

Learning with Covariance Matrices: Foundations and Applications to Network Neuroscience

Saurabh Sihag¹, Gonzalo Mateos², Elvin Isufi³, and Alejandro Ribeiro⁴

¹University at Albany – ssihag@albany.edu

²University of Rochester – gmateosb@ece.rochester.edu

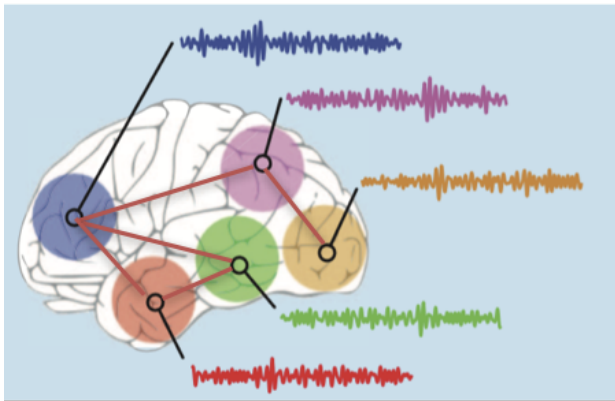
³Delft University of Technology – e.isufi-1@tudelft.nl

⁴University of Pennsylvania – aribeiro@seas.upenn.edu

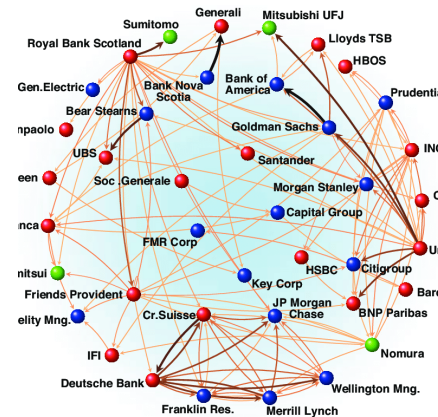
IEEE International Workshop on Machine Learning
for Signal Processing (MLSP), 2025

Covariance Matrix

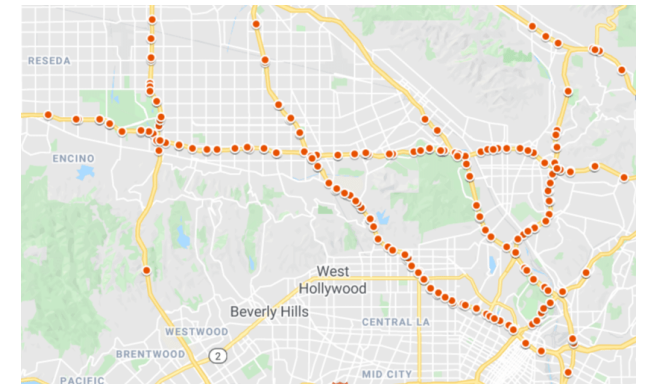
- Covariance matrix captures the **redundancies** between data points (features)
 - **Brain datasets:** some areas of the brain activate together
 - **Financial datasets:** stock prices fluctuate in tandem
 - **Traffic datasets:** traffic volume is correlated across intersections



Brain



Finance



Traffic

Covariance Matrix

➤ Evaluating a covariance matrix

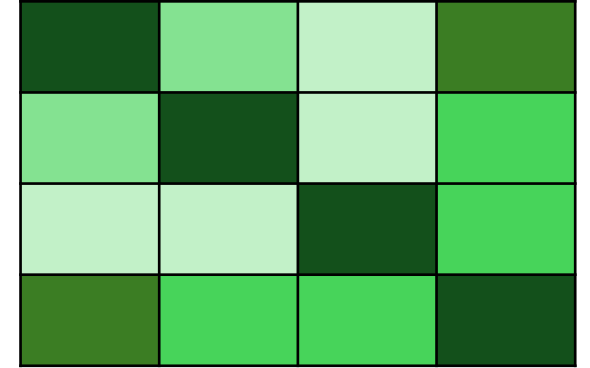
- Consider a random variable $\mathbf{x} \in \mathbb{R}^m$
- The covariance is

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

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

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

n : number of samples (size of a dataset)



Covariance Matrix

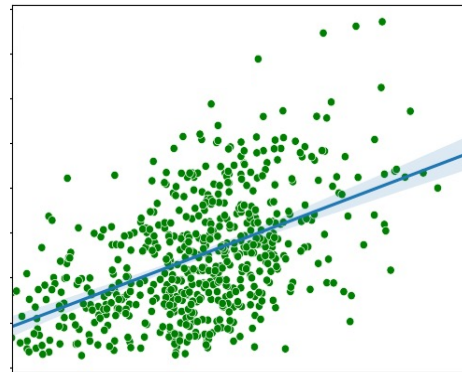
- Covariance matrix encodes **redundancies** between different features in data

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 $\sigma(r_1, r_2)$)

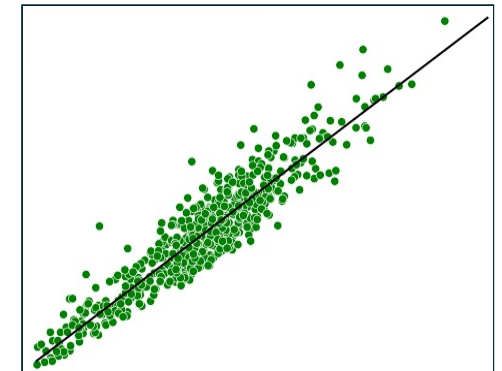
Feature r_1



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

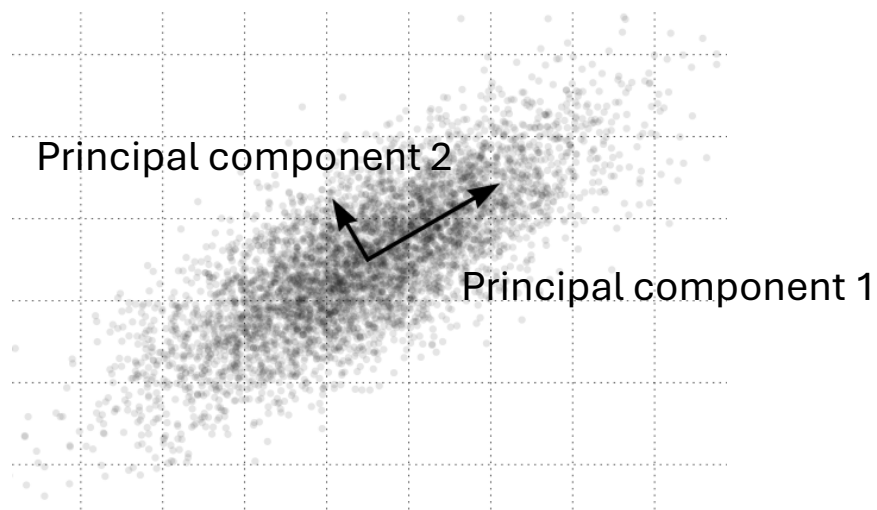
Covariance matrices are widespread in signal processing and machine learning

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

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



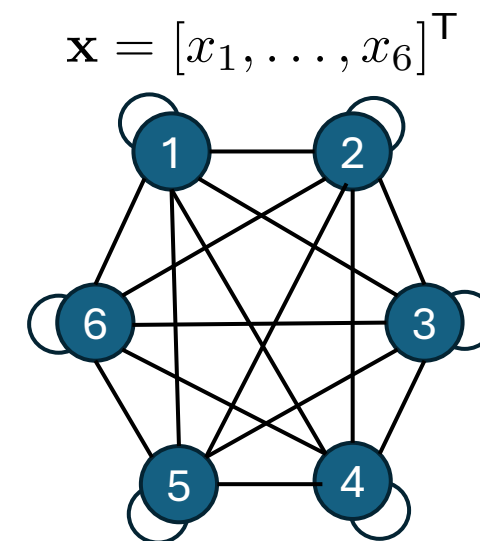
PCA transform in ML

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

Covariance matrices are widespread in signal processing and machine learning

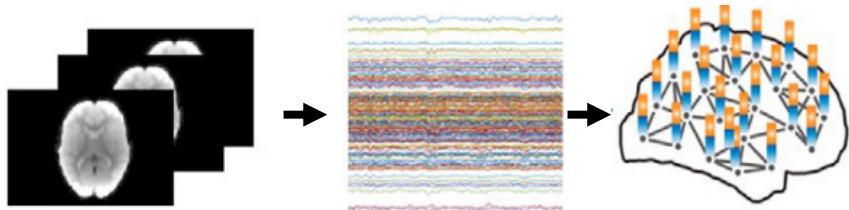
➤ Covariance matrices are leveraged as **graphical** representations of data

- 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

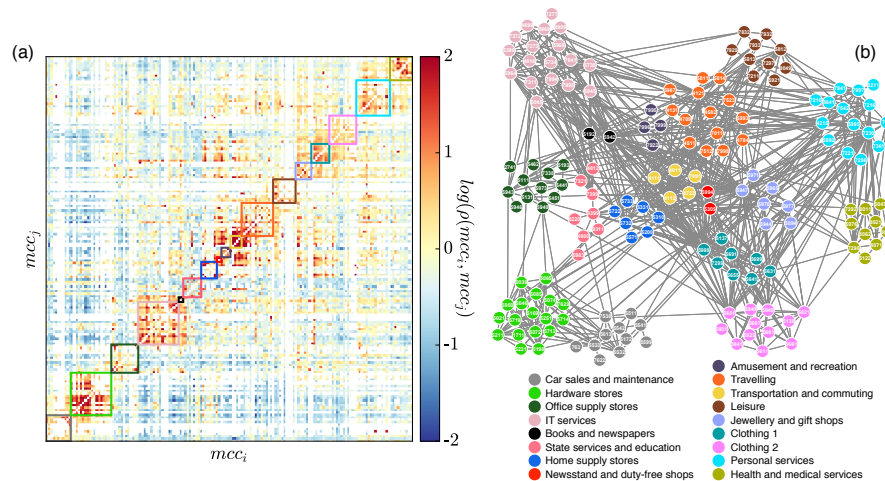


Covariance matrices are widespread in signal processing and machine learning

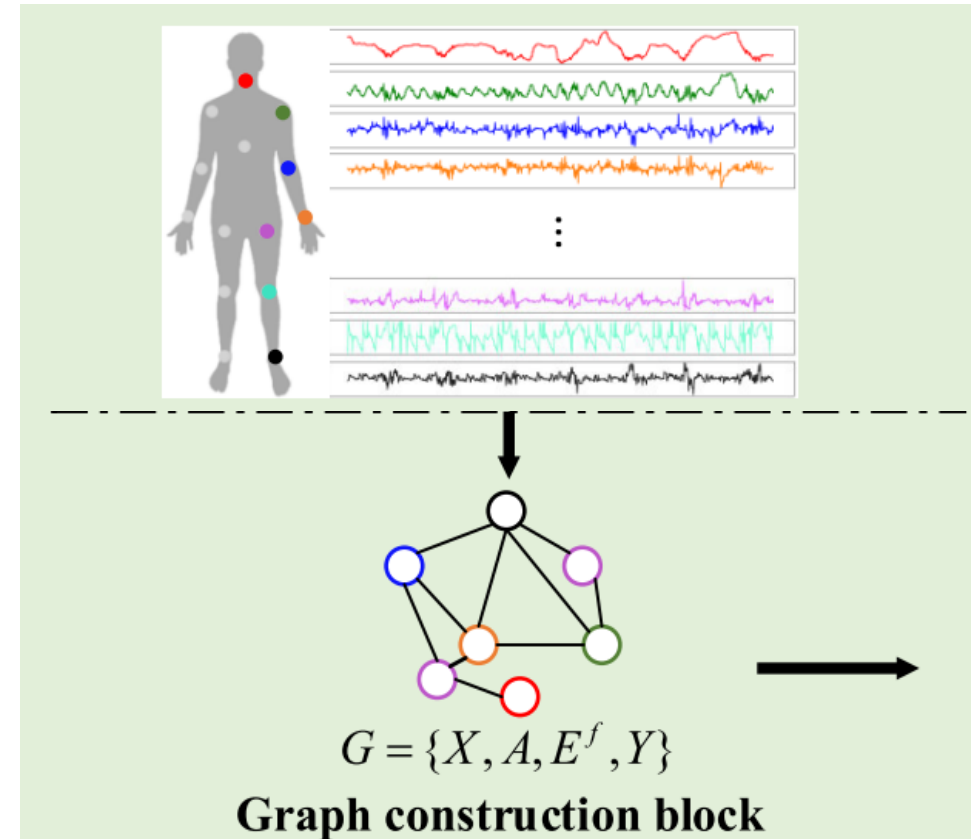
- Covariance matrices as **graphical** representations; used in graph neural nets



Brain connectome [Li, et al. 2021]



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



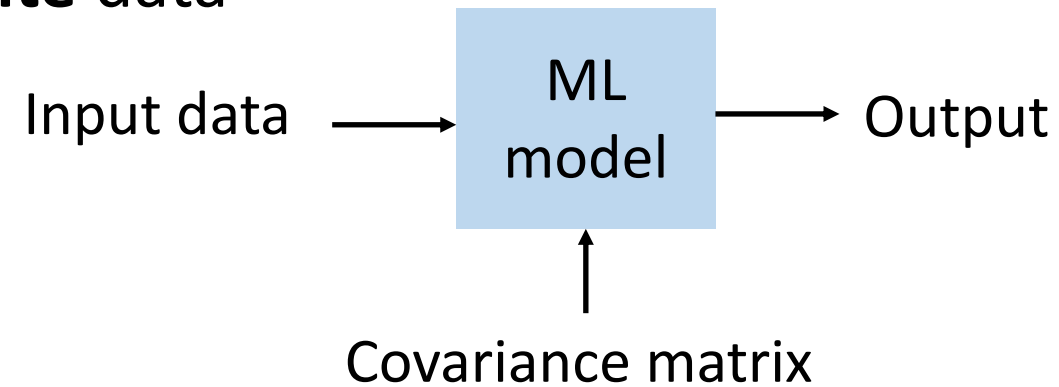
Wearable devices [Wang, et al. 2023]

Learning with covariance matrices: Challenges

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

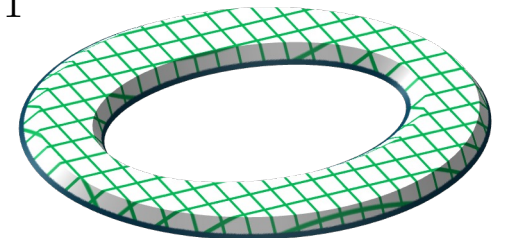
➤ ML model is trained on **training** dataset, deployed on **test** dataset

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

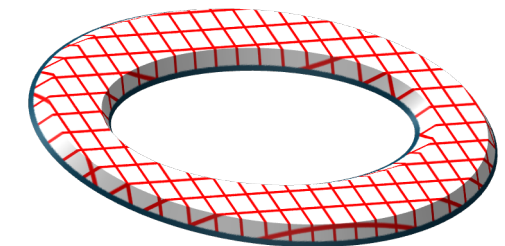


$$\hat{\mathbf{C}} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\top}$$

Representation of **training** dataset



Representation of **test** dataset



Learning with covariance matrices: Challenges

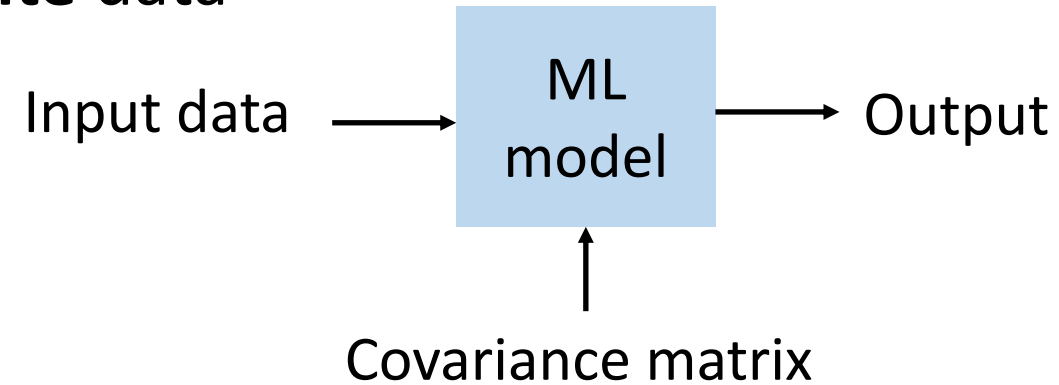
➤ Sample covariance matrix is estimate from **finite** data

➤ ML model is trained on **training** dataset, deployed on **test** dataset

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

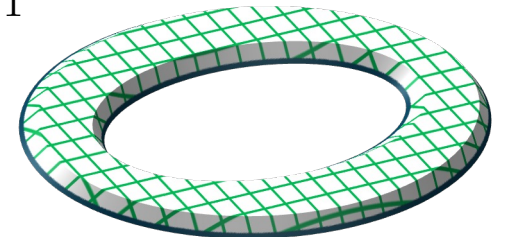
Challenge 1 (stability)

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

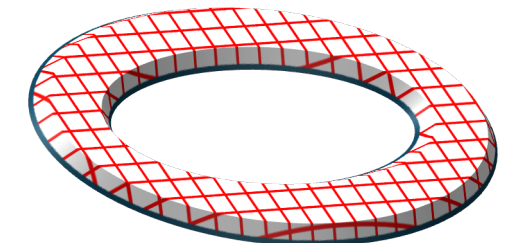


$$\hat{\mathbf{C}} = \frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^{\top}$$

Representation of **training** dataset



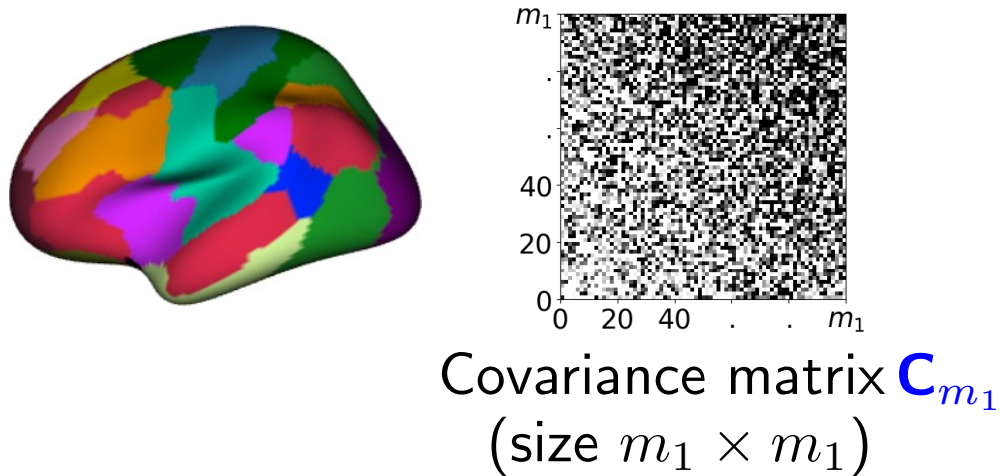
Representation of **test** dataset



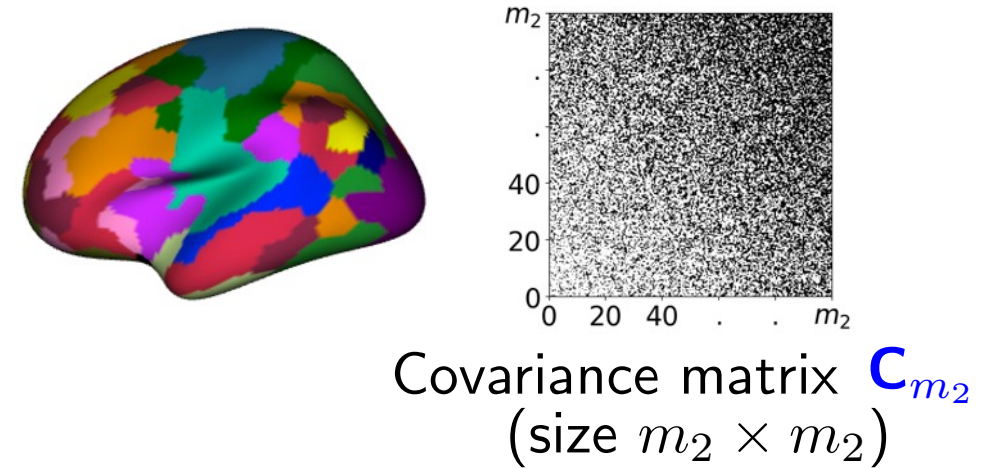
Learning with covariance matrices: Challenges

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

Dataset with m_1 features



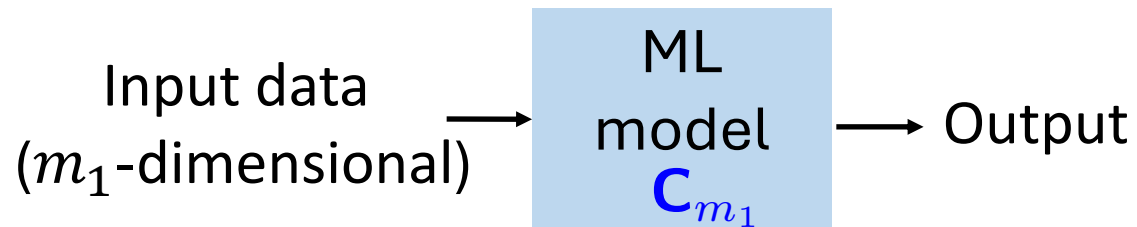
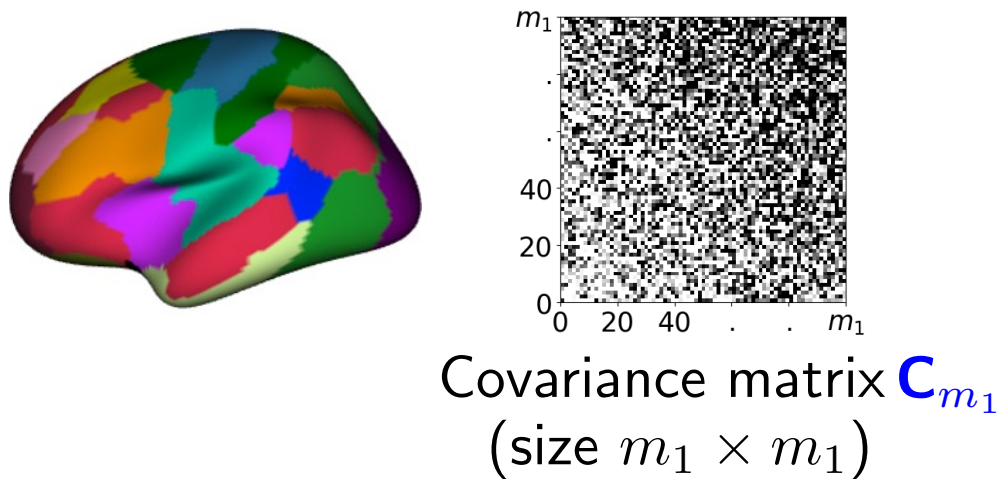
Dataset with m_2 features



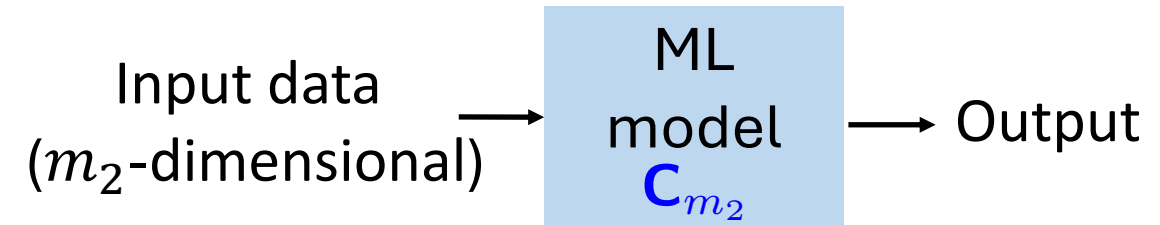
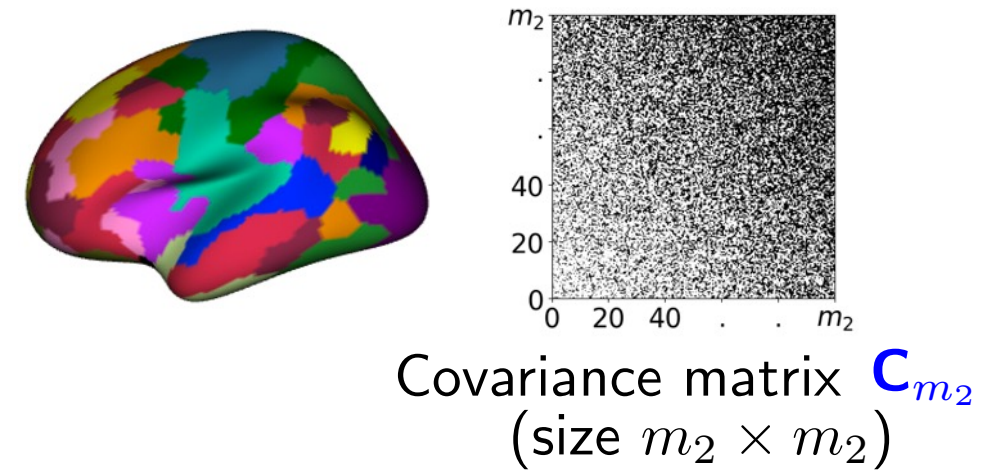
Learning with covariance matrices: Challenges

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

Dataset with m_1 features



Dataset with m_2 features



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



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

ABSTRACT | Research in graph signal processing (GSP) aims to develop tools for processing data defined on irregular graph domains. In this paper, we first provide an overview of core ideas in GSP and their connection to conventional digital signal processing, along with a brief historical perspective to highlight how concepts recently developed in GSP build on top of prior research in other areas. We then summarize recent advances in developing basic GSP tools, including methods for sampling, filtering, or graph learning. Next, we review progress in several application areas using GSP, including processing and analysis of sensor network data, biological data, and applications to image processing and machine learning.

