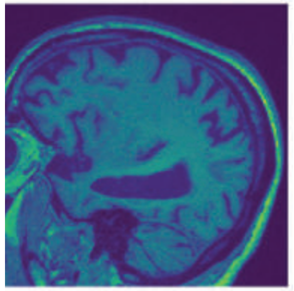


Brain age gap $\propto$ individual risks for neurological, neuropsychiatric and neurodegenerative diseases

Neurodegeneration (in terms of cortical atrophy)

- Neurodegeneration is **accelerated decline** of structure or function of the brain
- **Cortical atrophy**: reduction in cortical **thickness**/volume/area

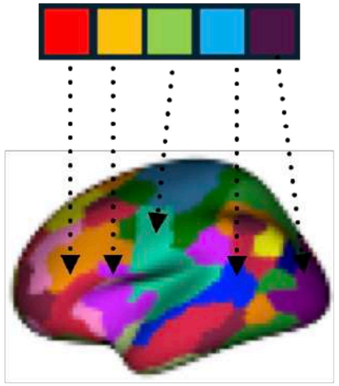
(characteristic of healthy aging and disorders like Alzheimer's disease (AD), frontotemporal dementia (FTD), etc.)



MRI scan

Preprocessing
(e.g., Freesurfer)

$\mathbf{x} = [x_1, \ldots, x_m]$

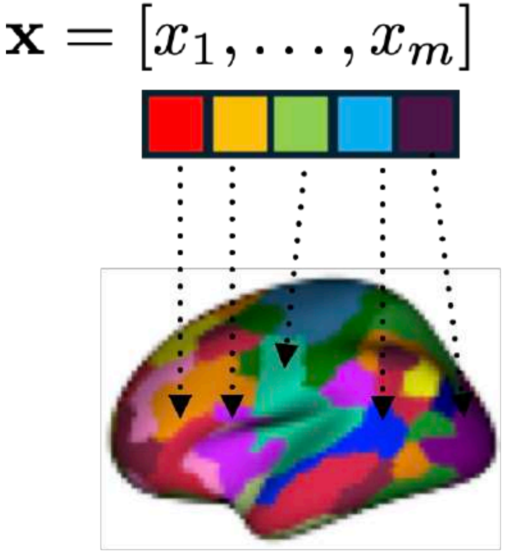


Cortical thickness features

x_i is cortical thickness for brain region i

Neuroimaging Data: Basics

- Data sample corresponds to measurement associated with brain (cortical) surface



Anatomic features

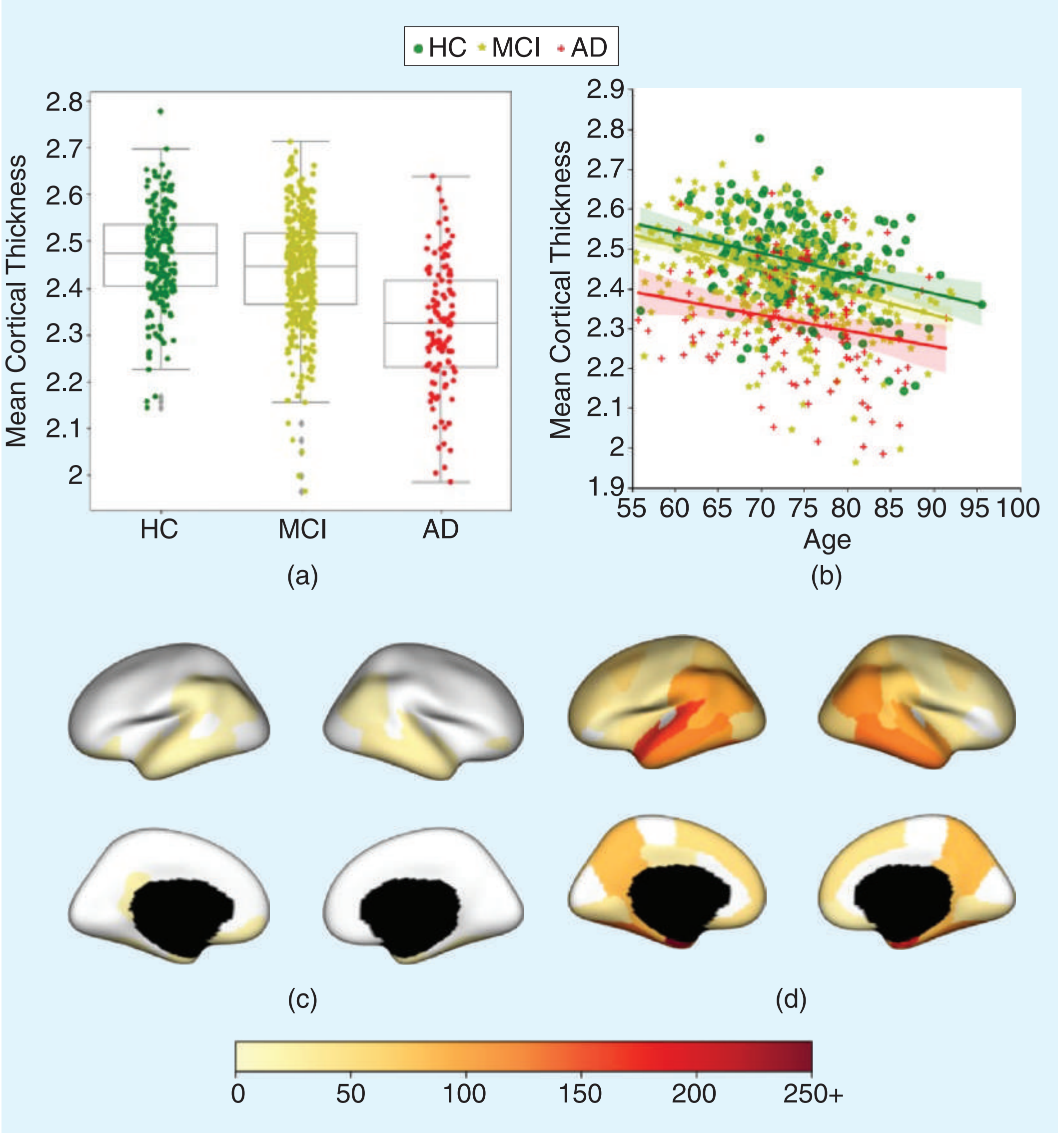
- Brain surface is divided according to **brain atlases**

  ⇨ datasets may have **distinct** dimensionalities

- **Multi-resolution** brain atlas discretizes brain surface at multiple resolutions (for e.g., Schaefer's atlas has resolutions 100-1000)

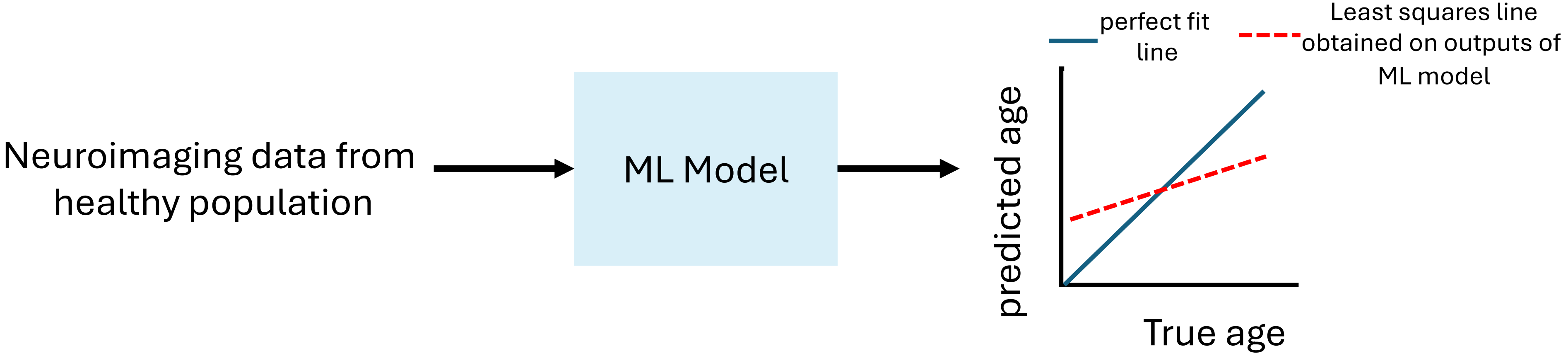
Case study (Neurodegeneration)

- **Data:** cortical thickness from 3 cohorts
  - HC (healthy)
  - MCI (Mild cognitive impairment)
  - AD (Alzheimer's disease)
- Larger **cortical atrophy** is feature of AD
- MCI is precursor to AD
  ⇨ shows intermediate cortical atrophy between HC and AD
- **Aging** also leads to cortical atrophy



Brain age gap evaluation using ML

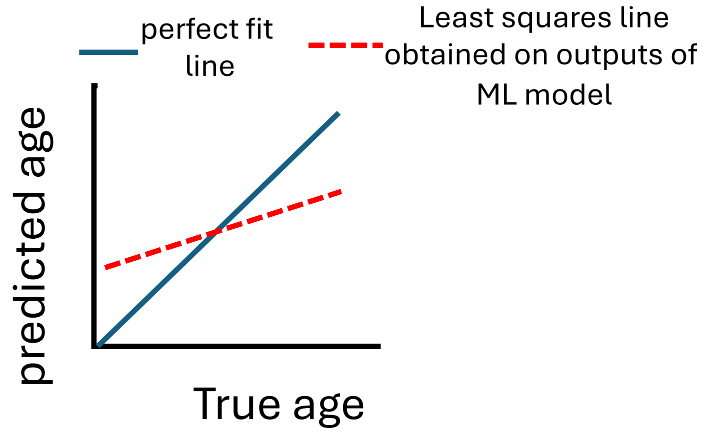
Step 1. Train ML model to predict chronological age for healthy controls from cortical thickness features



Brain age gap evaluation using ML

Step 1. Train ML model to predict chronological age for healthy controls from cortical thickness features

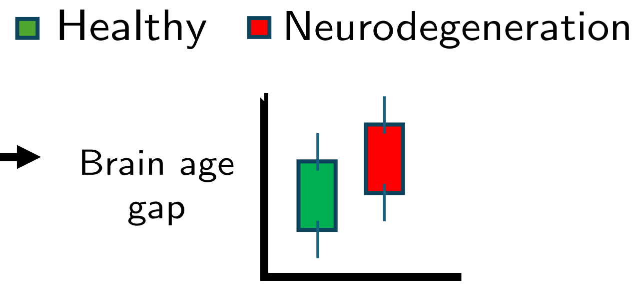
Neuroimaging data from healthy population → ML Model →



Step 2. Linear regression-based age-bias correct for outputs of ML model

Step 3. Obtain brain age gap for healthy controls and individuals with neurodegenerative condition.

Data from all cohorts → ML Model → Age-bias correction → Brain age gap = Bias-corrected ML model output – True age →



Brain age gap prediction is a transfer learning problem

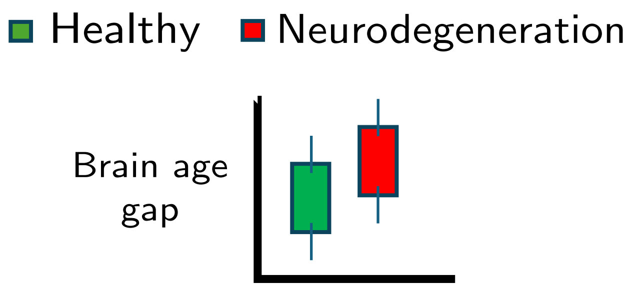
- ➢ Train ML model to predict age on a large dataset (healthy population)

Pre-training

Neuroimaging data from healthy population → ML Model → predicted age

- ➢ Apply the **pre-trained** ML model on a target dataset (neurodegeneration)

Data from all cohorts → ML Model →

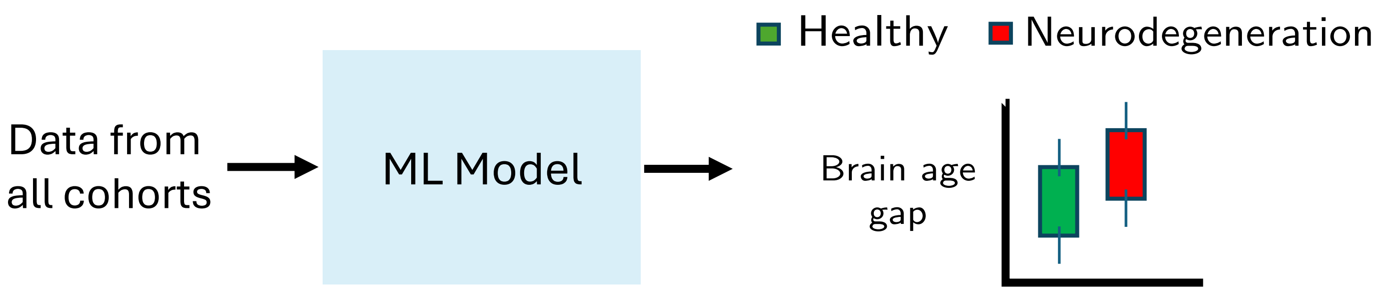


- ➢ Brain age gap is the **residual** of the model

Brain age gap prediction is a transfer learning problem

➢ **Some observations about a meaningful brain age gap**

- We expect model performance to **degrade** in target population
  - ✓ Degradation in performance (residuals) must be in a **specific direction**
  - ✓ Degradation in performance (residuals) $\propto$ **disease severity/status**

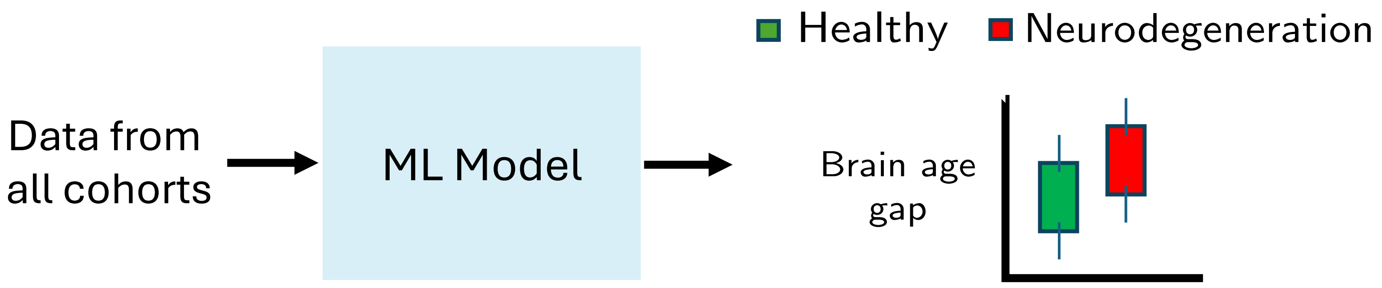


Choice of learning parametrization

- Choice of ML model dictates how data is leveraged to gauge brain age gap

- Prevalent approaches focus on achieving perfect pre-training performance
  - **Performance-driven approaches**

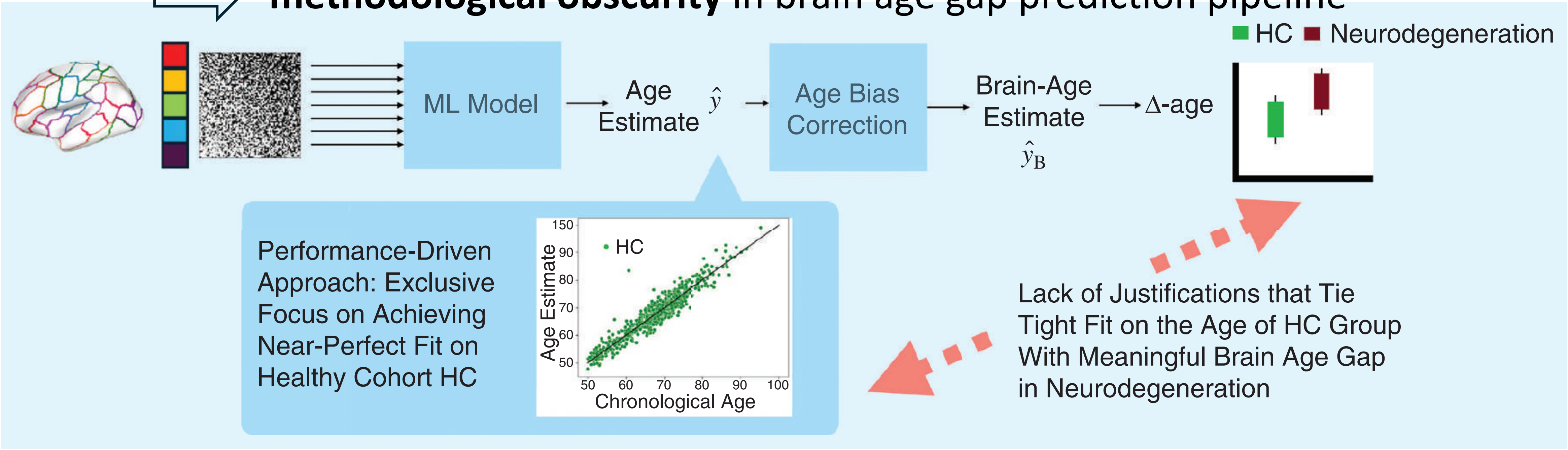
- **Performance-driven approaches** do not guarantee `**meaningful**' brain age gap



Choice of learning parametrization

- Neural networks are prevalent in performance-driven approaches
- A Neural Network may **not be interpretable** and prone to **overfitting**

⇨ **methodological obscurity** in brain age gap prediction pipeline



Choice of learning parametrization

- Neural networks are prevalent in performance-driven approaches

- A Neural Network may **not be interpretable** and prone to **overfitting**

$\Rightarrow$ **methodological obscurity** in brain age gap prediction pipeline

Performance in pre-training does not dictate **meaningful residuals** in target population

A principled approach to brain age gap prediction

- **Focus on residuals** of the ML model, not prediction performance

- **Qualitative evaluation** during pre-training

  what does the model learn during pre-training on healthy population?

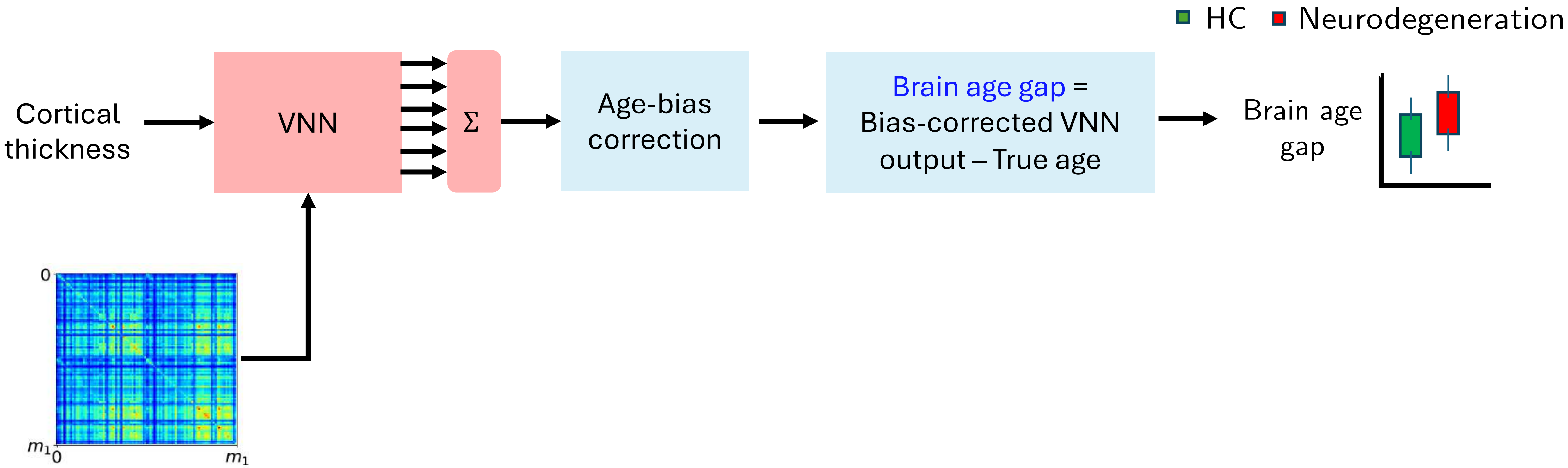
- **Interpretability/explainability:**

  what's driving elevated brain age gap (residuals) in neurodegeneration?

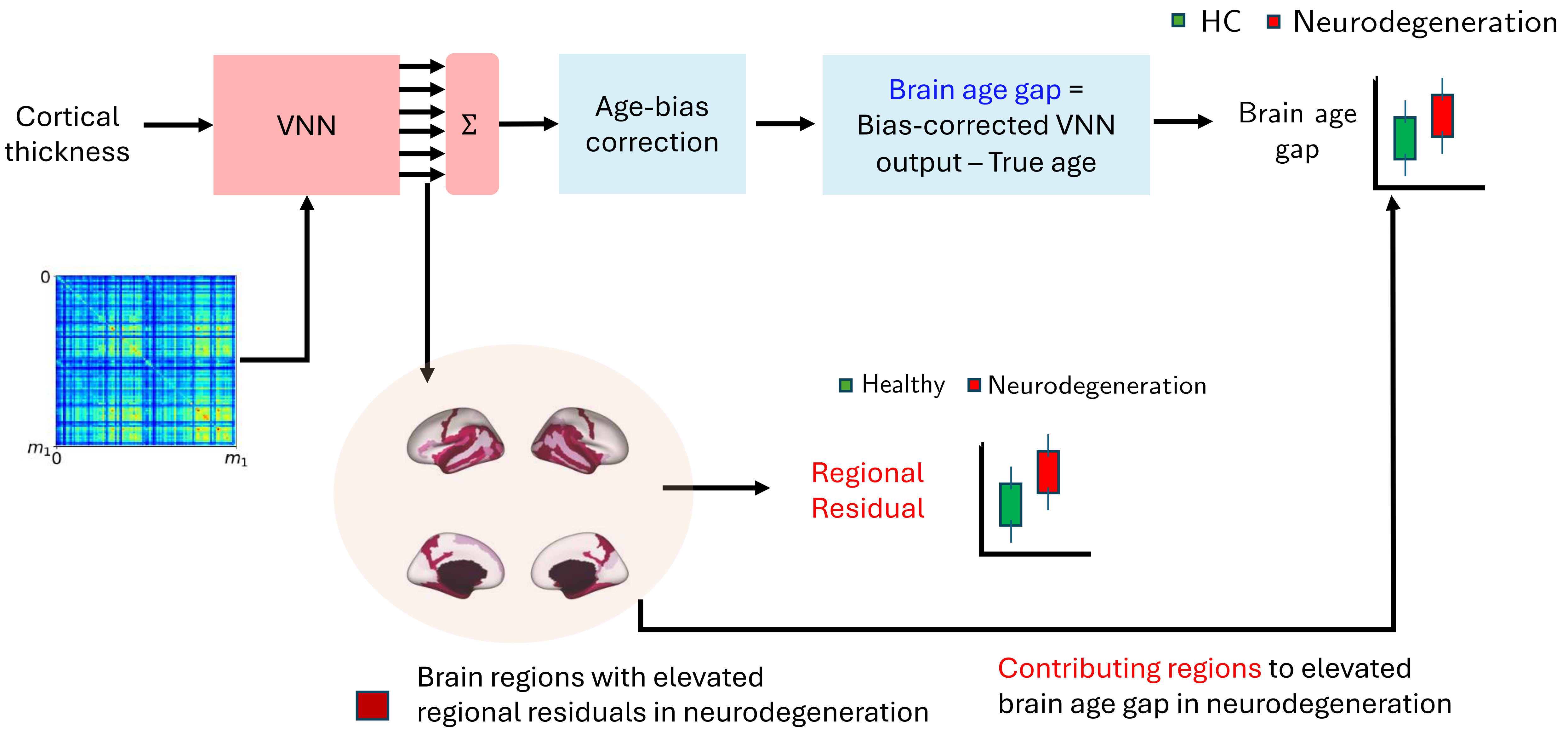
- **Generalizability** to diverse target populations

Sihag et al., 2025 (SPM, to appear)

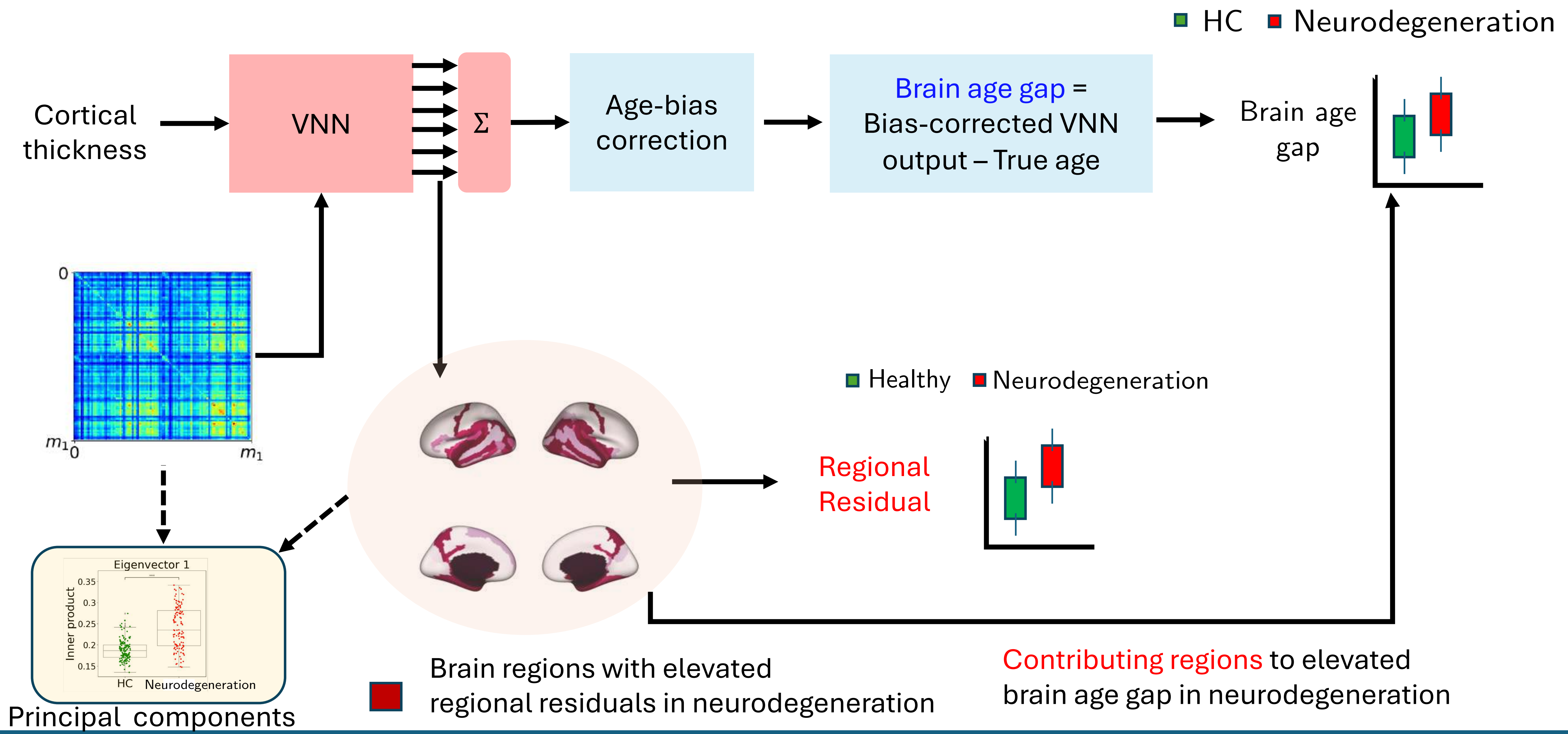
VNNs provide an anatomically interpretable and explainable brain age gap



