Tensor Core Computing: an example on Independent Component Analysis

May 2024

Zeyu Wang, Zoltan Juhasz

University of Pannonia
Veszprem, Hungary

Content outline

1. Tensor core

1.1 Abstract and hierarchy

1.2 Programming paradigm

2. Independent Component Analysis (ICA)

2.1 Algorithm flow

2.2 Parallel implementation

2.3 Variables analysis

3. Implementation strategy

3.1 Locating matrix multiplication

3.2 Numerical correctness

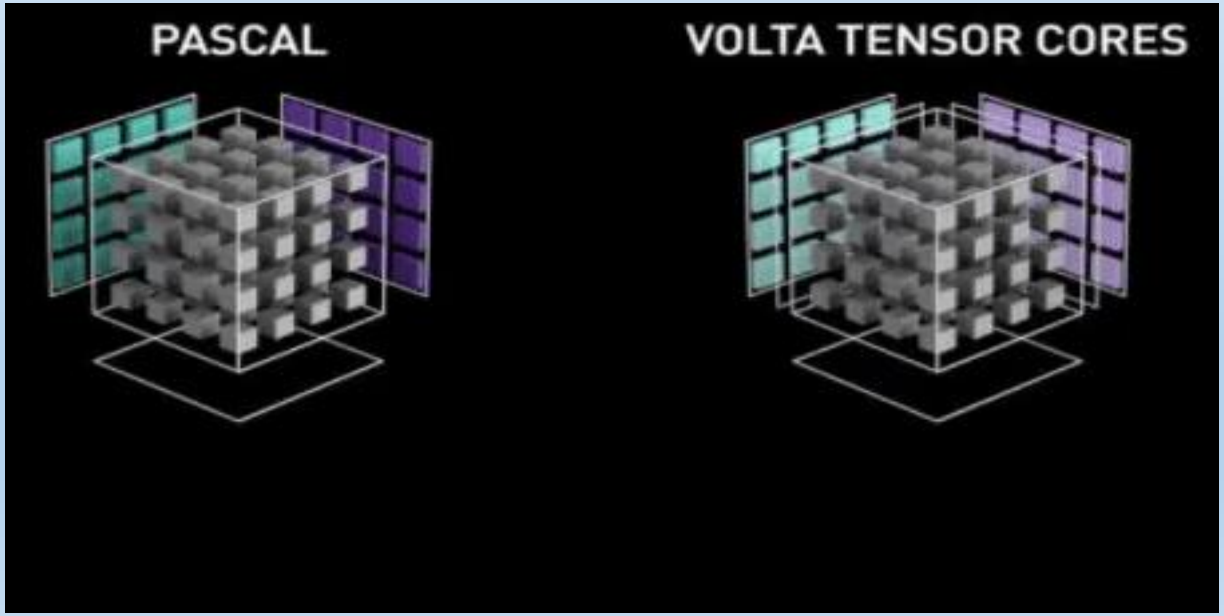