KEYWORDS | Graph signal processing (GSP); network science and graphs; sampling; signal processing

I. INTRODUCTION AND MOTIVATION

Data is all around us, and massive amounts of it. Almost every aspect of human life is now being recorded at all levels: from the marking and recording of processing inside the cells starting with the advent of fluorescent markers, to our personal data through health monitoring devices and apps, financial and banking data, our social networks, mobility and traffic patterns, marketing preferences, fads, and many more. The complexity of such networks [1] and interactions means that the data now reside on irregular and complex structures that do not lend themselves to standard tools.

Manuscript received November 26, 2022; revised March 10, 2024; accepted March 28, 2024. **Date of current version** April 24, 2024. (Corresponding author: Antonio Ortega.)
A. Ortega is with the University of Southern California, Los Angeles, CA 90089, USA (e-mail: antonio@usc.edu).
P. Frossard, and P. Vandergheynst are with EPFL, Lausanne, Switzerland 1015, Switzerland.
J. Kováčević, and J. M. F. Moura are with Carnegie Mellon University, Pittsburgh, PA 15213, USA.

Digital Object Identifier: 10.1109/SPRS.2024.2402024

0018-9219 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. Authorized licensed use limited to: UNIVERSITY AT ALBANY. Downloaded from August 2, 2025 at 19:52:10 UTC from IEEE Xplore. Restrictions apply.

808

Graphs offer the ability to model such data and complex interactions among them. For example, users on Twitter can be modeled as nodes while their friend connections can be modeled as edges. This paper explores adding attributes to such nodes and modeling them as signals on a graph, for example, year of graduation in a social network, temperature in a given city on a given day in a weather network, etc. Doing so requires us to extend classical signal processing concepts and tools such as Fourier transform, filtering, and frequency response to data residing on graphs. It also leads us to tackle complex tasks such as sampling in a principled way. The field that gathers all these questions under a common umbrella is graph signal processing (GSP) [2], [3].

While the precise definition of a graph signal will be given later in the paper, let us assume for now that a graph signal is a set of values residing on a set of nodes. These nodes are connected via (possibly weighted) edges. As in classical signal processing, such signals can stem from a variety of domains; unlike in classical signal processing, however, the underlying graphs can tell a fair amount about those signals through their structure. Different types of graphs model different types of networks that these nodes represent.

Typical graphs that are used to represent common real-world data include Erdős-Rényi graphs, ring graphs, random geometric graphs, small-world graphs, power-law graphs, nearest-neighbor graphs, scale-free graphs, and many others. These model networks with random connections (Erdős-Rényi graphs), networks of brain neurons (small-world graphs), social networks (scale-free graphs), and others.

As in classical signal processing, graph signals can have properties, such as smoothness, that need to be appropriately defined. They can also be represented via basic atoms and can have a spectral representation. In particular, the graph Fourier transform allows us to develop the intuition gathered in the classical setting and extend it to graphs; we can talk about the notions of frequency and bandwidth.

Geert Lau[✉], Antonio G. Marques[✉], José M.F. Moura[✉], Antonio Ortega[✉], and David I. Shuman[✉]

75TH ANNIVERSARY OF SIGNAL PROCESSING SOCIETY SPECIAL ISSUE

Graph Signal Processing

History, development, impact, and outlook



Digital Object Identifier: 10.1109/SPRS.2023.1282006
Date of current version: 1 June 2023

0018-9219 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

Authorized licensed use limited to: UNIVERSITY AT ALBANY. Downloaded from August 2, 2025 at 19:52:10 UTC from IEEE Xplore. Restrictions apply.

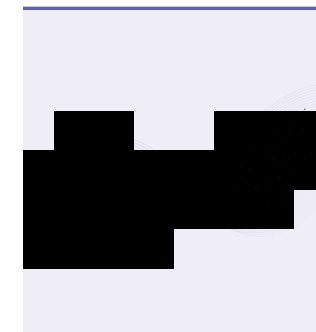
809

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

GRAPH SIGNAL PROCESSING: FOUNDATIONS AND EMERGING DIRECTIONS

Graph Signal Processing for Machine Learning

A review and new perspectives



The effective representation, processing, analysis, and visualization of large-scale structured data, especially those related to complex domains, such as networks and graphs, are one of the key questions in modern machine learning. Graph signal processing (GSP), a vibrant branch of signal processing models and algorithms that aims at handling data supported on graphs, opens new paths of research to address this challenge. In this article, we review a few important contributions made by GSP concepts and tools, such as graph filters and transforms, to the development of novel machine learning algorithms. In particular, our discussion focuses on the following three aspects: exploiting data structure and relational priors; improving data and computational efficiency; and enhancing model interpretability. Furthermore, we provide new perspectives on the future development of GSP techniques that may serve as a bridge between applied mathematics and signal processing on one side and machine learning and network science on the other. Cross-fertilization across these different disciplines may help unlock the numerous challenges of complex data analysis in the modern age.

Introduction

We live in a connected society. Data collected from large-scale interactive systems, such as biological, social, and financial networks, become largely available. In parallel, the past few decades have seen a significant amount of interest in the machine learning community for network data processing and analysis. Networks have an intrinsic structure that conveys very specific properties to data, e.g., interdependencies between data entities in the form of pairwise relationships. These properties are traditionally captured by mathematical representations such as graphs.

In this context, new trends and challenges have been developing fast. Let us consider, for example, a network of protein-protein interactions and the expression level of individual genes at every point in time. Some typical tasks in network biology related to this type of data are 1) discovery of key genes (via protein grouping) affected by the infection and 2) prediction of how the host organism reacts (in terms of gene expression)

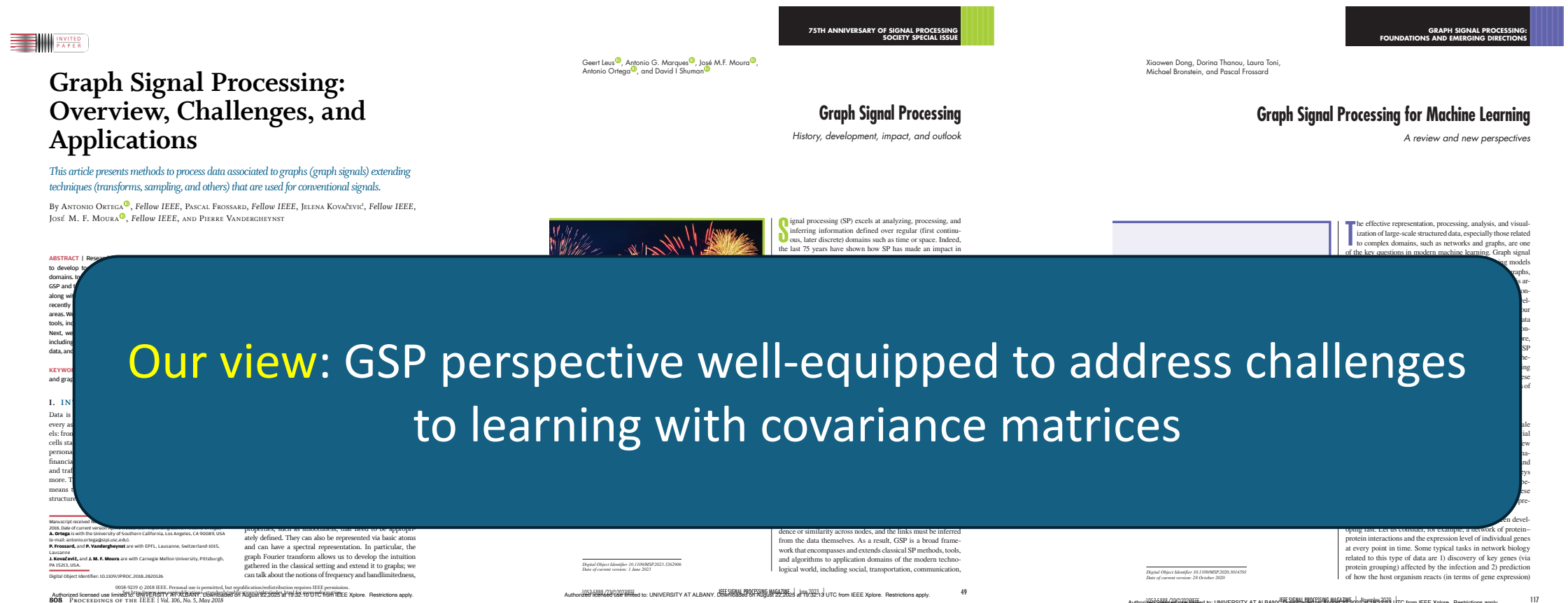
Digital Object Identifier: 10.1109/SPRS.2023.1282006
Date of current version: 24 October 2023

Authorized licensed use limited to: UNIVERSITY AT ALBANY. Downloaded from August 2, 2025 at 19:52:10 UTC from IEEE Xplore. Restrictions apply.

117

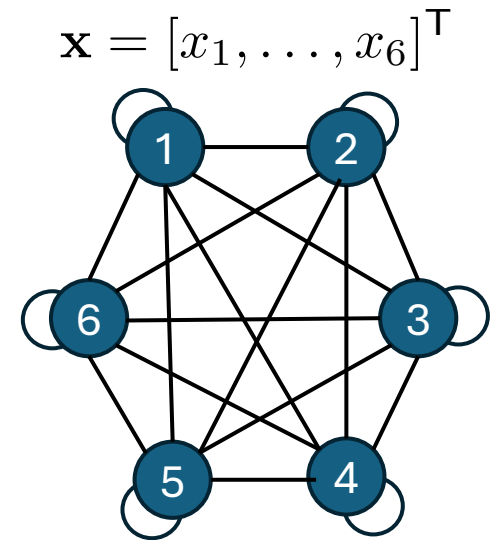
Learning with covariance matrices: A GSP approach

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



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



Outline

- PCA and the graph Fourier transform
- CoVariance neural networks (VNNs)
- Theory of VNNs: Stability and transferability
- Variants of VNNs

Outline

- PCA and the graph Fourier transform
- CoVariance neural networks (VNNs)
- Theory of VNNs: Stability and transferability
- Variants of VNNs

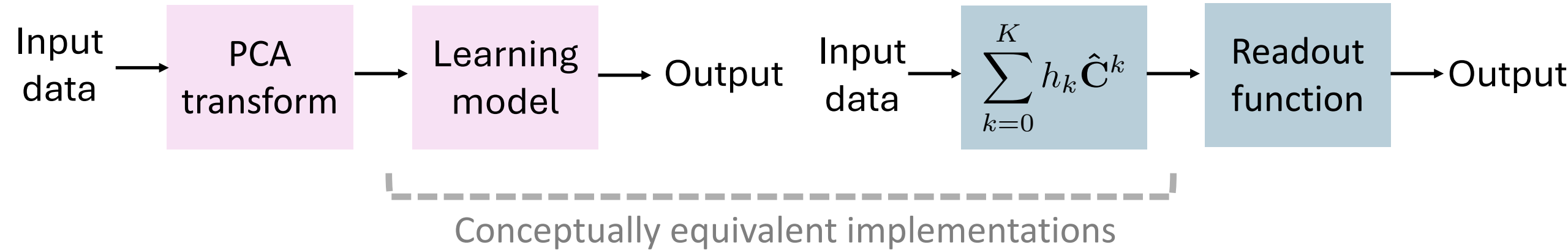
Key takeaways:

- VNNs offer a novel GSP-inspired perspective to PCA
 - ⇒ addressing challenges in modern data analysis
- Principled deep learning solution for finite-data regimes

PCA and Graph Fourier Transform

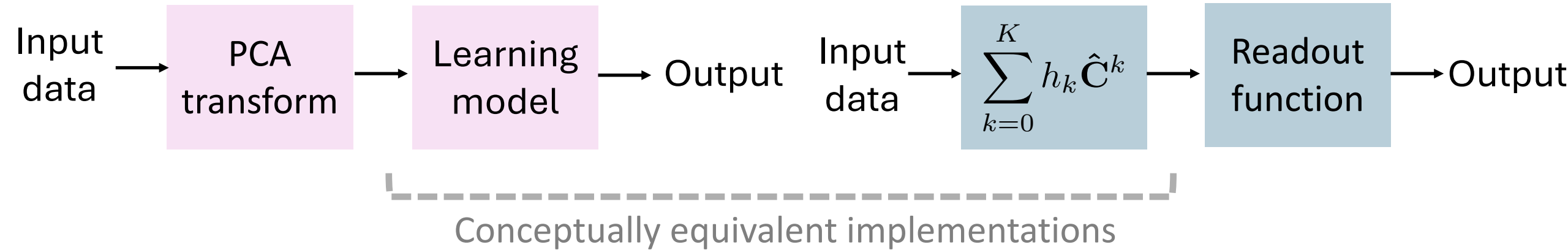
A graph filter implementation of PCA inference

- **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**

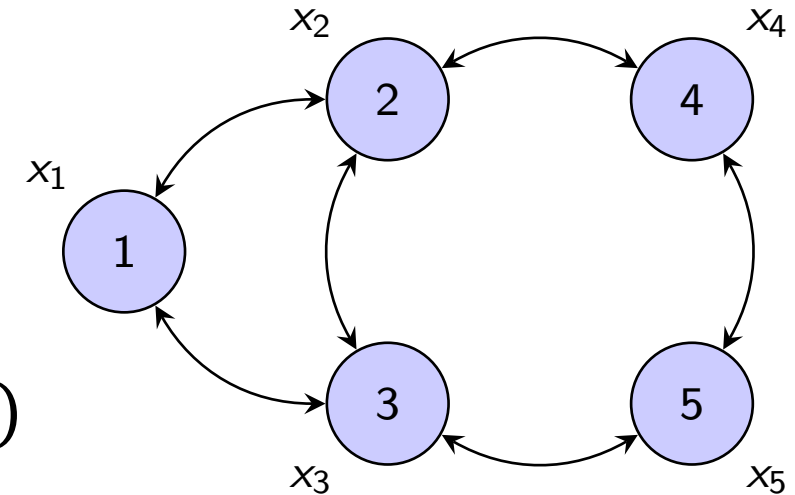
Preliminaries: Graph

➤ **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)

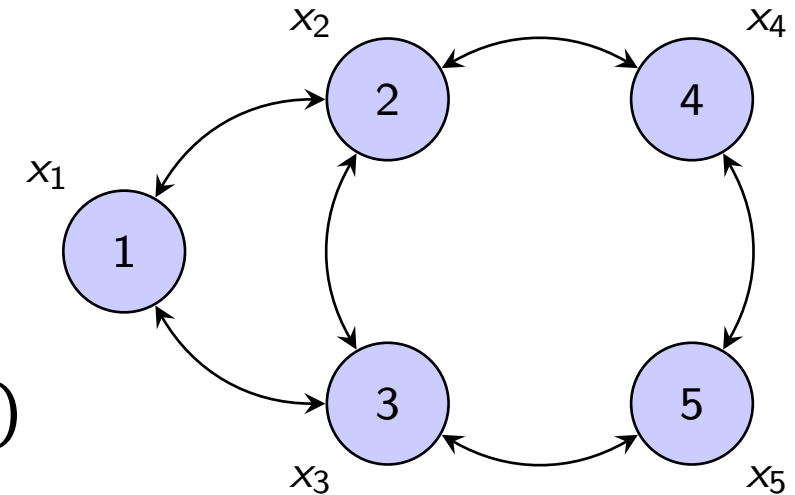
- A set of **nodes** $V = \{1, \dots, m\}$
- A set of (undirected) **edges** $E \subseteq V \times V$

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 $w_{ij} \in \mathbb{R}$

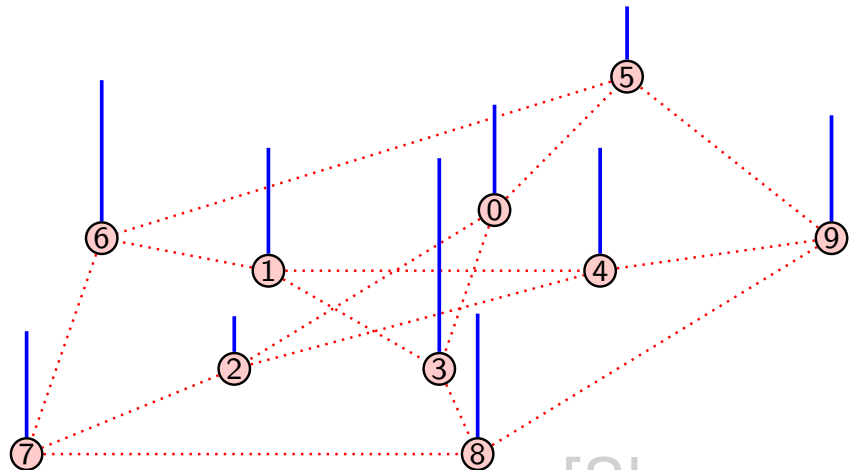
➤ **Adjacency matrix** representation of graph

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



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



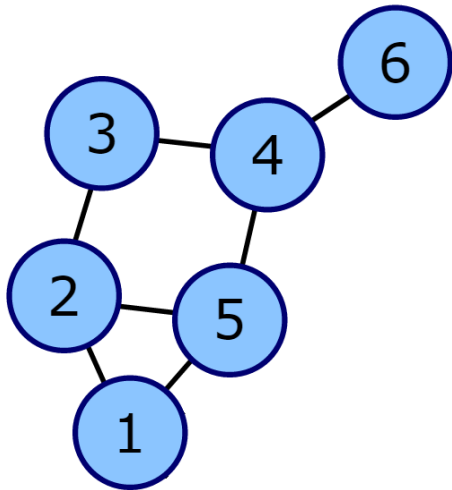
[Shuman, 2013]

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

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** $\mathbf{S} \in \mathbb{R}^{m \times m}$

$\Rightarrow [\mathbf{S}]_{ij} = 0$ 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 $\mathbf{S} = \mathbf{U}\mathbf{\Phi}\mathbf{U}^{-1}$
- **GFT** is the projection of graph signal on the eigenvector space \mathbf{U}

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

- **Inverse GFT** is defined as



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

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

When GSO is covariance matrix...

- GFT over covariance matrix

Given eigendecomposition

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

GFT of \mathbf{x} is

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

When GSO is covariance matrix...

- GFT over covariance matrix

Given eigendecomposition

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

GFT of \mathbf{x} is

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

- PCA transform

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

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

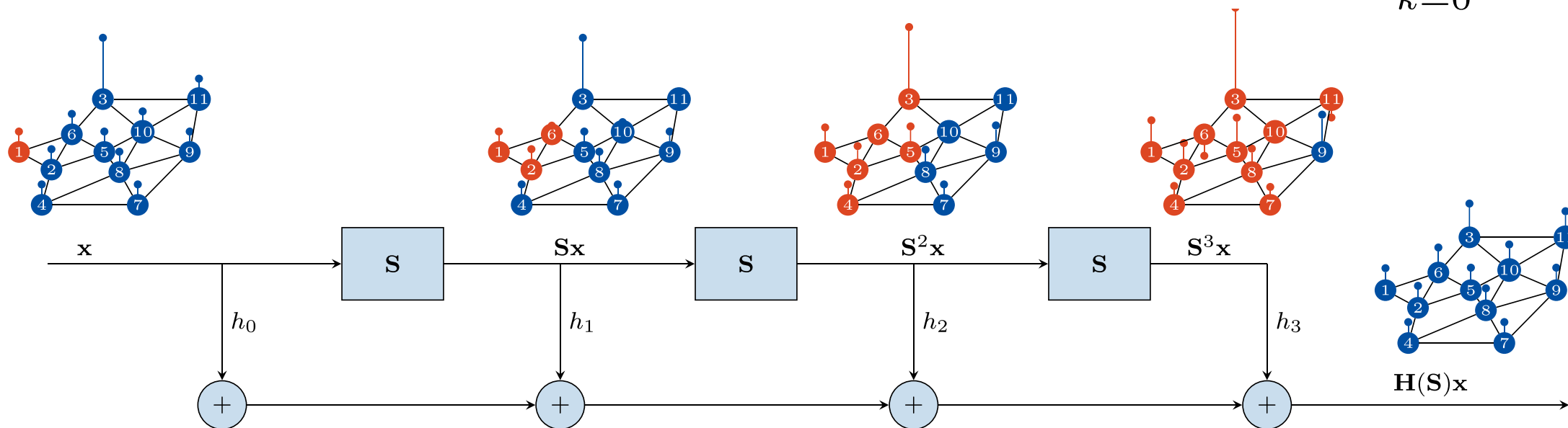
PCA transform is GFT with respect to the covariance graph!