VNNs provide an anatomically interpretable and explainable brain age gap

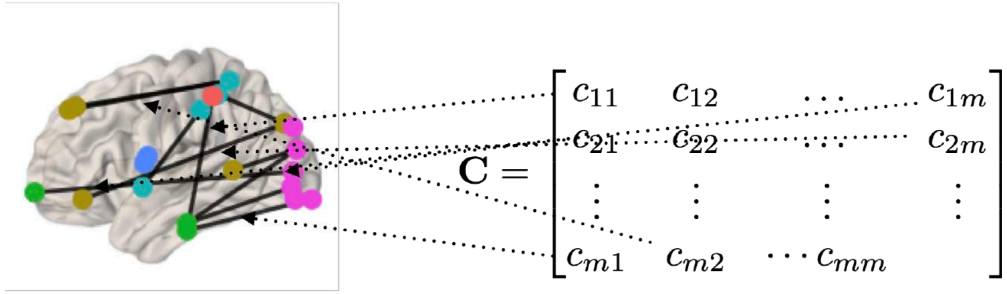


VNNs provide an anatomically interpretable and explainable brain age gap

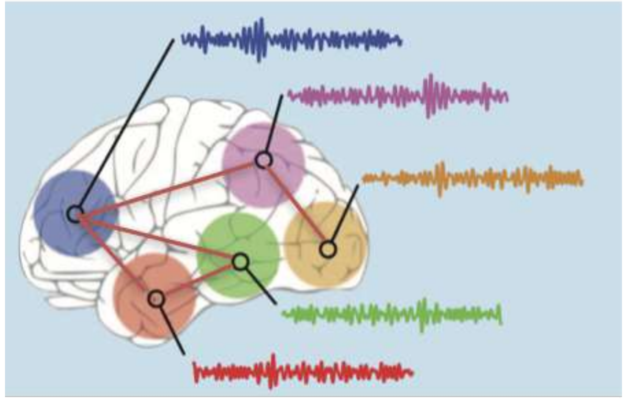


Network neuroscience

- Modeling brain as a network (**connectomes**)



Anatomical covariance matrix
(structural connectome)

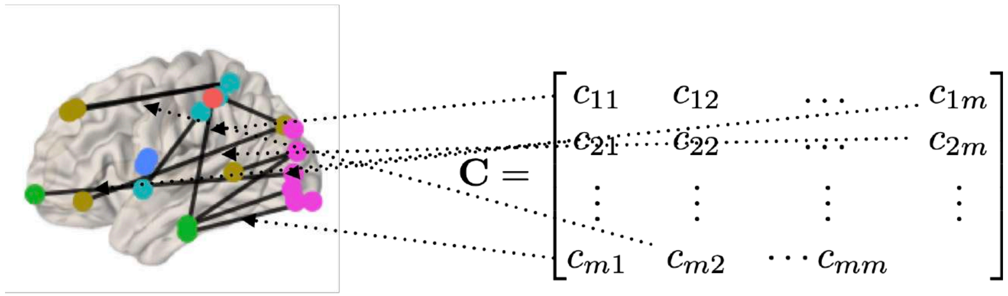


Functional connectome

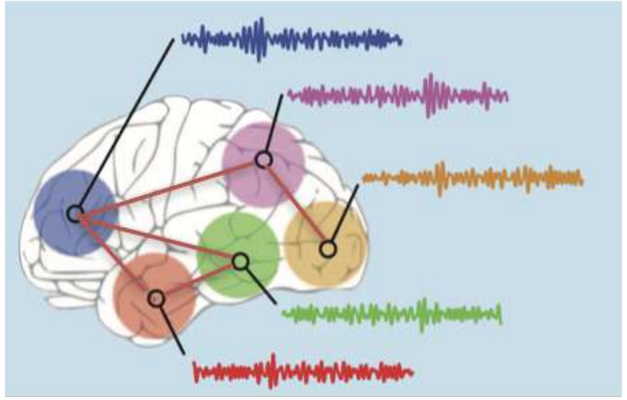
- **Motivation**
  - Significant redundancies in brain structural/functional features
  - Brain structure/function is compromised in neurodegeneration

Covariance matrices in network neuroscience

- Covariance matrices appear commonly in network neuroscience



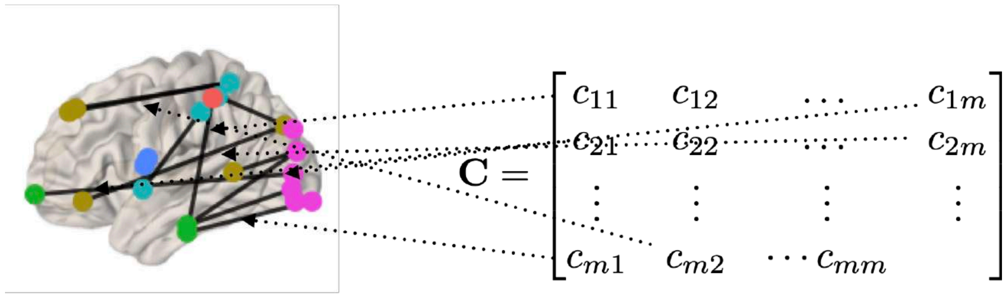
Anatomical covariance matrix



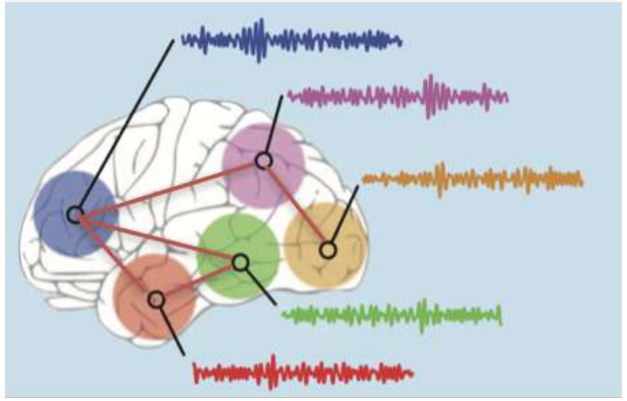
Functional connectome

Covariance matrices in network neuroscience

- Covariance matrices appear commonly in network neuroscience

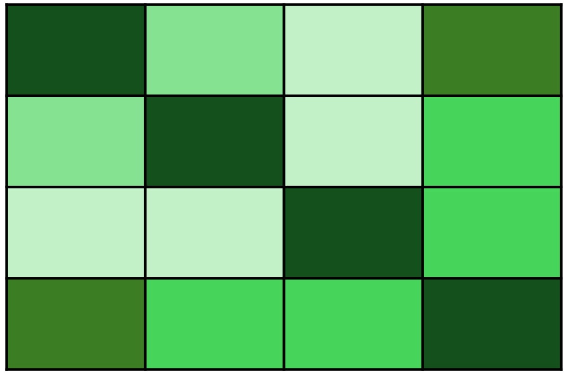


Anatomical covariance matrix



Functional connectome

- Inference over covariance matrices in network neuroscience
  - **Traditional** statistical approaches (for e.g., PCA)
    - Interpretable, suitable for low data regimes
  - **Deep learning** approaches (for e.g., GNNs)
    - Enhanced expressivity, improved performance



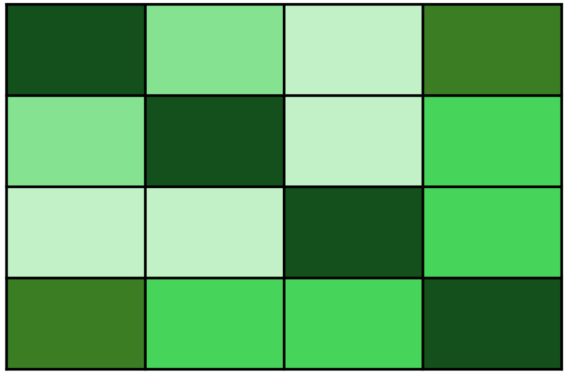
VNNs are well suited for neuroimaging data analysis

➢ Properties of VNNs make them appealing for neuroimaging data analysis

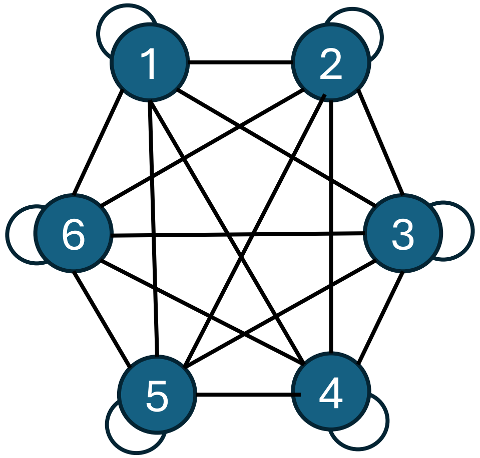
- **Connections with PCA** ⇨ transparent outcomes by leveraging spectrum of covariance matrix
- **Stability** ⇨ reproducible outcomes in limited data settings
- **Transferability** ⇨ enhanced generalizability and robustness to choice of brain atlases

Anatomical covariance matrix as a graph

➢ Covariance matrix is a **data-driven** graph

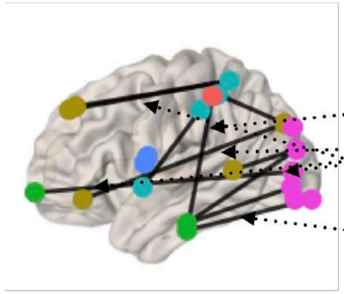


$$\mathbf{x} = [x_1, \ldots, x_6]^\top$$



Covariance matrix as a fully-connected graph

$$\hat{\mathbf{C}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\top, \text{ where } \hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

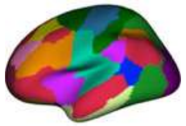


$$\mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} \\ c_{21} & c_{22} & \cdots & c_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} & \cdots c_{mm} & \end{bmatrix}$$

Anatomical covariance matrix
(estimated from cortical features)

VNN vs PCA on age prediction task

- Regression task

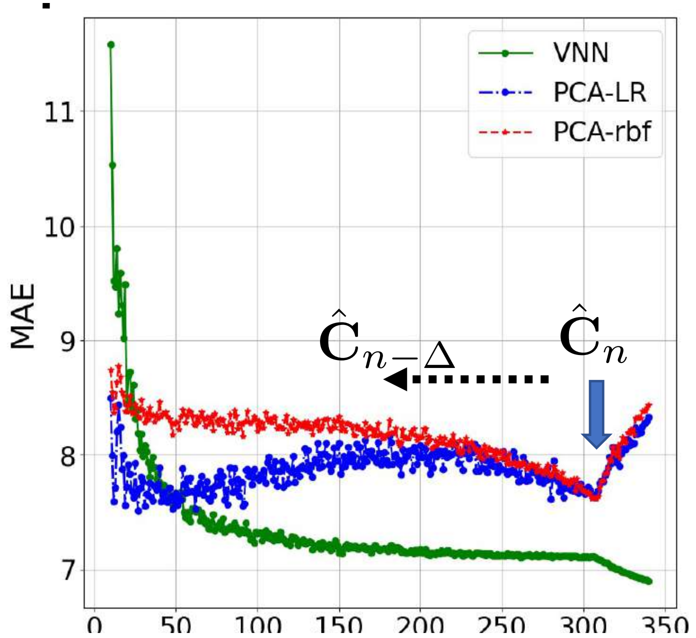


Cortical thickness data → VNN → Estimate of age

- Comparison against PCA-regression

Data: cortical thickness dataset (m = 104) from (n = 341) human subjects

- **Metric**: MAE (mean absolute error)



VNN: coVariance Neural Network

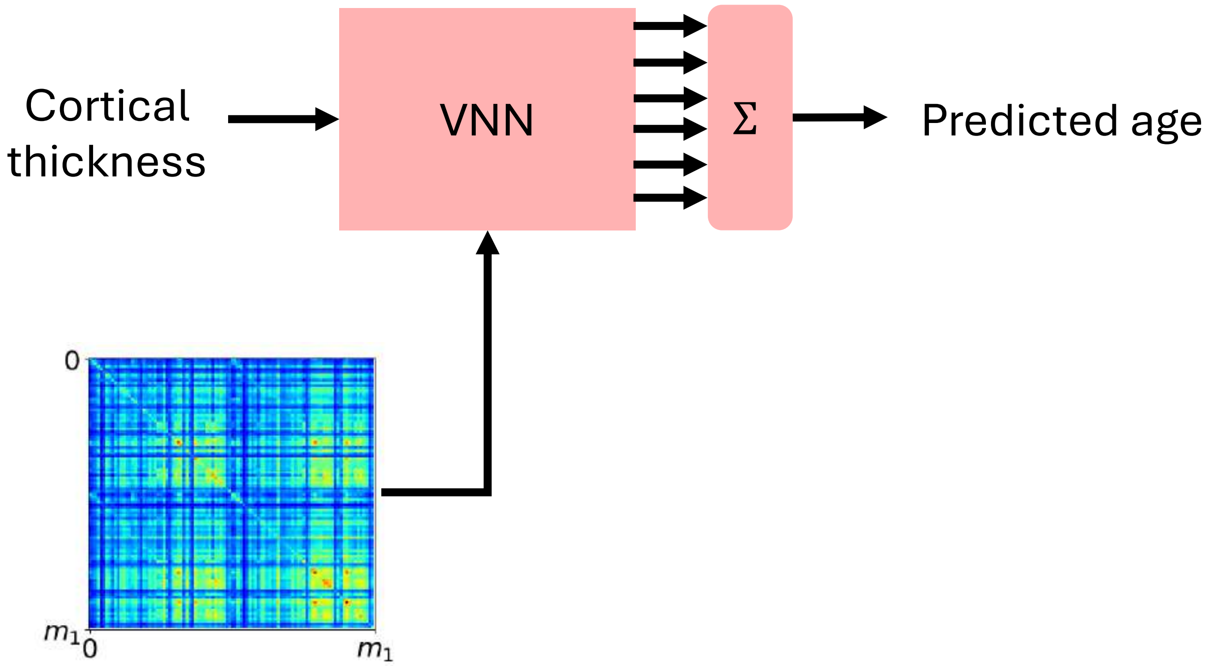
PCA-LR: PCA-regression with linear kernel

PCA-rbf: PCA regression with rbf kernel

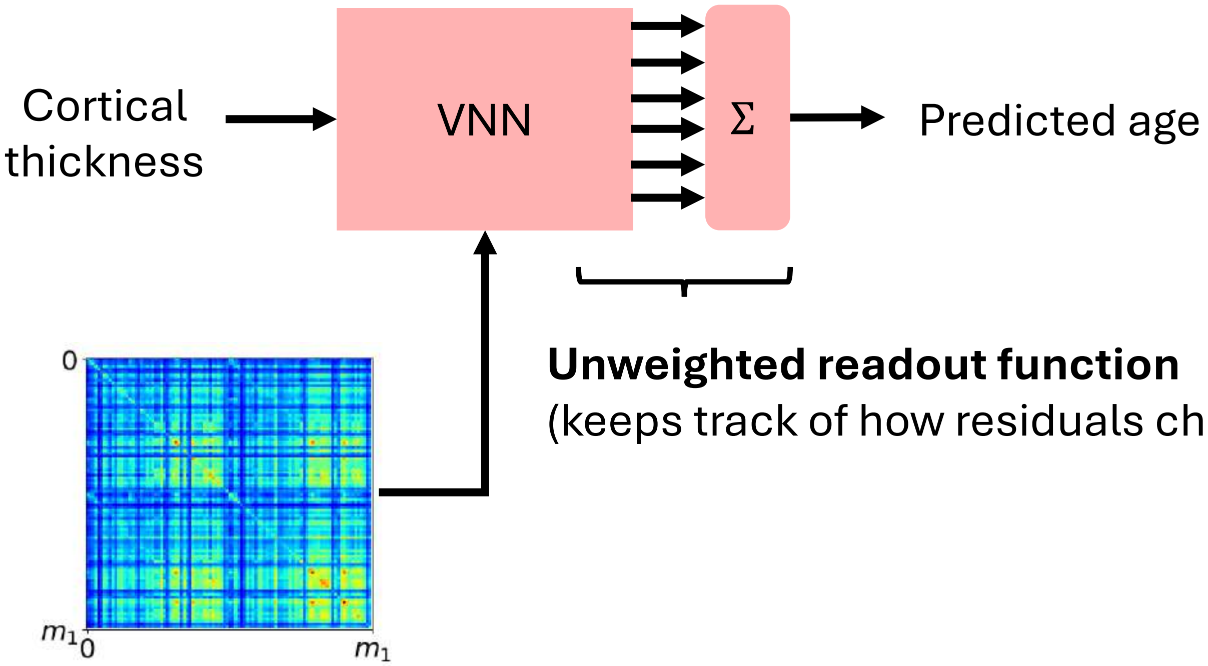
VNN outperforms PCA and is **more stable**

[Sihag et al., 2022]

VNNs provide an anatomically interpretable and explainable brain age gap

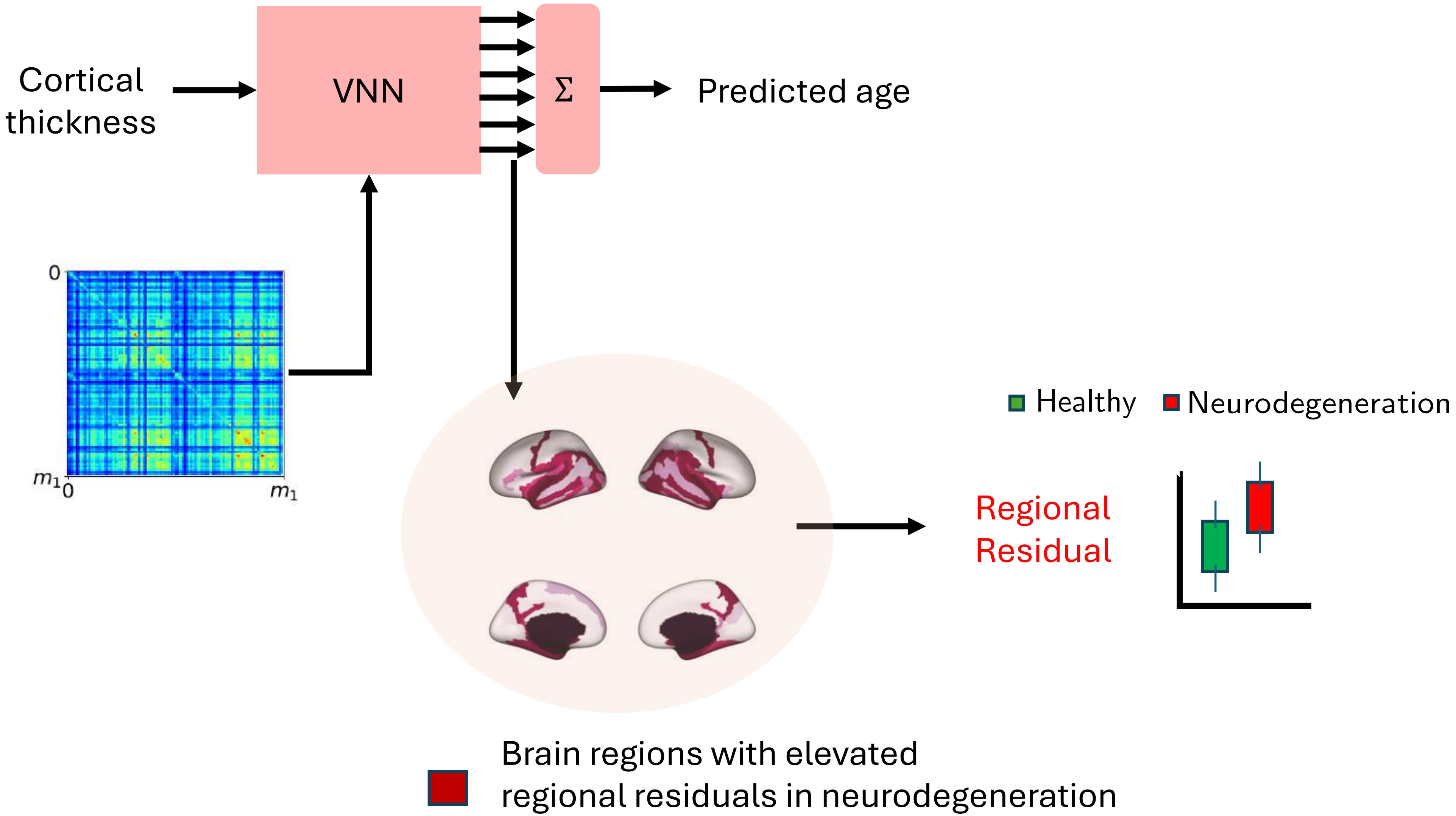


VNNs provide an anatomically interpretable and explainable brain age gap

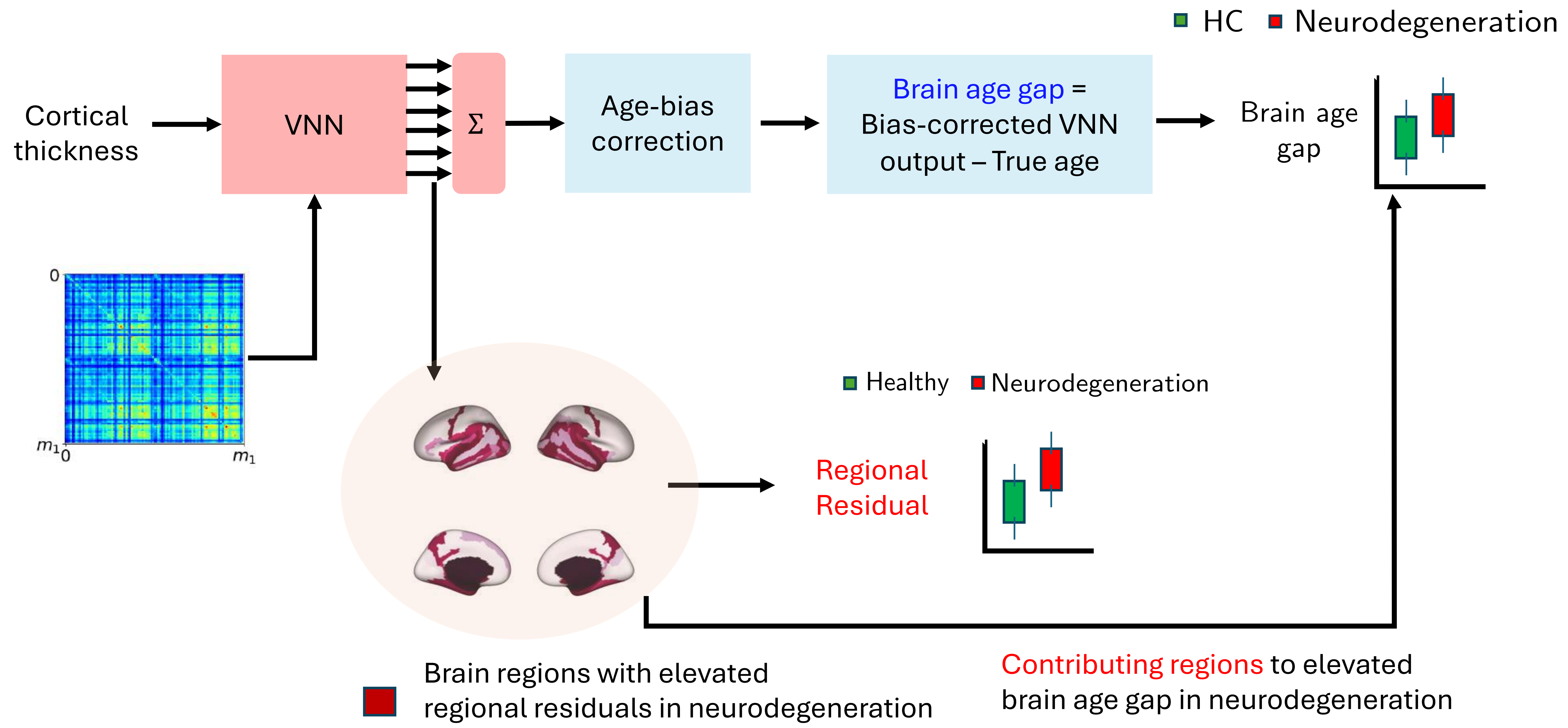


Unweighted readout function
(keeps track of how residuals change in target population)