3.3 Performance preview

4. Future works

1.1 Abstract and hierarchy of the tensor core

The structure of GA100 SM

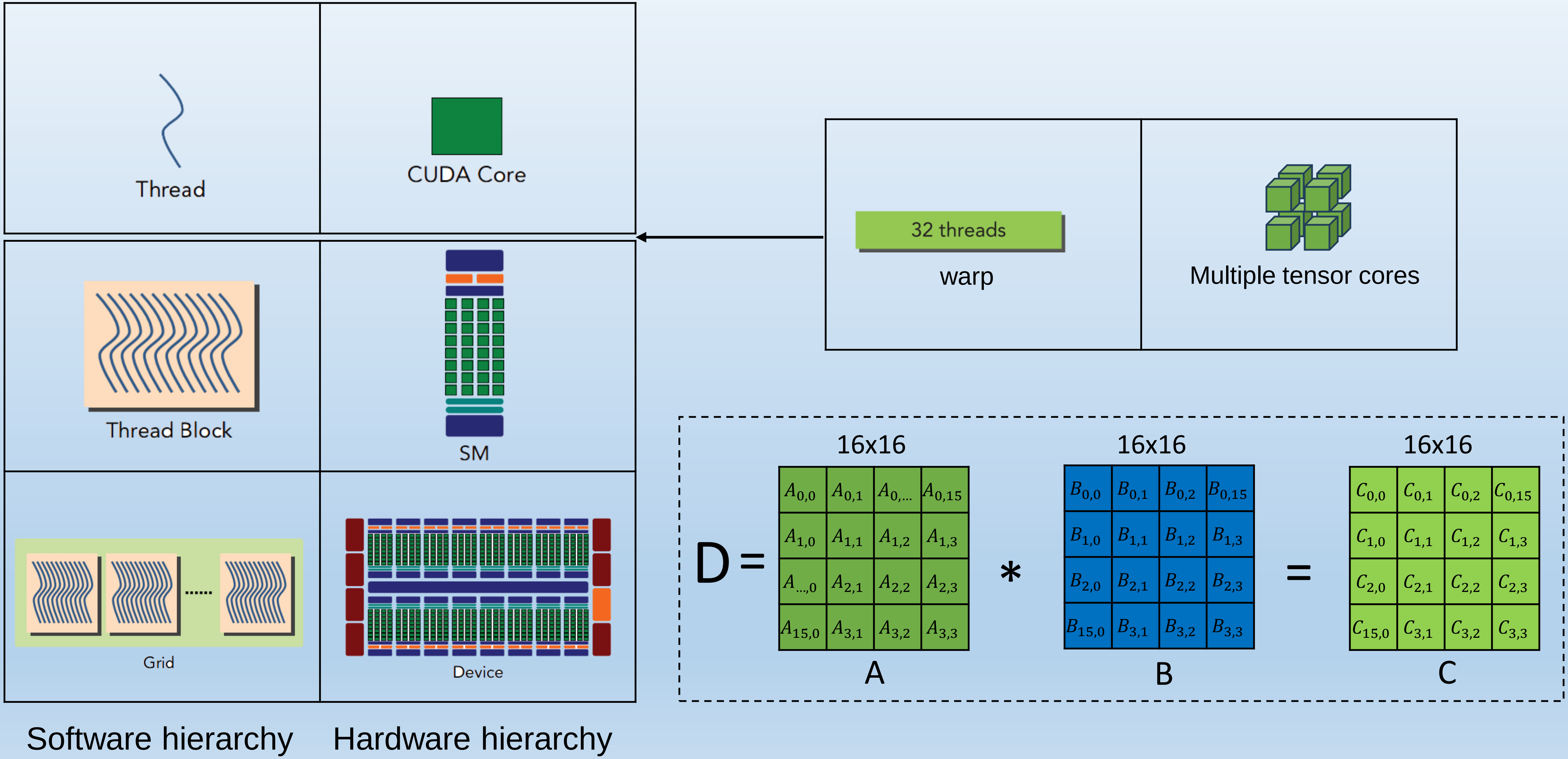
$$D = \underbrace{\begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix}}_{A}^{4\times4} * \underbrace{\begin{bmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix}}_{B}^{4\times4} = \underbrace{\begin{bmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{bmatrix}}_{C}^{4\times4}$$