Preliminaries: Graph filter

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

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

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

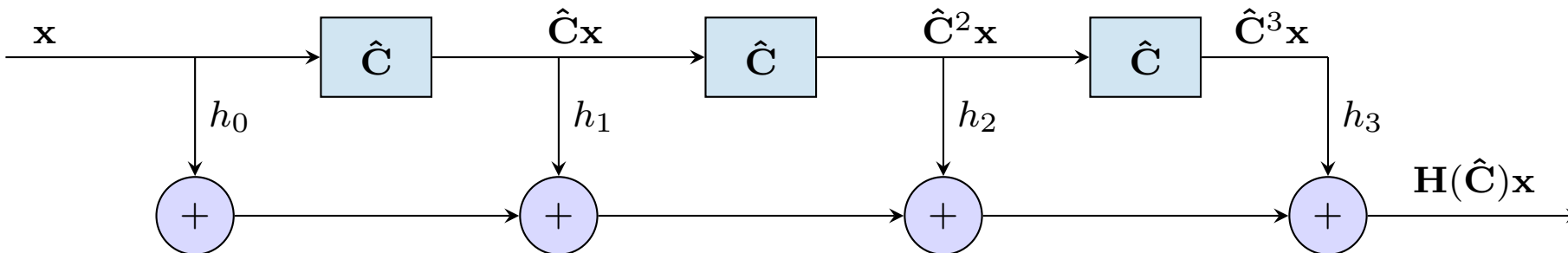
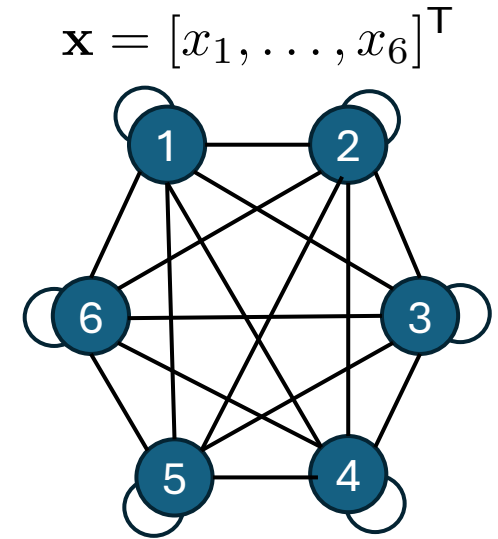


[Isufi et. al, IEEE TSP, 2024]

Graph filter on covariance matrix

- Covariance matrix forms a fully-connected graph where
 - nodes are features
 - edges are covariance values
- 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 $\hat{\mathbf{C}} = \hat{\mathbf{V}} \hat{\mathbf{\Lambda}} \hat{\mathbf{V}}^\top$, it follows that

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

CoVariance filter

➤ Analogy between $\mathbf{H}(\hat{\mathbf{C}})$ and PCA

- Using eigendecomposition $\hat{\mathbf{C}} = \hat{\mathbf{V}} \hat{\mathbf{\Lambda}} \hat{\mathbf{V}}^\top$, it follows that

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

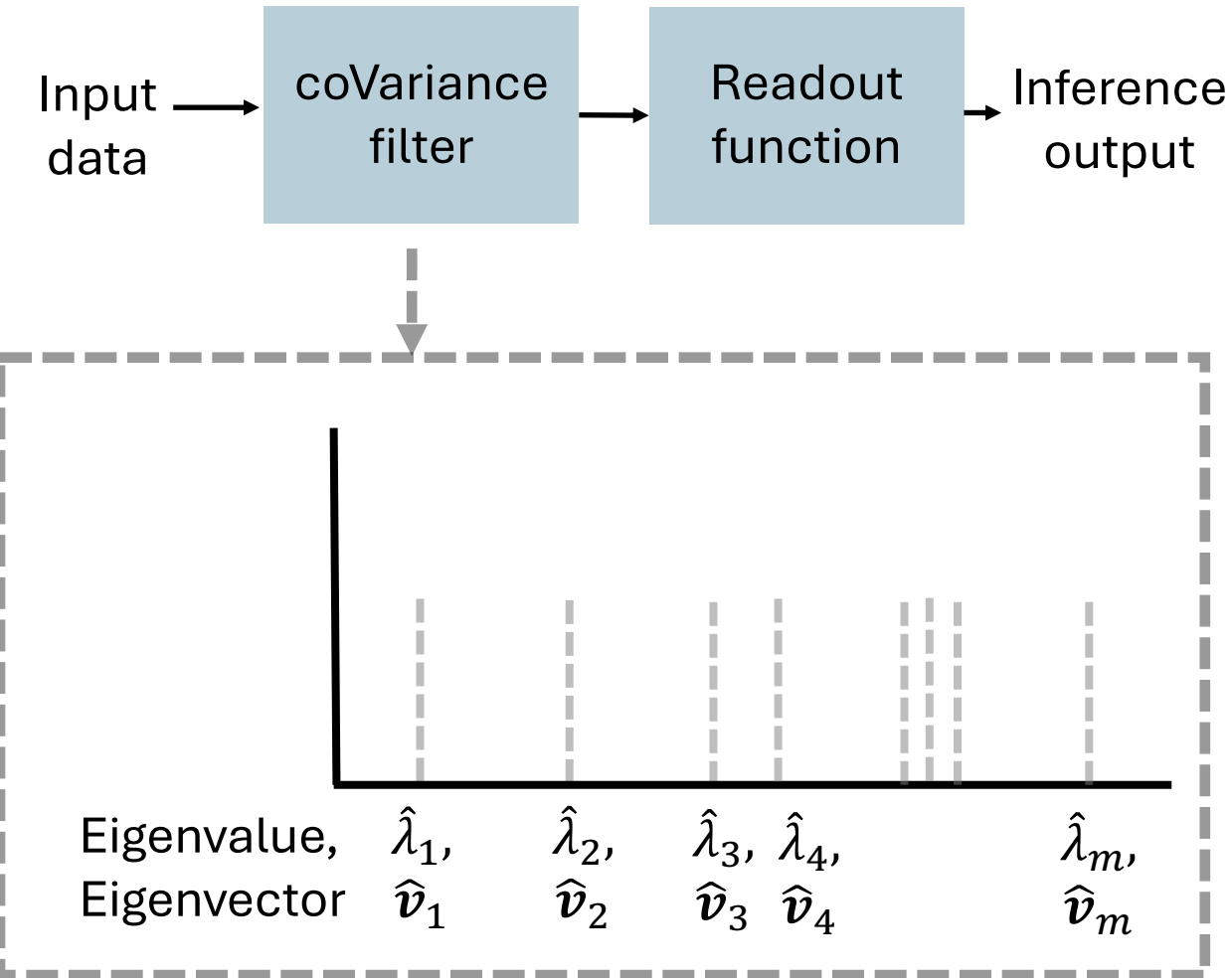
- GFT of coVariance filter output \mathbf{z} and PCA are **equivalent**

$$\tilde{\mathbf{z}} = \left(\sum_{k=0}^K h_k \hat{\mathbf{\Lambda}}^k \right) \hat{\mathbf{V}}^\top \mathbf{x}$$

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

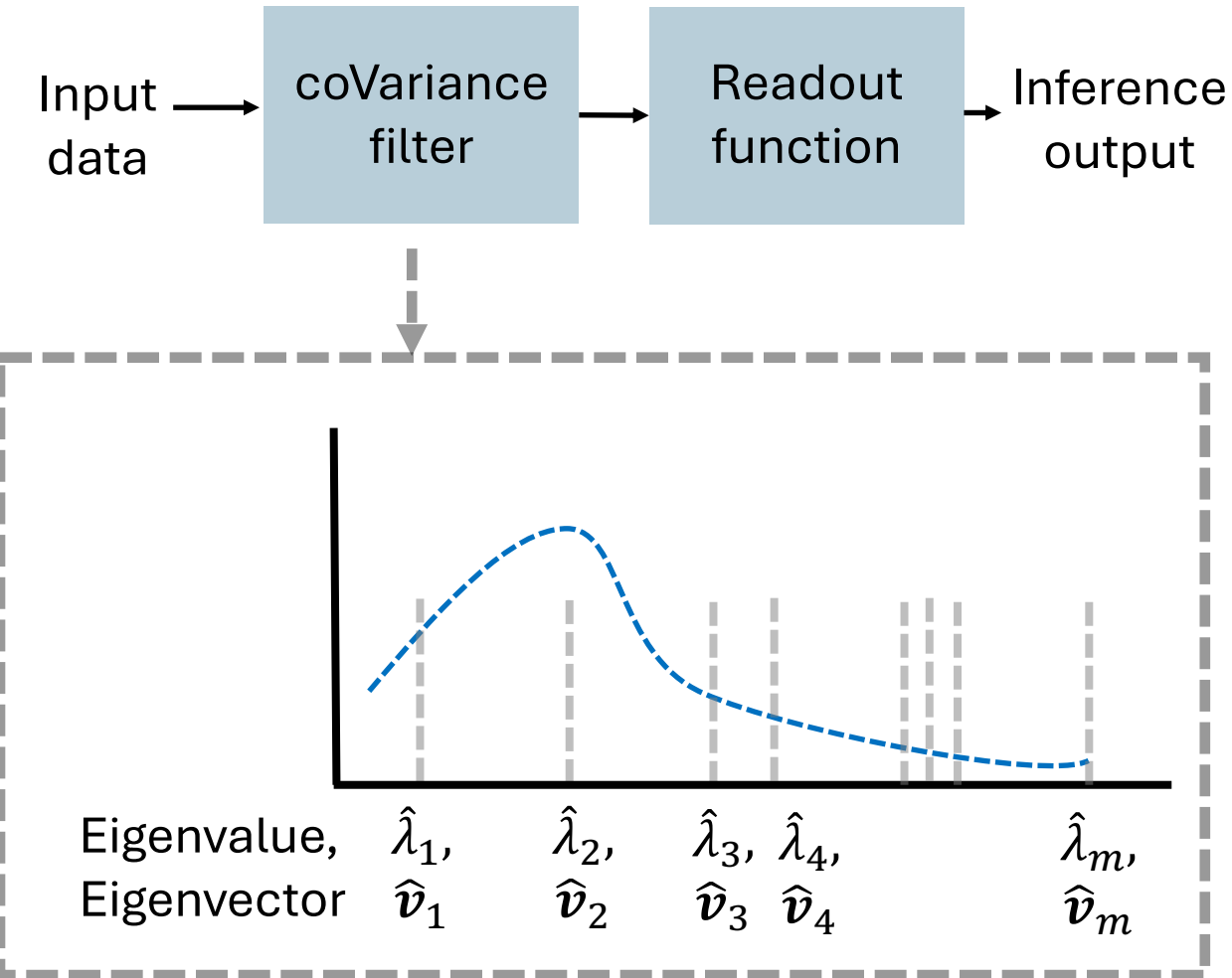
Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter



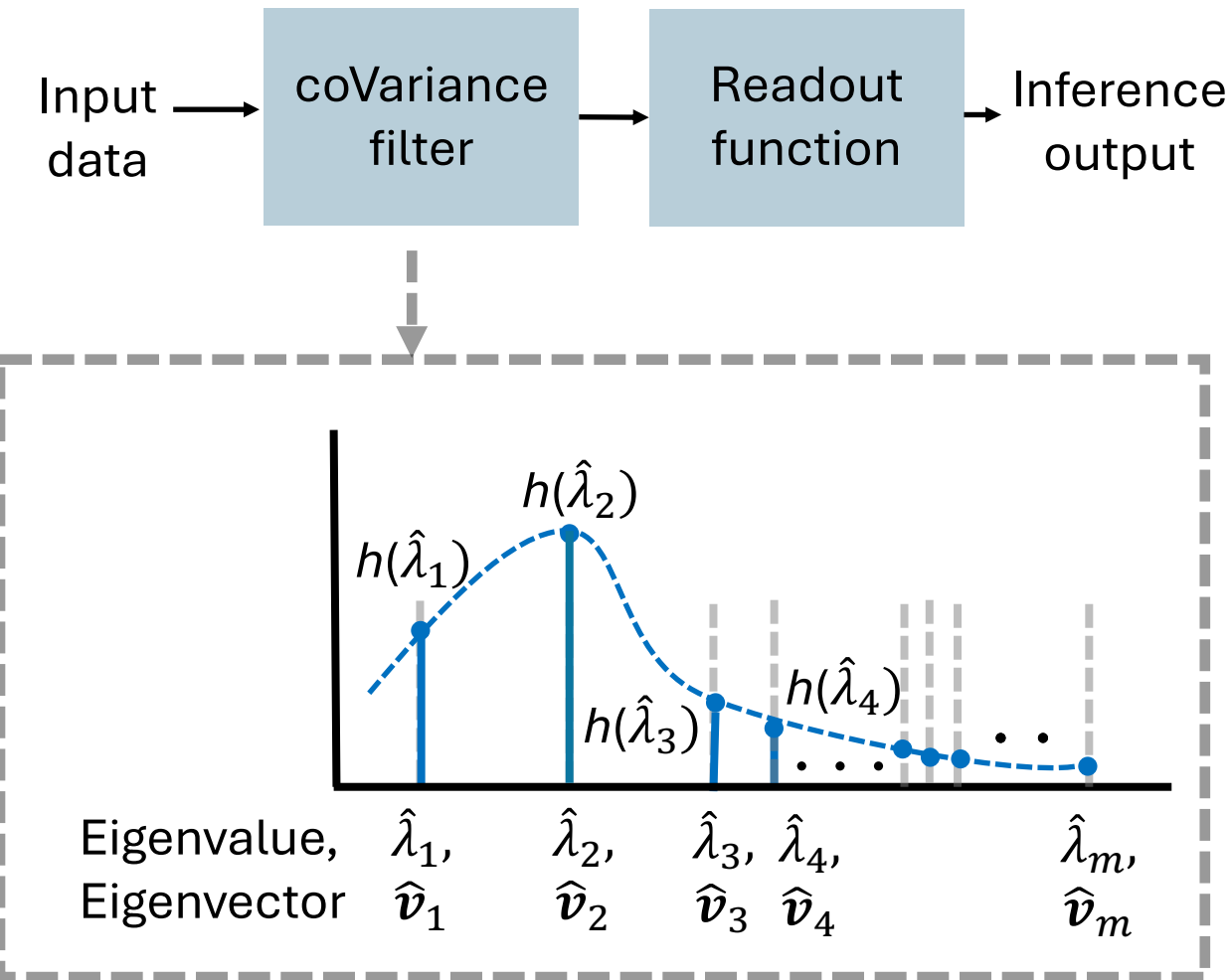
Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter



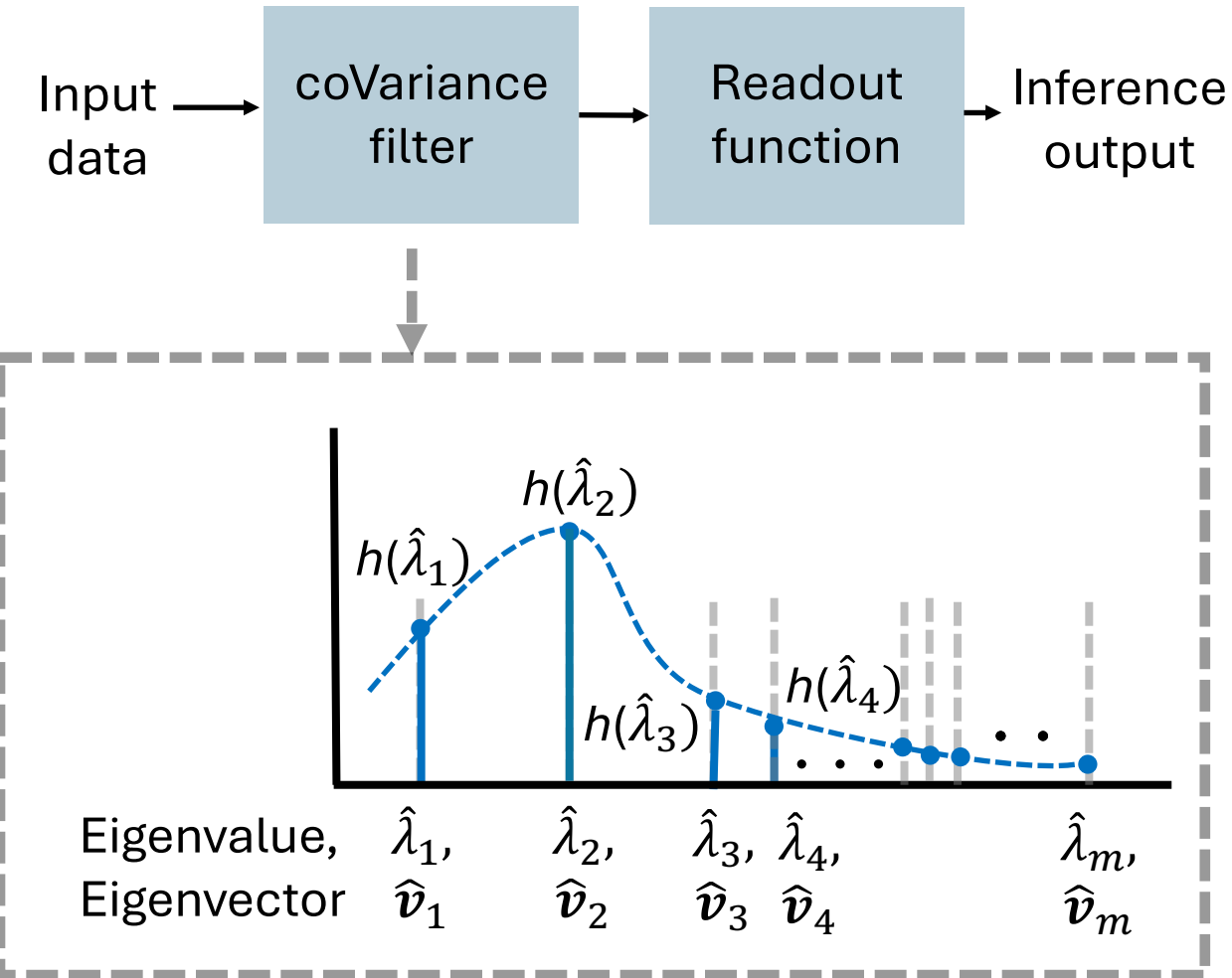
Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter

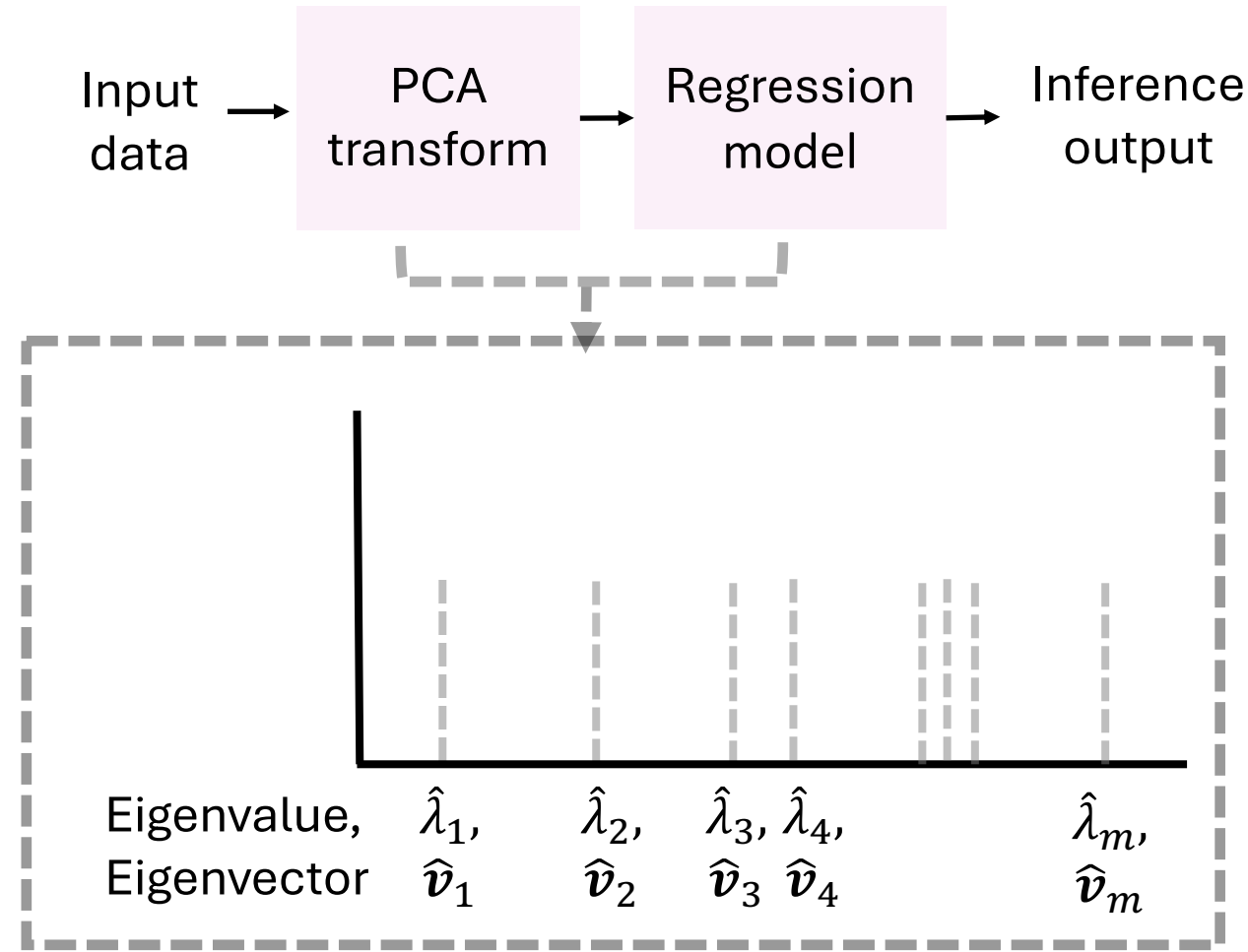


Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter

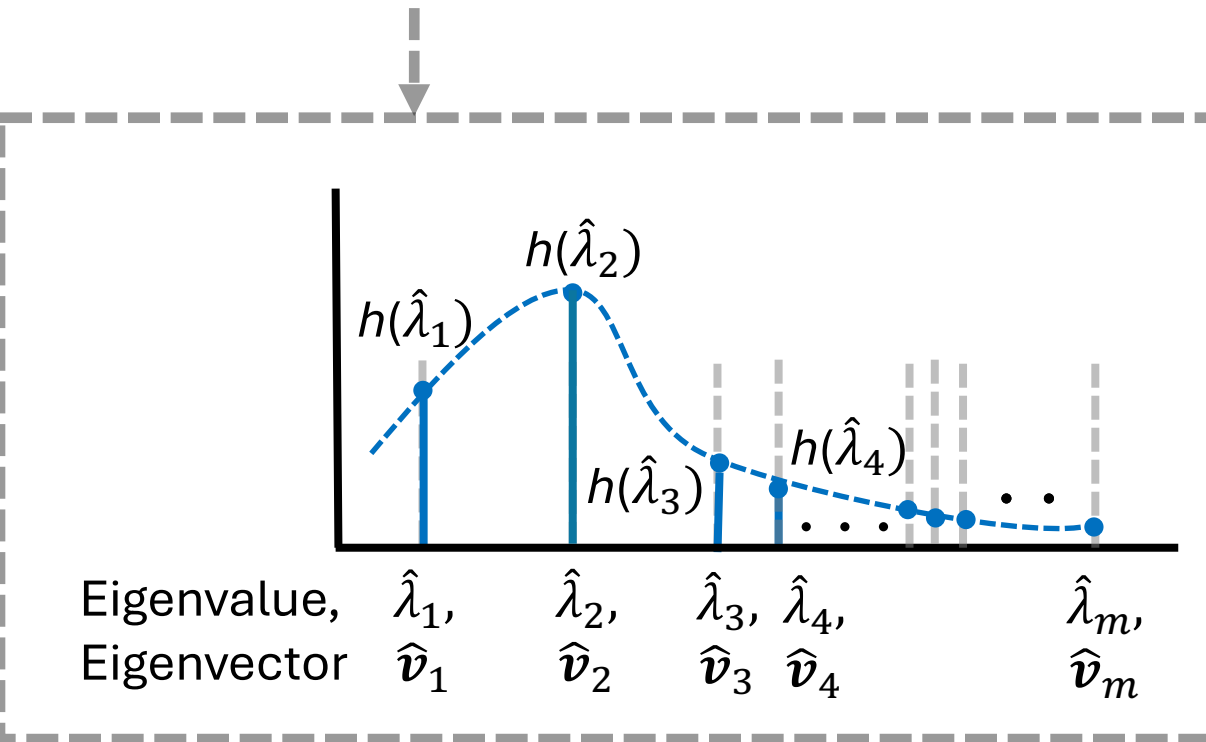
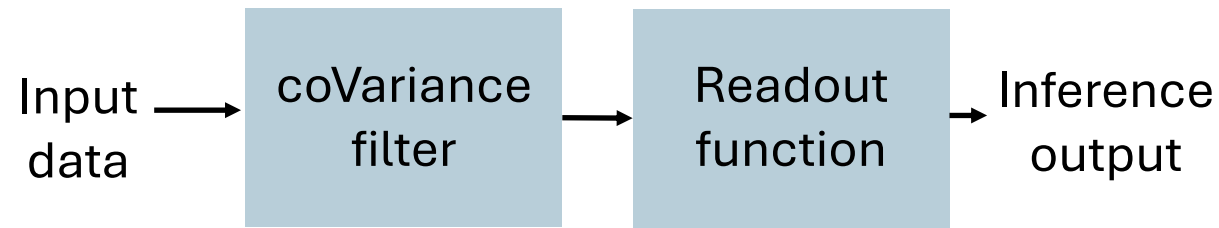