VNNs provide an anatomically interpretable and explainable brain age gap



VNNs provide an anatomically interpretable and explainable brain age gap



Experiments

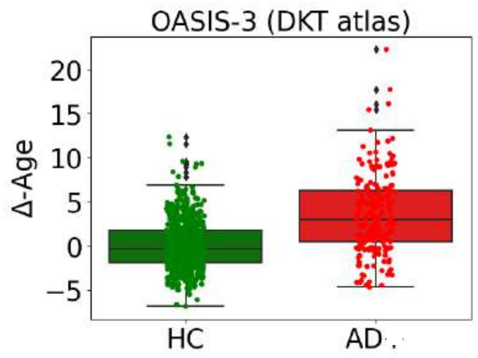
➢ Participants from OASIS-3 dataset, 148 cortical thickness features per individual

(Distrieux brain atlas)

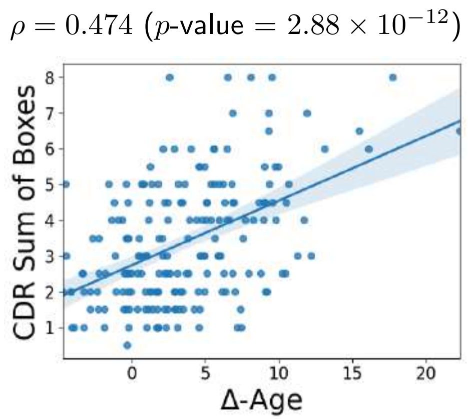
	HC	AD
Number	611	194
Age	68.38 (7.62)	74.72 (7.02)
Sex (m/f)	260/351	100/94
CDR sum of boxes	0	3.45 (1.74)

HC group: cognitively normal
AD group: AD diagnosis
CDR: Clinical dementia rating

➢ Brain age gap is elevated in AD group and correlated with CDR sum of boxes

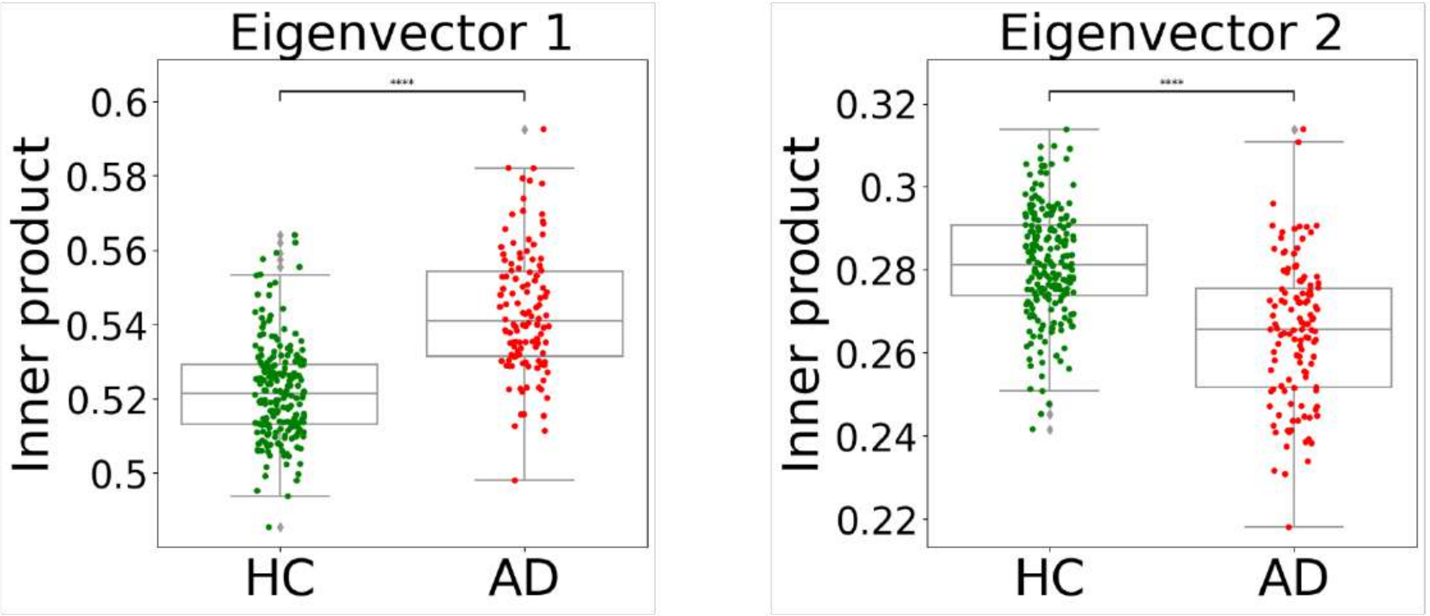


Anatomical interpretability



Experiments

➢ VNN **distinctly** exploits eigenvectors in AD and HC groups



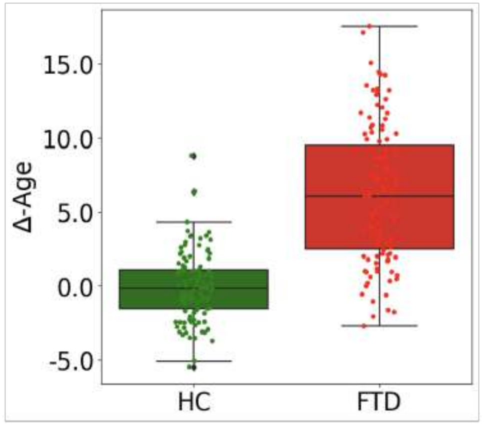
⇨ explains anatomical interpretability of brain age gap in AD

Experiments

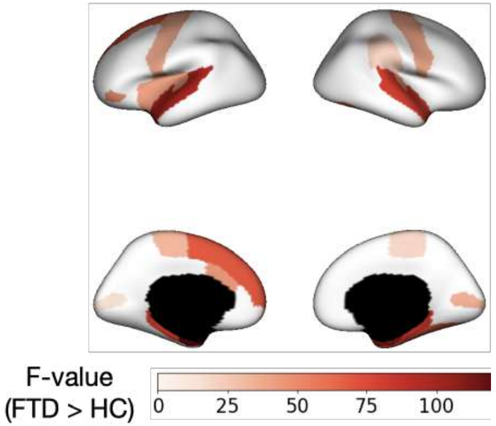
- Whole brain cortical thickness dataset for Frontotemporal Dementia (FTD)
  - Healthy controls (HC, n =114, age = 64.51 ± 6.51 years, 65 females)
  - FTD diagnosis (FTD, n =119, age = 64.72 ± 6.78 years, 47 females)

- 68 cortical thickness features (Desikan-Killiany atlas)

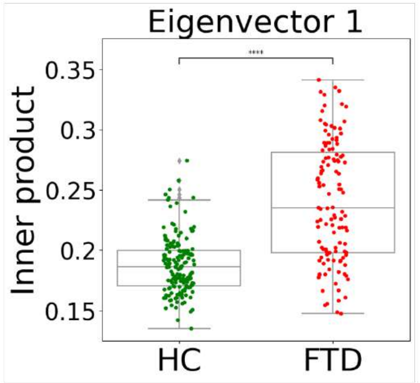
Brain age gap distributions

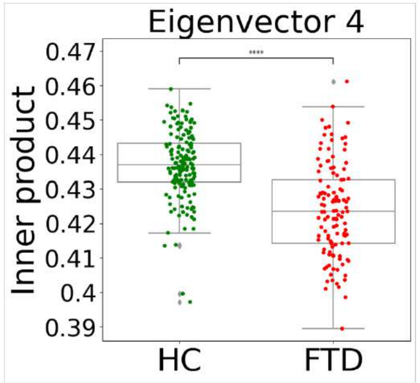


Anatomic interpretability

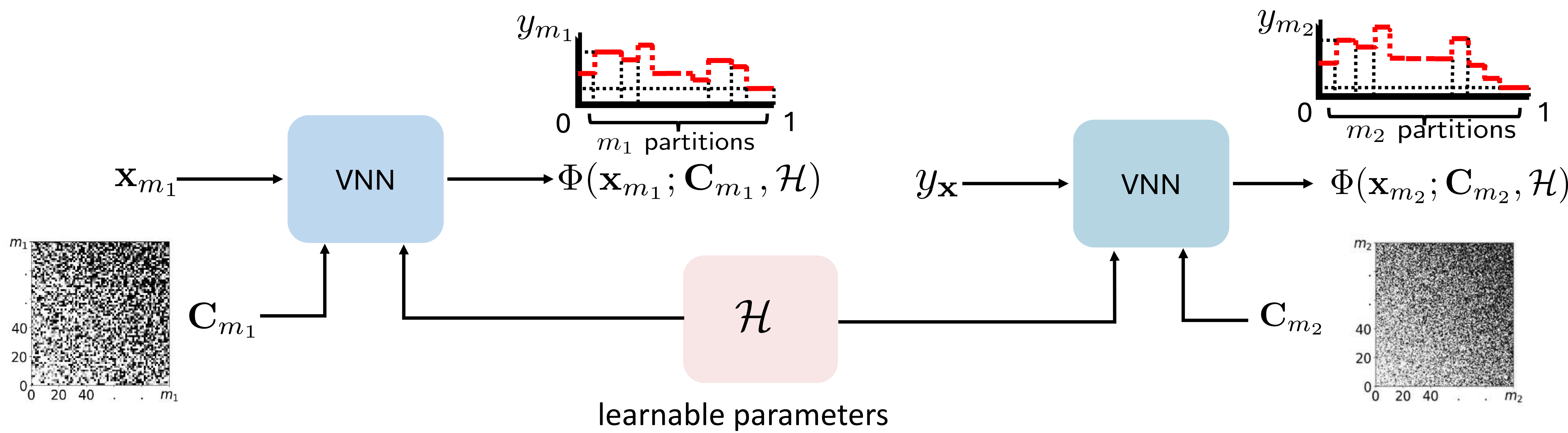


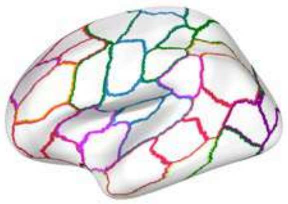
Explaining anatomic interpretability

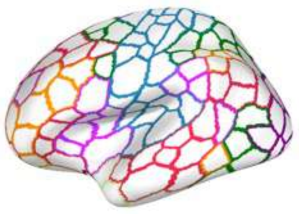




VNNs are provably transferable





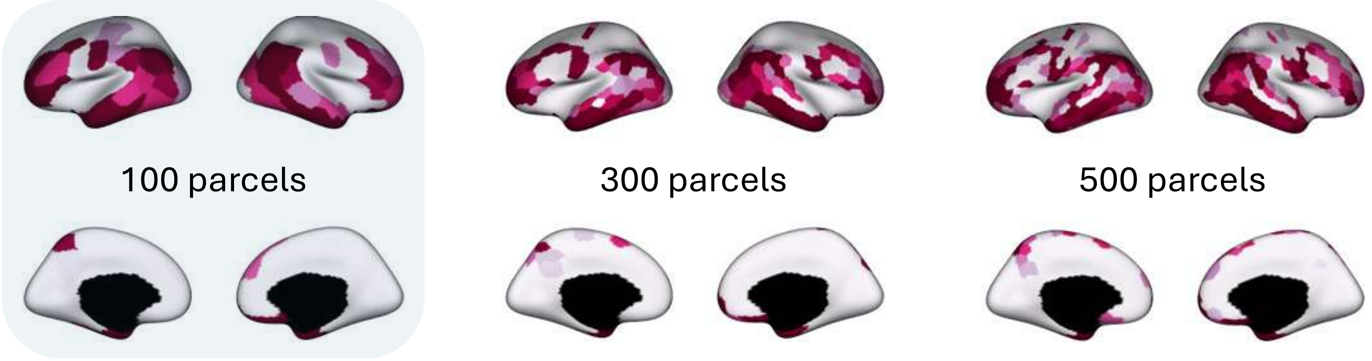


Transferability bound

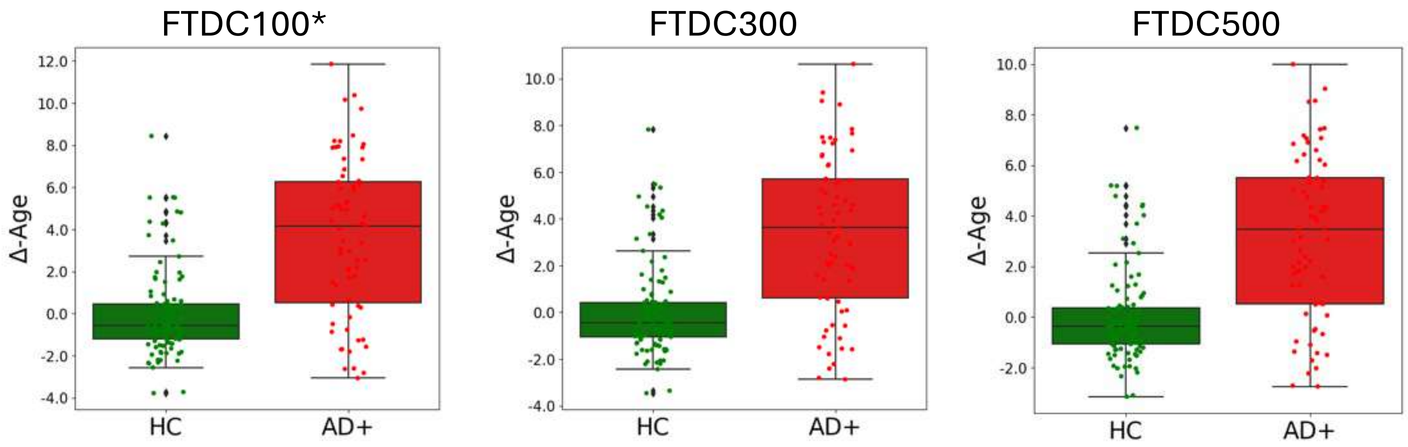
$$\|y_{m_1} - y_{m_2}\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}} + \frac{1}{m_2^{3\zeta/2-1}}\right), \text{ for } \zeta \in (2/3, 1]$$

Recap: Transferability of VNNs cross-validates brain age gap in multi-resolution setting

Objective: Brain age gap prediction in HC (healthy) and AD+ (Alzheimer's) cohorts from VNNs trained on 100-feature dataset



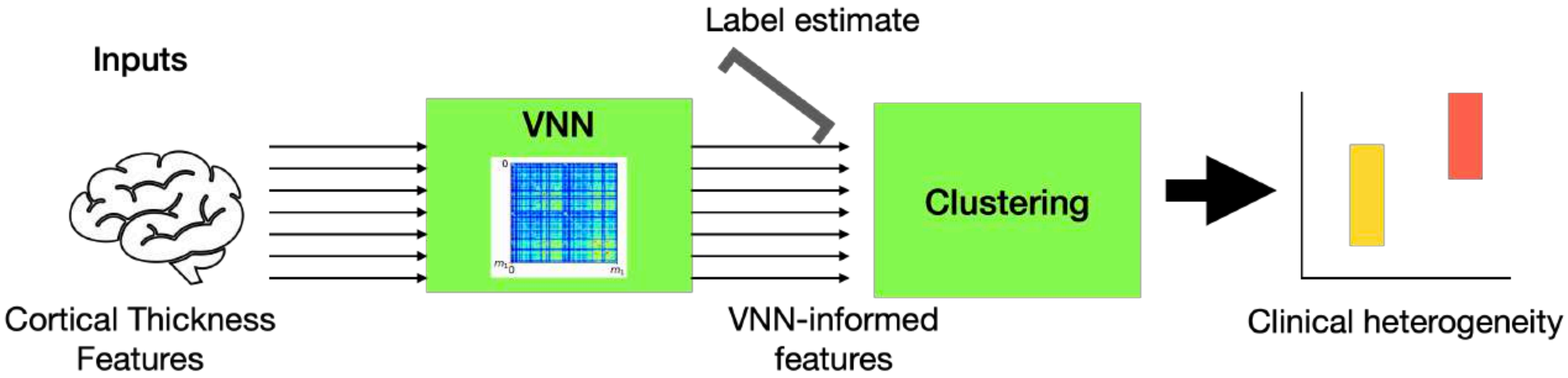
- ROIs contributing to elevated brain age gap in AD+ across different resolutions



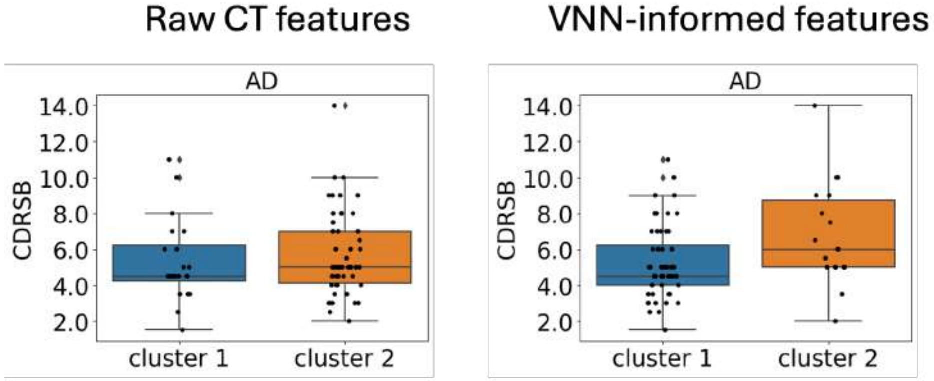
- Brain age gap is elevated in AD+ w.r.t HC cohort in 100-feature dataset
- Results on brain age gap retained after transferring VNN to 300 and 500-feature datasets

VNNs as pre-trained models...

- Uncovering **disease heterogeneity** with VNNs as pre-trained models



- VNNs offer more **significant clinical stratification** than raw anatomical features

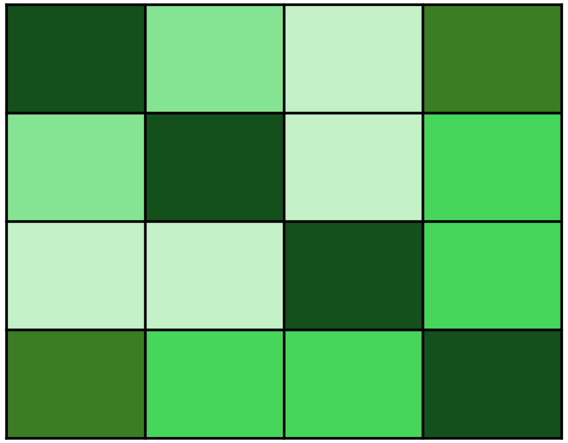


VNN enhances the clinical relevance of anatomical features

Alignment of covariance with learning: An NTK perspective

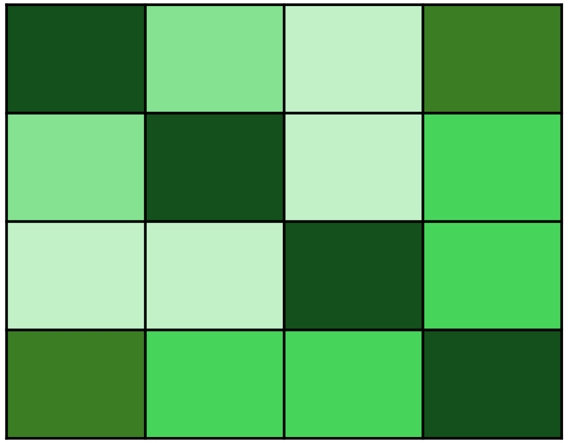
Are covariance matrices suitable for a learning task?

- Covariance matrices add a meaningful inductive bias to neural nets
  - Covariance matrix captures the (linear) structure
- VNNs provide the bridge between PCA and GNNs



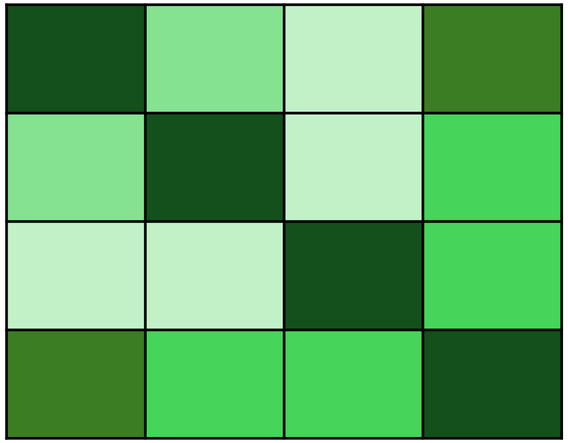
Are covariance matrices suitable for a learning task?

- Covariance matrices add a meaningful inductive bias to neural nets
  - Covariance matrix captures the (linear) structure
- VNNs provide the bridge between PCA and GNNs
- Can we quantify the suitability of covariance to learning objective?
  - Is performance good?
  - good generalization?



Are covariance matrices suitable for a learning task?

- Covariance matrices add a meaningful inductive bias to neural nets
  - Covariance matrix captures the (linear) structure
- VNNs provide the bridge between PCA and GNNs
- Can we quantify the suitability of covariance to learning objective?
  - Is performance good?
  - good generalization?
- **Neural tangent kernels (NTKs)**-driven insights for VNNs

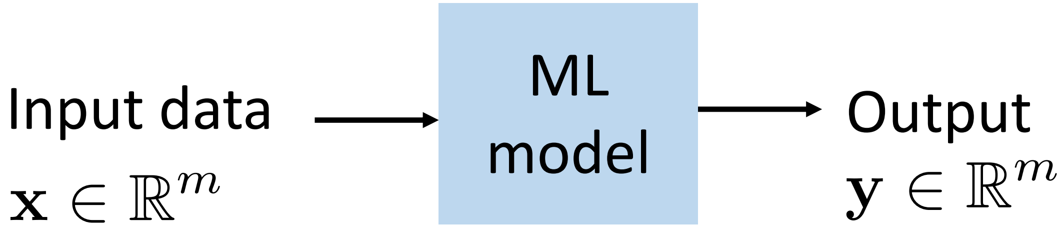


Neural tangent kernels (NTK)

- **NTKs** describe the **evolution of neural nets** during training by **gradient descent**
- **Example:**

  Predict $\mathbf{y}$ from $\mathbf{x}$ using

  ML model $\mathbf{f}$ (parameters $\mathbf{h}$)



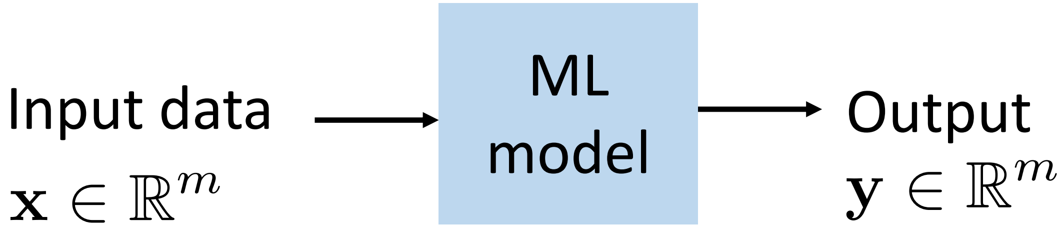
$$\mathbf{f} : \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}^m$$

Neural tangent kernels (NTK)

- **NTKs** describe the **evolution of neural nets** during training by **gradient descent**
- **Example:**

  Predict $\mathbf{y}$ from $\mathbf{x}$ using

  ML model $\mathbf{f}$ (parameters $\mathbf{h}$)



$$\mathbf{f} : \mathbb{R}^m \times \mathbb{R}^p \mapsto \mathbb{R}^m$$

$$\text{Loss :} \quad \mathcal{L} = \text{ mean squared error (MSE)}$$

$$= \sum_{i \in \text{Data}} \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i, \mathbf{h}))\|^2$$

- Optimize parameters $\mathbf{h}$ using **gradient descent**

Neural tangent kernels (NTK)

Input data $\longrightarrow$ ML model $\longrightarrow$ Output

- **Evolution of gradient descent**

(linear approximation)

$$\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t+1)}) = \mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - \eta \boldsymbol{\Theta}(\mathbf{x}, \mathbf{h}^{(t)}) \cdot (\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - y)$$

η : learning rate

$\boldsymbol{\Theta}(\mathbf{x}, \mathbf{h}^{(t)})$: NTK matrix

$$\Theta(\mathbf{x}_i, \mathbf{x}_j) = \nabla_{\mathbf{h}} \mathbf{f}(\mathbf{x}, \mathbf{h})^{\mathsf{T}} \nabla_{\mathbf{h}} \mathbf{f}(\mathbf{x}, \mathbf{h})$$

Depends on model architecture + input

Neural tangent kernels (NTK)

Input data $\longrightarrow$ ML model $\longrightarrow$ Output

- **Evolution of gradient descent**

  (linear approximation)

$$\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t+1)}) = \mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - \eta \boldsymbol{\Theta}(\mathbf{x}, \mathbf{h}^{(t)}) \cdot (\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - y)$$

η : learning rate

$\boldsymbol{\Theta}(\mathbf{x}, \mathbf{h}^{(t)})$: NTK matrix

$$\Theta(\mathbf{x}_i, \mathbf{x}_j) = \nabla_{\mathbf{h}} \mathbf{f}(\mathbf{x}, \mathbf{h})^{\mathsf{T}} \nabla_{\mathbf{h}} \mathbf{f}(\mathbf{x}, \mathbf{h})$$

Depends on model architecture + input

- For neural networks with **infinite width**, NTK matrix is constant w.r.t $\mathbf{h}$

$$\boldsymbol{\Theta}(\mathbf{x}, \mathbf{h}^{(t)}) \to \boldsymbol{\Theta}(\mathbf{x})$$

Neural tangent kernels (NTK)

- **Convergence of gradient descent** dictated by **alignment** between NTK and data

$$\|\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - \mathbf{y}\|_2^2 \propto \mathcal{A} \quad \text{where} \quad \mathcal{A} = \mathbf{y}^\top \boldsymbol{\Theta} \mathbf{y}$$