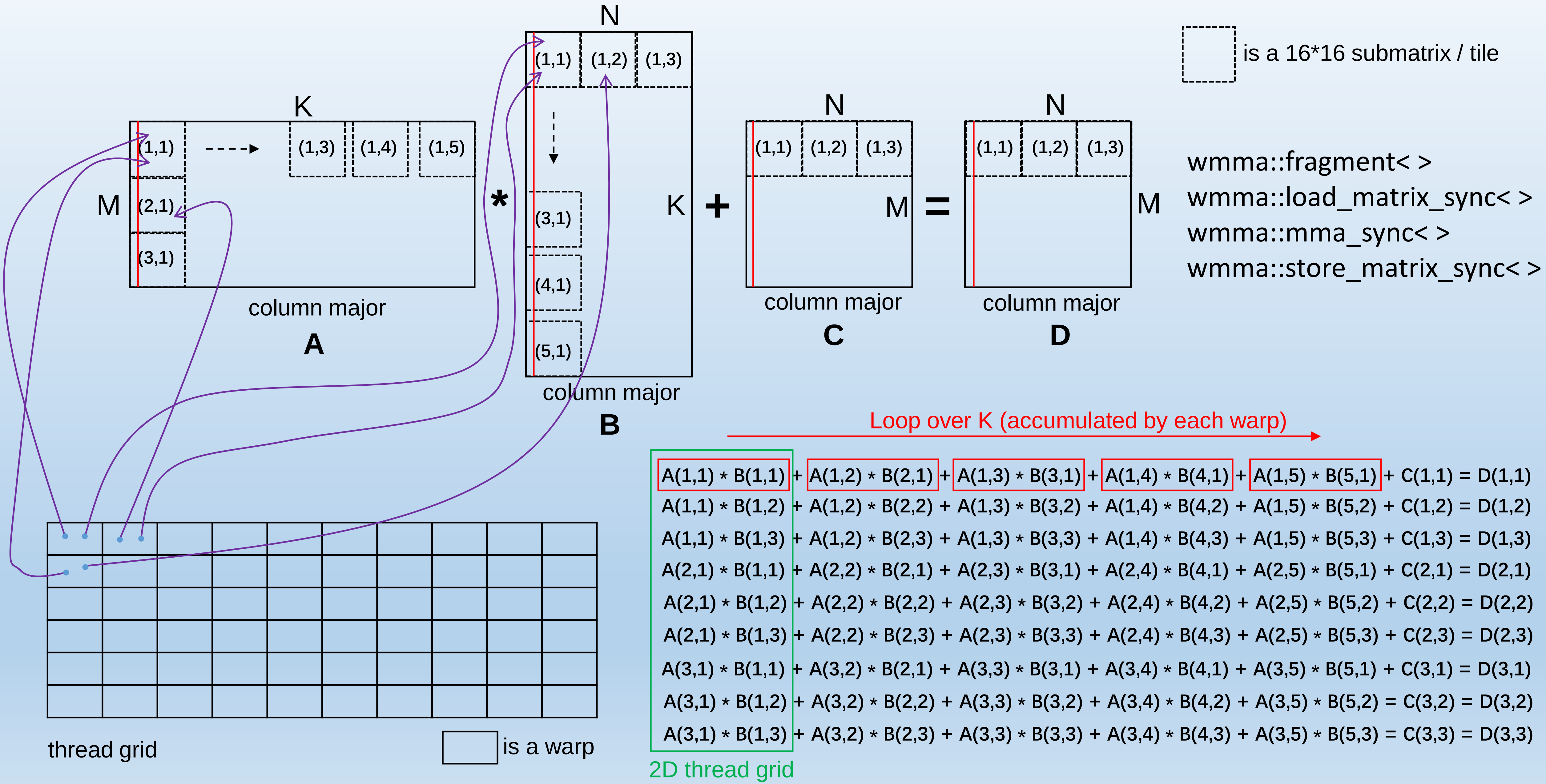
Some GPUs equipped with tensor cores

Model	Micro-architecture	Launch	CUDA cores	Memory size (GB)	Processing power (TFLOPS)		
					Tensor core FP32	Single precision	Double precision
V100	Volta	Jun 21, 2017	5120	16/32	112.2	14.0	7.0
T4	Turing	Sept 12, 2018	2560	16	64.8	8.1	0.2
A100	Ampere	May 14, 2020	6912	40/80	312.0	19.5	9.7
H100	Hopper	Mar 22, 2022	14592	80	756.4	51.2	25.6