➤ PCA-based learning

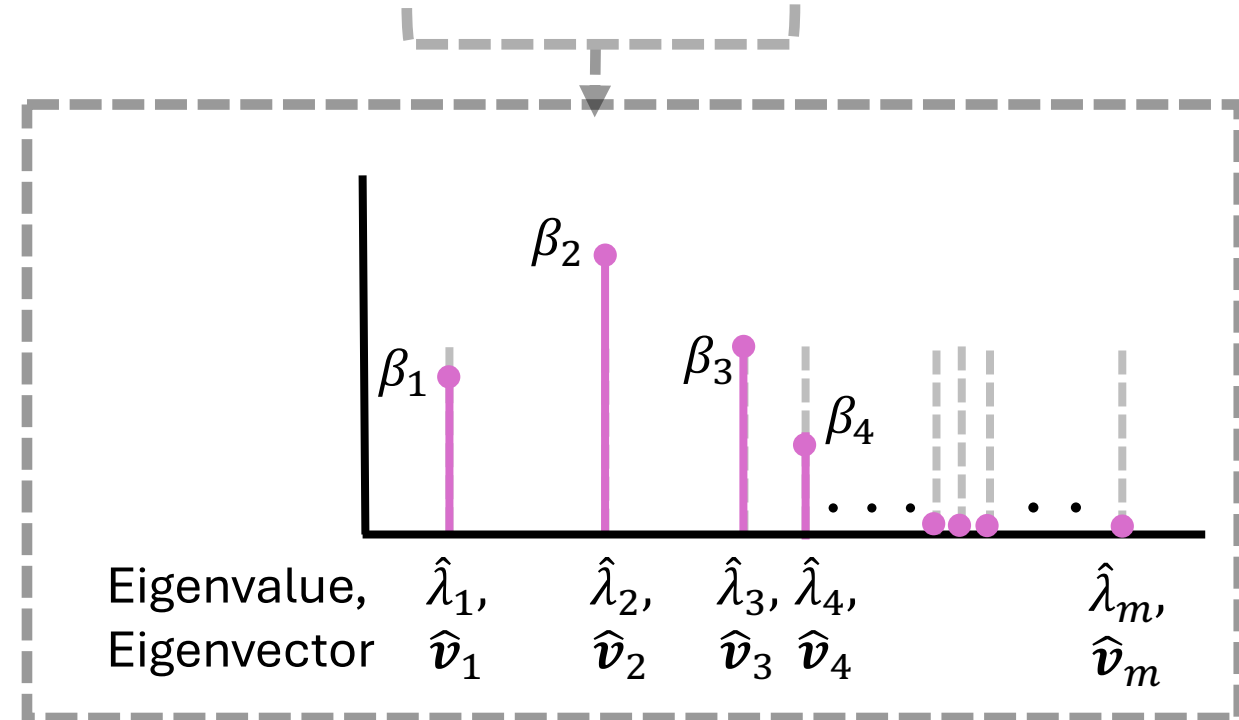
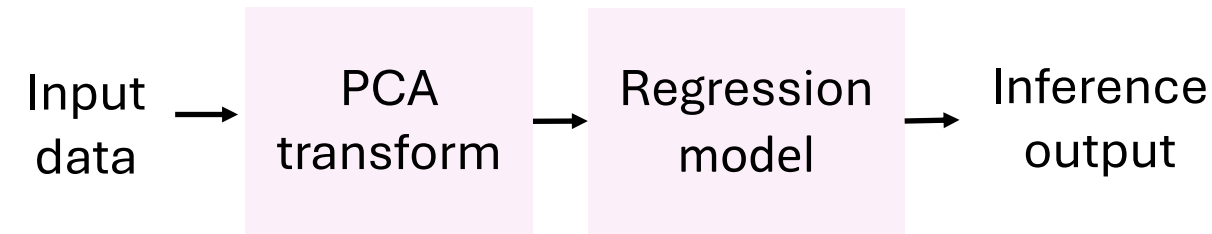


Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter



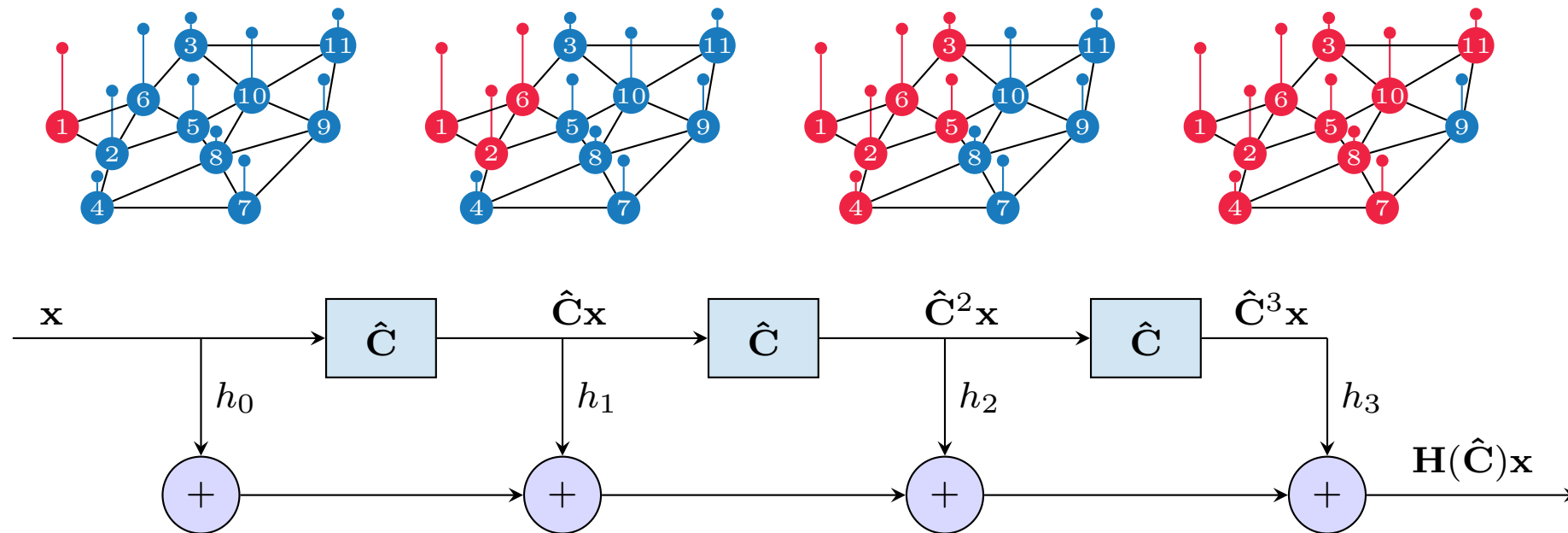
➤ PCA-based learning



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 $\{\mathbf{h}_k\}$ are called **filter taps**, are **scalars** and **learnable** parameters

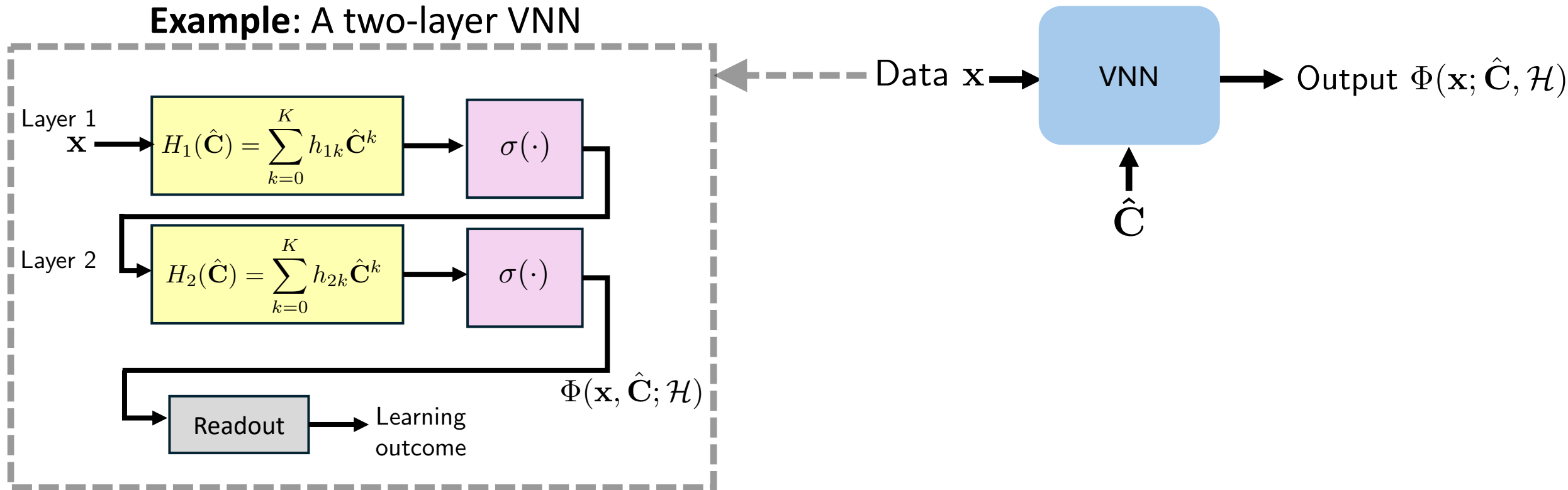
CoVariance Neural Networks (VNNs)

- 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.)

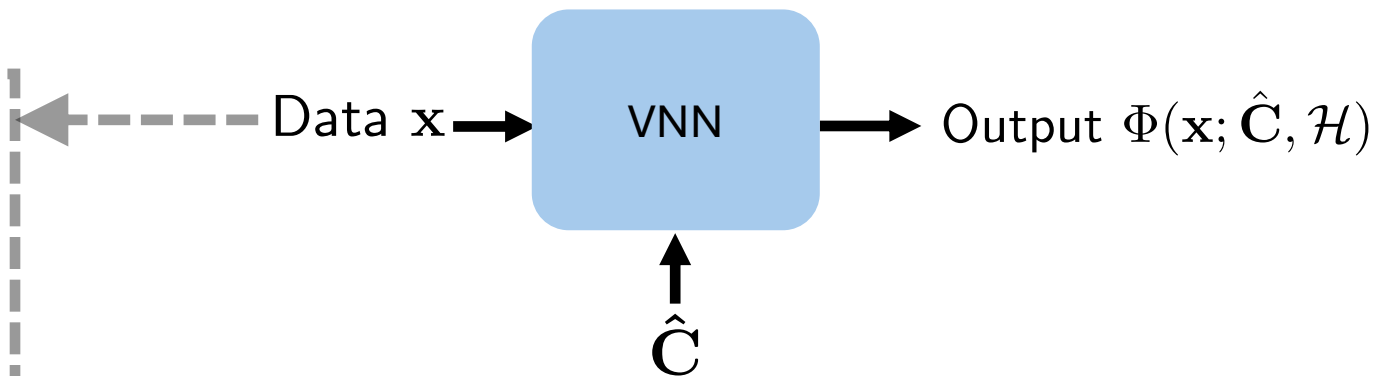
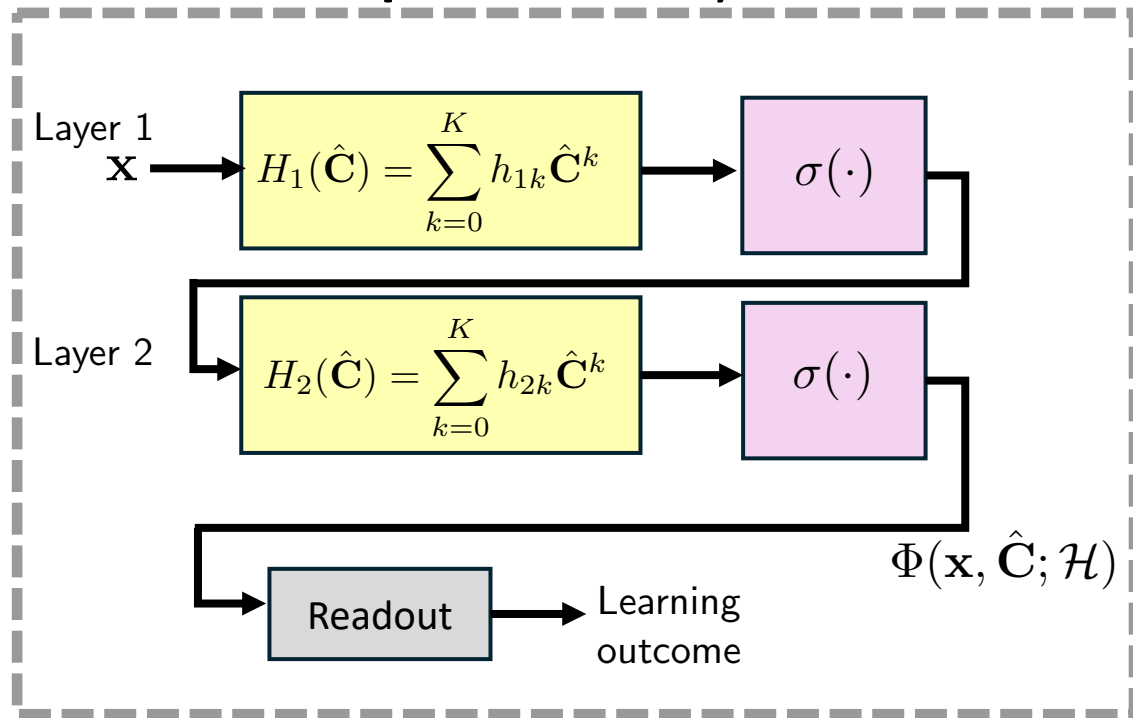
Example: A two-layer VNN



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.)

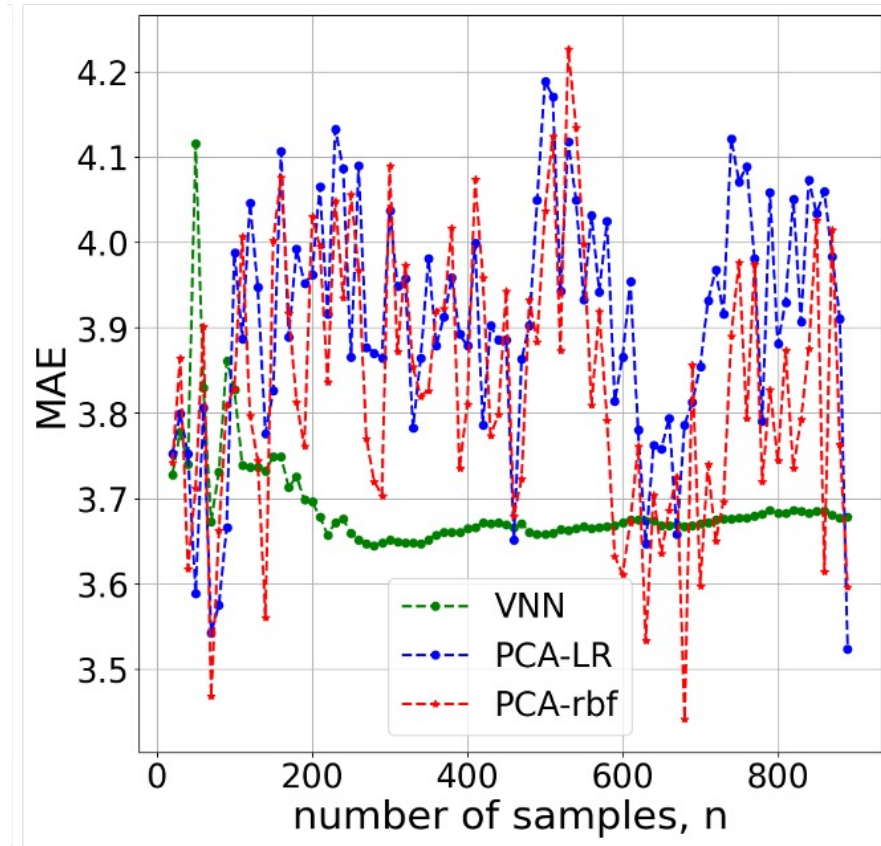
Example: A two-layer VNN



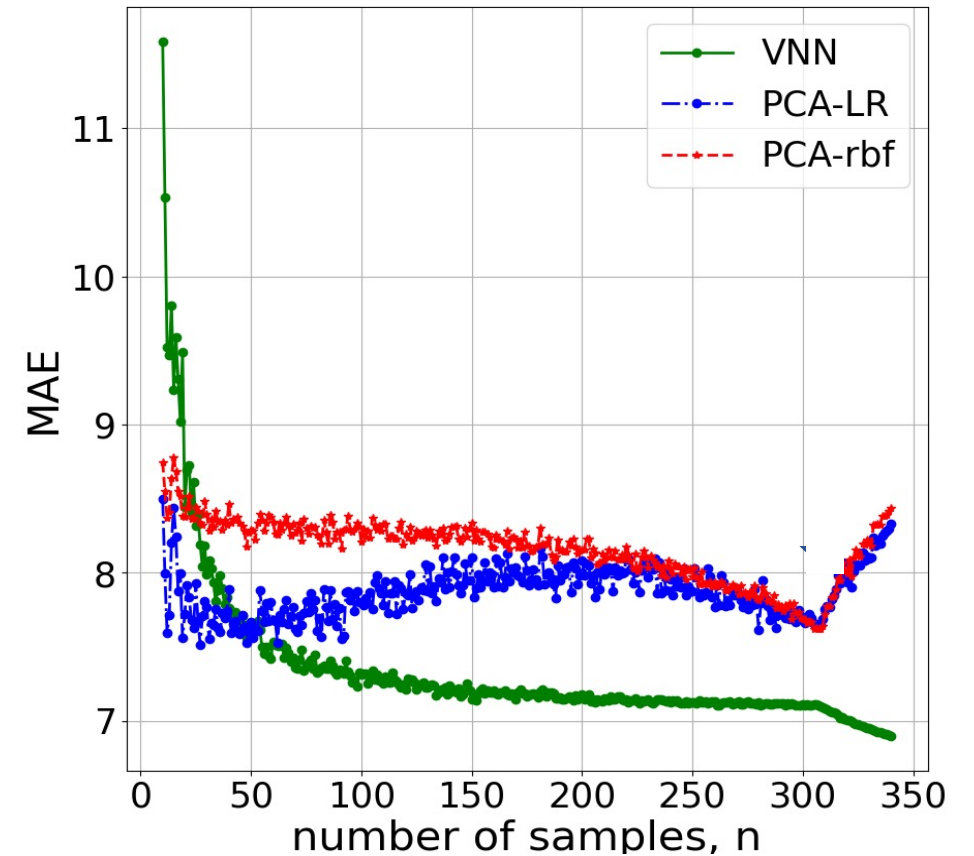
- $\Phi(\mathbf{x}; \hat{\mathbf{C}}, \mathcal{H})$ represents VNN output
- \mathcal{H} is set of all filter taps

VNNs outperform PCA (regression task)

Synthetic data
(Friedman regression problem)



Neuroimaging data
(age prediction task)

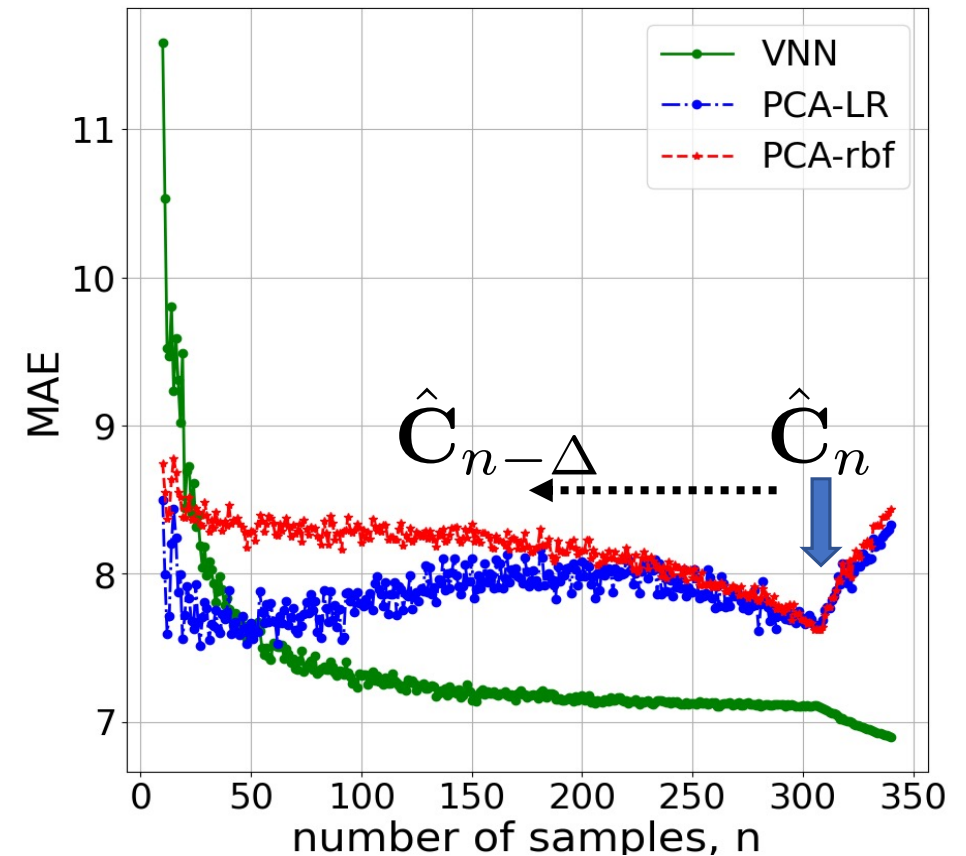


Stable Inference with VNNs

Stability of inference with PCA and VNNs

- PCA-driven inference can be **unstable** to stochastic perturbations in sample covariance matrix (**finite sample effect**)
- VNNs provide **stable** outcomes
⇒ enhanced reproducibility

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}

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

\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}

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

\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}

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

⇒ 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 $\mathbf{V}, \boldsymbol{\Lambda}$ [*]

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

⇒ **Unstable** PCA transform when eigenvalues of covariance are **close**

[*] Loukas, Andreas, 2017

Stability of coVariance filter

➤ How to gauge stability?

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

⇒ 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?

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

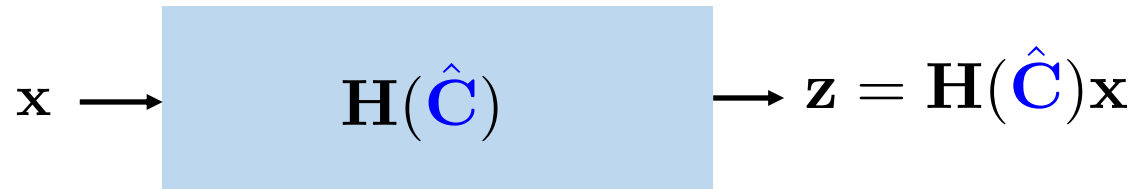
⇒ Output \mathbf{z} must be robust to number of samples n used to estimate $\hat{\mathbf{C}}$

- Compare filter outputs for **sample** and **true** covariance matrix

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

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

Stability of coVariance filter

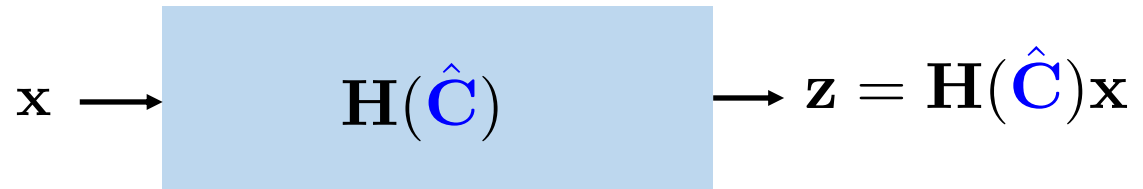


Stability result [Sihag et al., 2022]

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

} coVariance filter output is asymptotically consistent

Stability of coVariance filter



Stability result [Sihag et al., 2022]

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

Assumption.