Larger $\mathcal{A}$ $\Longrightarrow$ faster convergence

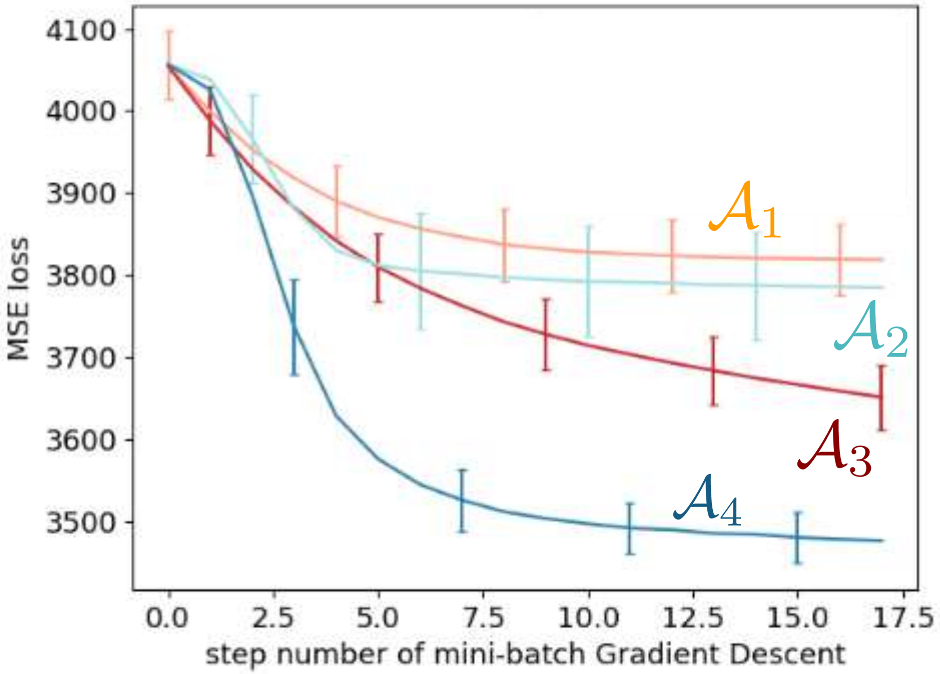
Neural tangent kernels (NTK)

- **Convergence of gradient descent** dictated by **alignment** between NTK and data

$$\|\mathbf{f}(\mathbf{x}, \mathbf{h}^{(t)}) - \mathbf{y}\|_2^2 \propto -\mathcal{A} \quad \text{where} \quad \mathcal{A} = \mathbf{y}^\mathsf{T} \boldsymbol{\Theta} \mathbf{y}$$

Larger $\mathcal{A}$ $\Rightarrow$ faster convergence

Example

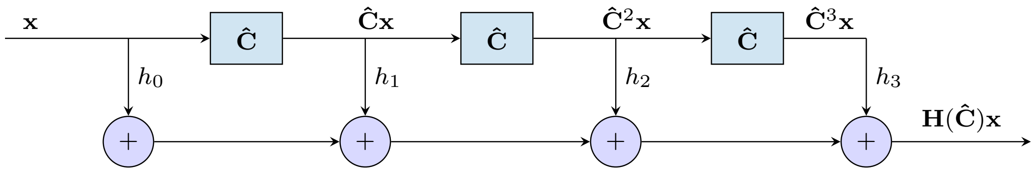


$\mathcal{A}_1 < \mathcal{A}_2 < \mathcal{A}_3 < \mathcal{A}_4$

Larger alignment implies better convergence/ loss

NTK for covariance filter

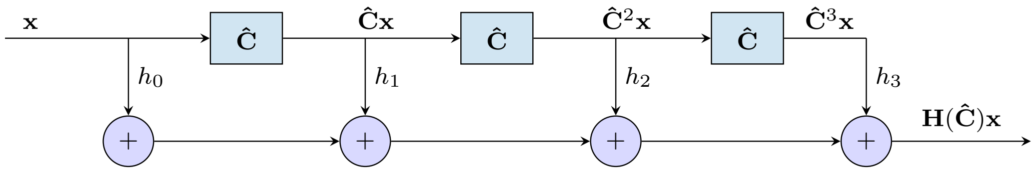
- For a covariance filter $\mathbf{H}(\hat{\mathbf{C}}) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k$,



  - NTK is $\Theta = \sum_{k=0}^{K} \hat{\mathbf{C}}^k \mathbf{x}\mathbf{x}^\mathsf{T} \hat{\mathbf{C}}^k$

Alignment for covariance filter

- For a covariance filter $\mathbf{H}(\hat{\mathbf{C}}) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k$,



  - NTK is $\Theta = \sum_{k=0}^{K} \hat{\mathbf{C}}^k \mathbf{x}\mathbf{x}^\mathsf{T} \hat{\mathbf{C}}^k$

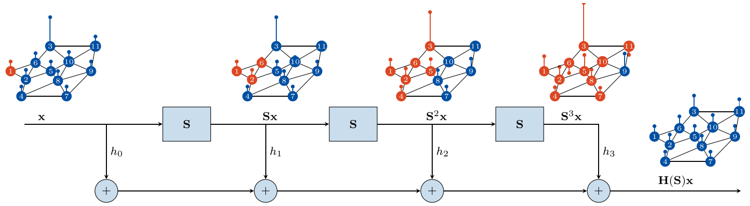
⇨ convergence of learning with covariance filter is dictated by

$$\mathcal{A} = \mathbf{y}^\mathsf{T} \left(\sum_{k=0}^{K} \hat{\mathbf{C}}^k \mathbf{x}\mathbf{x}^\mathsf{T} \hat{\mathbf{C}}^k \right) \mathbf{y}$$

Alignment between $\mathbf{x}$, $\hat{\mathbf{C}}$, and $\mathbf{y}$

Khalafi et al., 2024

Data-driven graph by optimizing alignment



- Treating alignment as an optimization objective
  - **Goal**: find the *optimal* graph shift operator

$$\mathbf{S}^* = \max_{\mathbf{S}} \mathcal{A}(\mathbf{S})$$

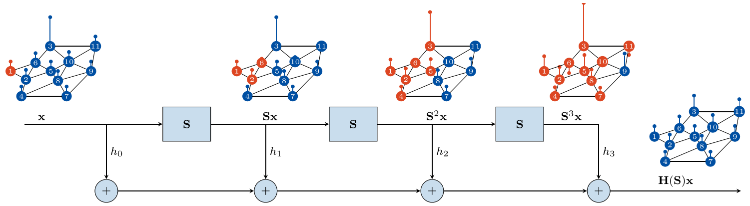
$$= \max_{\mathbf{S}} \mathbf{y}^\top \left(\sum_{k=0}^{K} \mathbf{S}^k \mathbf{x}\mathbf{x}^\top \mathbf{S}^k \right) \mathbf{y}$$

⇨ Find **S** that maximizes

$$\sum_{k=0}^{K} (\mathbf{y}^\top \mathbf{S} \mathbf{x})^2$$

Khalafi et al., 2024

Data-driven graph by optimizing alignment



- Treating alignment as an optimization objective
  - **Goal**: find the *optimal* graph shift operator

$$\mathbf{S}^* = \max_{\mathbf{S}} \mathcal{A}(\mathbf{S})$$
$$= \max_{\mathbf{S}} \mathbf{y}^\mathsf{T} \left(\sum_{k=0}^{K} \mathbf{S}^k \mathbf{x} \mathbf{x}^\mathsf{T} \mathbf{S}^k \right) \mathbf{y}$$

⇒ Find $\boldsymbol{S}$ that maximizes

$$\boxed{\sum_{k=0}^{K} (\mathbf{y}^\mathsf{T} \mathbf{S} \mathbf{x})^2}$$

Correlation between

- Graph shift operator $\boldsymbol{S}$
- Input $\mathbf{x}$
- Output $\mathbf{y}$

Khalafi et al., 2024

Data-driven covariance graph by optimizing alignment

- **Cross-covariance** graph optimizes alignment

$$\mathbf{S}^* = \frac{1}{2}(\mathbf{x}\mathbf{y}^\top + \mathbf{y}\mathbf{x}^\top)$$

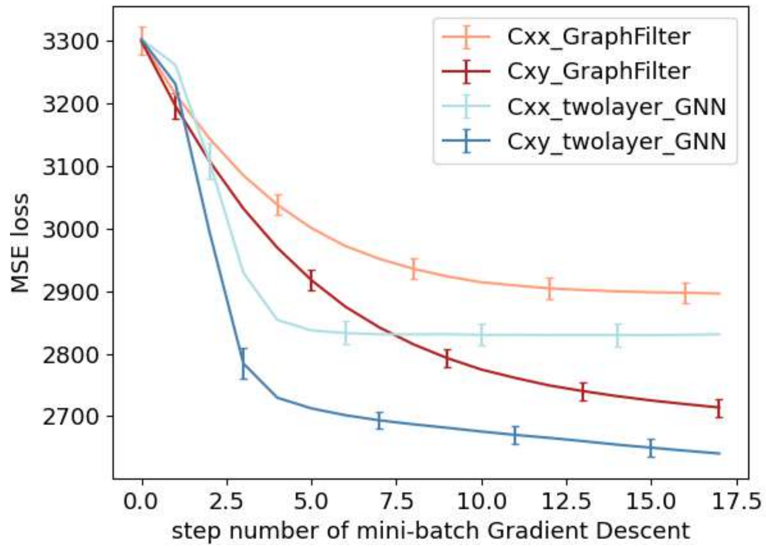
Khalafi et al., 2024

Data-driven covariance graph by optimizing alignment

- **Cross-covariance** graph optimizes alignment

$$\mathbf{S}^* = \frac{1}{2}(\mathbf{x}\mathbf{y}^\top + \mathbf{y}\mathbf{x}^\top)$$

- **Numerical results**
  - Time series forecasting: predicting next time step



- Cross-covariance achieves better loss
- GNN with cross-covariance outperforms VNN

Khalafi et al., 2024

Variants of VNNs

Are VNNs enough?

- **Limitations** of VNNs
  - Sample covariance could be poor quality in **low data, high dimensionality setting**
  - High **computational cost** (quadratic in size for dense covariance)
  - No considerations of **temporal, evolving** data
  - Prone to **undesired bias** within the data

Low data, high dimensional settings

- Sample covariance matrix is dense

  ⇨ **noisy** entries in low data, high dimensional settings

  ⇨ computationally inefficient VNNs (quadratic complexity)

Low data, high dimensional settings

- Sample covariance matrix is dense

  ⇒ **noisy** entries in low data, high dimensional settings

  ⇒ computationally inefficient VNNs (quadratic complexity)

- **Solution:** sparsify the sample covariance matrix

  - If true covariance is sparse:

    - Improve estimation quality

    - Common in real world
      (brain imaging, finance, etc.)

Low data, high dimensional settings

- Sample covariance matrix is dense

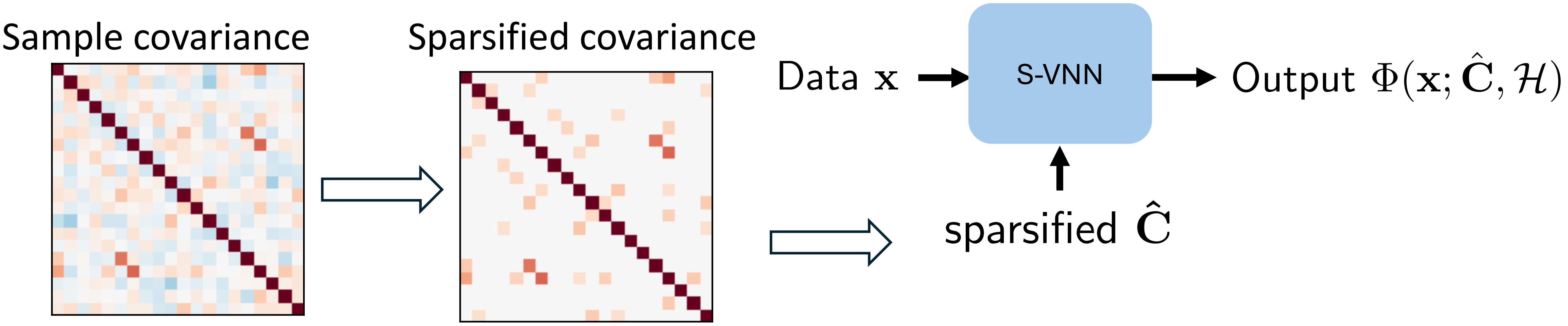
  ⇨ **noisy** entries in low data, high dimensional settings

  ⇨ computationally inefficient VNNs (quadratic complexity)

- **Solution:** sparsify the sample covariance matrix
  - If true covariance is sparse:
    - Improve estimation quality
    - Common in real world (brain imaging, finance, etc.)
  - For generic covariance:
    - Improve computational efficiency

Sparse VNNs

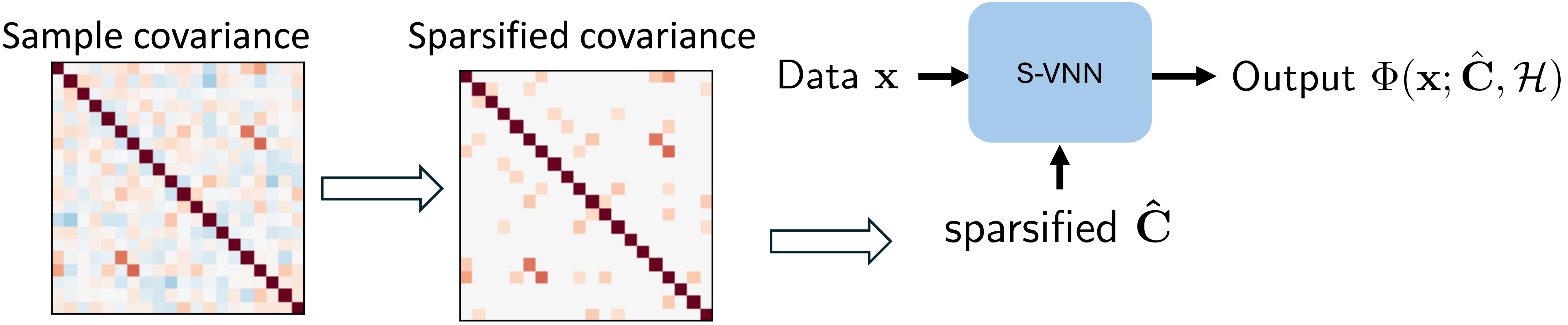
- **Sparse VNNs:** sparsify the covariance matrix with **thresholding** techniques



[Cavallo et al., 2024]

Sparse VNNs

➢ **Sparse VNNs:** sparsify the covariance matrix with **thresholding** techniques



- What thresholding techniques?
- Are sparse VNNs stable?

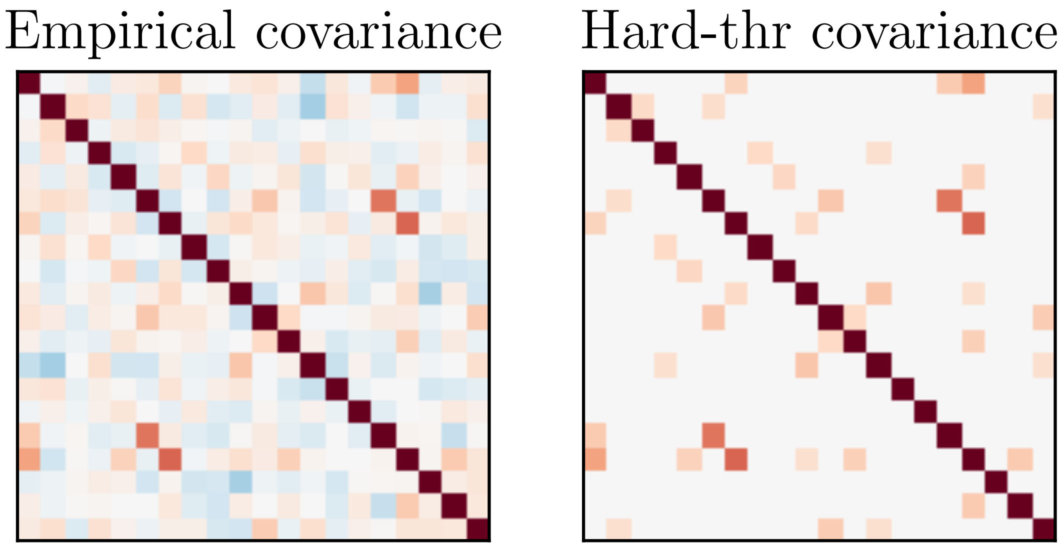
questions to address

[Cavallo et al., 2024]

Hard thresholding

- **Definition**

$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$



[Cavallo et al., 2024]

Hard thresholding

- **Definition**

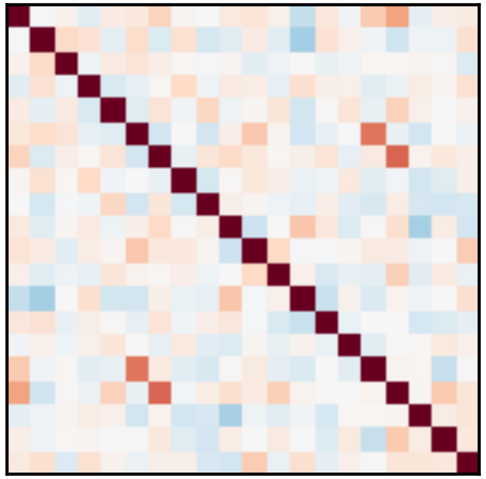
$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$

- **Stability bound**

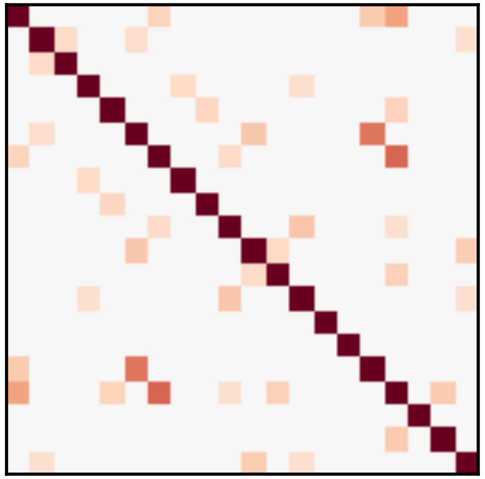
$$\|\mathbf{H}(\hat{\mathbf{C}}_{\text{thr}}) - \mathbf{H}(\hat{\mathbf{C}})\| = \mathcal{O}\left(\frac{c_0}{n^{1/2}}\right)$$

c_0: number of non-zero elements in $\hat{\mathbf{C}}_{\text{thr}}$

Empirical covariance



Hard-thr covariance



[Cavallo et al., 2024]

Hard thresholding

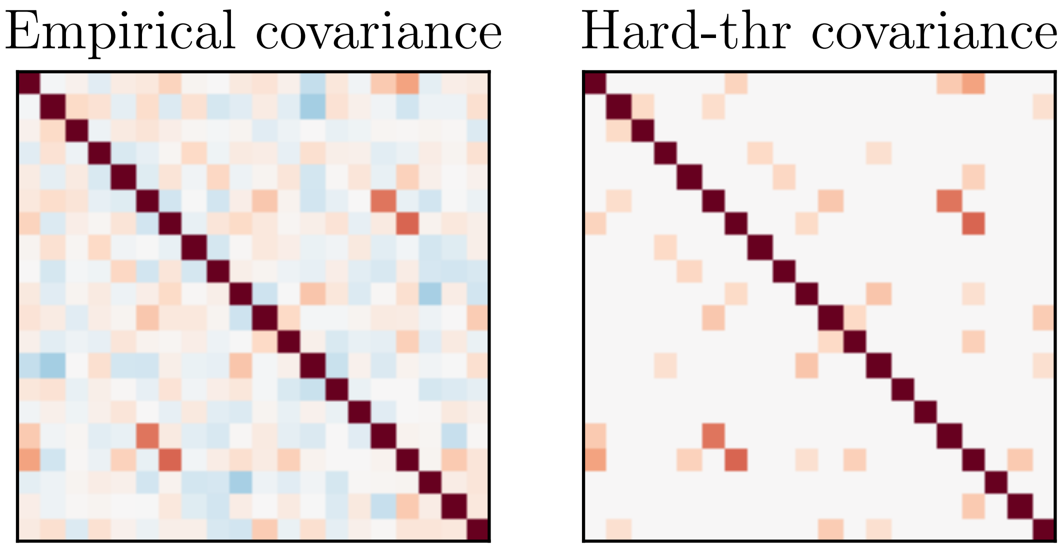
- **Definition**

$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$

- **Stability bound**

$$\|\mathbf{H}(\hat{\mathbf{C}}_{\text{thr}}) - \mathbf{H}(\hat{\mathbf{C}})\| = \mathcal{O}\left(\frac{c_0}{n^{1/2}}\right)$$

c_0: number of non-zero elements in $\hat{\mathbf{C}}_{\text{thr}}$



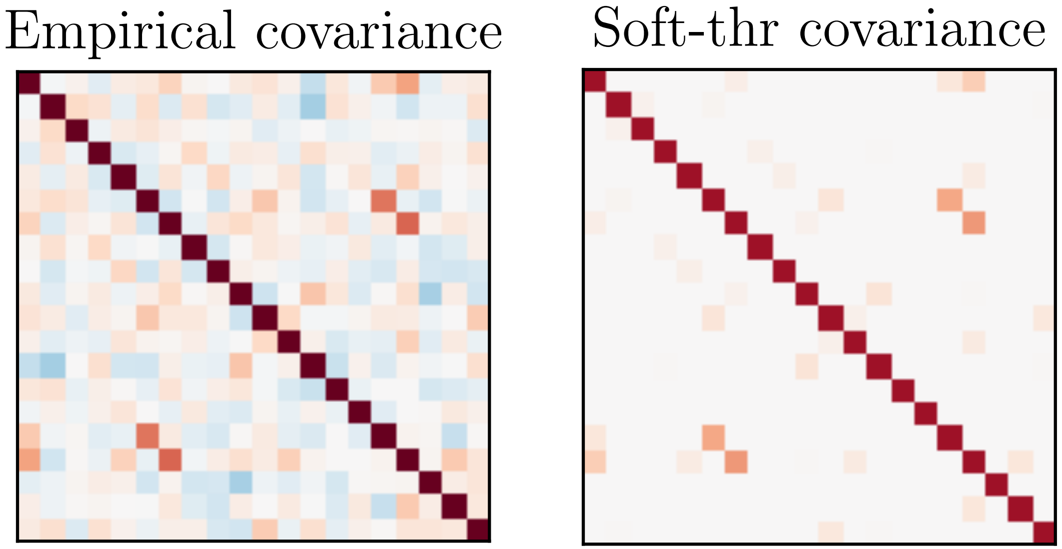
- Stability bound for S-VNNs is **tighter** than *dense* VNNs

[Cavallo et al., 2024]

Soft thresholding

- **Definition**

$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} - \mathsf{sign}(\hat{c}_{ij})\tau/n \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$



[Cavallo et al., 2024]

Soft thresholding

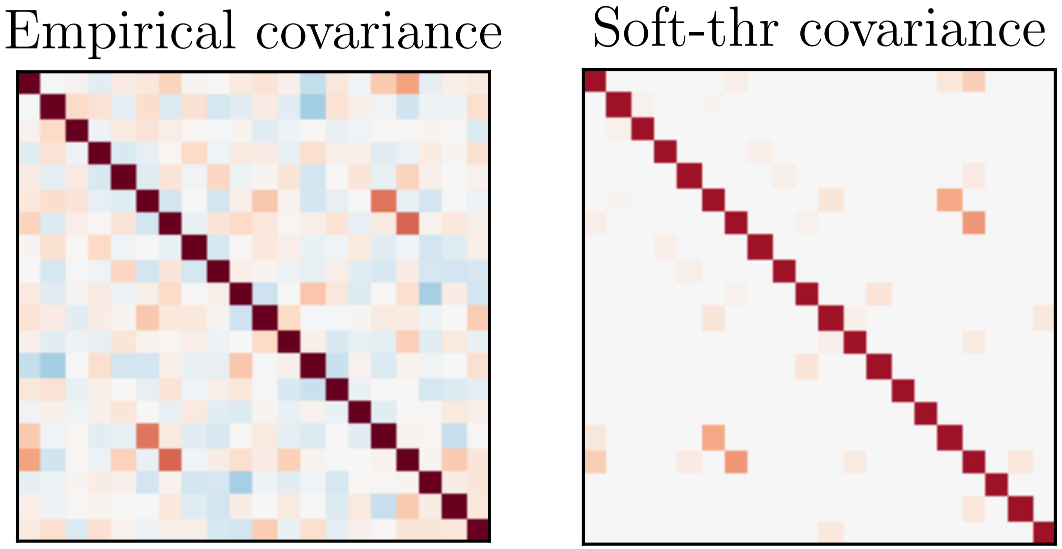
- **Definition**

$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} - \mathsf{sign}(\hat{c}_{ij})\tau/n \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$