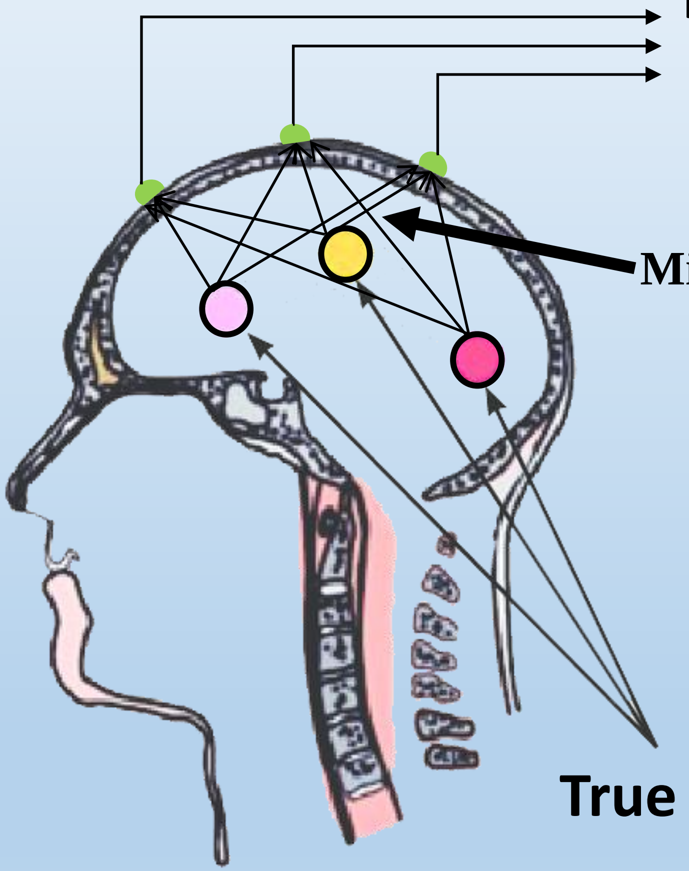
1.1 Abstract and hierarchy of the tensor core



1.2 Tensor core programming paradigm



2.1 Independent Component Analysis (ICA)



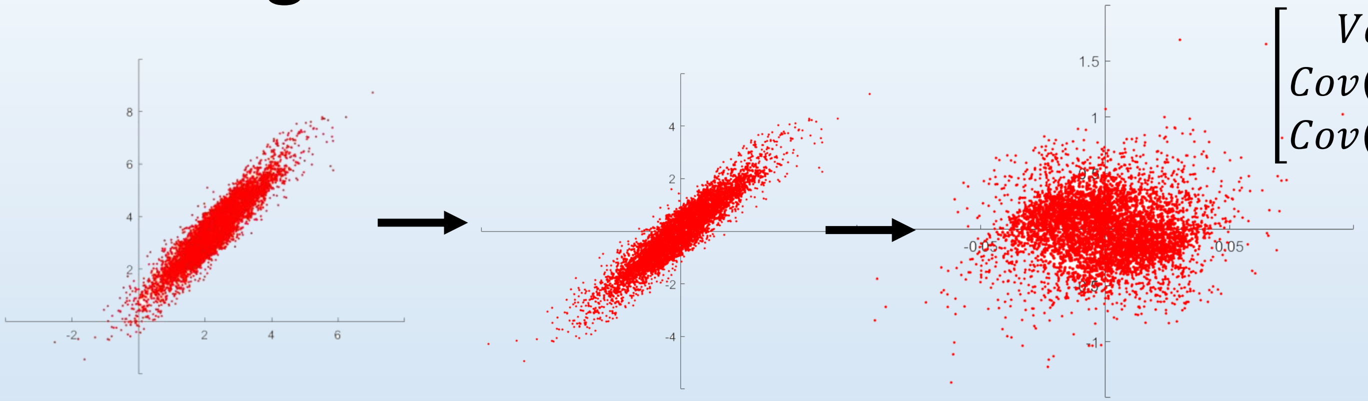
Unknown

$$\begin{bmatrix} x_{1(t)} \\ x_{2(t)} \\ x_{3(t)} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{32} \end{bmatrix} \cdot \begin{bmatrix} s_{1(t)} \\ s_{2(t)} \\ s_{3(t)} \end{bmatrix}$$