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

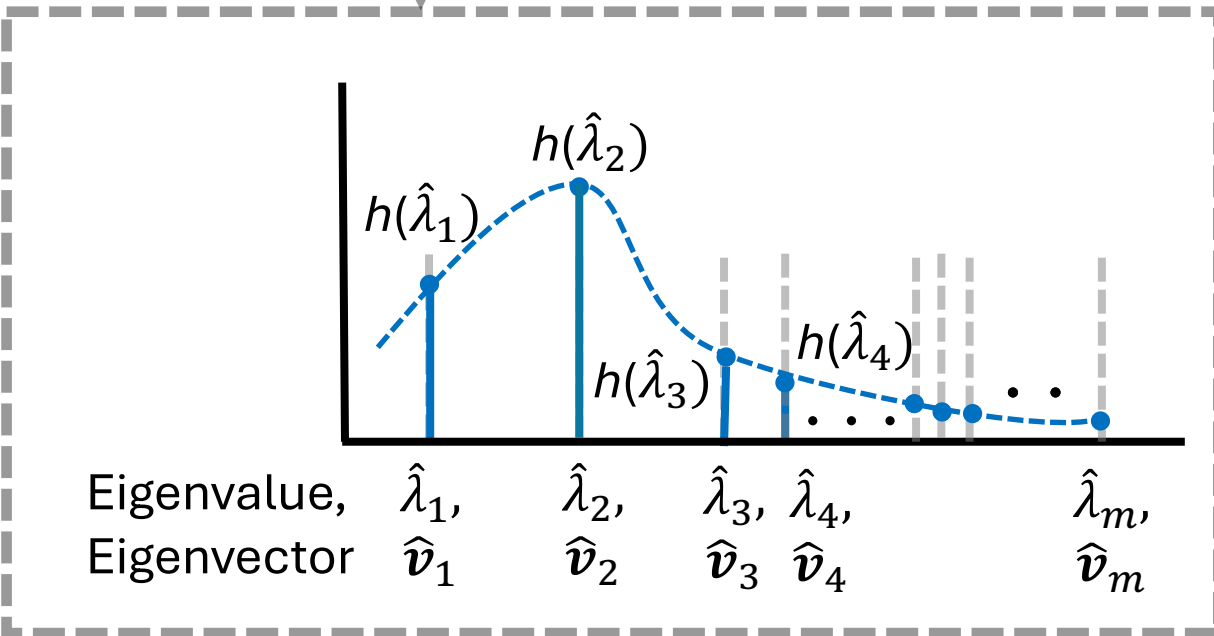
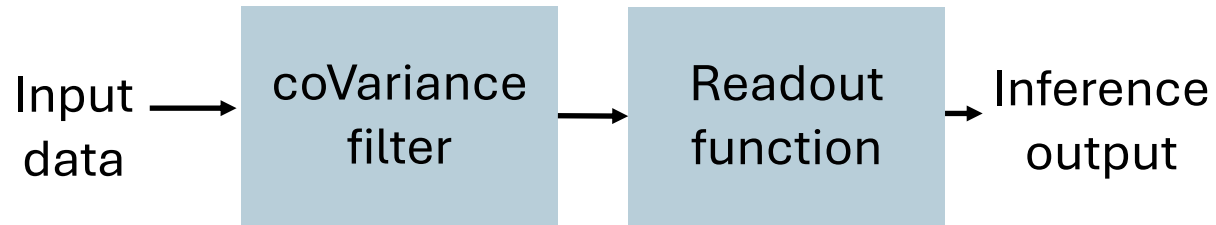
$$|h(\lambda_i) - h(\lambda_j)| \leq Q \frac{|\lambda_i - \lambda_j|}{k_i}$$

coVariance filter output is asymptotically consistent

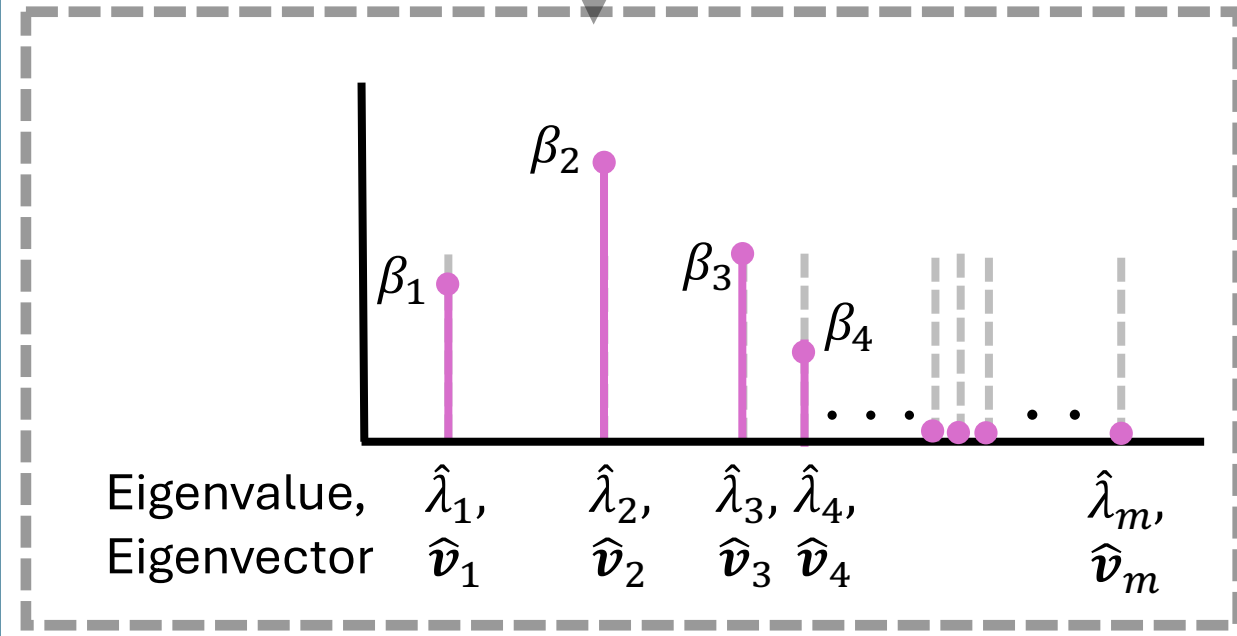
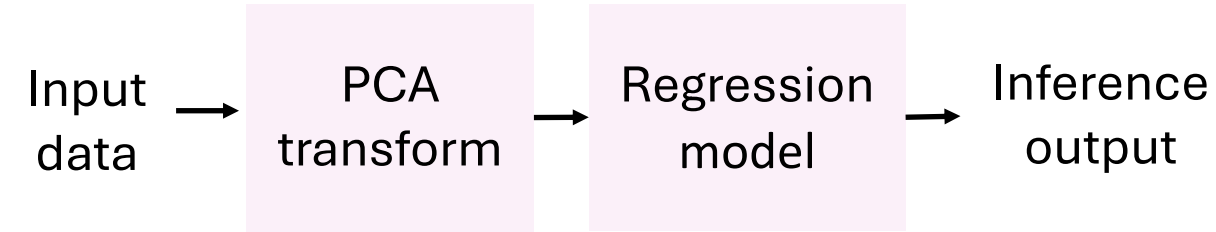
coVariance filter sacrifices discriminability between close eigenvalues for stability

Recall: Learning with coVariance filter versus PCA-based learning

➤ Learning with a coVariance filter

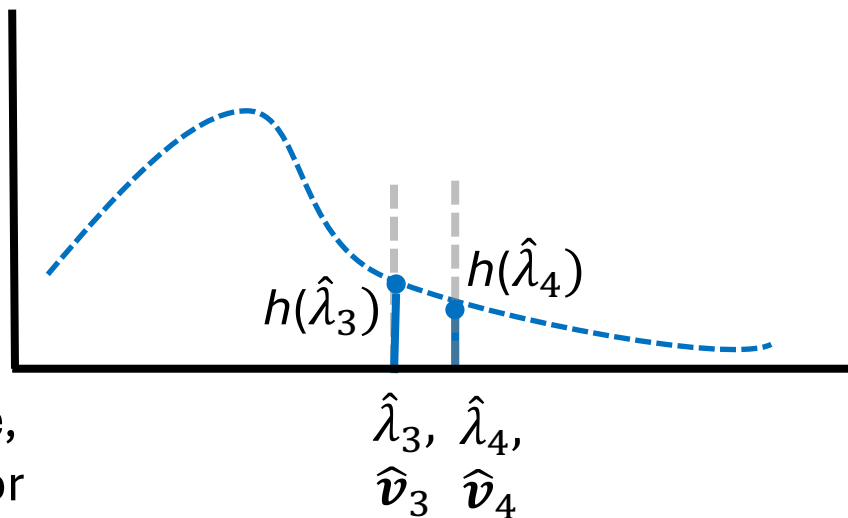
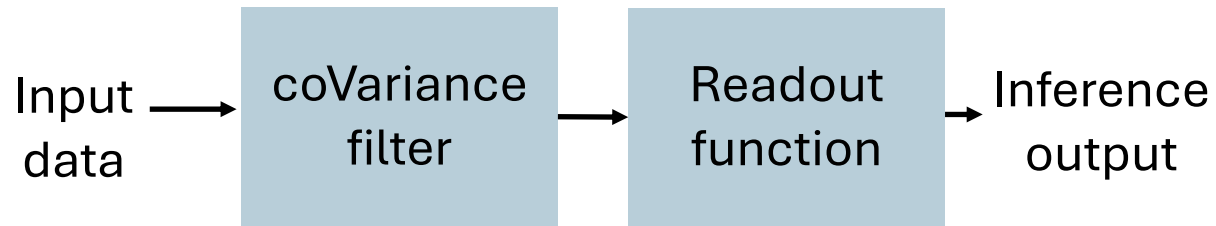


➤ PCA-based learning

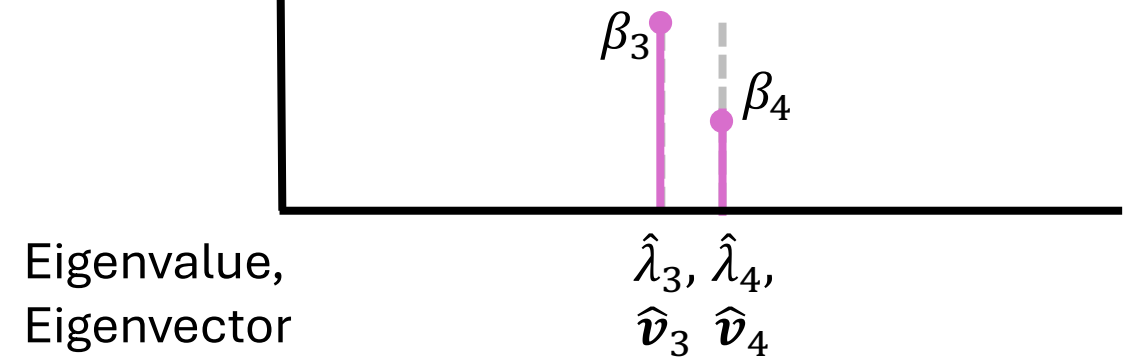
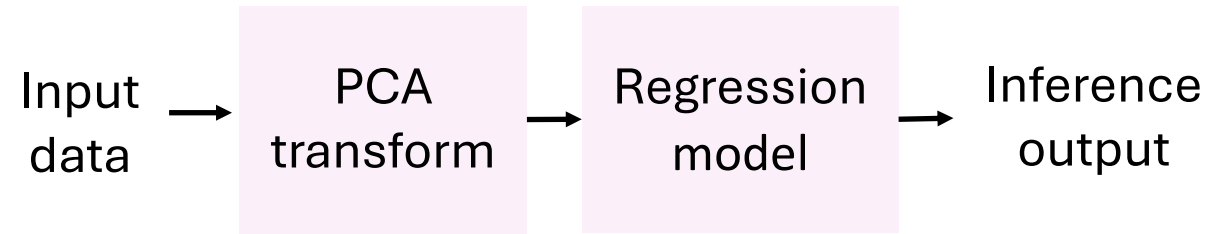


Why is coVariance filter more stable than PCA?

➤ Learning with a coVariance filter

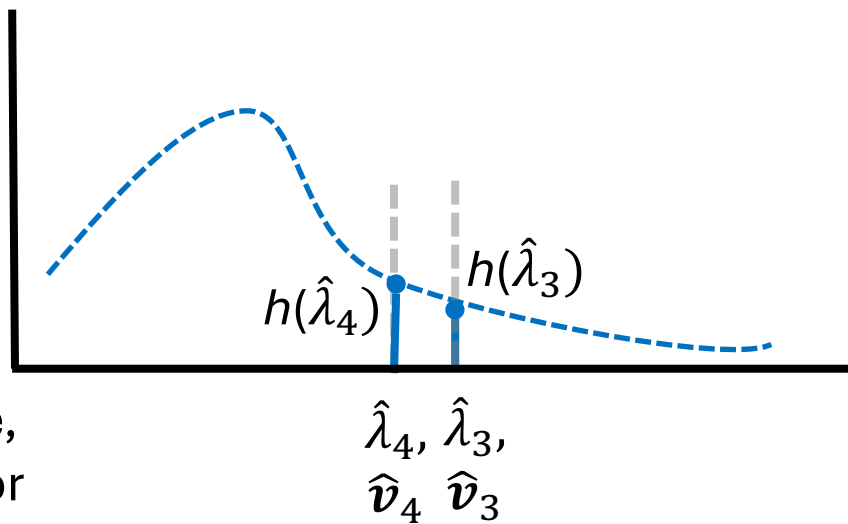
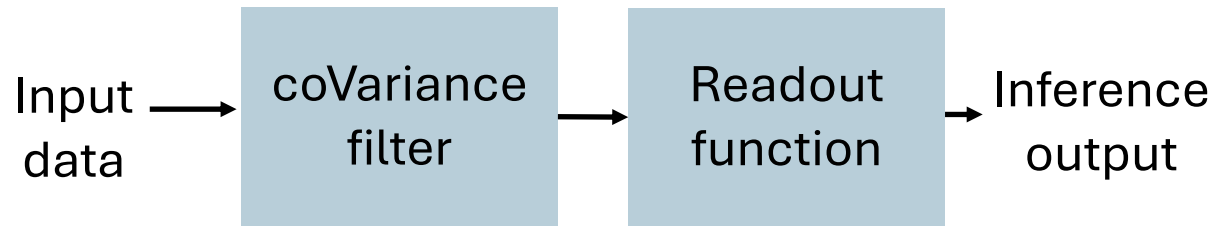


➤ PCA-based learning

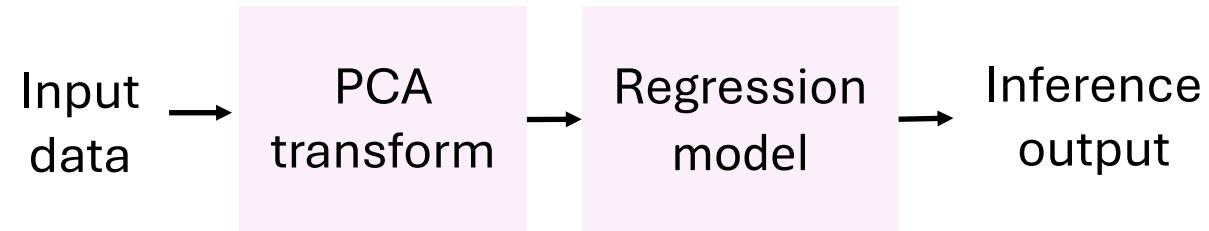


Why is coVariance filter more stable than PCA?

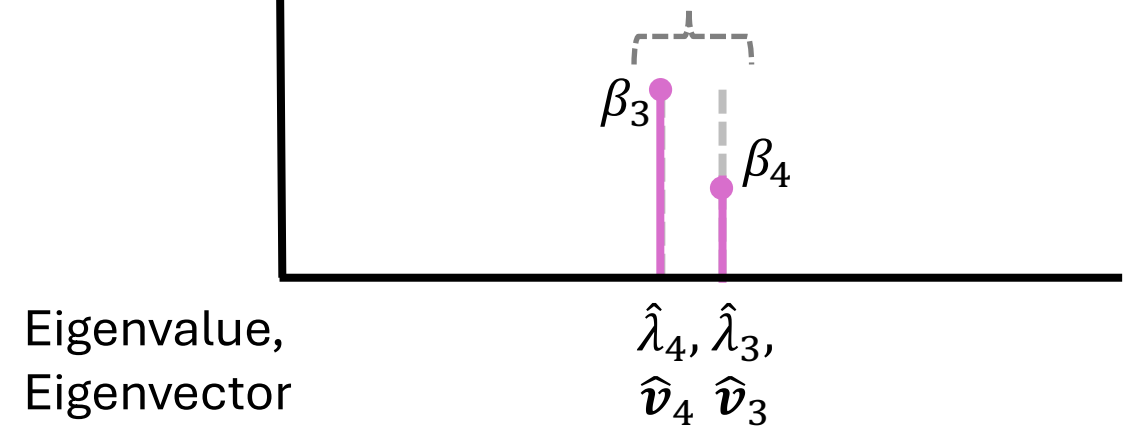
➤ Learning with a coVariance filter



➤ PCA-based learning



Overfitting on the ordering of eigenvalues is source of instability



Stability of VNNs

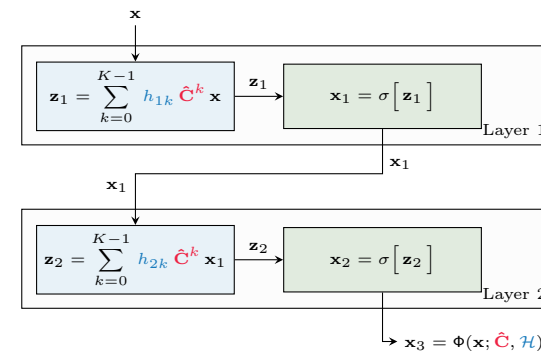
- VNNs inherit the stability from coVariance filters
 - Stability bound depends on the bound for filters

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

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

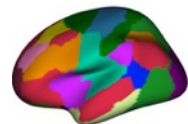
$$\left\| \Phi(\mathbf{x}, \hat{\mathbf{C}}; \mathcal{H}) - \Phi(\mathbf{x}, \mathbf{C}; \mathcal{H}) \right\| \leq LF^{L-1} \alpha_n$$