- **Stability bound**

$$\|\mathbf{H}(\hat{\mathbf{C}}_{\mathsf{thr}}) - \mathbf{H}(\hat{\mathbf{C}})\| = \mathcal{O}\left(\frac{c_0}{n^{1/2}}\right)$$

c_0: number of non-zero elements in $\hat{\mathbf{C}}_{\mathsf{thr}}$

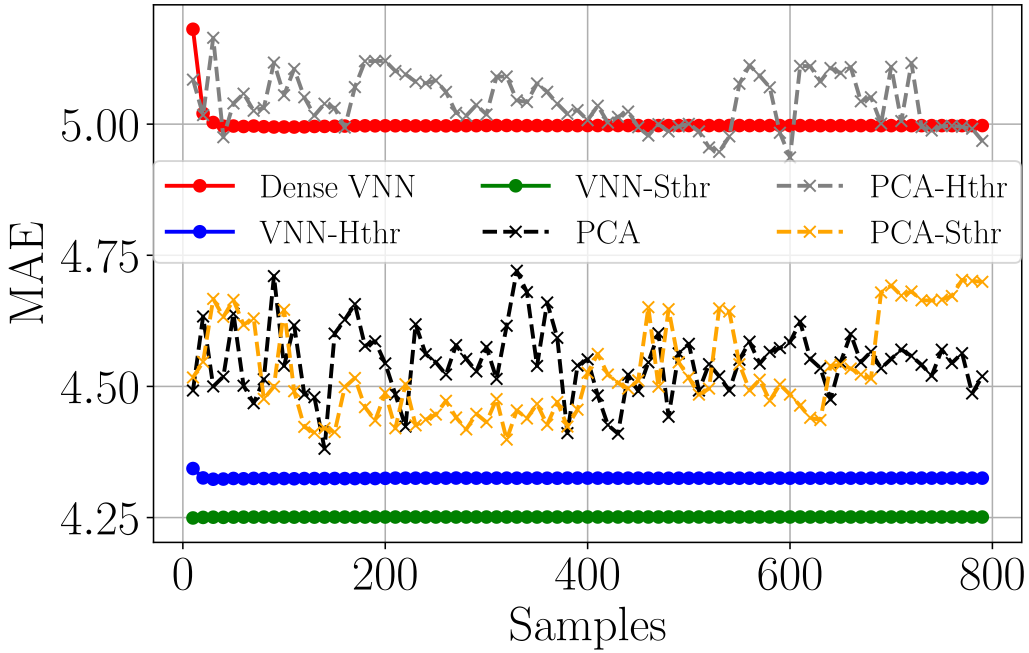


- Stability bound for S-VNNs is **tighter** than *dense* VNNs

[Cavallo et al., 2024]

Sparse VNNs: Numerical results

- Train VNNs/PCA on one covariance and test on another covariance estimated from less samples (synthetic dataset)



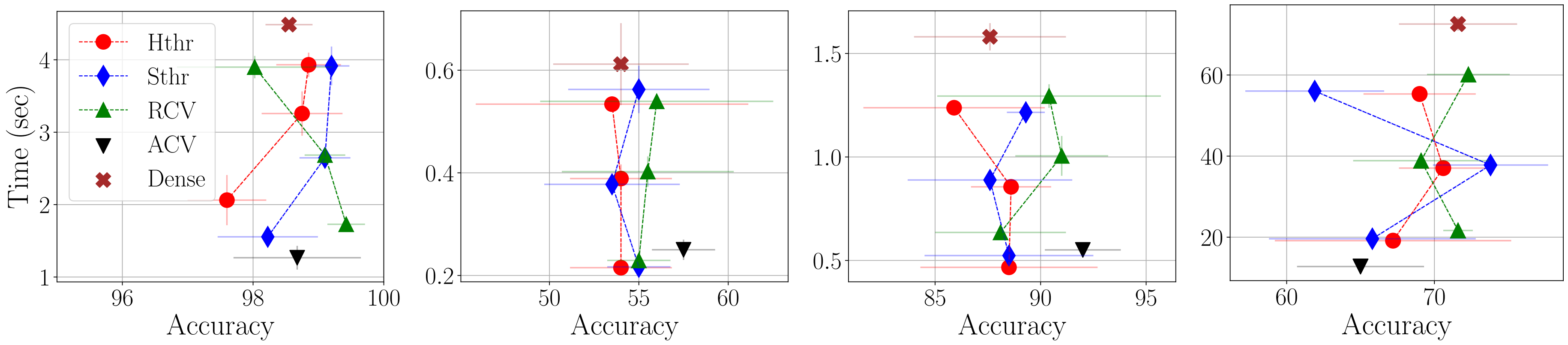
Results

- S-VNN (both soft and hard thresholding) **outperform** PCA and *dense* VNNs
- VNNs **more stable** than PCA

[Cavallo et al., 2024]

Sparse VNNs: Numerical results

- Classification task on real data
- **Datasets** (from left to right)
  - **Brain recordings:** Epilepsy and CNI – classify patient condition
  - **Human action recognition:** MHEALTH and Realdisp – classify action



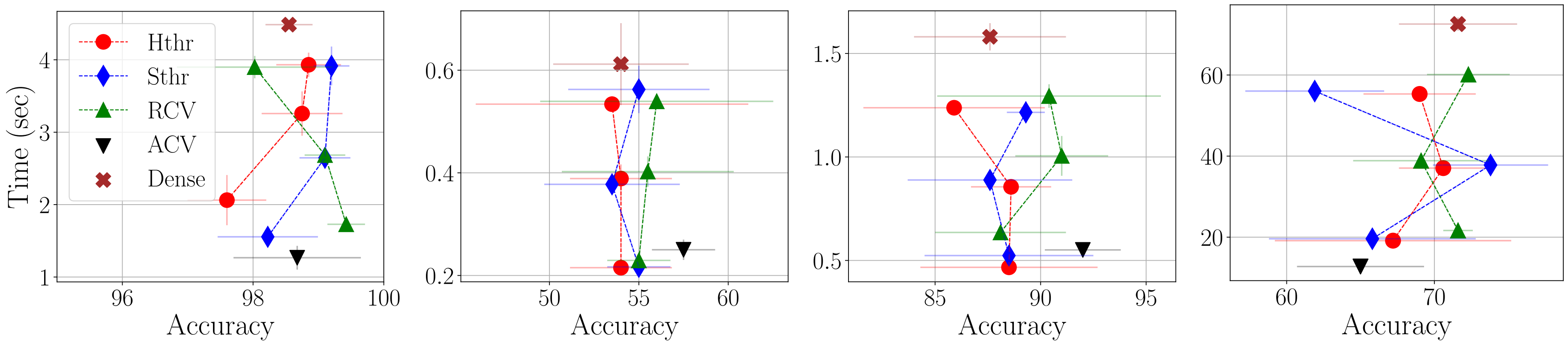
[Cavallo et al., 2024]

Sparse VNNs: Numerical results

- ➢ Classification task on real data

- ➢ **Datasets** (from left to right)

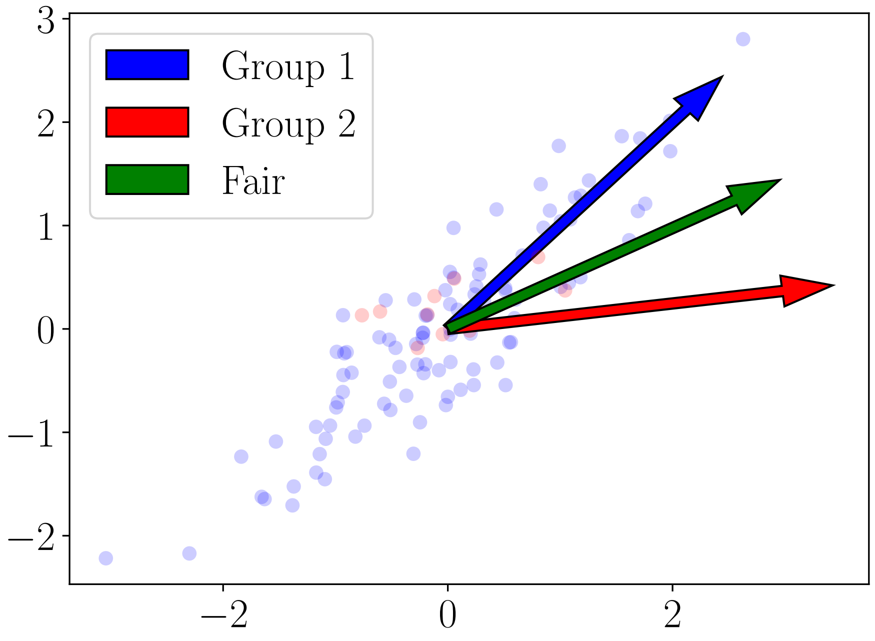
  - **Brain recordings:** Epilepsy and CNI – classify patient condition
  - **Human action recognition:** MHEALTH and Realdisp – classify action



- ➢ S-VNNs are **faster** and achieve **better performance** than *dense* VNNs

Limitations of VNNs - 2

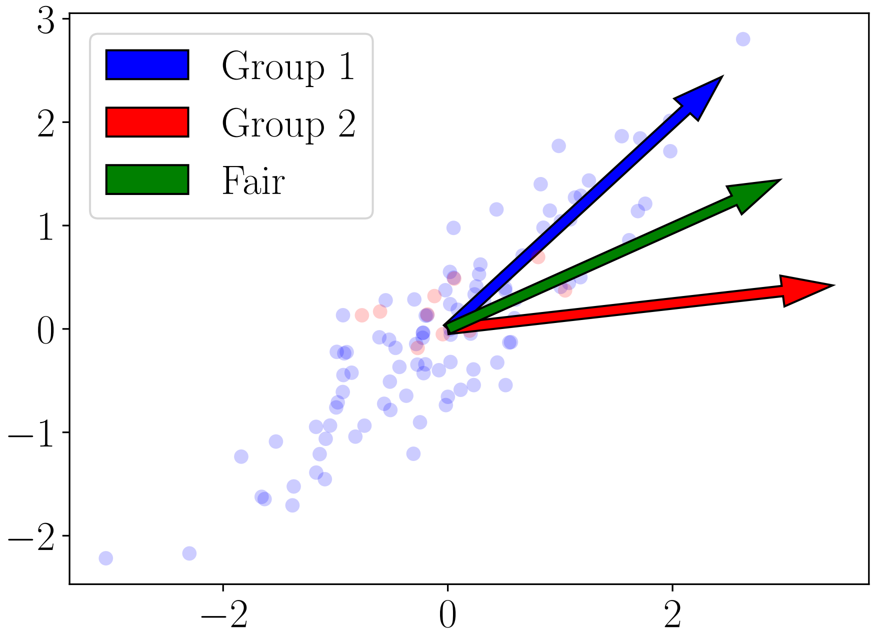
- Datasets may contain harmful **biases**
  - For e.g., under-represented groups
  - Biased (unfair) performance
  - Fair PCA might be **unstable**



[Cavallo et al., 2025]

Limitations of VNNs - 2

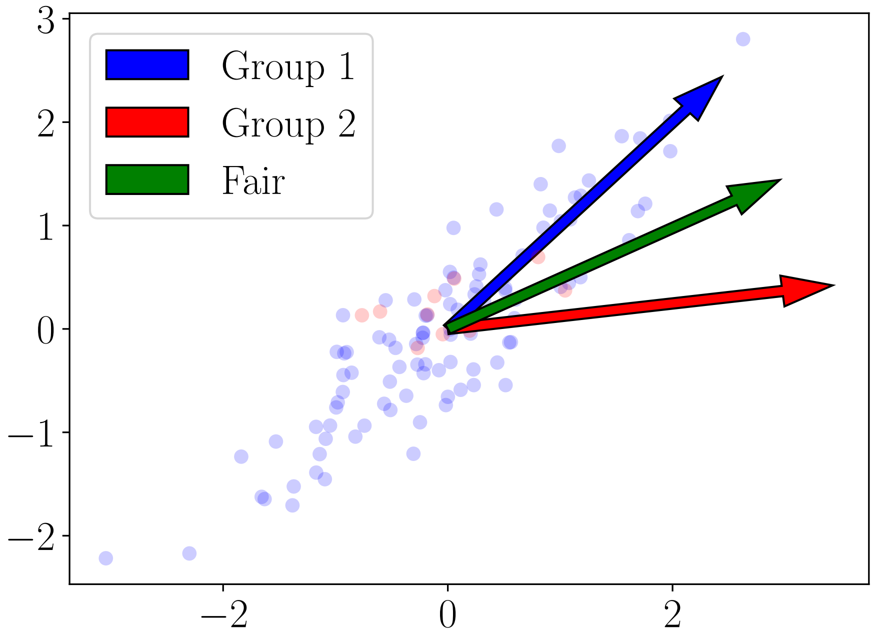
- Datasets may contain harmful **biases**
  - For e.g., under-represented groups
  - Biased (unfair) performance
  - Fair PCA might be **unstable**
- **Fair VNNs (F-VNNs)**
  - *Fairness*: parity in performance across groups within data



[Cavallo et al., 2025]

Limitations of VNNs - 2

- Datasets may contain harmful **biases**
  - For e.g., under-represented groups
  - Biased (unfair) performance
  - Fair PCA might be **unstable**



- **Fair VNNs (F-VNNs)**
  - *Fairness*: parity in performance across groups within data
  - How to make VNNs fair?
  - Are Fair VNNs stable?

questions to address

[Cavallo et al., 2025]

Fair covariance estimates

- **Balanced covariance**

For two groups g and h,

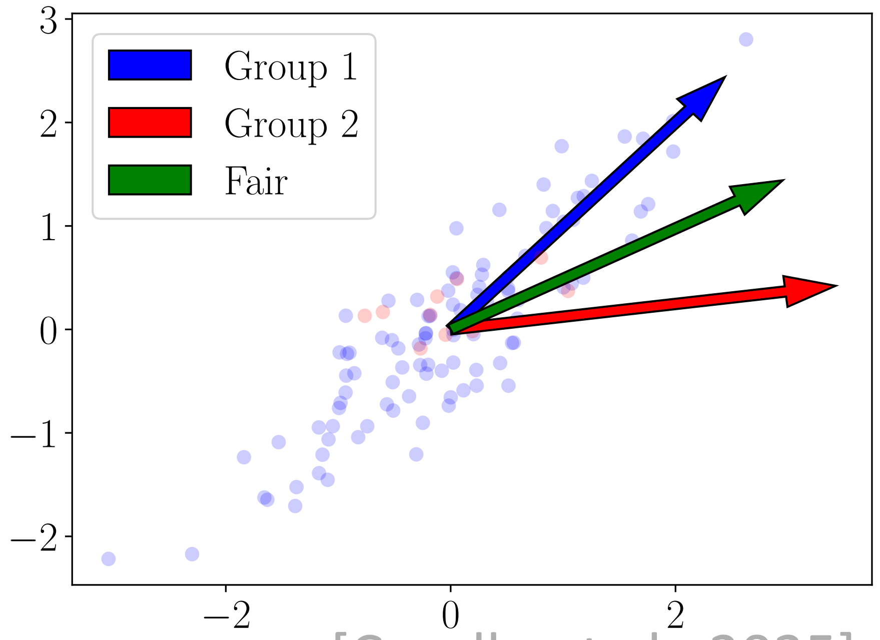
$$\hat{\mathbf{C}}_{\text{bal}} = \alpha\hat{\mathbf{C}} + (1-\alpha)(\hat{\mathbf{C}}_h - \hat{\mathbf{C}}_g) = \alpha_g\hat{\mathbf{C}}_g + \alpha_h\hat{\mathbf{C}}_h$$

- **Debiased covariance**

$$\hat{\mathbf{C}}_{\text{deb}} = \mathbf{X}^{\mathsf{T}}(\mathbf{I}_m + \beta\mathbf{Z}\mathbf{Z}^{\mathsf{T}})^{-1}\mathbf{X}/n$$

$\mathbf{X}$: data matrix

$\mathbf{Z}$: groups of samples



[Cavallo et al., 2025]

Bias-mitigation penalty

- F-VNNs are trained with a **loss penalty** that encourages fairness

$$\min_{\mathcal{H}} \; \gamma\mathcal{L}(\mathbf{X}, \mathbf{y}, \Phi) + (1-\gamma)\mathcal{R}(\mathbf{X}, \mathbf{y}, \mathbf{z}, \Phi)$$

$\mathcal{L}$: task-specific loss (for e.g., cross-entropy, MAE)

$\mathcal{R}$: bias penality (for e.g., performance difference across groups)

γ: balancing term

[Cavallo et al., 2025]

Stability of F-VNNs

- Fair covariance estimates
  - $\hat{\mathbf{C}}_{\text{deb}}$ and $\hat{\mathbf{C}}_{\text{bal}}$ are subject to covariance estimation errors
  - PCA with fair covariance estimates (Fair PCA) may be **unstable**

$\Rightarrow$ biased treatment

[Cavallo et al., 2025]

Stability of F-VNNs

- Fair covariance estimates
  - $\hat{\mathbf{C}}_{\text{deb}}$ and $\hat{\mathbf{C}}_{\text{bal}}$ are subject to covariance estimation errors
  - PCA with fair covariance estimates (Fair PCA) may be **unstable**

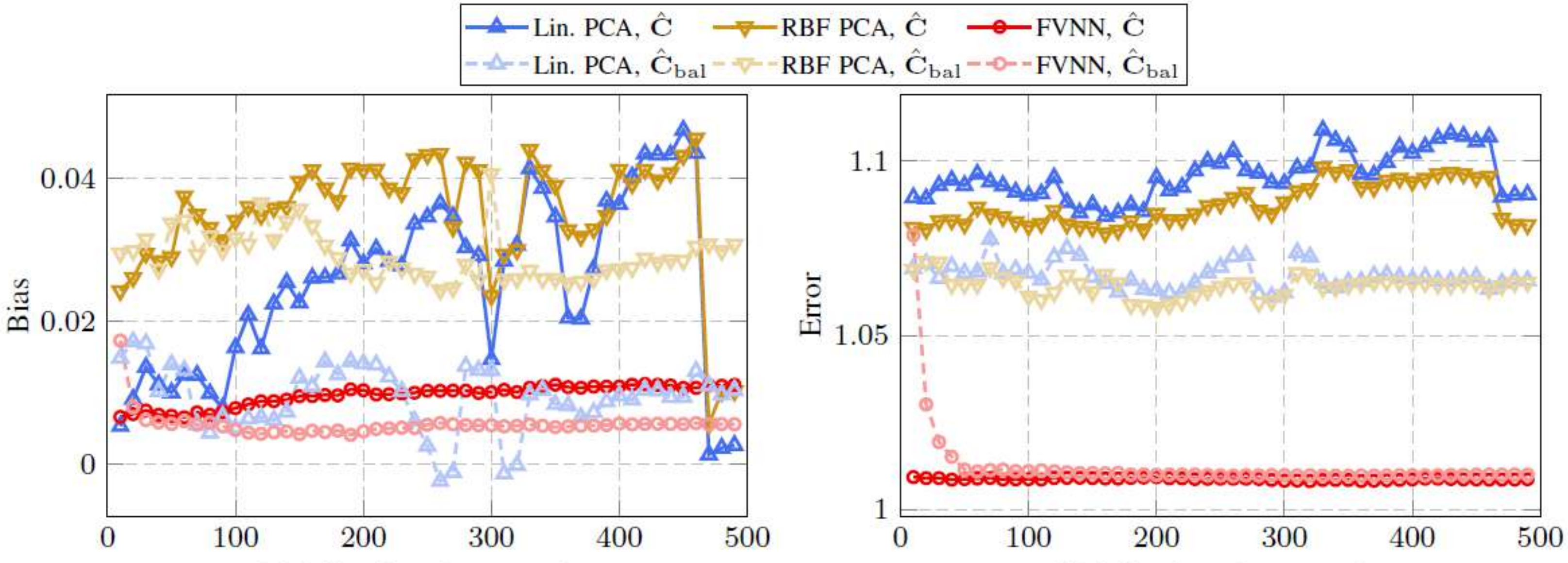
  $\Rightarrow$ biased treatment

- **F-VNNs are stable**
  - Stability of F-VNNs with **balanced** covariance $\propto \mathcal{O}\left(\frac{1}{n_g^{1/2}}\right) + \mathcal{O}\left(\frac{1}{n_h^{1/2}}\right)$
  - Stability of F-VNNs with **debiased** covariance $\propto \mathcal{O}\left(\frac{1}{n^{1/2}}\right)$

[Cavallo et al., 2025]

Numerical results

- **Stability**: synthetic biased data
  - Train on unbiased dataset
  - During test, replace covariance with unbalanced/fair version
  - Compare PCA+SVM with VNNs

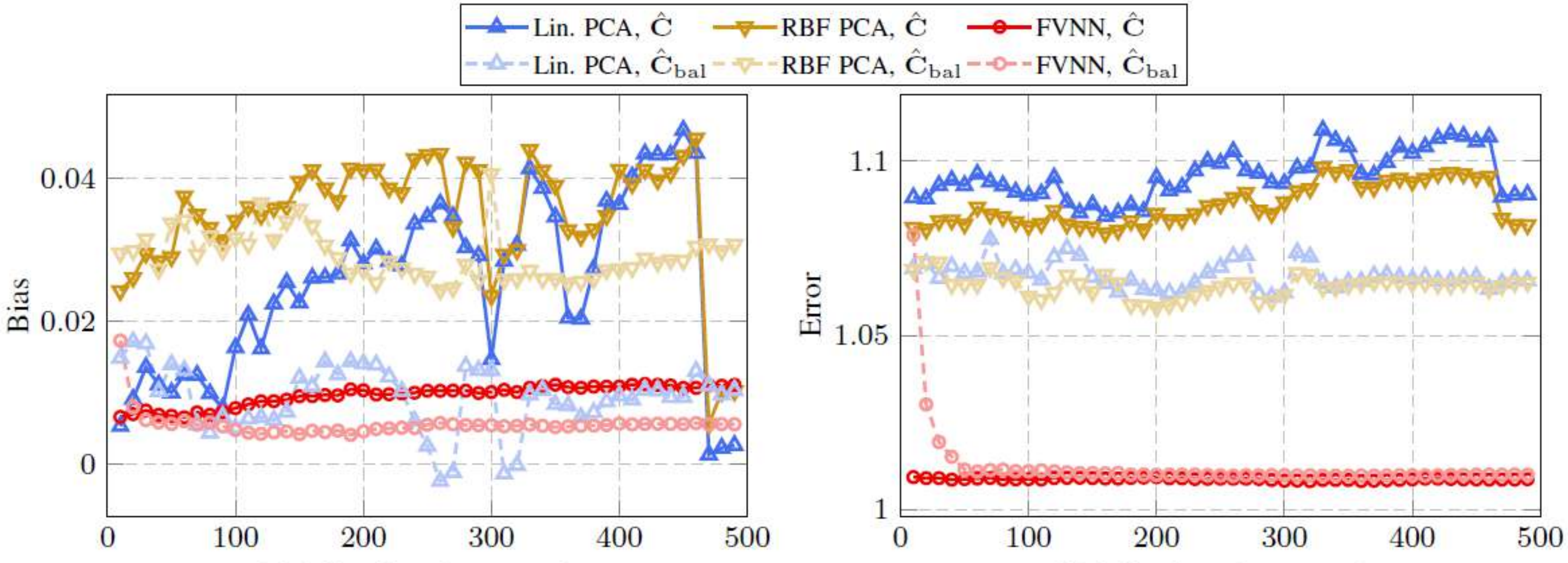


(a) Minority class samples (b) Minority class samples

[Cavallo et al., 2025]

Numerical results

- **Stability**: synthetic biased data
  - Train on unbiased dataset
  - During test, replace covariance with unbalanced/fair version
  - Compare PCA+SVM with VNNs



(a) Minority class samples

(b) Minority class samples

F-VNNs are more **stable**

F-VNNs achieve **less bias**

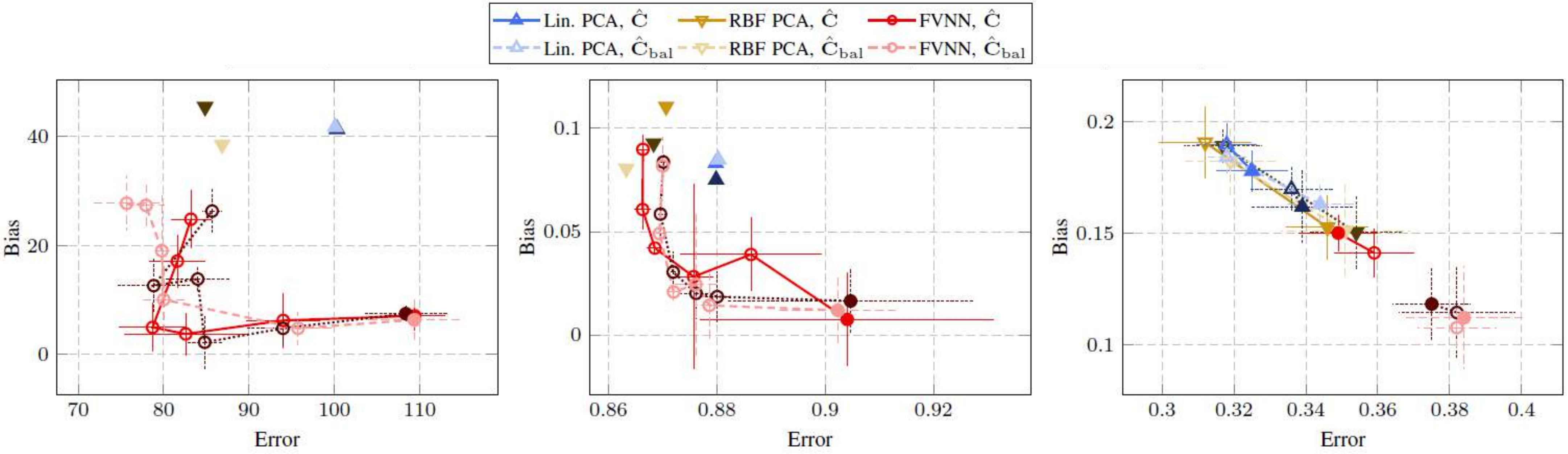
F-VNNs outperform PCA

[Cavallo et al., 2025]

Numerical results

- Real world datasets

Dataset	Description	Task	Sensitive attribute
Parkinson (left)	Medical records of patients	Regression for Parkinson's level	Sex of patient
LSAC (center)	Law school students' features	Regression for GPA	Race of students
German credit (right)	Features of individuals applying for credit	Classification (good or bad)	Sex of individual