$$\boldsymbol{X} \qquad \boldsymbol{A} \qquad \boldsymbol{S}$$

$$\boldsymbol{X} = \boldsymbol{A} \cdot \boldsymbol{S} \Longrightarrow \begin{cases} \boldsymbol{S} = \boldsymbol{W} \cdot \boldsymbol{X} \\ \boldsymbol{W} = \boldsymbol{A}^{-1} \end{cases}$$

2.1 Algorithm flow



Raw data Centralized data Whitened data

$$Cov = XX^T$$

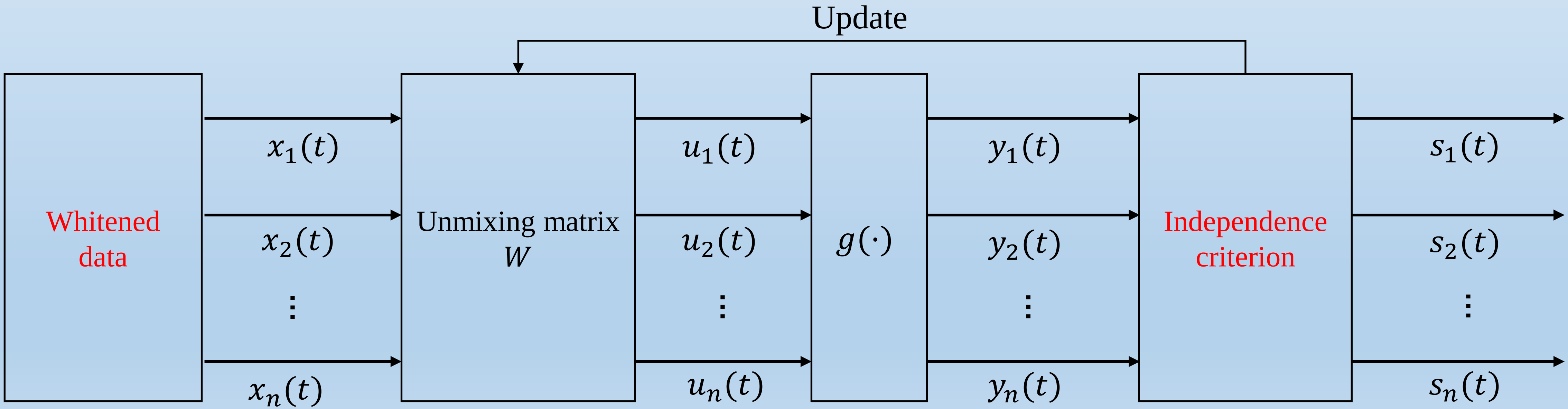
$$\begin{bmatrix} Var(CH1) & Cov(CH1,CH2) & Cov(CH1,CH3) \\ Cov(CH2,CH1) & Var(CH2) & Cov(CH2,CH3) \\ Cov(CH3,CH1) & Cov(CH3,CH2) & Var(CH3) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Singular Value Decomposition