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

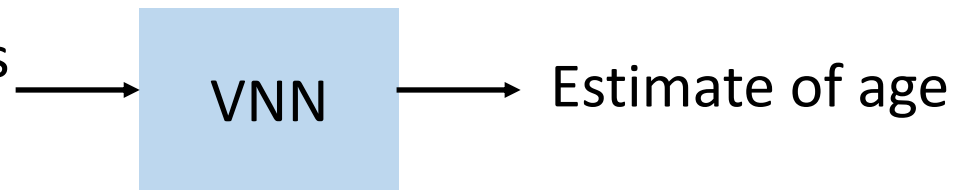


Stability of VNNs: Experiments

- Regression task



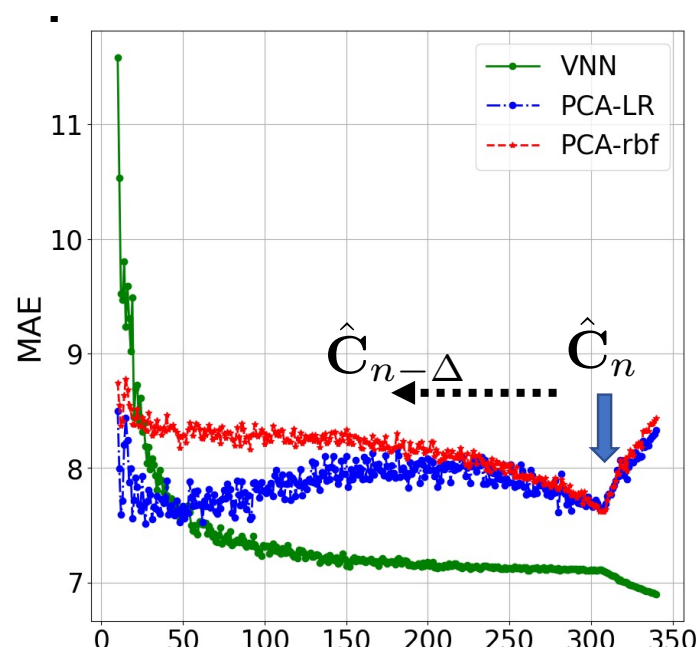
Cortical thickness
data



- 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

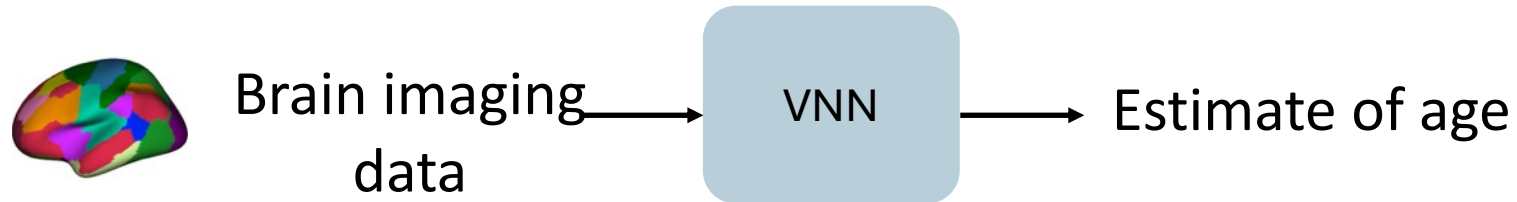
PCA-rbf: PCA regression with rbf kernel

VNN outperforms PCA and is more stable




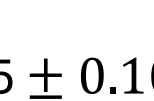
Transferability of VNNs

Empirical evidence of transferability across multiscale data

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

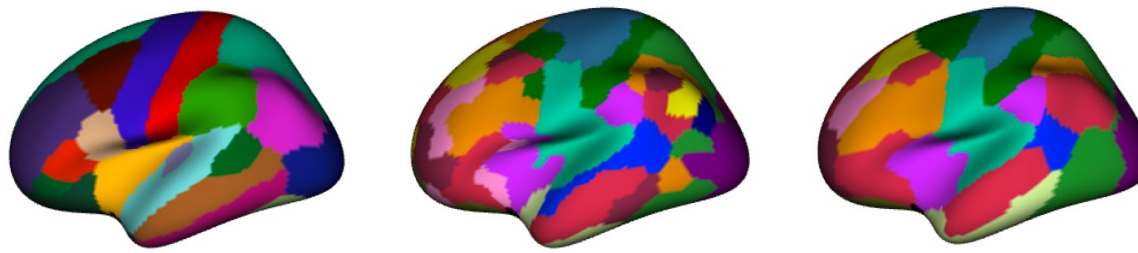


Transferability across datasets with different number of features

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

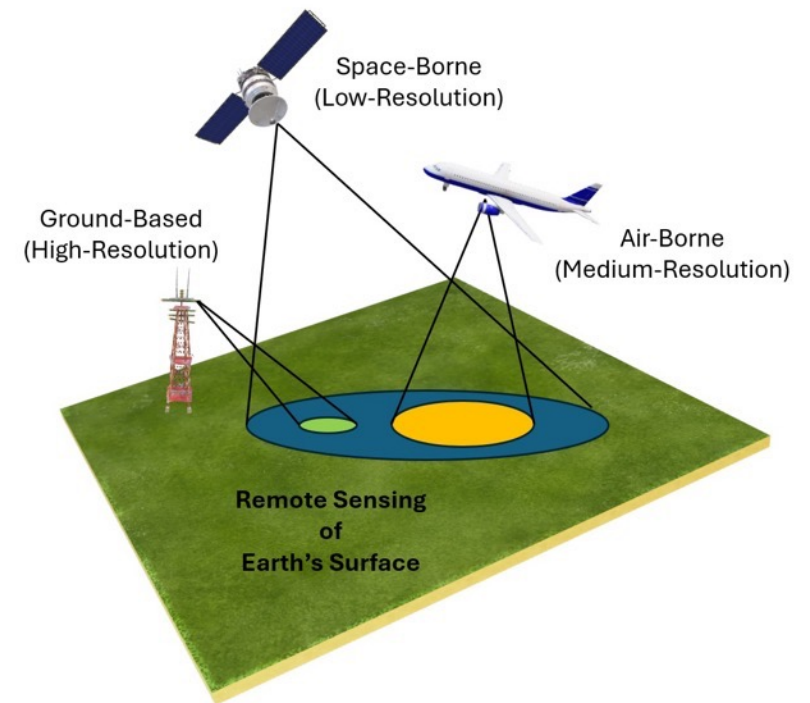
Transferability

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



Brain imaging data

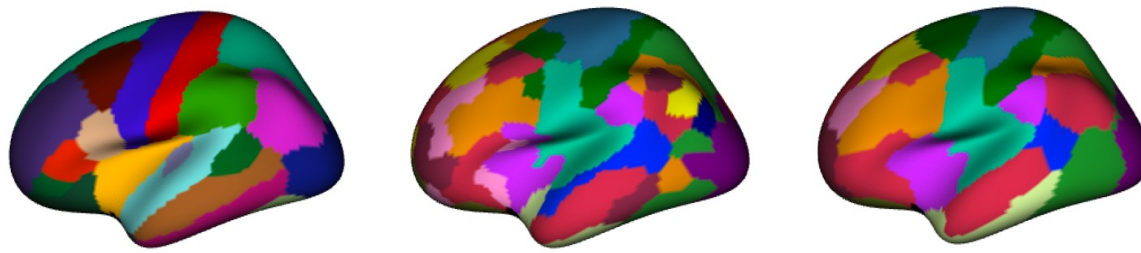
Remote sensing



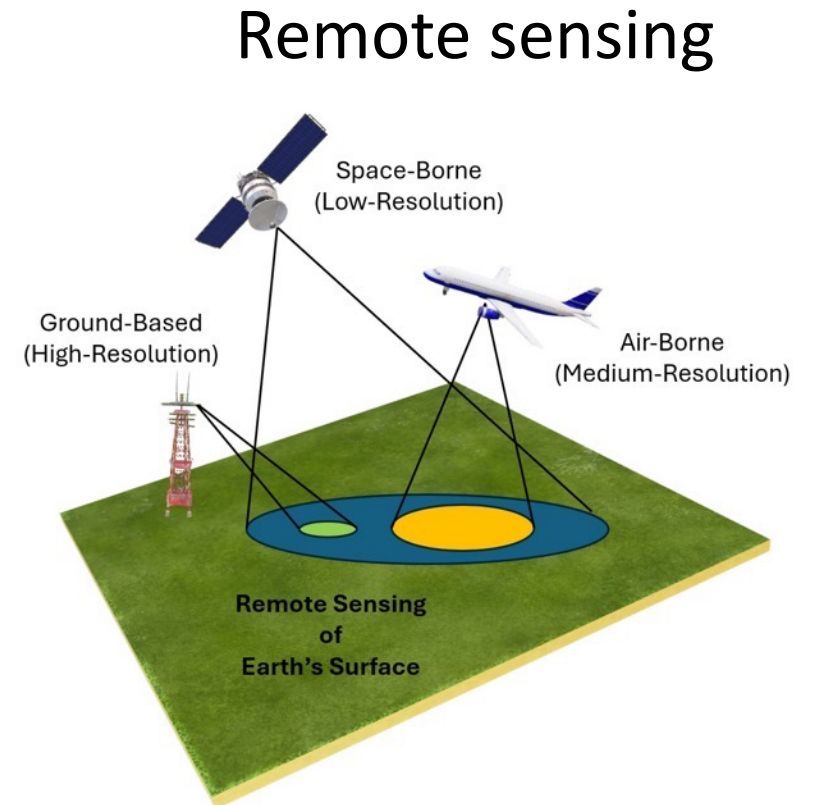
Credit: Mustafa Aksoy, UAlbany

Transferability

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



Brain imaging data



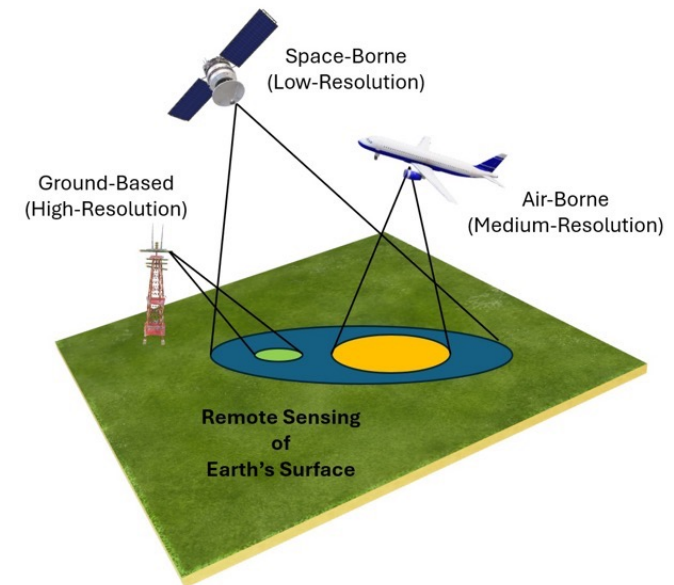
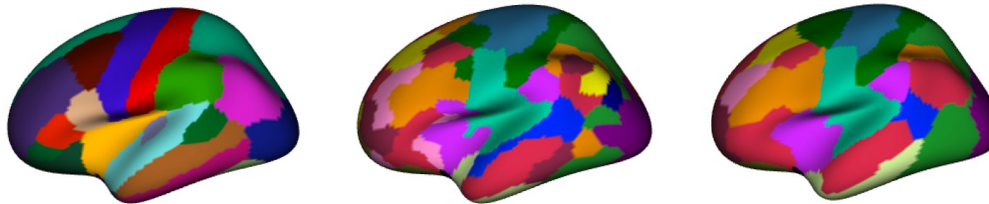
Credit: Mustafa Aksoy, UAlbany

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

Transferability

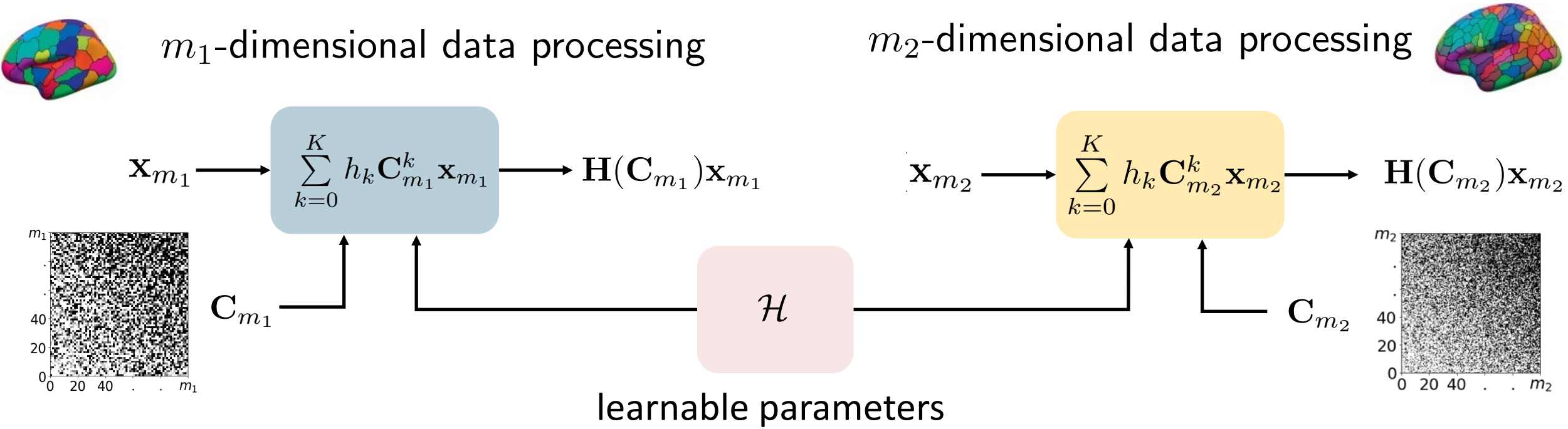
- Most statistical approaches, including PCA, operate within the dimensionality
 - ⇒ seamless transference not possible across different dimensionalities
- **This section:** How do VNNs transfer?

When is transference successful?



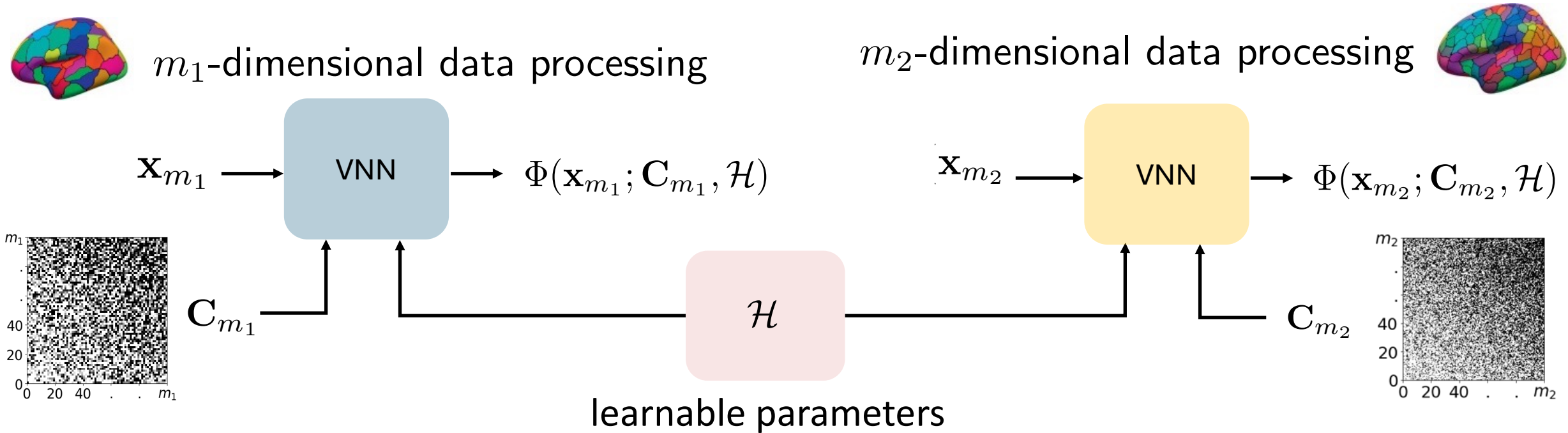
Credit: Mustafa Aksoy, UAlbany

coVariance filters are scale-free models



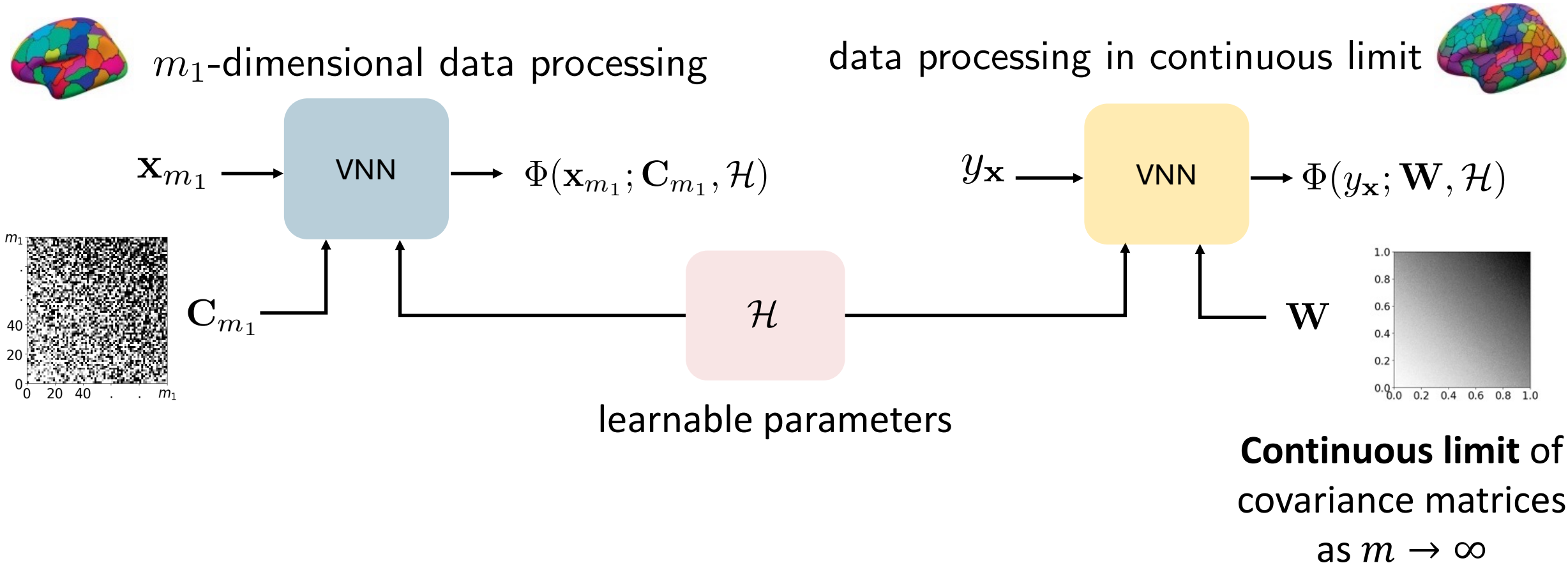
- A coVariance filter $\mathbf{H}(\cdot)$ with scalar filter taps $\{h_k\}$ can process dataset (covariance matrix) of any arbitrary dimensionality: **scale-free model**

VNNs as scale-free models



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

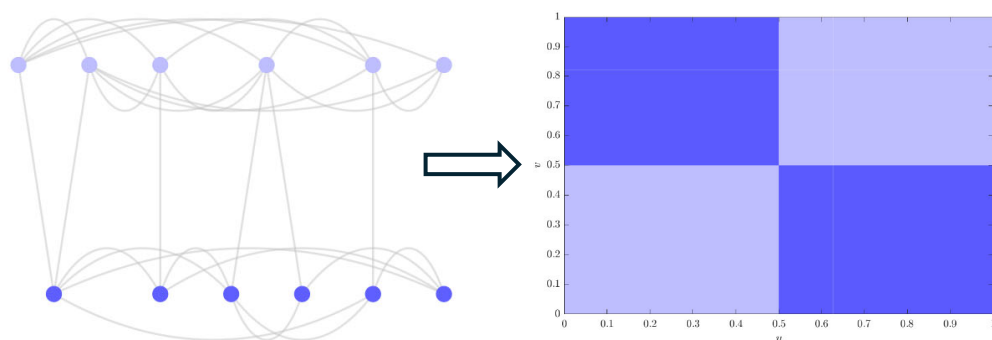
VNNs as scale-free models



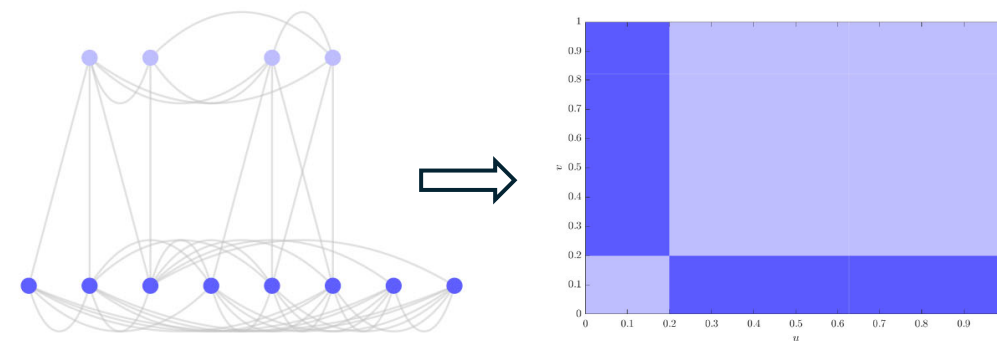
How to compare $\Phi(\mathbf{x}_{m_1}; \mathbf{C}_{m_1}, \mathcal{H})$ and $\Phi(y_{\mathbf{x}}; \mathbf{W}, \mathcal{H})$?

Graphons as continuous limits

- Graphs can have **limit objects** with uncountable number of nodes
- **Example:** Stochastic block models [Ruiz et al., TSP, 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 $\mathbf{W}(u, v) = \mathbf{W}(v, u)$

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

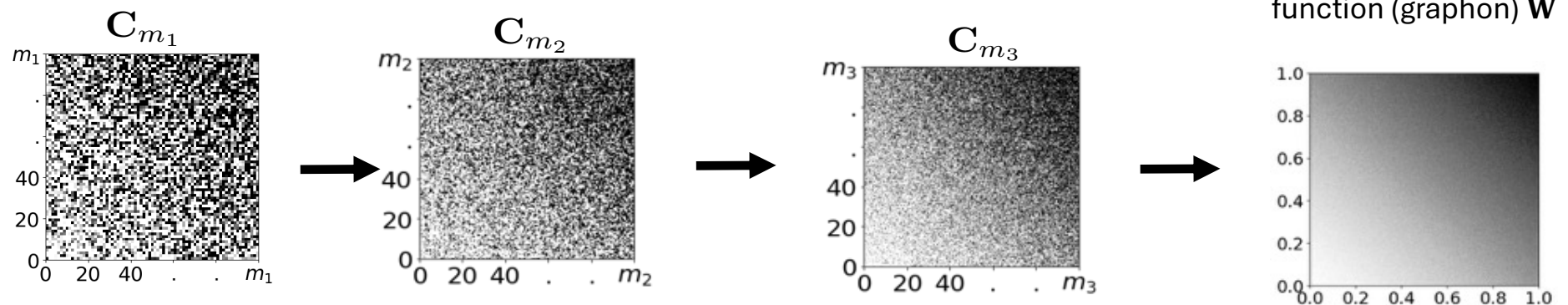
Graphons as continuous limits

➤ **Graphon:** A graphon is a symmetric, bounded measurable function

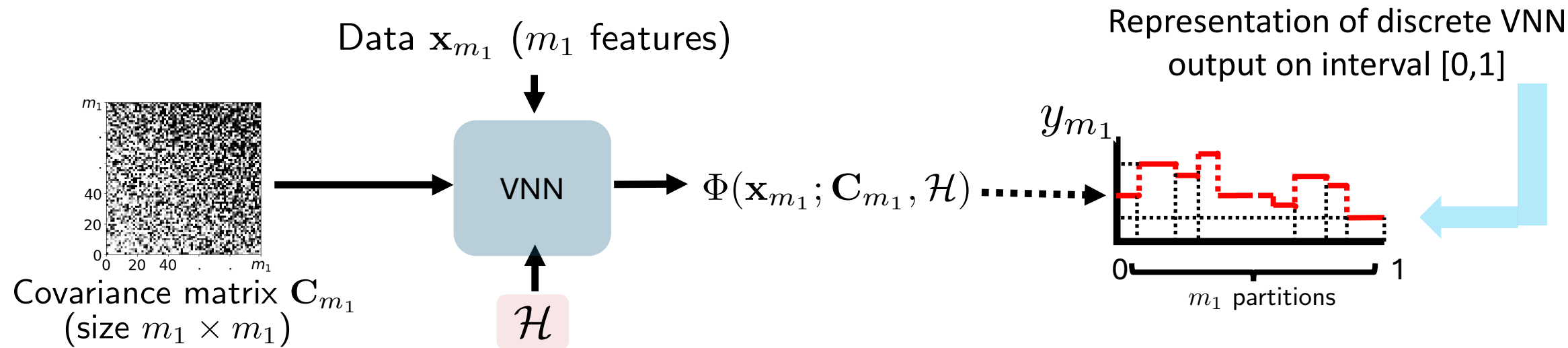
- Node labels are graphon arguments $u \in [0,1]$
- edge weights are graphon values $\mathbf{W}(u, v) = \mathbf{W}(v, u)$

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

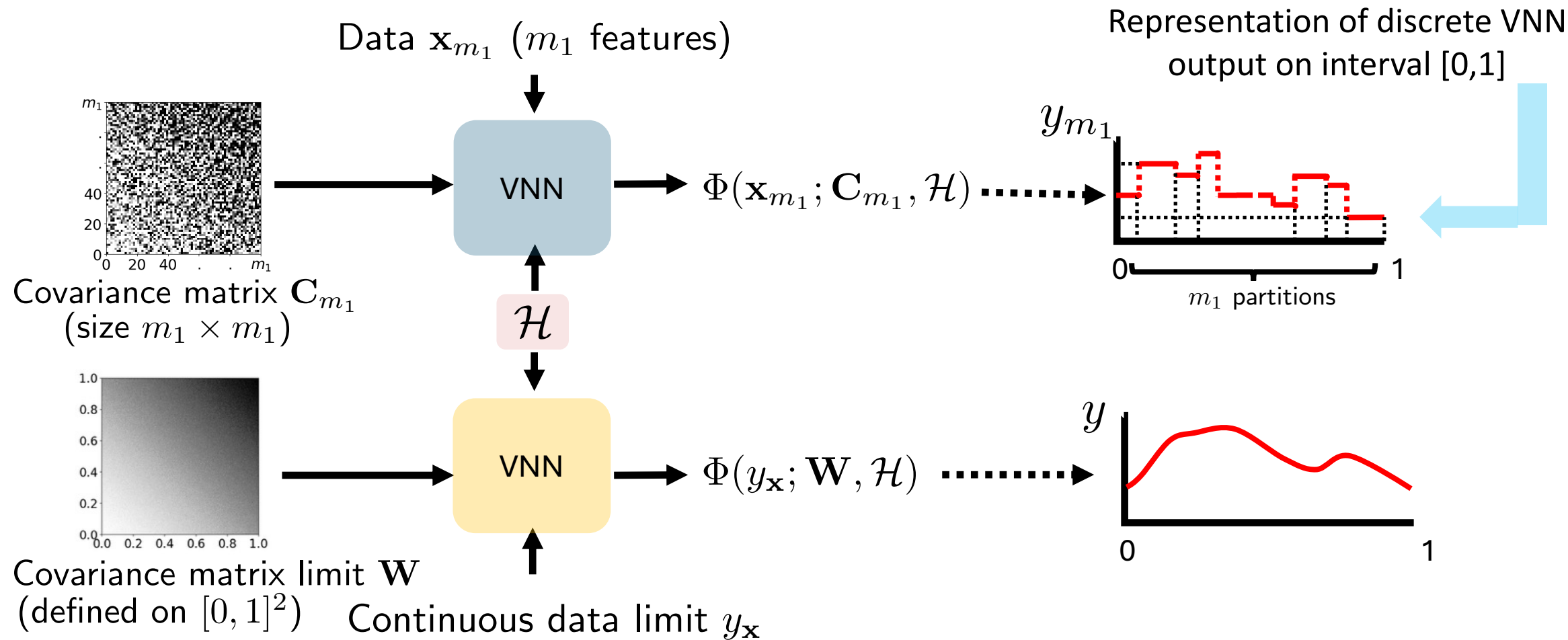
➤ Transferability when covariance matrix is part of some converging sequence