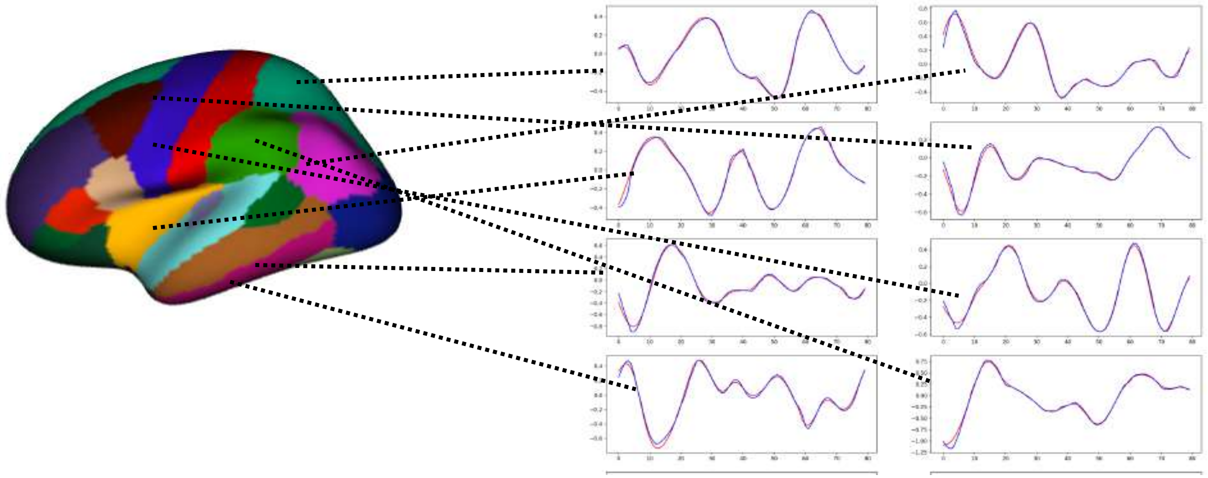


F-VNNs achieve better fairness and performance than PCA

[Cavallo et al., 2025]

Limitations-3

- VNN models discussed so far operate on *static* data
  - Real world applications have **dynamic** data
  - Non-trivial modifications needed to handle temporal, non-stationary data
  - Online estimates introduce additional source of errors

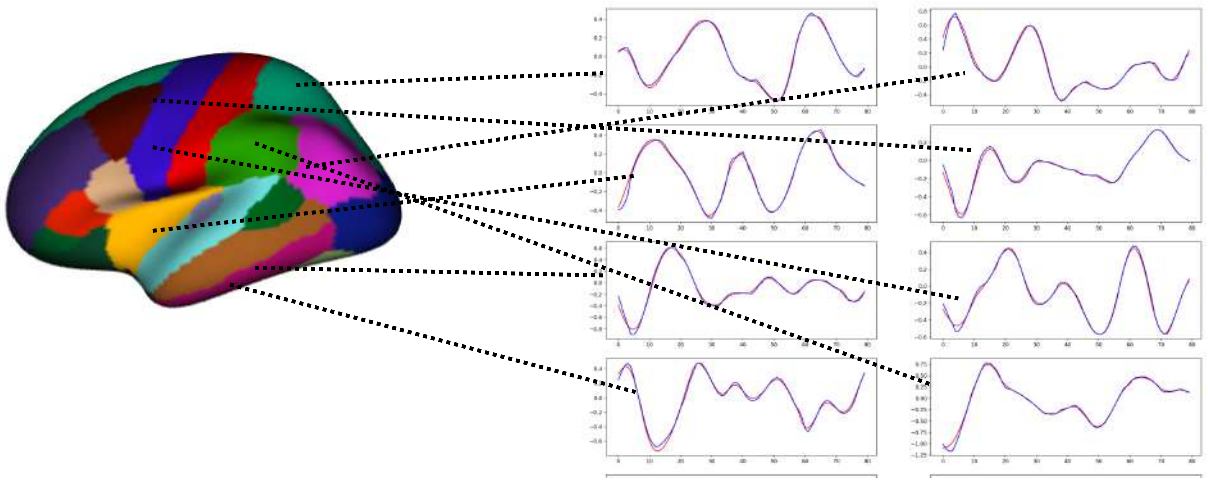


Limitations-3

- VNN models discussed so far operate on *static* data
  - Real world applications have **dynamic** data
  - Non-trivial modifications needed to handle temporal, non-stationary data
  - Online estimates introduce additional source of errors

- **Spatio-temporal VNNs (STVNNs)**

  VNNs for spatio-temporal datasets



Spatiotemporal VNNs

➢ **Model design**

- Online covariance matrix estimate

$$\hat{\mathbf{C}}_{t+1} = \zeta_t \hat{\mathbf{C}}_t + \beta_t (\mathbf{x}_{t+1})(\mathbf{x}_{t+1})^\mathsf{T}$$

[Cavallo et al., 2024]

Spatiotemporal VNNs

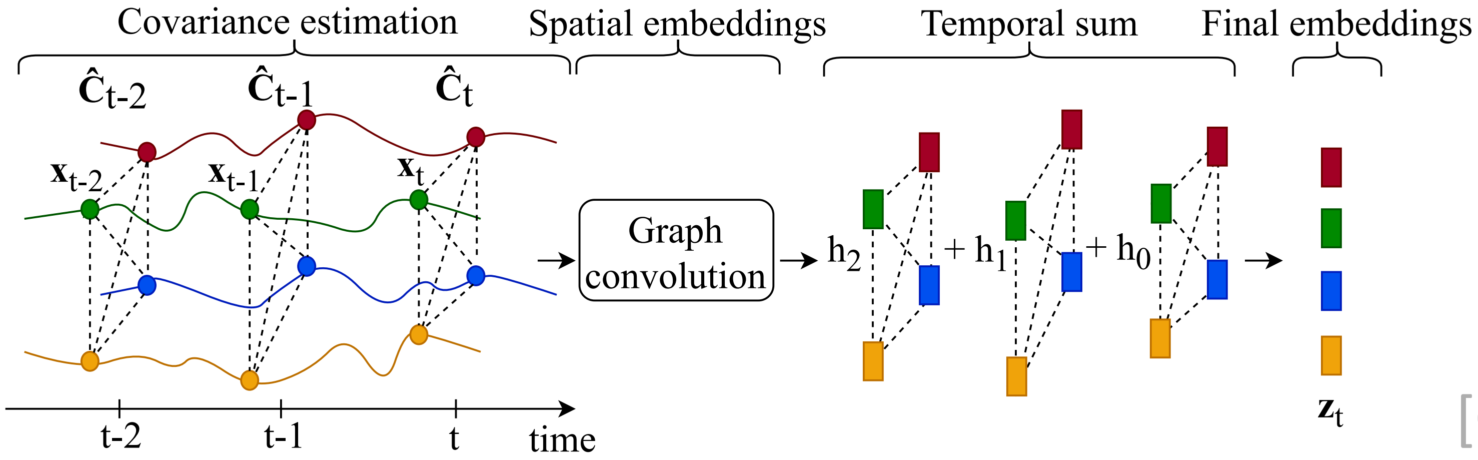
- **Model design**
  - Online covariance matrix estimate

$$\hat{\mathbf{C}}_{t+1} = \zeta_t \hat{\mathbf{C}}_t + \beta_t (\mathbf{x}_{t+1})(\mathbf{x}_{t+1})^{\mathsf{T}}$$

  - Spatio-temporal coVariance filter

$$\mathbf{z}_t := \mathbf{H}(\hat{\mathbf{C}}_t, \mathbf{h}_t, \mathbf{x}_{T:t}) = \sum_{t'=0}^{T-1} \sum_{k=0}^{K} h_{kt'} \hat{\mathbf{C}}_t^k \mathbf{x}_{t-t'}$$

Spatial and temporal convolution



[Cavallo et al., 2024]

Spatiotemporal VNNs

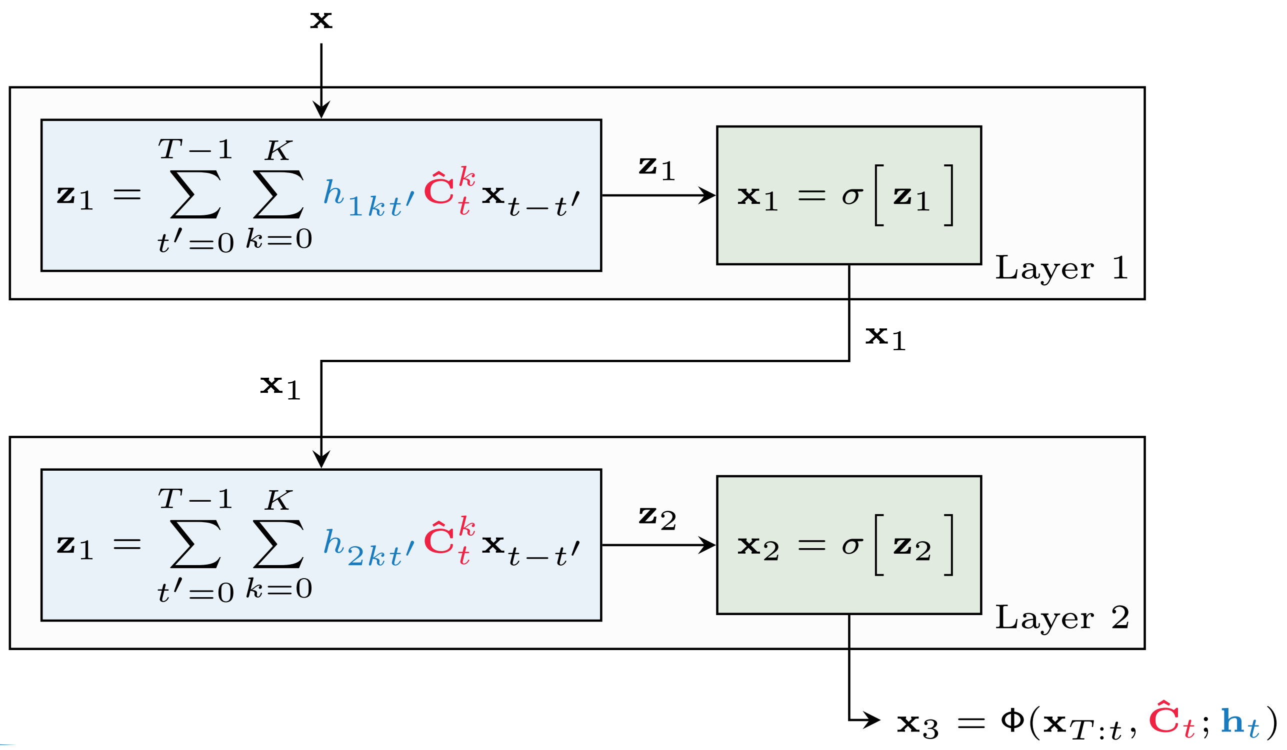
➢ STVNN

- Sequences of spatio-temporal covariance filters followed by non-linearity

$$\mathbf{z}_t^l = \sigma\left(\mathbf{H}^l(\hat{\mathbf{C}}_t, \mathbf{h}_t, \mathbf{z}_{T:t}^{l-1})\right)$$

- Online parameter updates

$$\mathbf{h}_{t+1} = \mathbf{h}_t - \eta\nabla_t\mathcal{L}(\Phi(\mathbf{x}_{T:t}, \hat{\mathbf{C}}_t; \mathbf{h}_t))$$



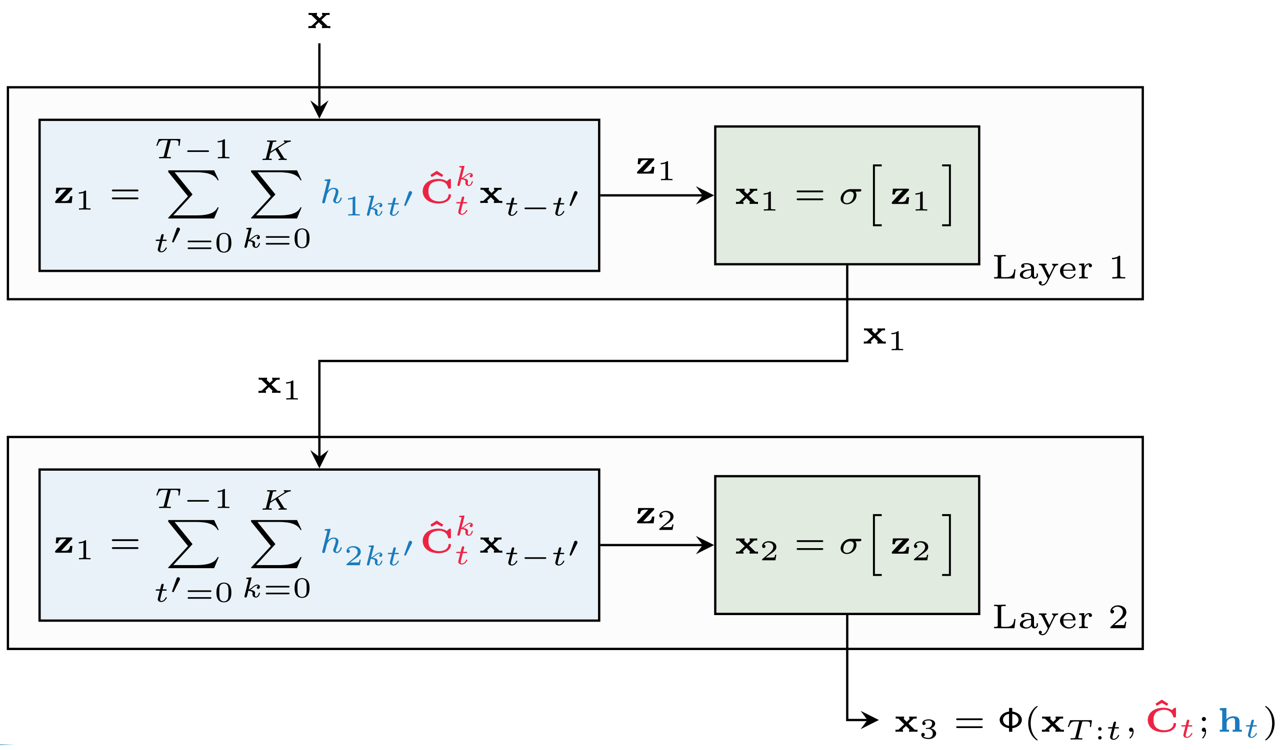
[Cavallo et al., 2024]

Spatiotemporal VNNs

➢ **STVNN**

- Sequences of spatio-temporal covariance filters followed by non-linearity

$$\mathbf{z}_t^l = \sigma\left(\mathbf{H}^l(\hat{\mathbf{C}}_t, \mathbf{h}_t, \mathbf{z}_{T:t}^{l-1})\right)$$



- Online parameter updates

$$\mathbf{h}_{t+1} = \mathbf{h}_t - \eta \nabla_t \mathcal{L}(\Phi(\mathbf{x}_{T:t}, \hat{\mathbf{C}}_t; \mathbf{h}_t))$$

- STVNNs are **stable**

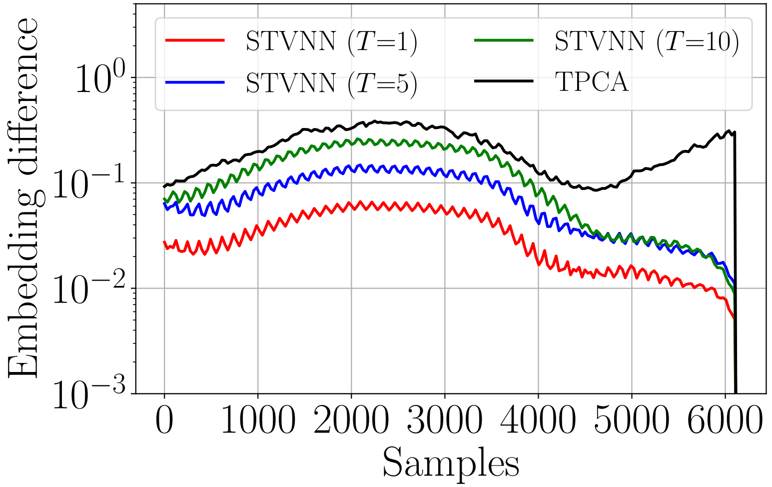
Stability bound $\propto \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$

[Cavallo et al., 2024]

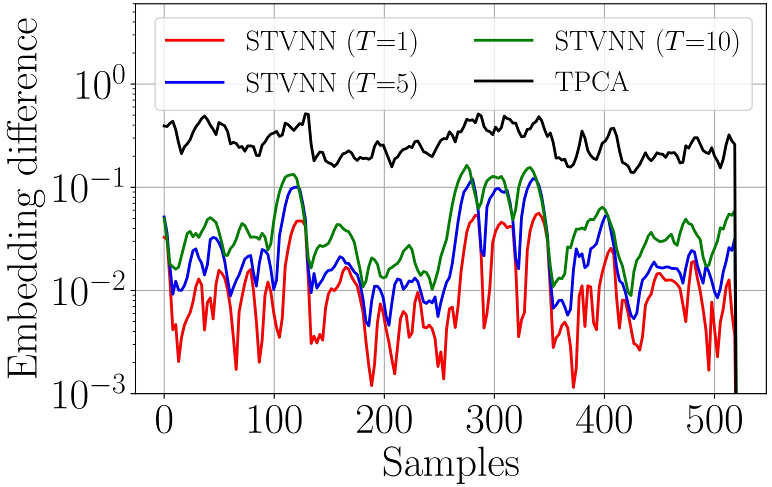
Numerical results

➢ **Time series forecasting task** (weather data and currency exchange rates)

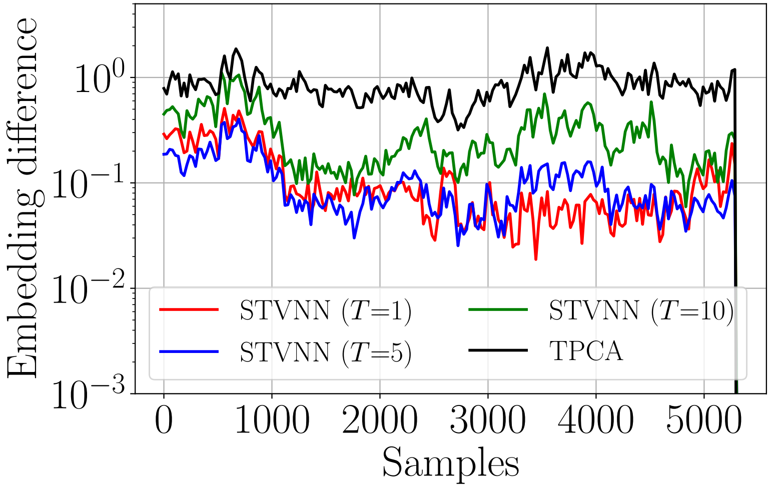
- Train with one covariance, test with another estimated from fewer samples



NOAA



Molene



Exchange rate

STVNNs are more stable than temporal-PCA (TPCA)

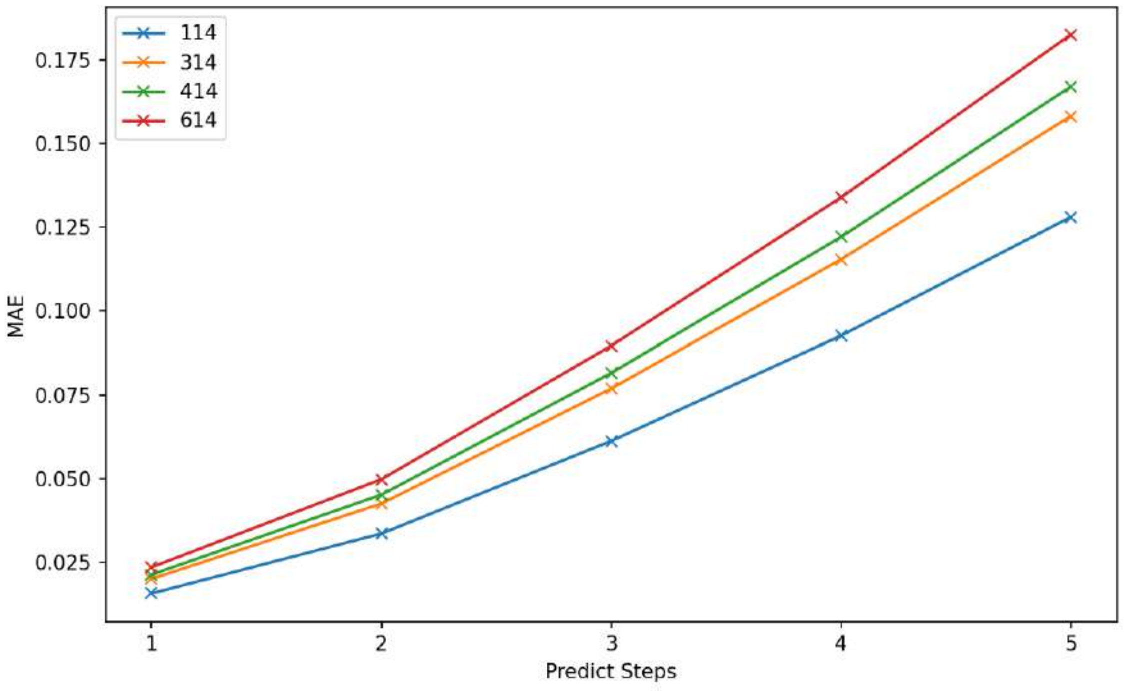
Higher T (temporal window size), lower stability

Numerical results

➢ **Time series forecasting task** (brain imaging data)

- **Data:** HCP Young-adult dataset
- BOLD data at at spatial scales

  of 114, 314, 414, 614 (Schaefer's)
- Train model on 314 resolution
  - Test on 114, 414, 614 resolutions

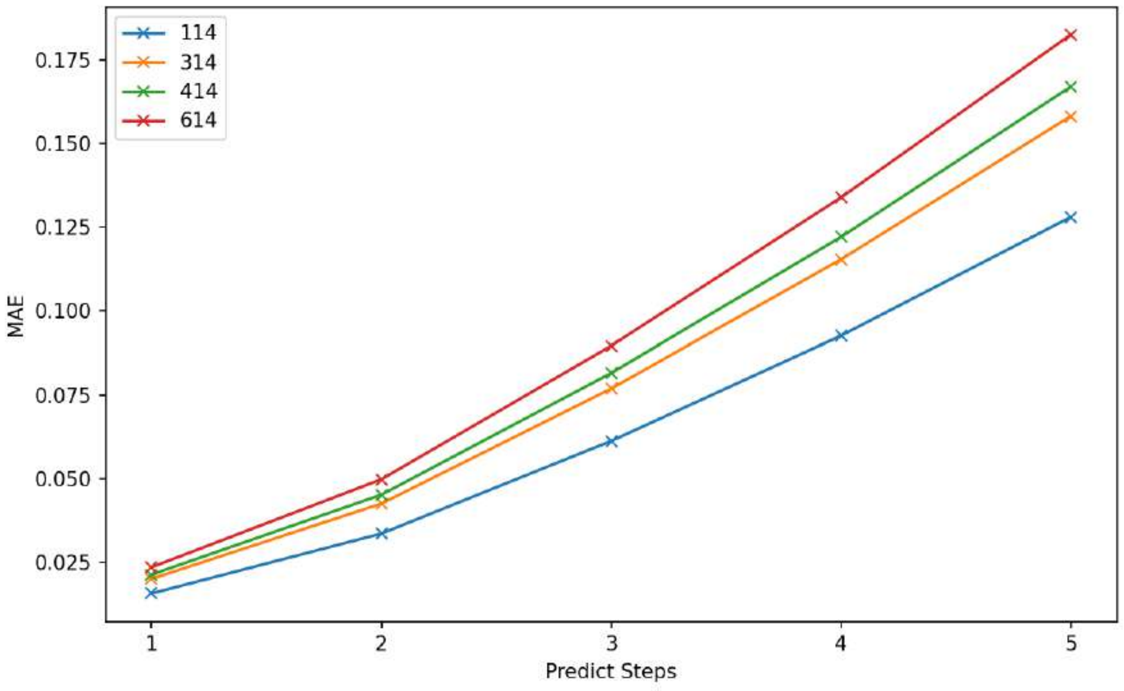


Numerical results

➢ **Time series forecasting task** (brain imaging data)

- **Data:** HCP Young-adult dataset
- BOLD data at at spatial scales

  of 114, 314, 414, 614 (Schaefer's)
- Train model on 314 resolution
  - Test on 114, 414, 614 resolutions



STVNN demonstrates **transferability** across **multi-scale spatio-temporal** datasets