$$X = U\Sigma V^T$$

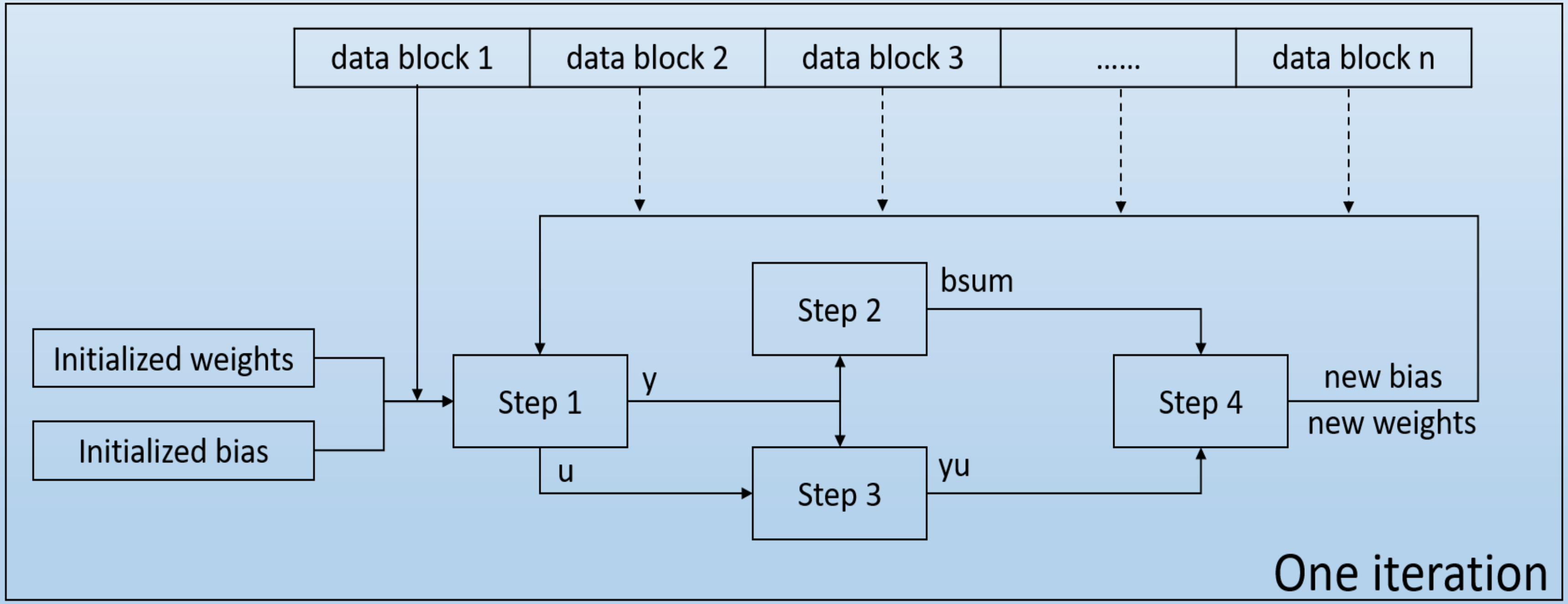
$$\Sigma = diag(\sigma_1, \sigma_2, \sigma_3, \dots)$$