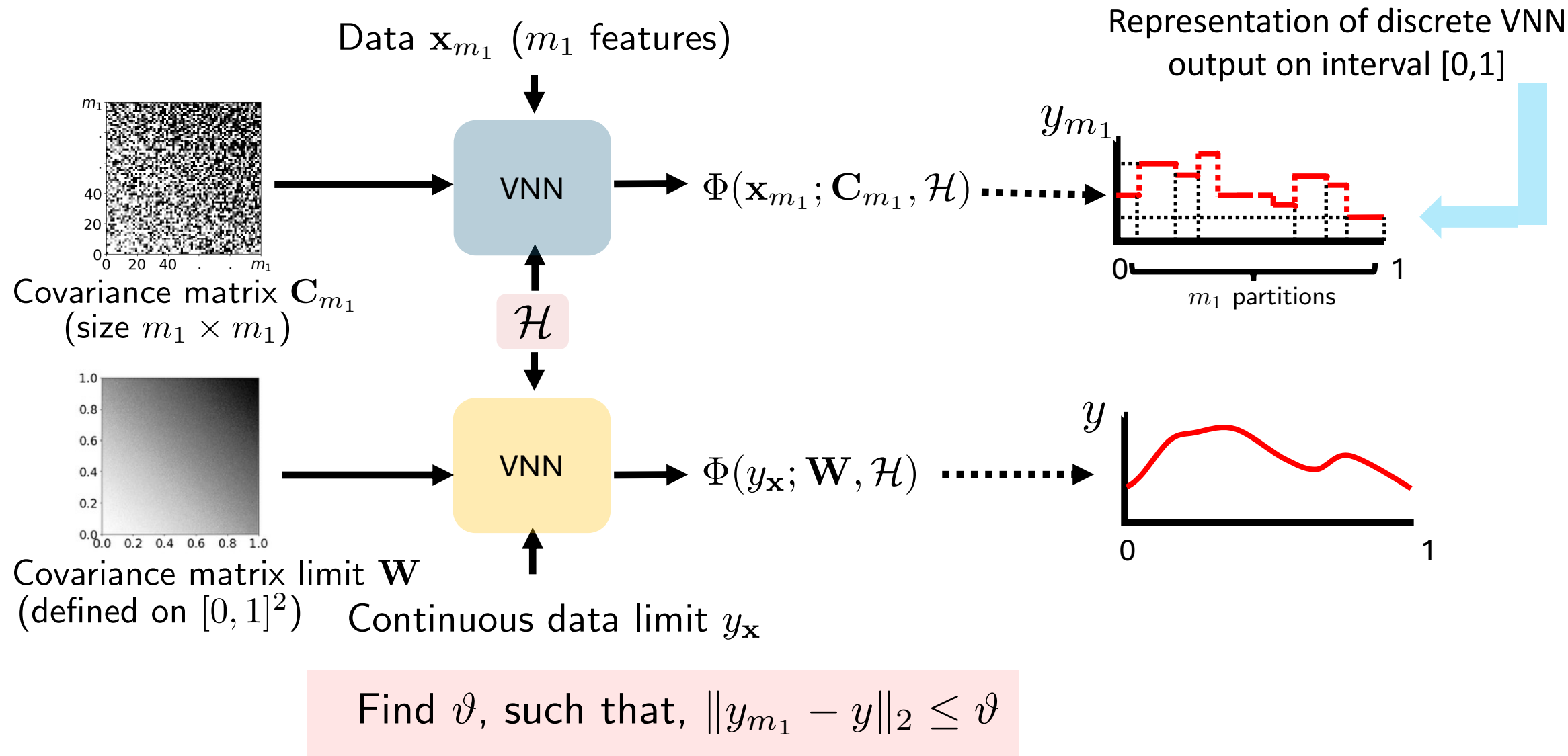
Problem formulation for transferability



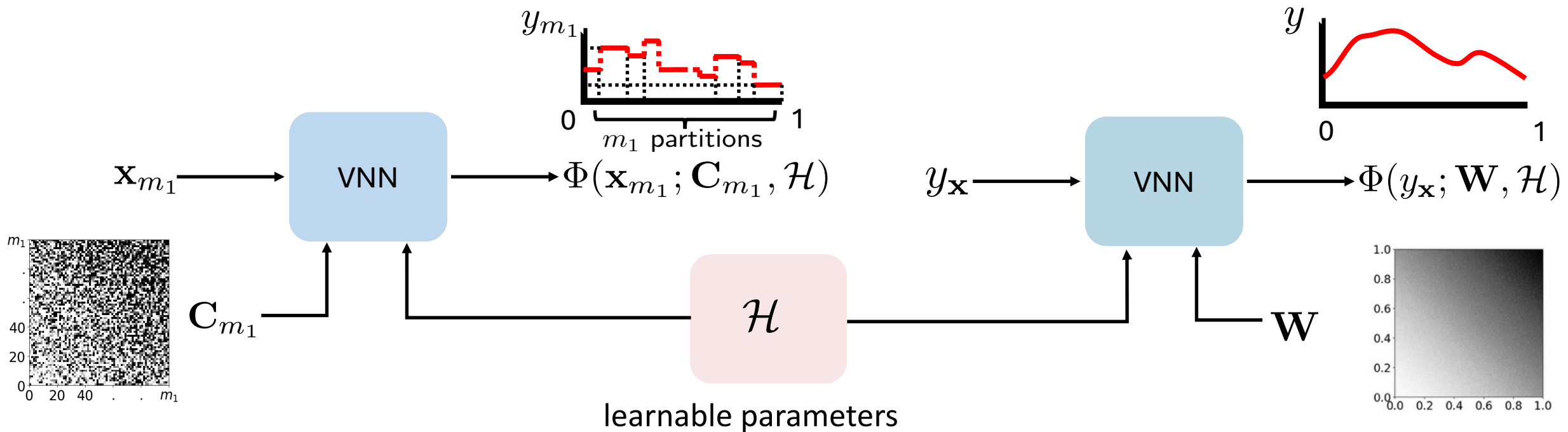
Problem formulation for transferability



Problem formulation for transferability



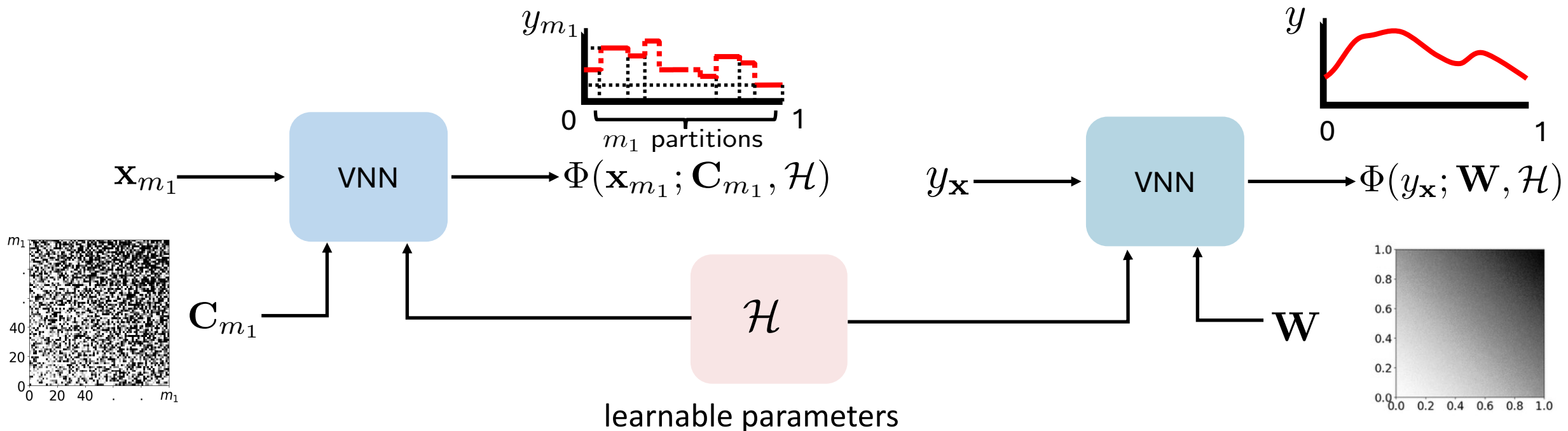
VNNs are provably transferable



Transferability bound* [Sihag et al., 2024]

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

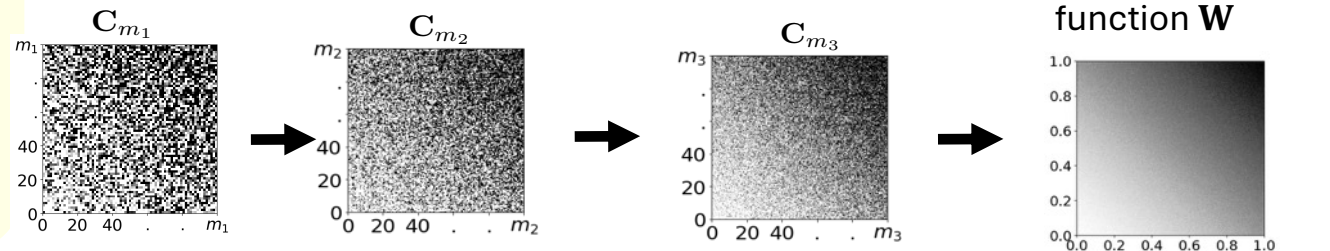
VNNs are provably transferable



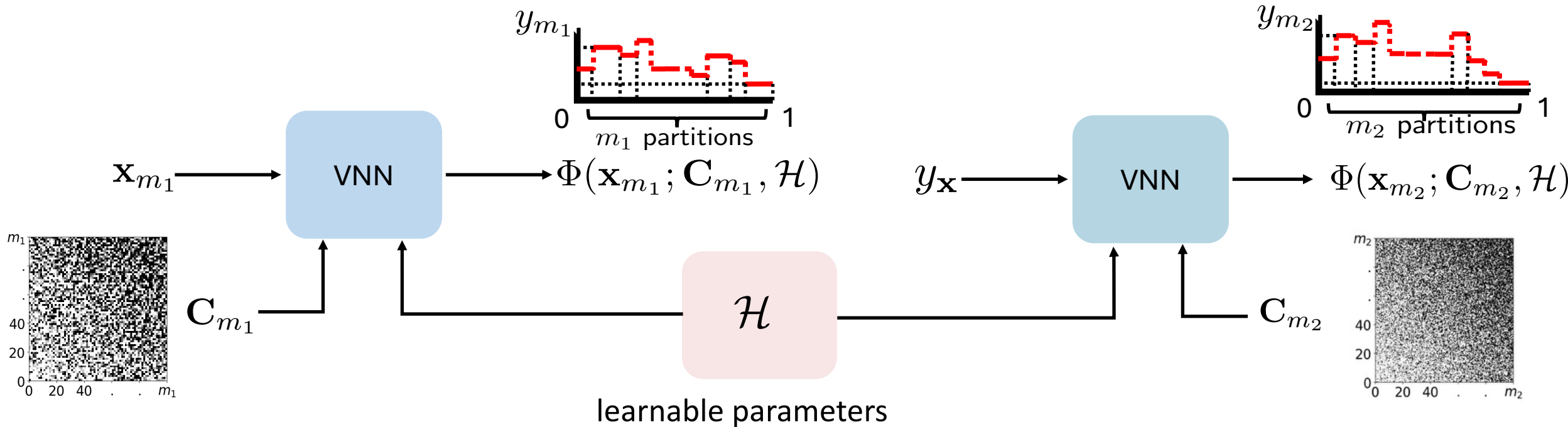
Transferability bound* [Sihag et al., 2024]

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

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

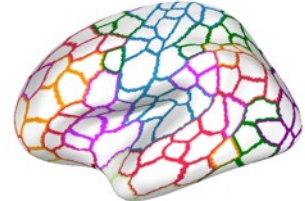
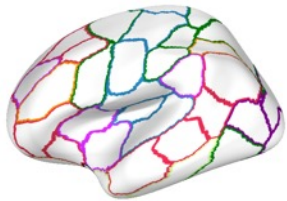


VNNs are provably transferable



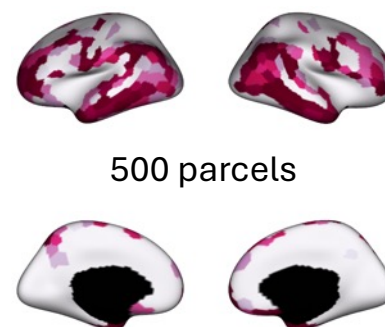
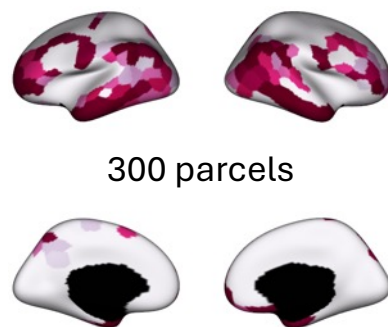
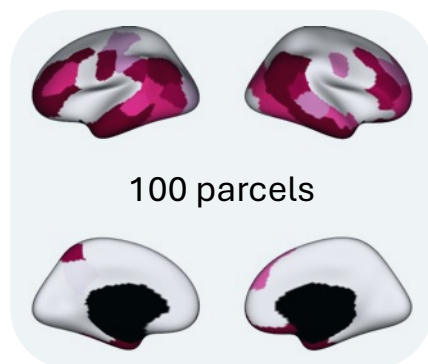
Transferability bound

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

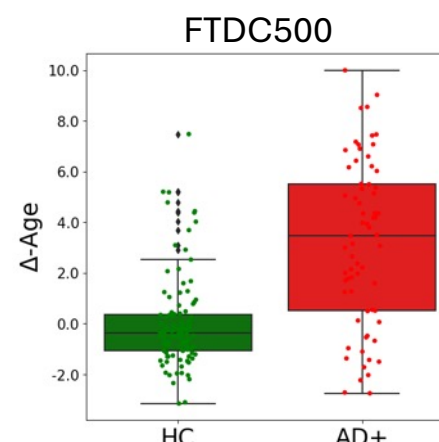
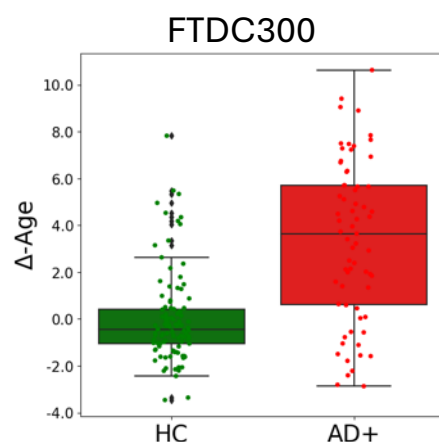
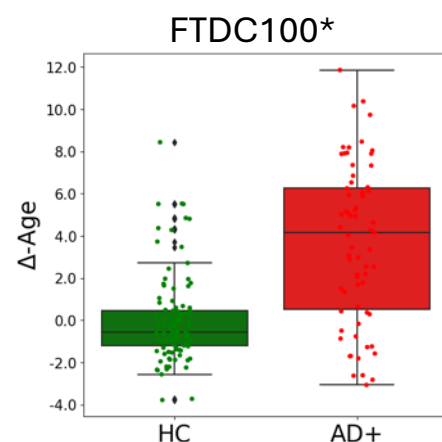


Experiments

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 and JSTSP 2024]



- 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

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 of matrix for dense covariance)
 - No considerations of **temporal, evolving** data

Sparse VNNs

- **Sparse** VNNs (S-VNN) rely on sparsification of sample covariance matrix
- Sparsification improves estimation quality
- Strategies to sparsify

- **Hard** thresholding

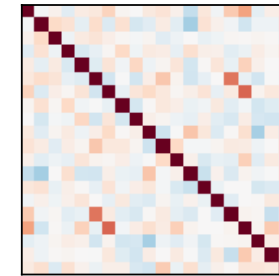
$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$

- **Soft** thresholding

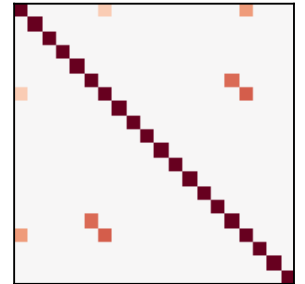
$$\eta(\hat{\mathbf{C}})_{ij} = \hat{c}_{ij} - \text{sign}(\hat{c}_{ij})\tau/n \text{ if } |\hat{c}_{ij}| \geq \tau/\sqrt{n}, 0 \text{ otherwise}$$

- Both thresholding strategies preserve stability in S-VNNs [Cavallo et al., 2024]

Empirical covariance

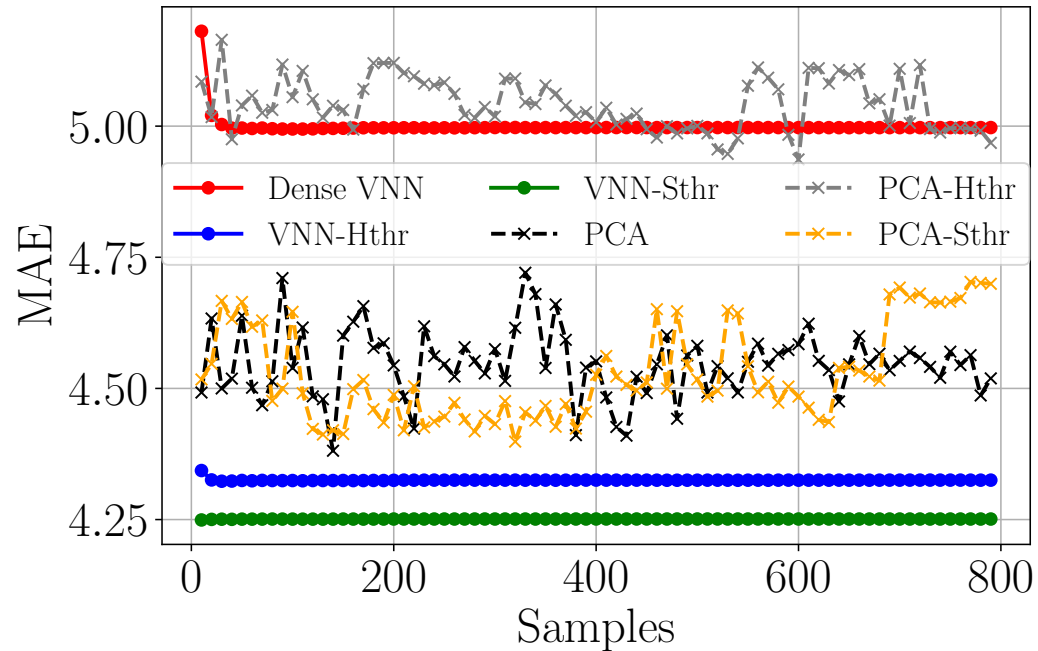


True covariance



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 nominal VNNs
- VNNs more stable than PCA

Spatiotemporal VNNs

- VNN models discussed so far operate on *static* data
 - ⇒ non-trivial modifications needed to handle temporal, non-stationary data
- **Spatio-temporal VNNs (STVNNs)**

- **Model design**

1. Online covariance matrix estimate

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

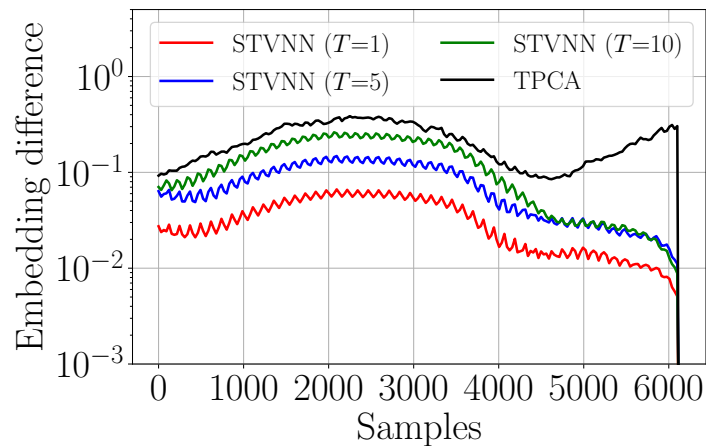
2. Spatio-temporal coVariance filter

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

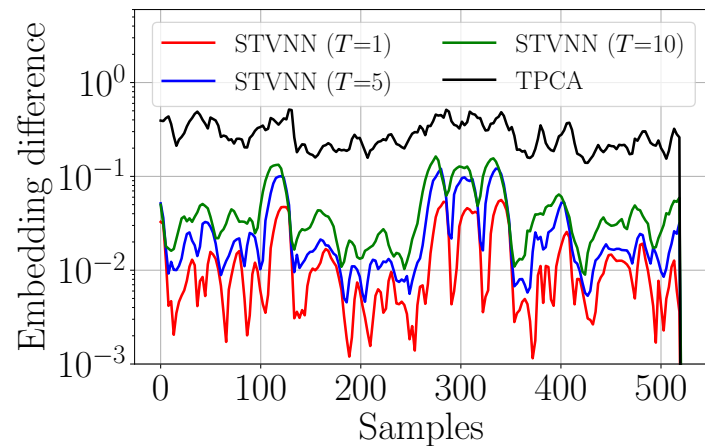
Spatial and **temporal**
convolution

Spatiotemporal VNNs

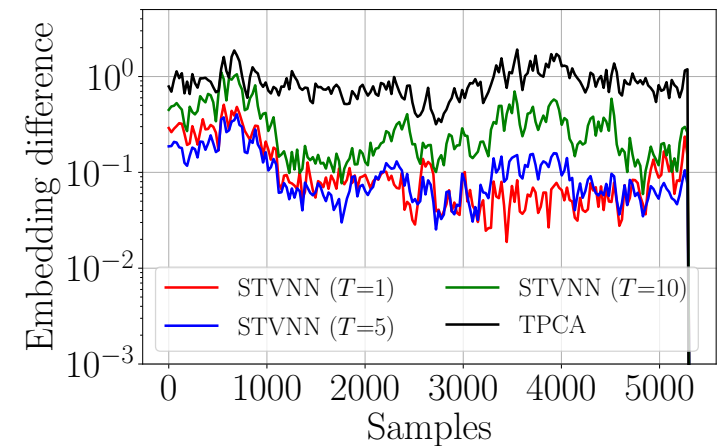
- STVNNs are stable to estimation errors in covariance [*]
- **Numerical results**
 - Time series forecasting task (weather data and currency exchange rates)
 - Train with one covariance, test with another estimated from fewer samples



NOAA



Molene



Exchange rate

[*] Cavallo et al., 2024

Concluding Remarks

- **Learning with covariance matrices**
 - Covariance matrices encode redundancies within dataset
 - their eigenvectors (principal components) inform the directions of maximum variance
 - PCA-driven methods can be **unstable**
 - PCA operates **restricted** to datasets of same dimensionality
- **CoVariance neural networks (VNNs)**
 - VNNs provide GSP-motivated implementation of PCA
 - Stable outcomes, transference across multiscale datasets

Concluding Remarks

- **Emerging areas** we did not cover in detail
 - **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]

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,” 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 Forty-first 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." ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2025.

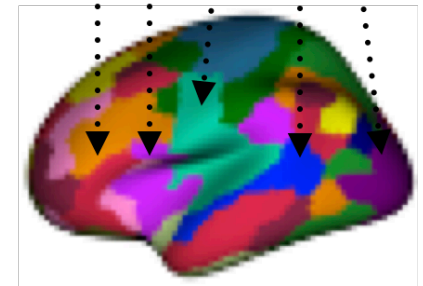
A. Cavallo, M. Sabbagi, and Isufi, Elvin, “Spatiotemporal covariance neural networks,” in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 18–34, Springer, 2024.