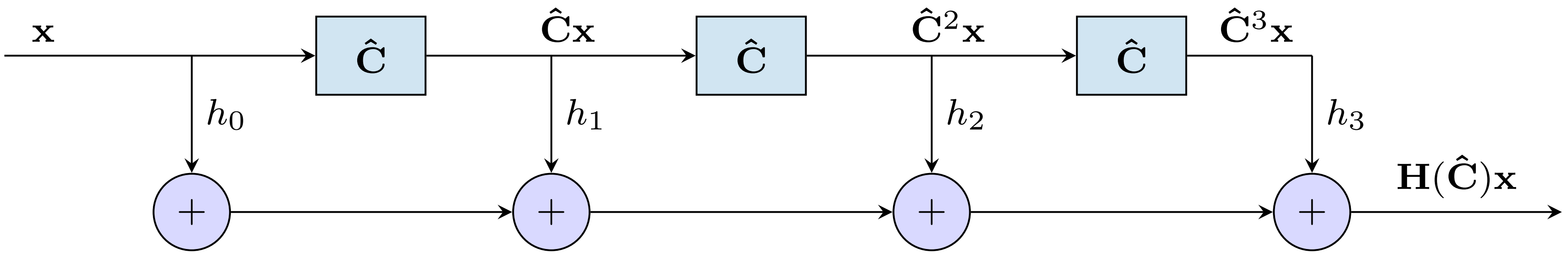
Conclusions and Future Directions

Covariance filters

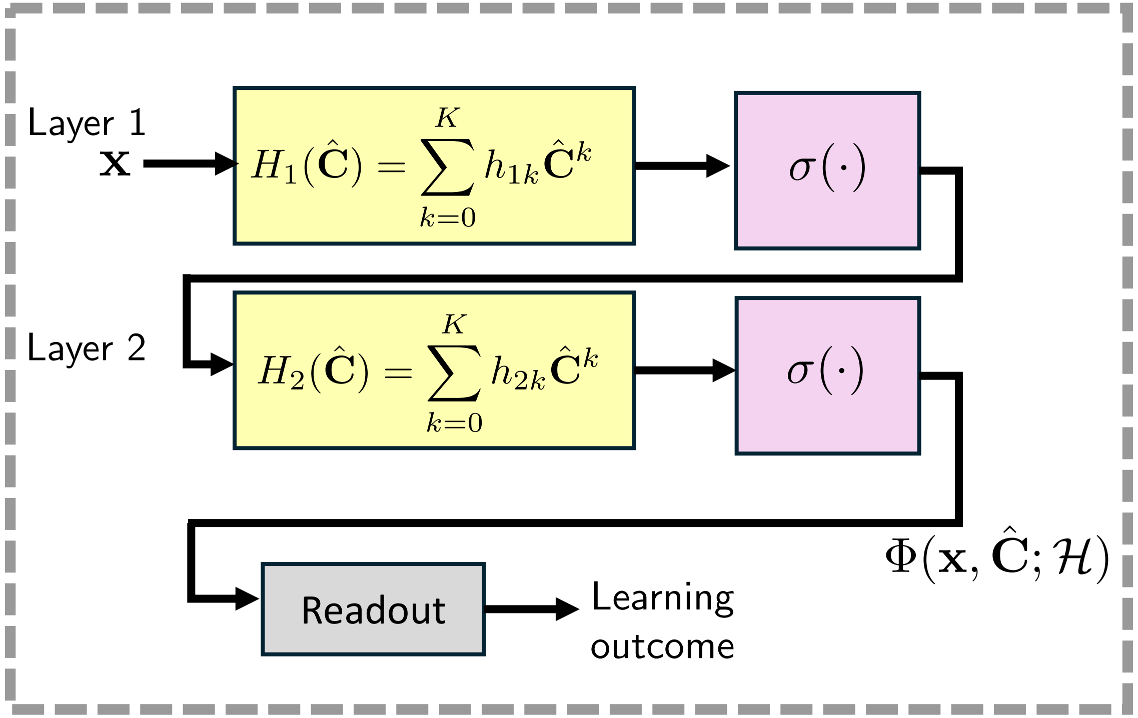
- A covariance filter is a polynomial in the covariance matrix $\hat{\mathbf{C}}$

$$\mathbf{H}(\hat{\mathbf{C}}) = \sum_{k=0}^{K} h_k \hat{\mathbf{C}}^k \mathbf{x}$$

- We train the filter coefficients h_k to accomplish some task



CoVariance Neural Networks (VNNs)



- A VNN is a composition of layers
- Each of which is a composition of

  … a covariance filter

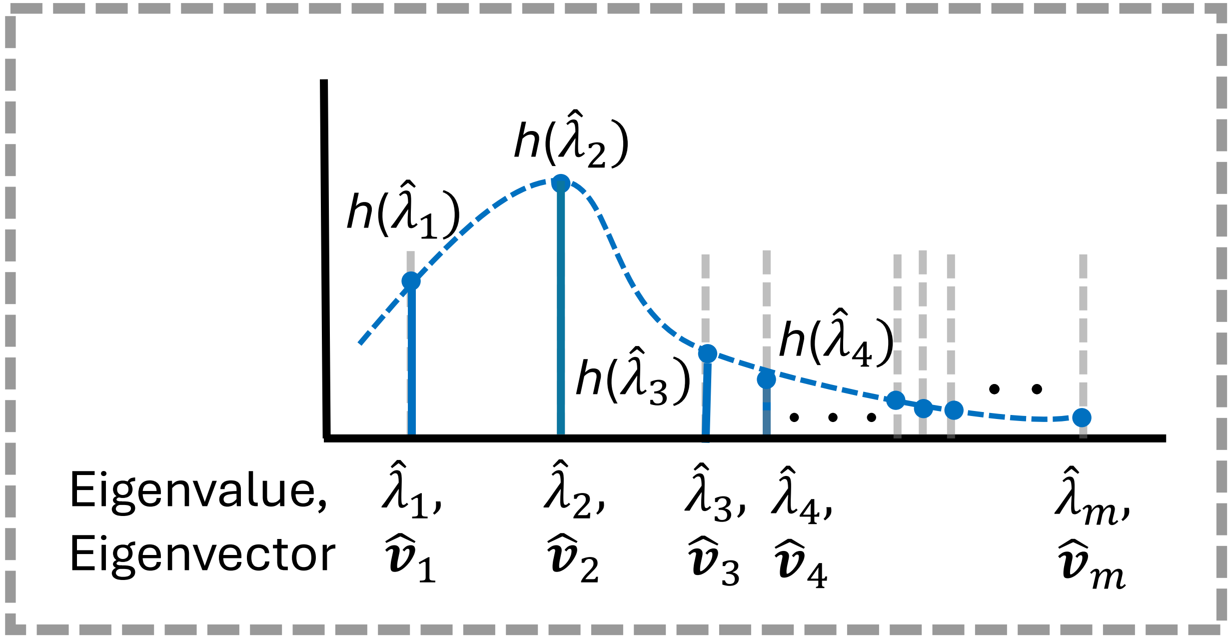
  … with a pointwise nonlinearity

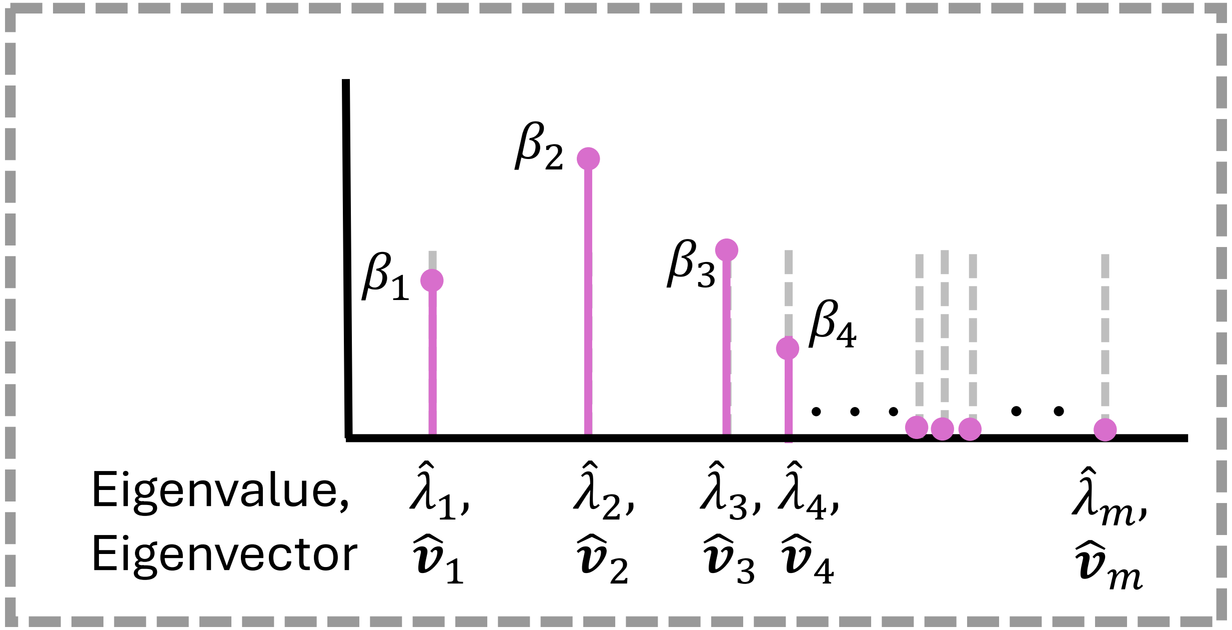
- $\Phi(\mathbf{x}; \hat{\mathbf{C}}, \mathcal{H})$ represents VNN output
- $\mathcal{H}$ is the set of trainable filter taps

Covariance Filters are **Implicitly** Equivalent to PCA

Input data → coVariance filter → Readout function → Inference output

Input data → PCA transform → Regression model → Inference output



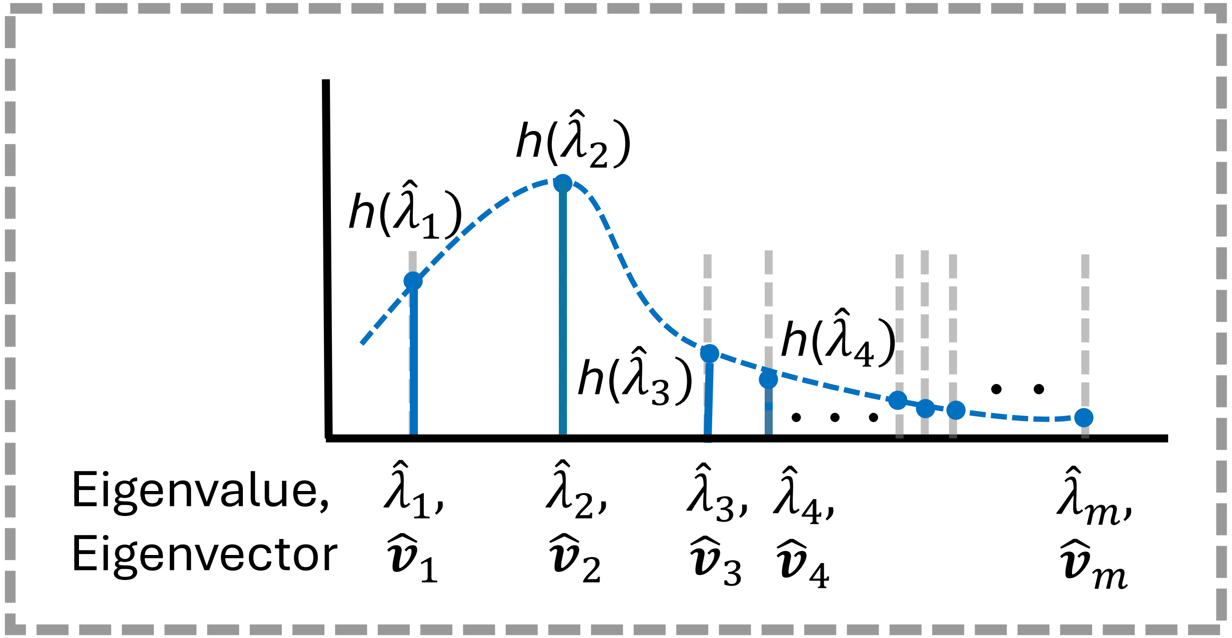


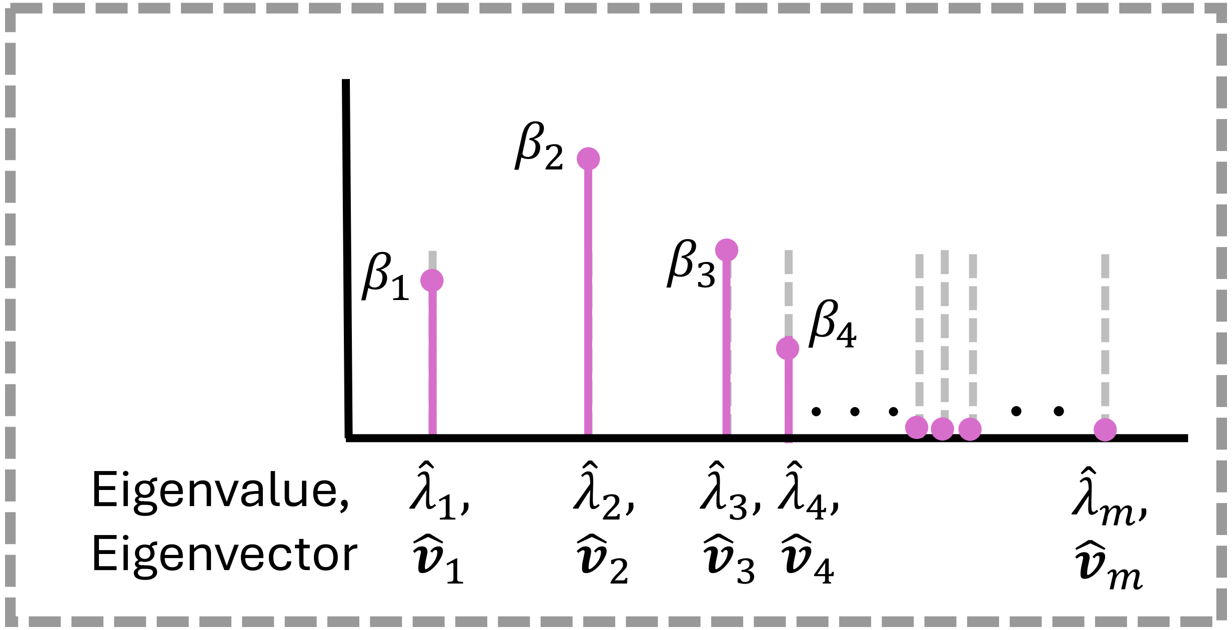
Covariance Filters are **Implicitly** Equivalent to PCA

- The difference is that covariance filters (and VNNs) do not require eigenvectors

  Stability: Leading to more stable signal processing

  Transferability: And the possibility of transferring trained filters across scales





Stability of coVariance filters and Neural Networks

- Outputs of coVariance filters NNs on true and estimated covariances are close

$$\left\|\mathbf{H}(\hat{\mathbf{C}}) - \mathbf{H}(\mathbf{C})\right\| = \mathcal{O}\left(\frac{1}{n^{1/2-\varepsilon}}\right) = \alpha_n \qquad \left\|\Phi(\mathbf{x}, \hat{\mathbf{C}}; \mathcal{H}) - \Phi(\mathbf{x}, \mathbf{C}; \mathcal{H})\right\| \leq LF^{L-1}\alpha_n$$

- Provided that the filters (at each layer) have Lipschitz frequency responses

$$|h(\lambda_i) - h(\lambda_j)| \leq Q\frac{|\lambda_i - \lambda_j|}{k_i}$$

- This requirement limits discriminability but it is a necessary limit

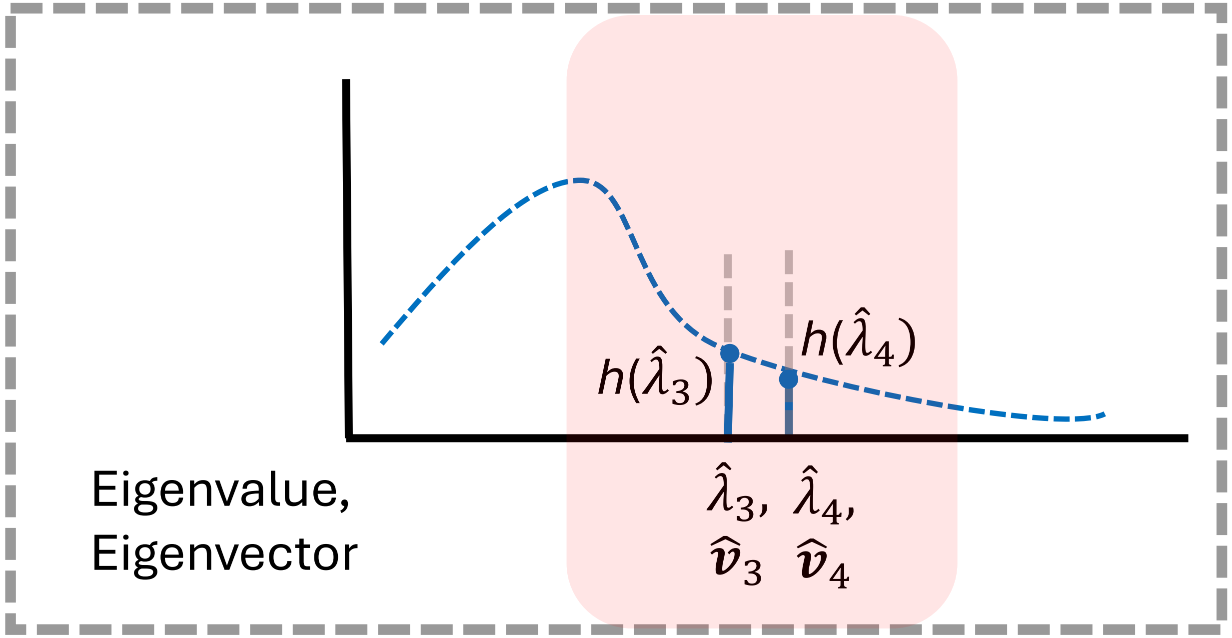
[Sihag et al., 2022]

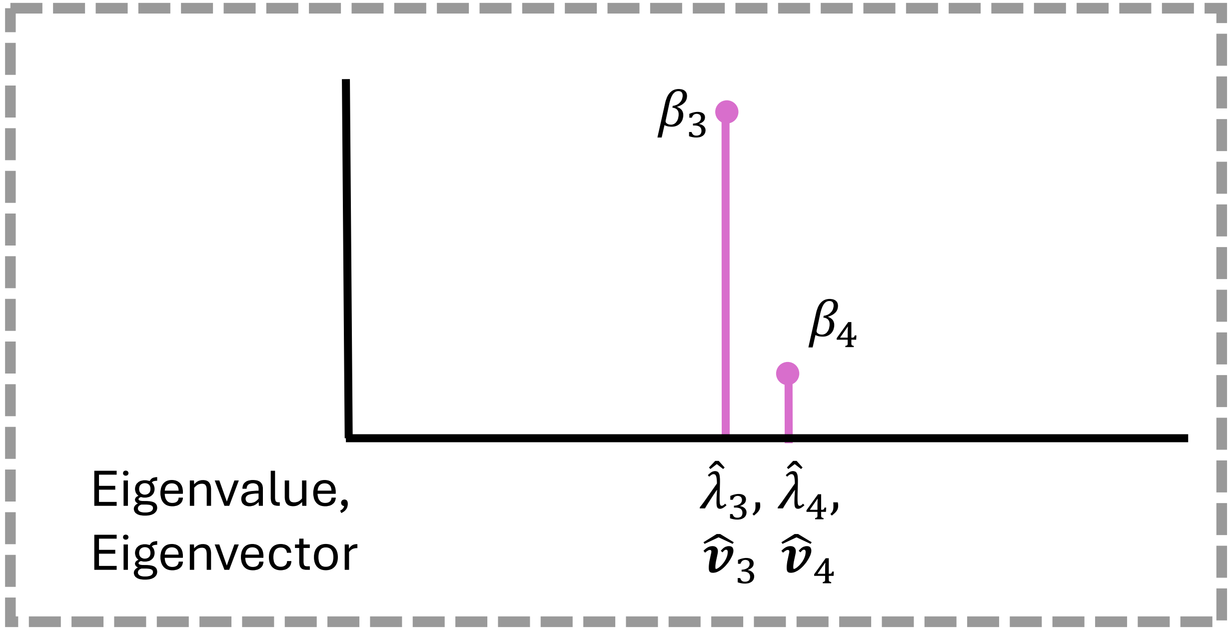
PCA responds catastrophically to eigenspace estimation

- Difference between true and estimated eigenvectors can be arbitrarily different

$$\|\hat{\mathbf{V}}\mathbf{x} - \mathbf{V}\mathbf{x}\| = \mathcal{O}\left(\frac{1}{n^{1/2}\min_{i\neq j}|\lambda_i - \lambda_j|}\right)$$

- Filters process similar eigenvalues with similar coefficients



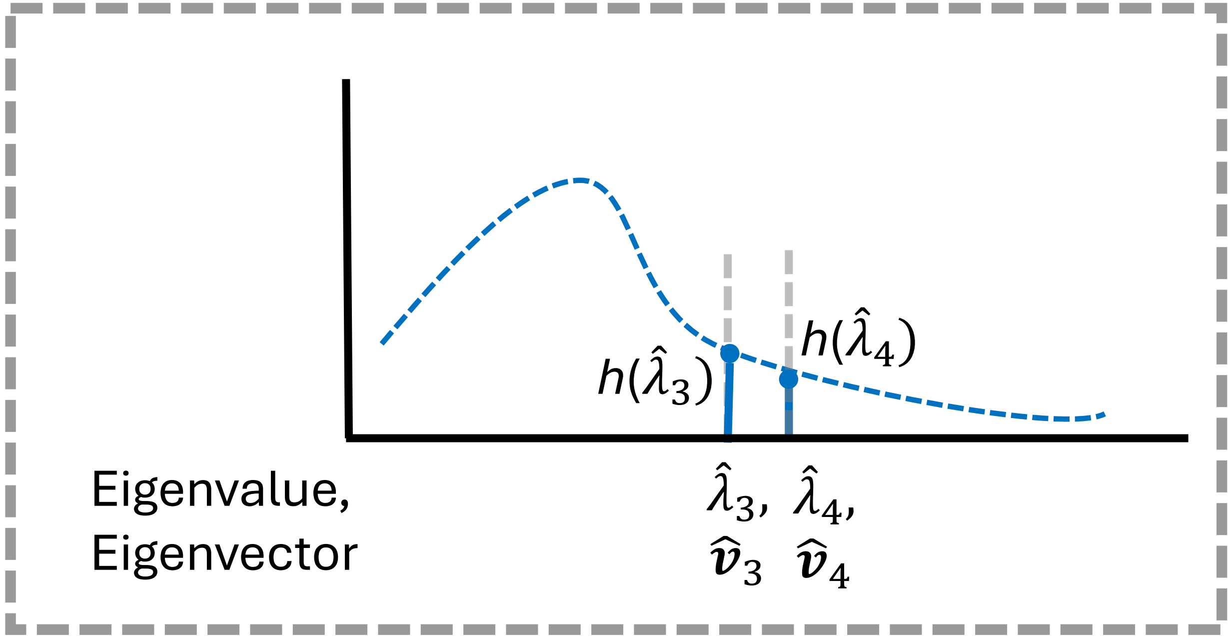


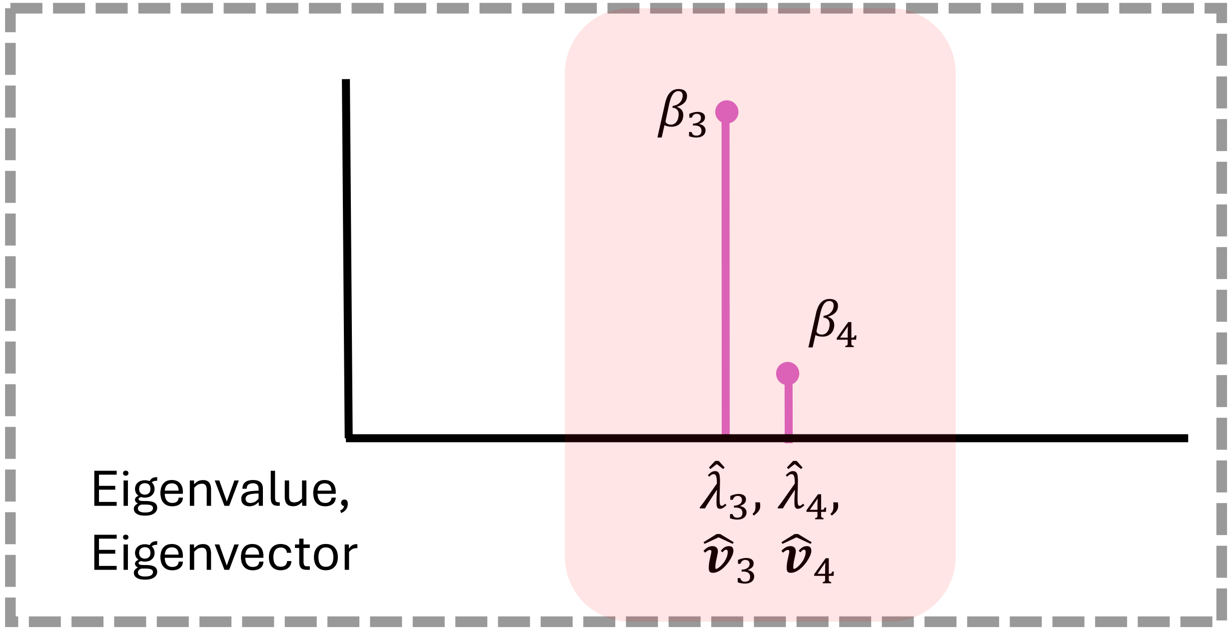
PCA responds catastrophically to eigenspace estimation

- Difference between true and estimated eigenvectors can be arbitrarily different

$$\|\hat{\mathbf{V}}\mathbf{x} - \mathbf{V}\mathbf{x}\| = \mathcal{O}\left(\frac{1}{n^{1/2}\min_{i\neq j}|\lambda_i - \lambda_j|}\right)$$

- Eigenvectors with similar eigenvalues can be processed with dissimilar coefficients