$$X_{deco} = \sqrt{U^T X}$$



The process of solving the unmixing matrix

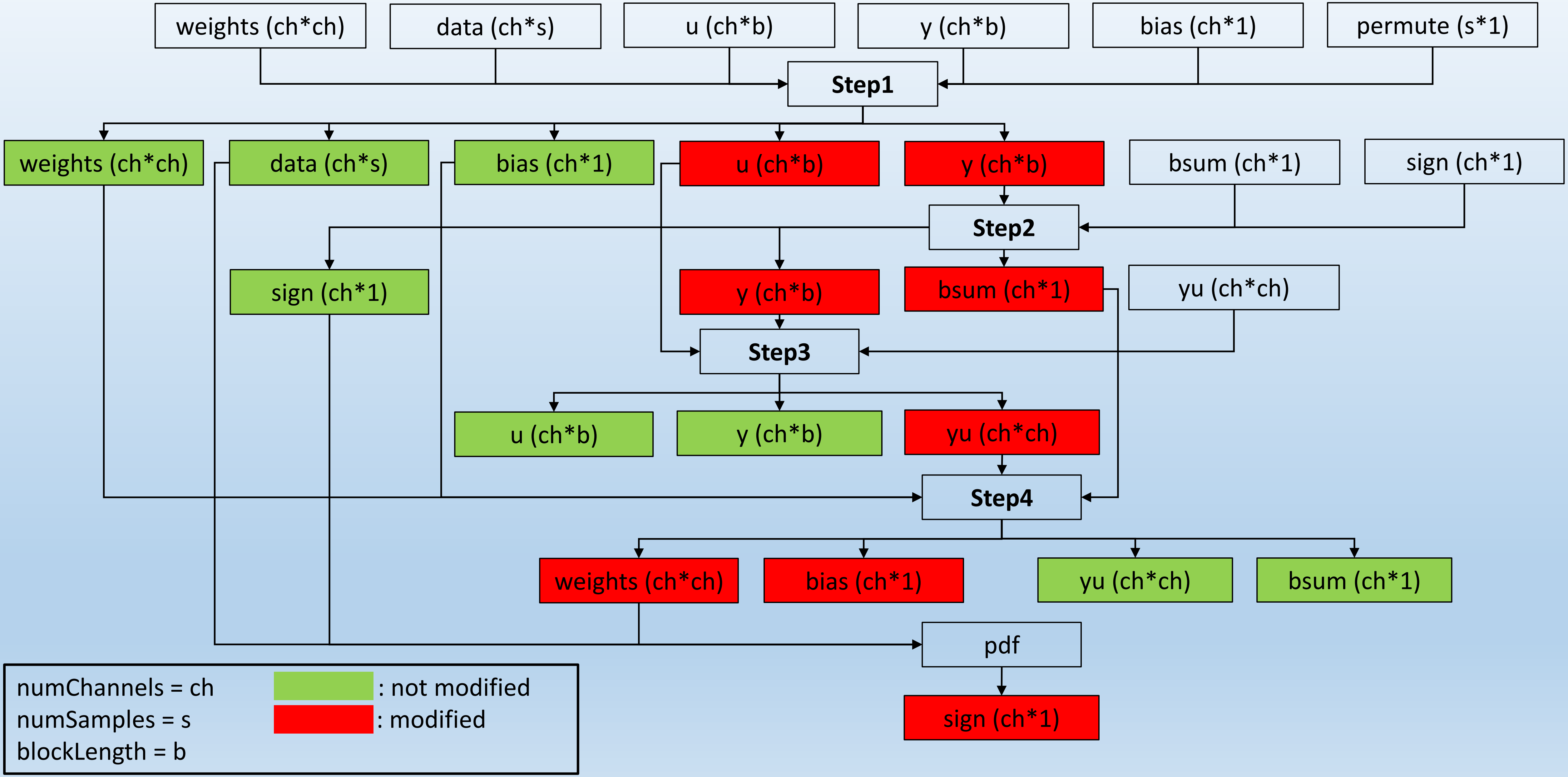
2.2 Parallel implementation



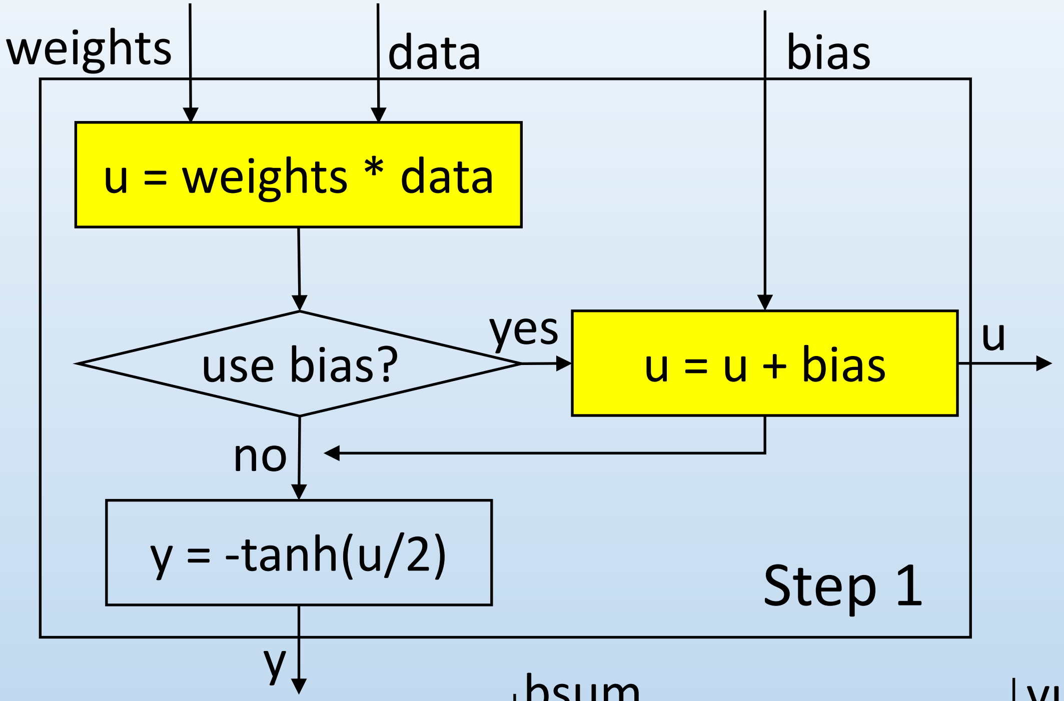
CUDA ICA

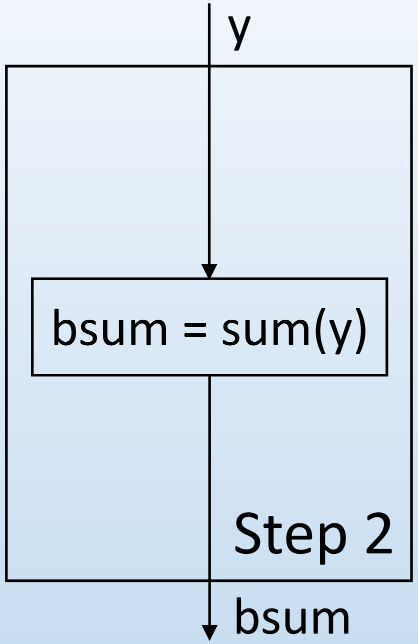
1. Initialize $W(0)$
2. **while** (not converged and t < limit)
    2.1 $\mathbf{Y}$ = permutate($\mathbf{X}$)