Principled brain age gap prediction with VNNs

Neuroimaging Data: Basics

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

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

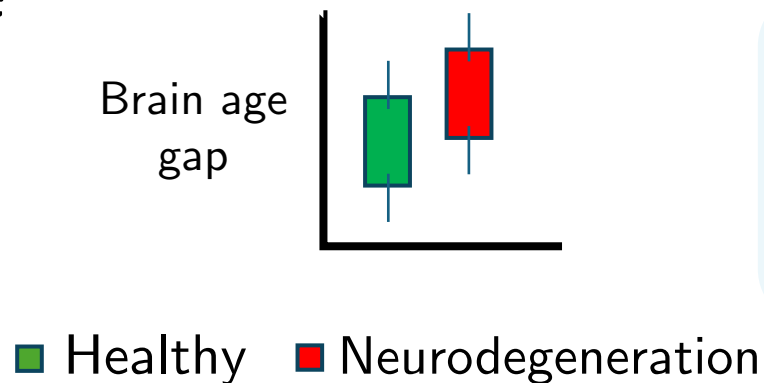


Anatomic features

- 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)

Brain age gap is a marker of neurodegeneration

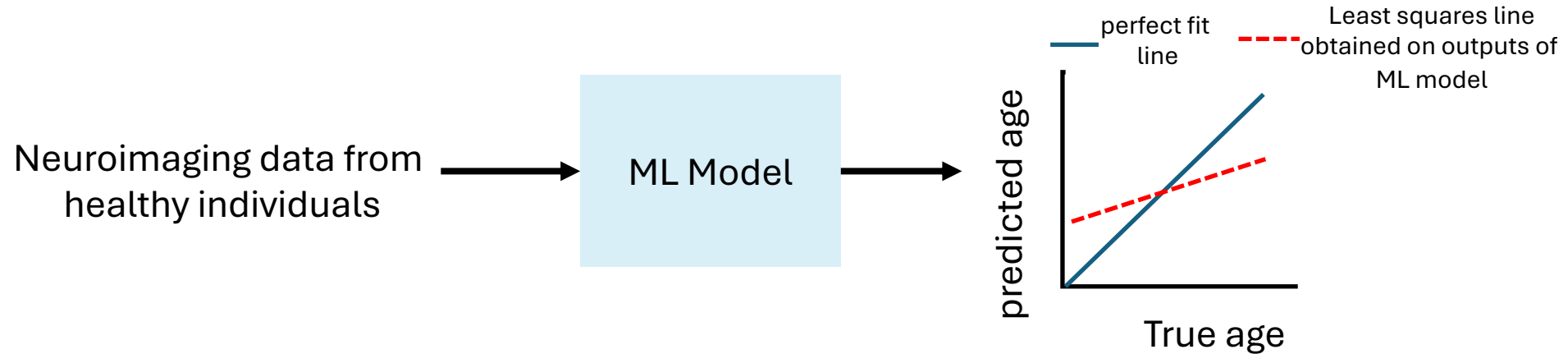
- Individual rate of “aging” is different from chronological rate of aging
 - Driven by environment, genetics, behavior, **neurodegeneration**
- **Brain age** provides a biological estimate brain age, derived from brain imaging modalities
- The **brain age gap** is the deviation between brain age and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



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

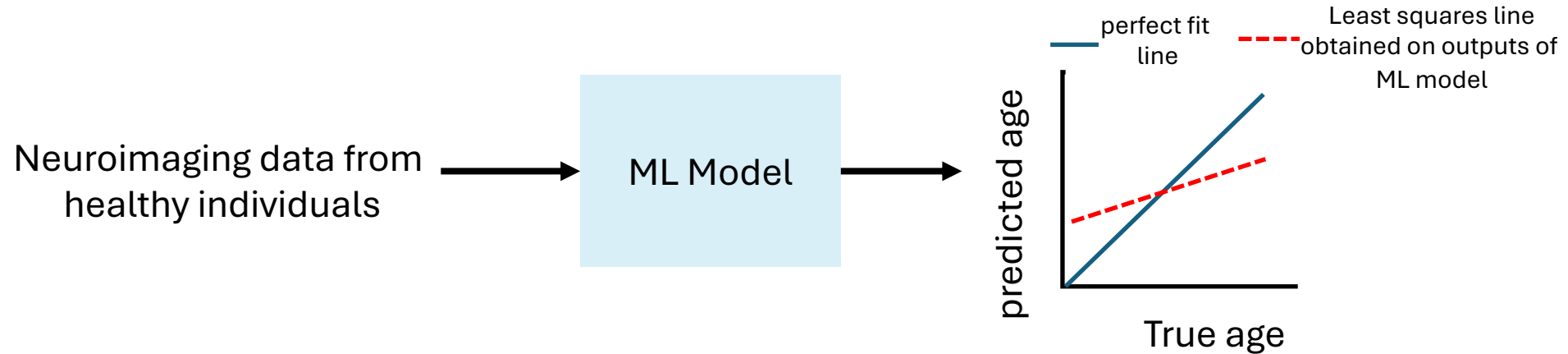
Brain age gap evaluation using ML

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



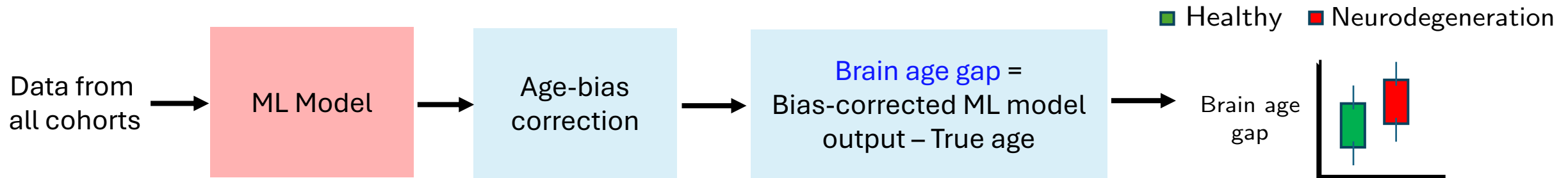
Brain age gap evaluation using ML

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

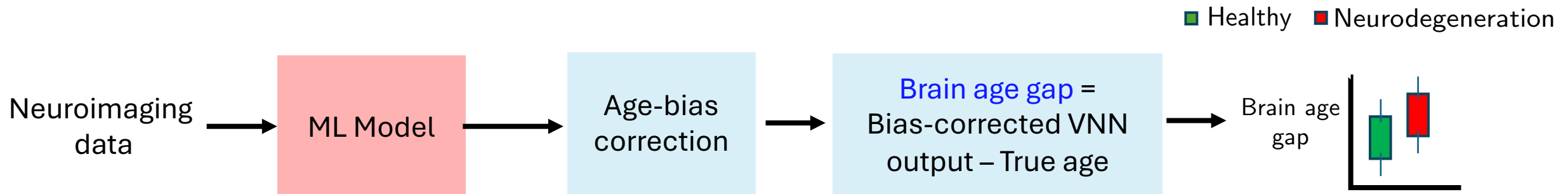


Step 2. Linear regression-based age-bias correct for outputs of ML model

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



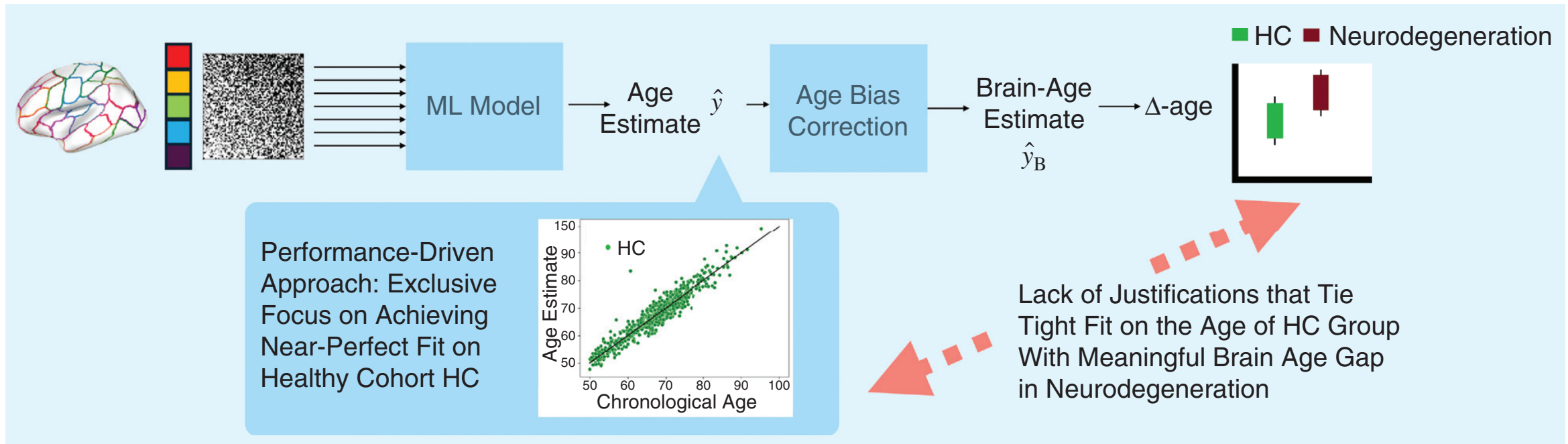
Choice of learning parametrization



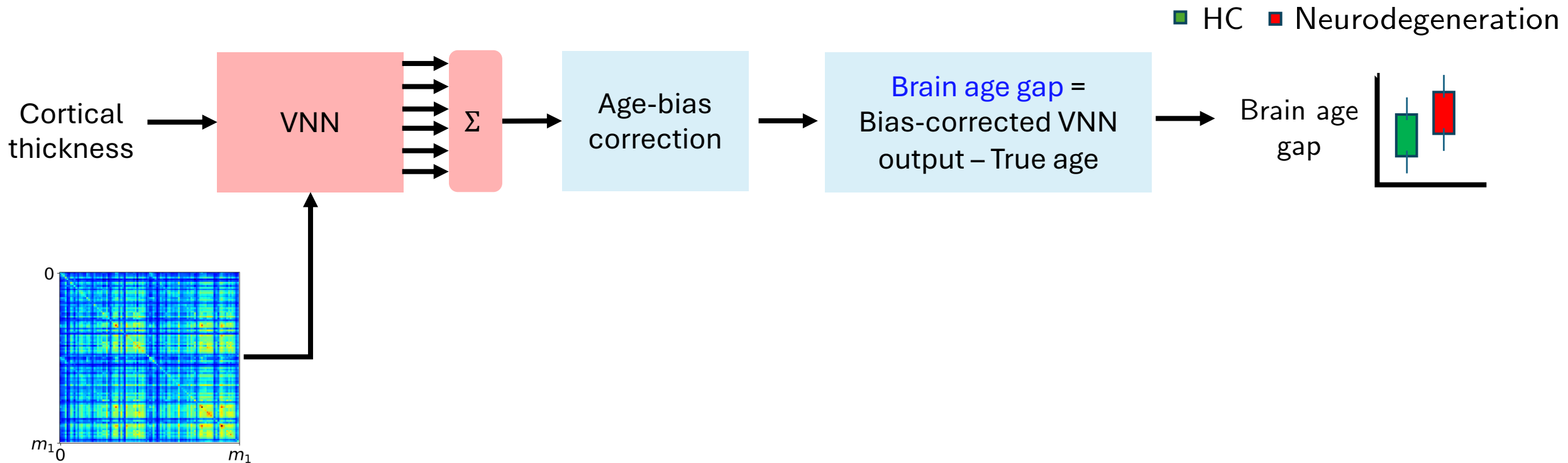
- Choice of ML model determines how information is leveraged to gauge brain age
- Prevalent approaches leverage neural networks as **ML model** to achieve best fit on healthy population: **Performance-driven approach**
- **Performance-driven approaches** do not necessarily lead to a 'meaningful' brain age gap

Choice of learning parametrization

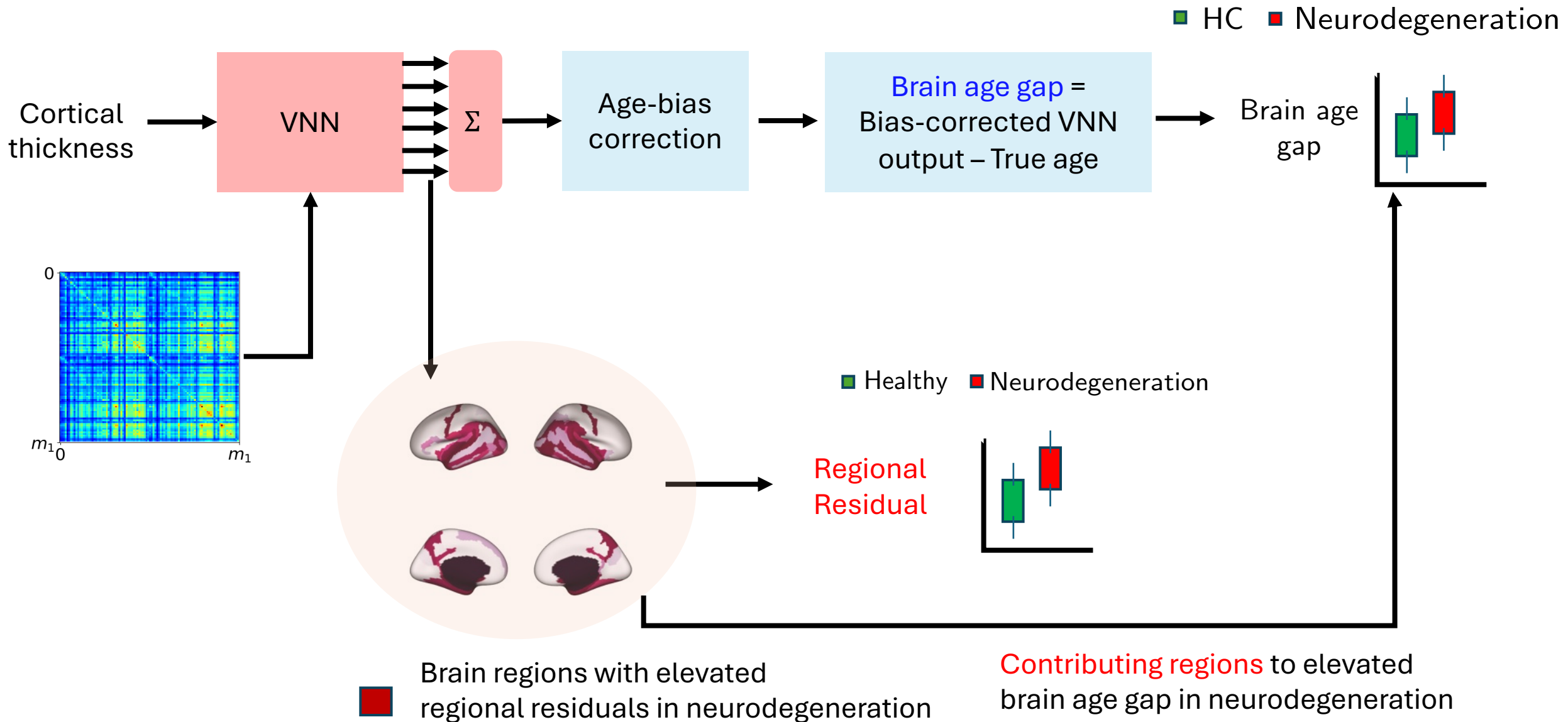
- Neural networks are prevalent in performance-driven approaches
 - A Neural Network may **not be interpretable** and prone to **overfitting**
- ⇒ **methodological obscurity** in brain age gap prediction pipeline



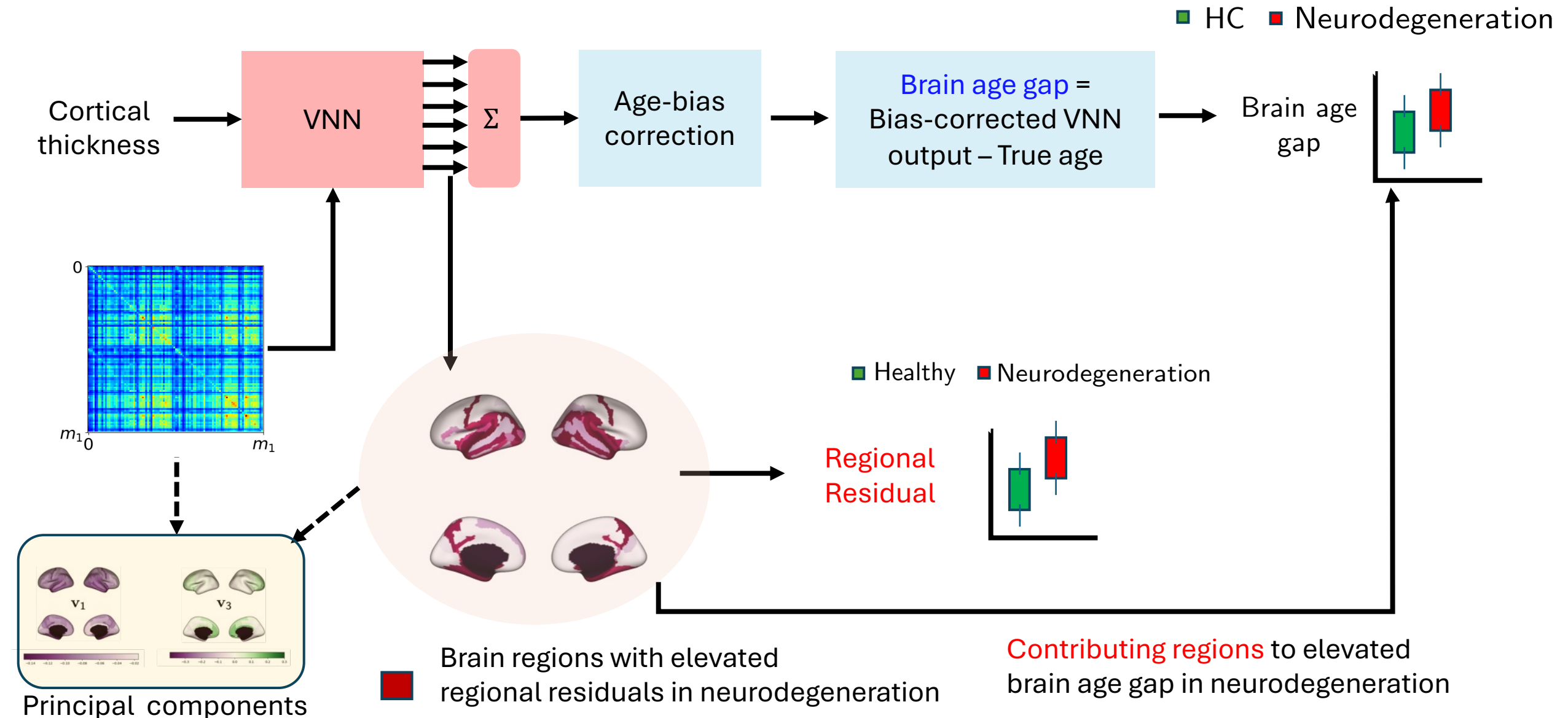
VNNs provide an anatomically interpretable and explainable brain age gap



VNNs provide an anatomically interpretable and explainable brain age gap



VNNs provide an anatomically interpretable and explainable brain age gap



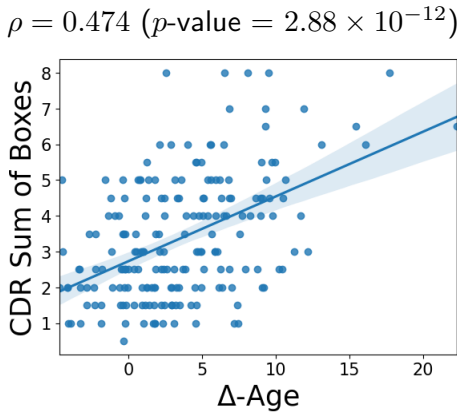
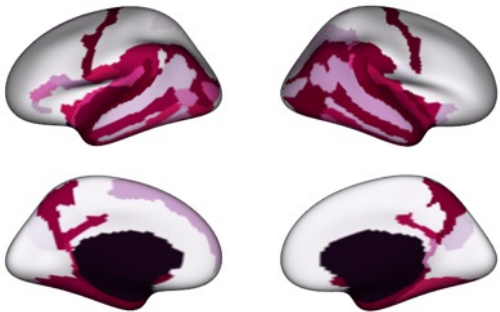
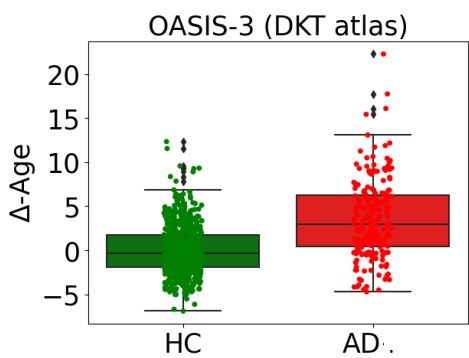
Experiments

- 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

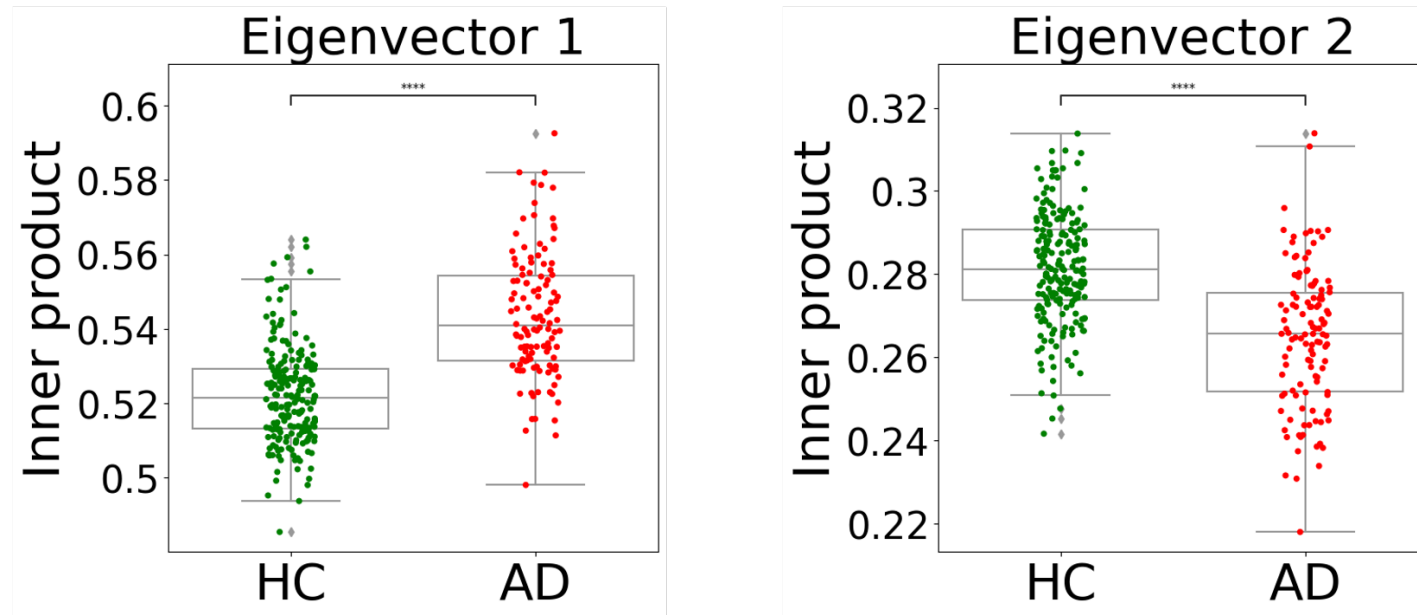
- Brain age gap is elevated in AD group and correlated with CDR sum of boxes



[*] Pamela J LaMontagne, et al. OASIS-3: longitudinal neuroimaging, clinical, and cognitive dataset for normal aging and Alzheimer disease. MedRxiv, 2019

Experiments

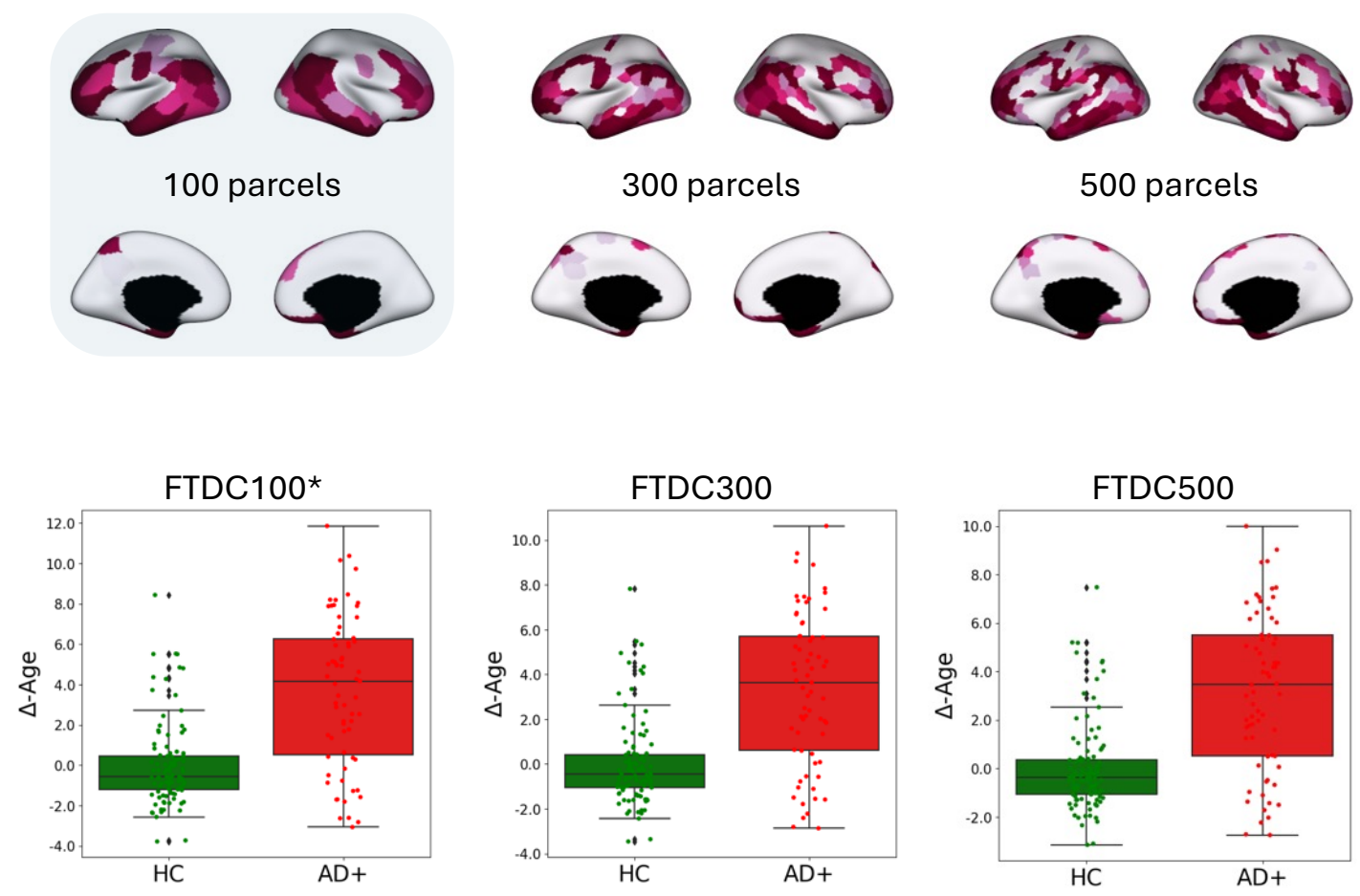
- VNN **distinctly** exploits eigenvectors in AD and HC groups



⇒ explains anatomical interpretability of brain age gap in AD

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