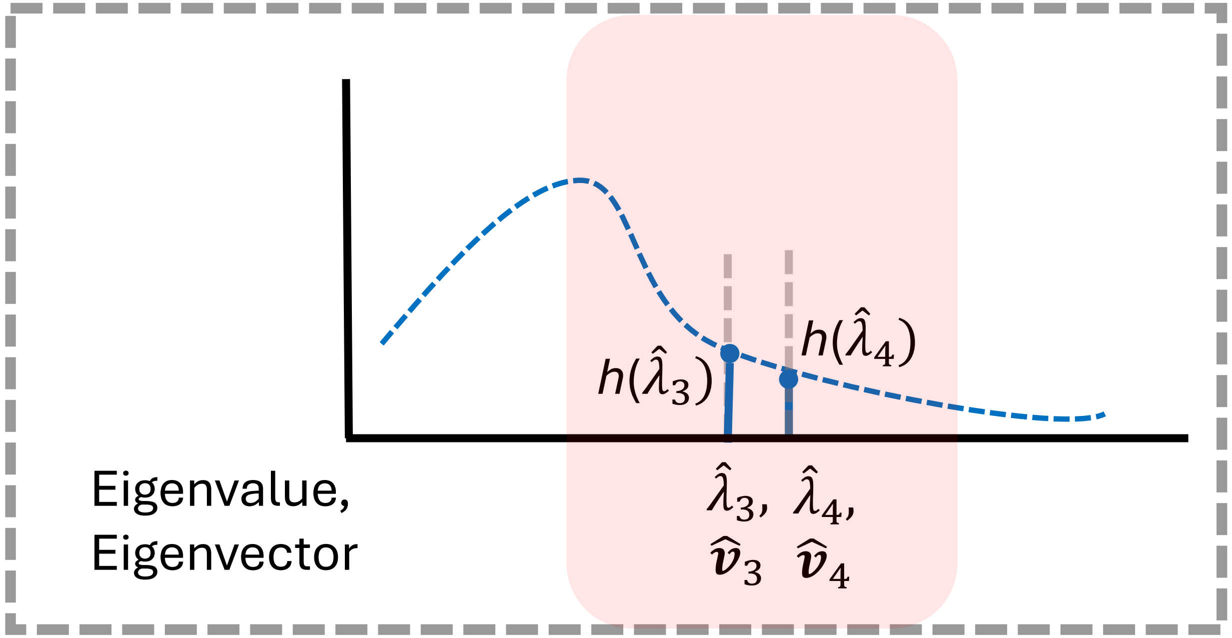


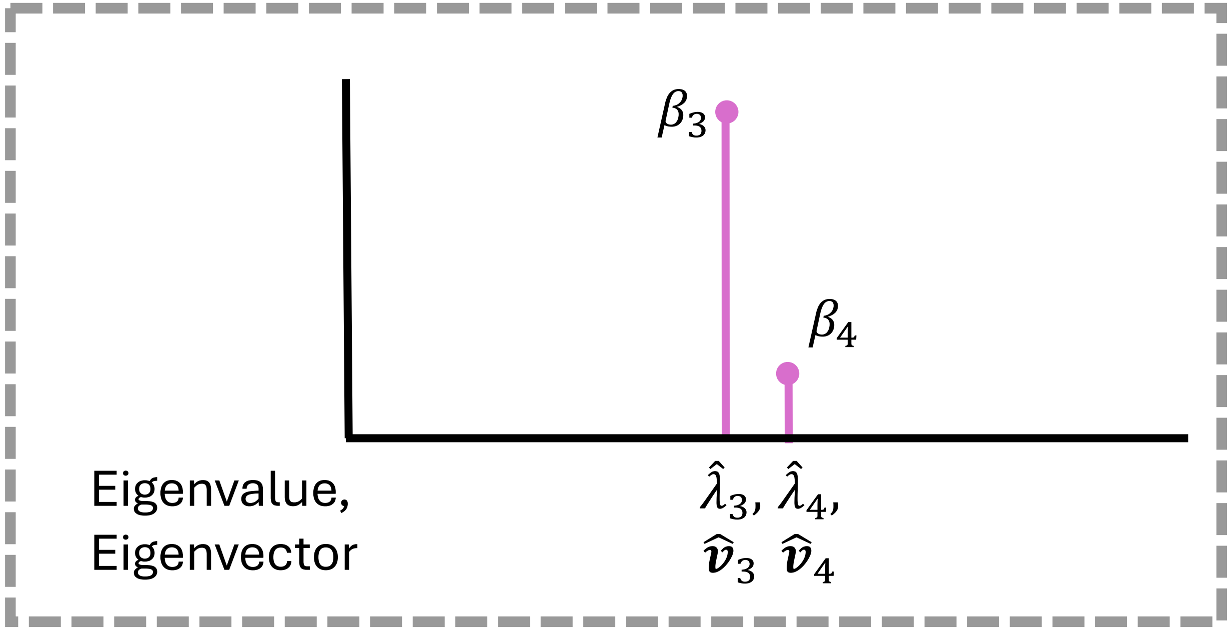
PCA responds catastrophically to eigenspace estimation

- Difference between true and estimated eigenvectors can be arbitrarily different

$$\|\hat{\mathbf{V}}\mathbf{x} - \mathbf{V}\mathbf{x}\| = \mathcal{O}\left(\frac{1}{n^{1/2}\min_{i\neq j}|\lambda_i - \lambda_j|}\right)$$

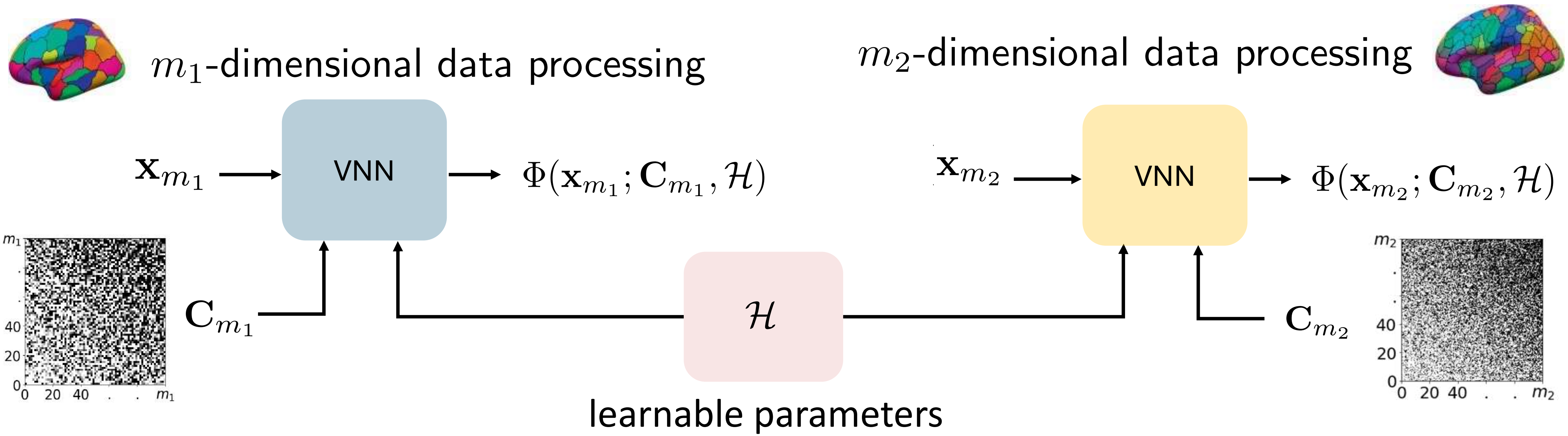
- Filters process similar eigenvalues with similar coefficients



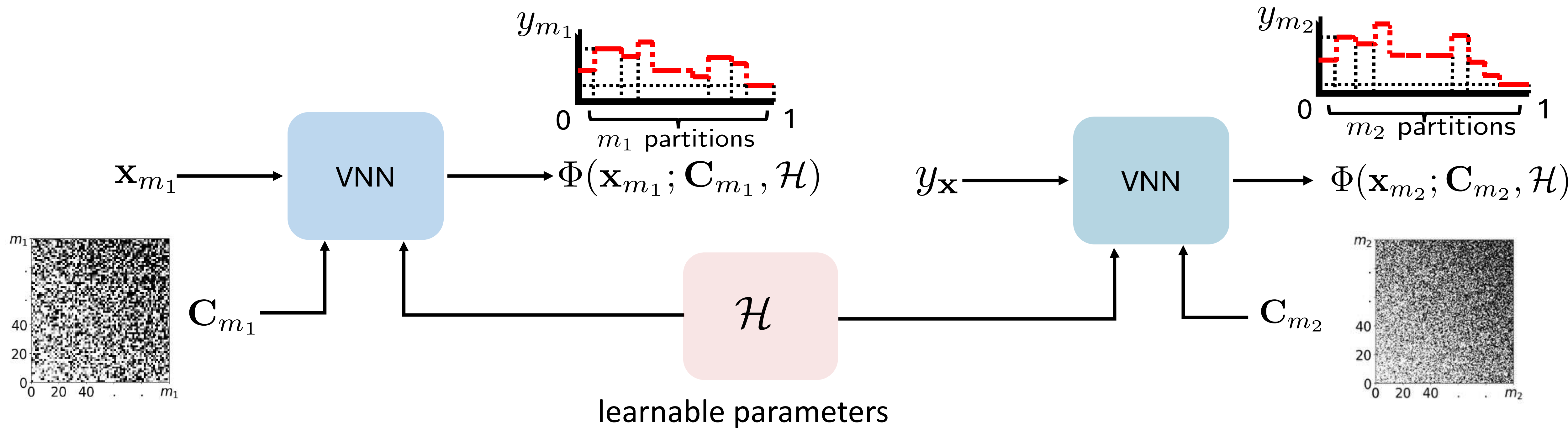


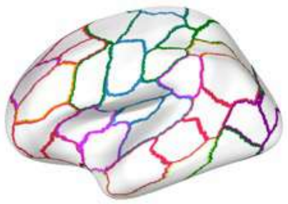
coVariance Filters and VNNs are Scale-Free Models

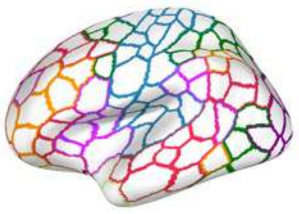
- Filters and VNNs are defined by coefficients that we can transfer across scales
  - Train at small scale and transfer to large scale
  - Train jointly across a heterogeneous range of scales



VNNs are provably transferable



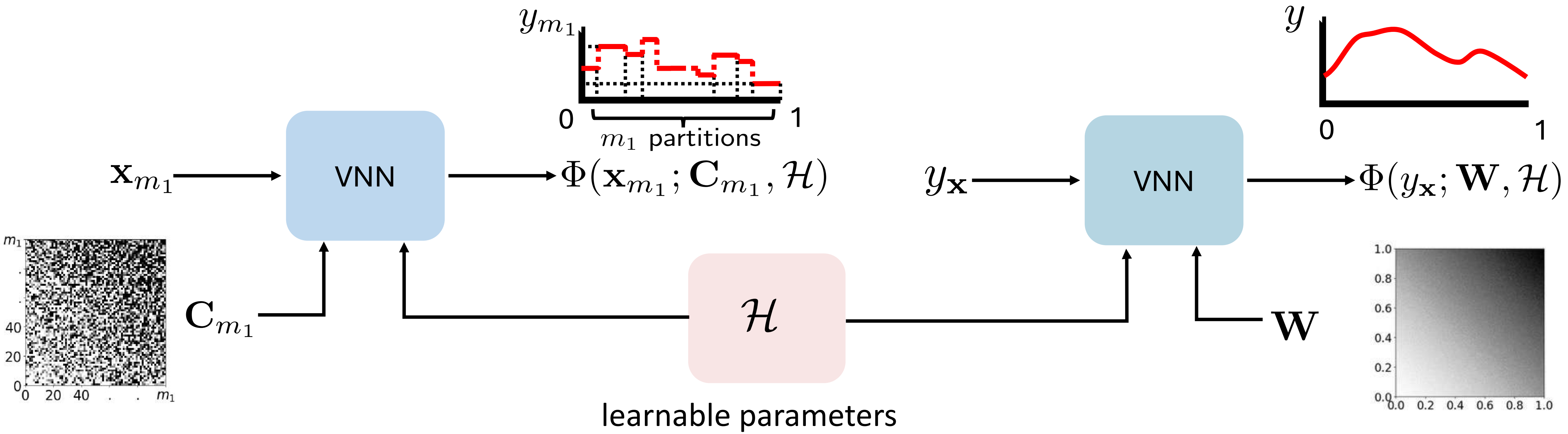




Transferability bound

$$\|y_{m_1} - y_{m_2}\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}} + \frac{1}{m_2^{3\zeta/2-1}}\right), \quad \text{for } \zeta \in (2/3, 1]$$

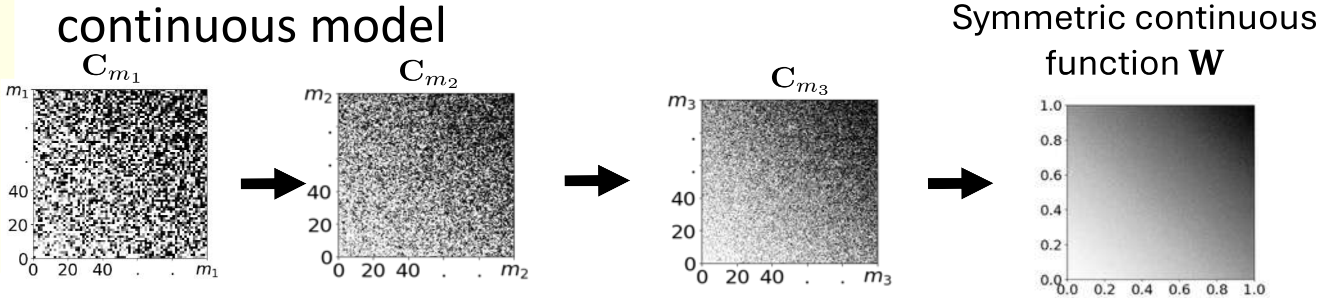
VNNs are provably transferable to limit models



Transferability bound* [Sihag et al., 2024]

$$\|y_{m_1} - y\| \propto \mathcal{O}\left(\frac{1}{m_1^{3\zeta/2-1}}\right) \quad , \text{ for } \zeta \in (2/3, 1]$$

***Assumption**: data is a discretization of a common continuous model



VNNs are well suited for neuroimaging data analysis

➢ Properties of VNNs make them appealing for neuroimaging data analysis

- **Connections with PCA** ⇨ transparent outcomes by leveraging spectrum of covariance matrix
- **Stability** ⇨ reproducible outcomes in limited data settings
- **Transferability** ⇨ enhanced generalizability and robustness to choice of brain atlases

Brain age gap prediction is a transfer learning problem

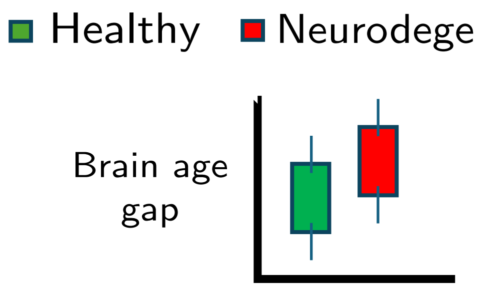
- Train ML model to predict age on a large dataset (**healthy** population)

Pre-training

Neuroimaging data from healthy population → ML Model → predicted age

- Apply the **pre-trained** ML model on a target dataset (**neurodegeneration**)

Data from all cohorts → ML Model →



- Brain age gap is the **residual** of the model

A principled approach to brain age gap prediction

- **Focus on residuals** of the ML model, not prediction performance
- **Qualitative evaluation** during pre-training

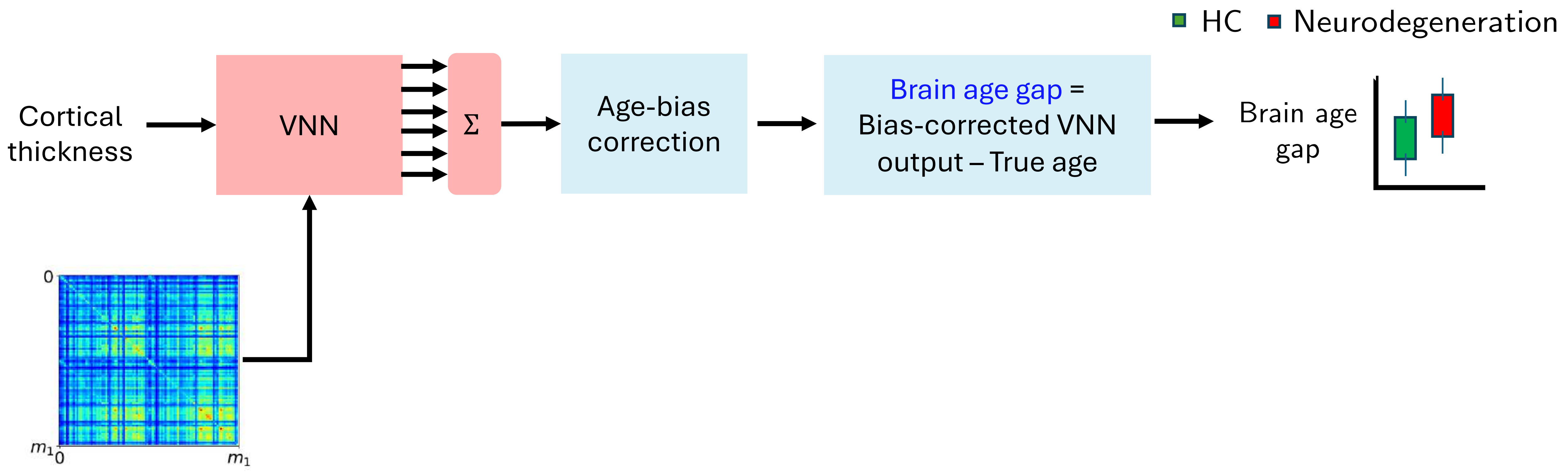
  what does the model learn during pre-training on healthy population?

- **Interpretability/explainability:**

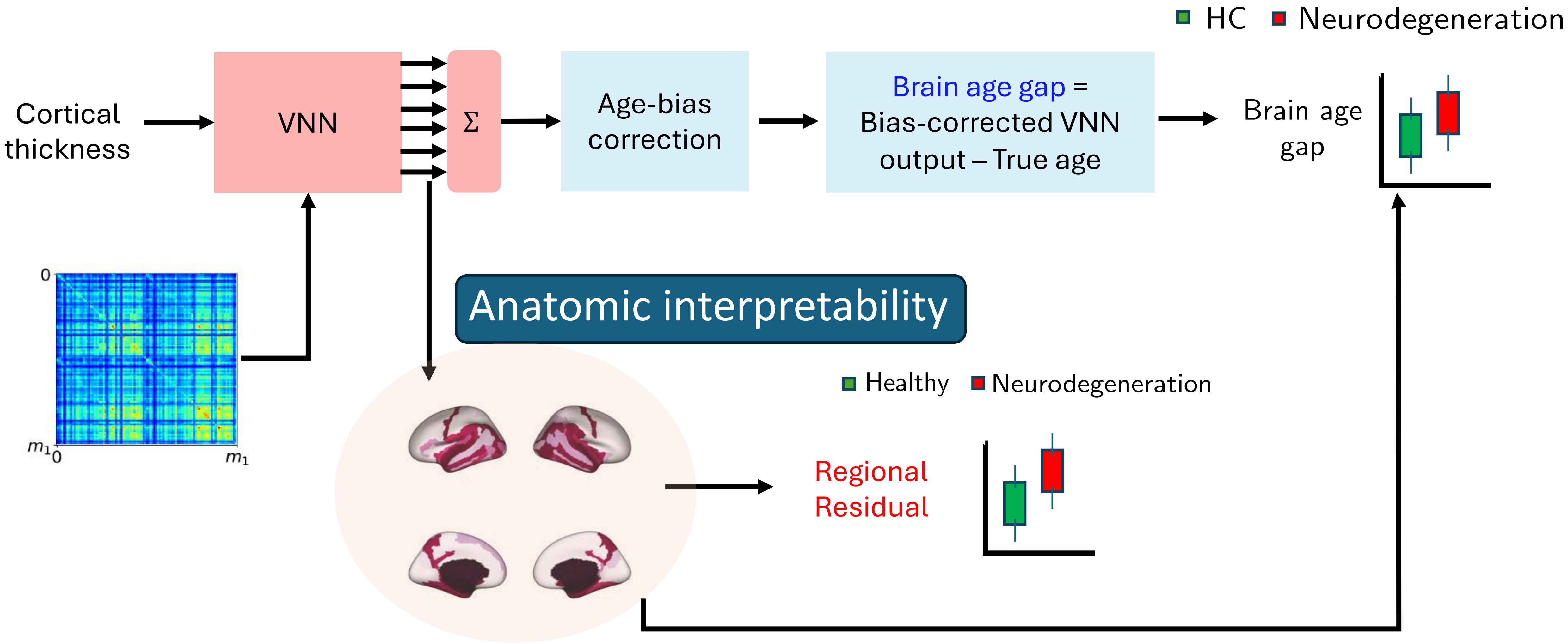
  what's driving elevated brain age gap (residuals) in neurodegeneration?

- **Generalizability** to diverse target populations

VNNs provide an anatomically interpretable and explainable brain age gap



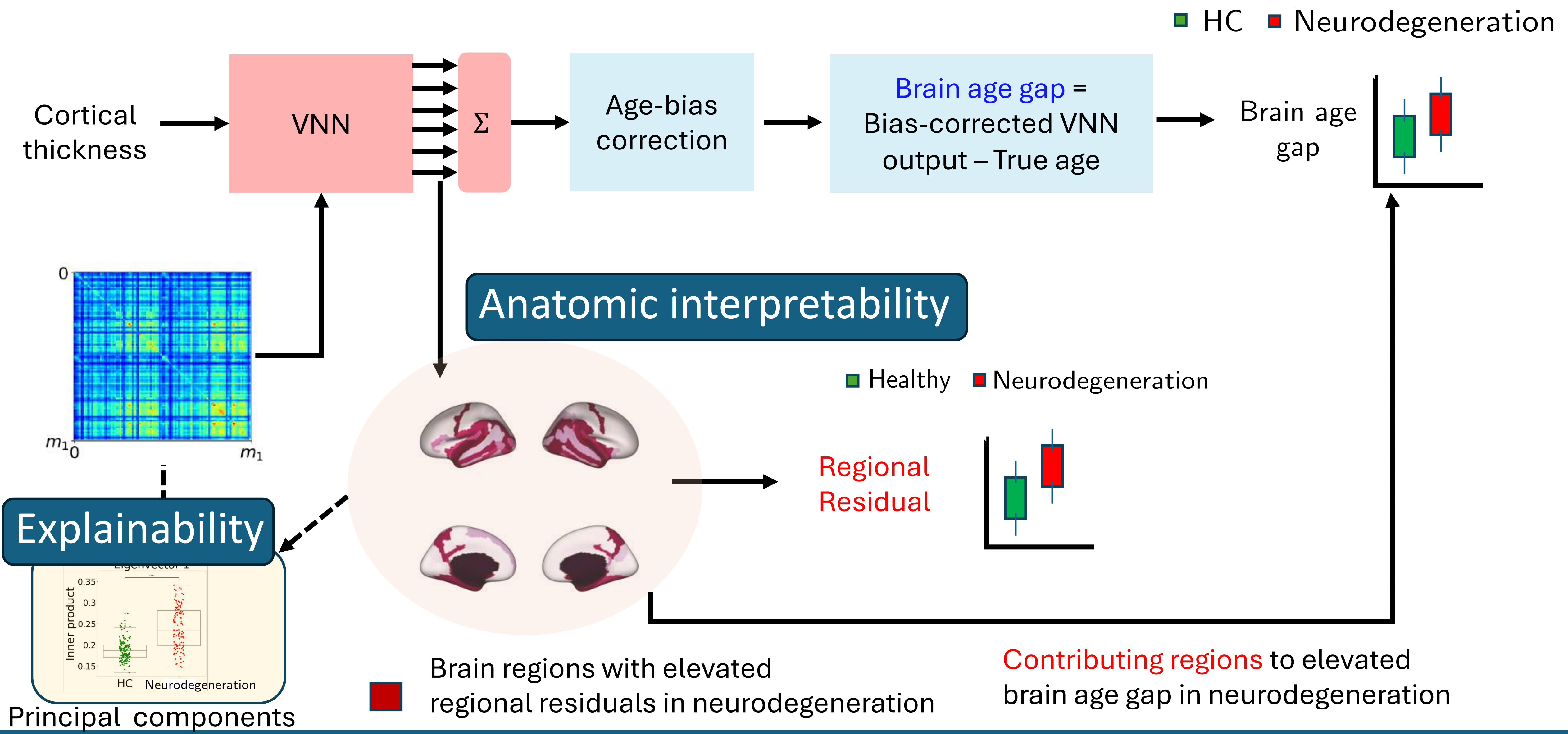
VNNs provide an anatomically interpretable and explainable brain age gap



Brain regions with elevated regional residuals in neurodegeneration

Contributing regions to elevated brain age gap in neurodegeneration

VNNs provide an anatomically interpretable and explainable brain age gap



Concluding Remarks

- **Emerging areas**
  - **Sparse VNNs:** sparsifying covariance matrix [Cavallo et al., 2024]
  - **Spatiotemporal VNNs:** temporal datasets [Cavallo et al., 2024]
  - **Fair VNNs:** unbiased outcomes with VNNs [Cavallo et al., 2025]
  - **Optimality of covariance matrices:** suitability of covariance to learning task [Khalafi et al., 2024]
  - **Application to brain age gap prediction** [Sihag et al., 2024; 2025]

Future directions

- Learning with **cross-covariance** graphs
  - Links with partial least squares/ canonical correlation analysis
- Expand **interpretability/explainability** of VNNs
  - How are eigenvectors exploited in STVNNs on dynamic datasets?
- Building **interpretable biomarkers**
  - Using other modalities (for e.g., fMRI)

References

Sihag, Saurabh, Mateos, Gonzalo, C. McMillan, and Ribeiro, Alejandro, “coVariance neural networks,” in Proc. Conference on Neural Information Processing Systems, Nov. 2022.

Saurabh Sihag, Gonzalo Mateos, C. McMillan, and Alejandro Ribeiro, “Explainable brain age prediction using covariance neural networks,” in Proc. Conference on Neural Information Processing Systems, 2023.

Saurabh Sihag, Gonzalo Mateos, and Alejandro Ribeiro, “Disentangling neurodegeneration with brain age gap prediction models: A graph signal processing perspective,” in IEEE Signal Processing Magazine, 2025 (to appear).

Sihag, Saurabh, Mateos, Gonzalo, C. McMillan, and Ribeiro, Alejandro, “Transferability of covariance neural networks,”, in IEEE Journal of Selected Topics in Signal Processing, pp. 1–16, 2024.

S. Khalafi, Saurabh Sihag, and Alejandro Ribeiro, “Neural tangent kernels motivate cross-covariance graphs in neural networks,” in International Conference on Machine Learning, 2024.

Sihag, Saurabh, Mateos, Gonzalo, and Ribeiro, Alejandro, “Explainable brain age gap prediction in neurodegenerative conditions using covariance neural networks,” IEEE International Symposium on Biomedical Imaging, 2025.

A. Cavallo, Z. Gao, and Elvin Isufi, “Sparse covariance neural networks,” arXiv:2410.01669, vol. cs.LG, 2024.

Cavallo, Andrea, et al. "Fair covariance neural networks." IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2025.

A. Cavallo, M. Sabbaqi, and Isufi, Elvin, “Spatiotemporal covariance neural networks,” in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 18–34, Springer, 2024.

Slides available at