    2.2 **for** each block $\mathbf{B}$ in signal $\mathbf{Y}$
        2.2.1 $\mathbf{U} = \mathbf{WY} + bias$
        2.2.2 $\mathbf{Y} = tanh(\mathbf{U})$
        2.2.3 $\mathbf{W}^* =$
        $\mathbf{W} + \eta(\mathbf{I} - \mathbf{YU}^\mathbf{T} - \mathbf{UU}^\mathbf{T})\mathbf{W}$
        2.2.4 update bias
    end for
end while

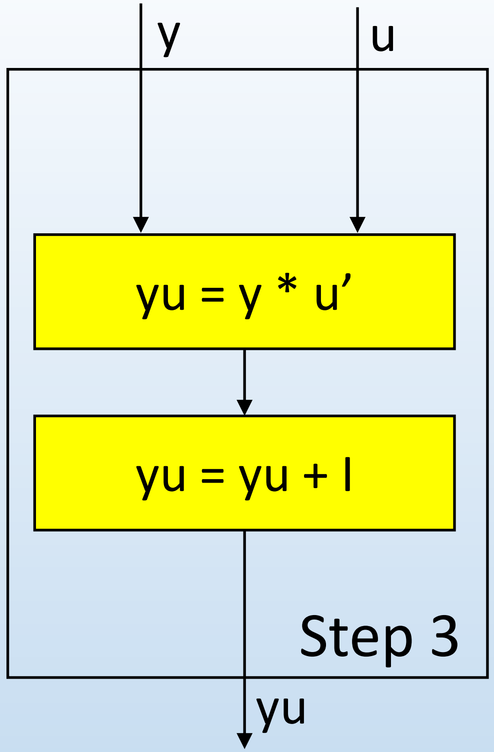
2.3 Variables analysis

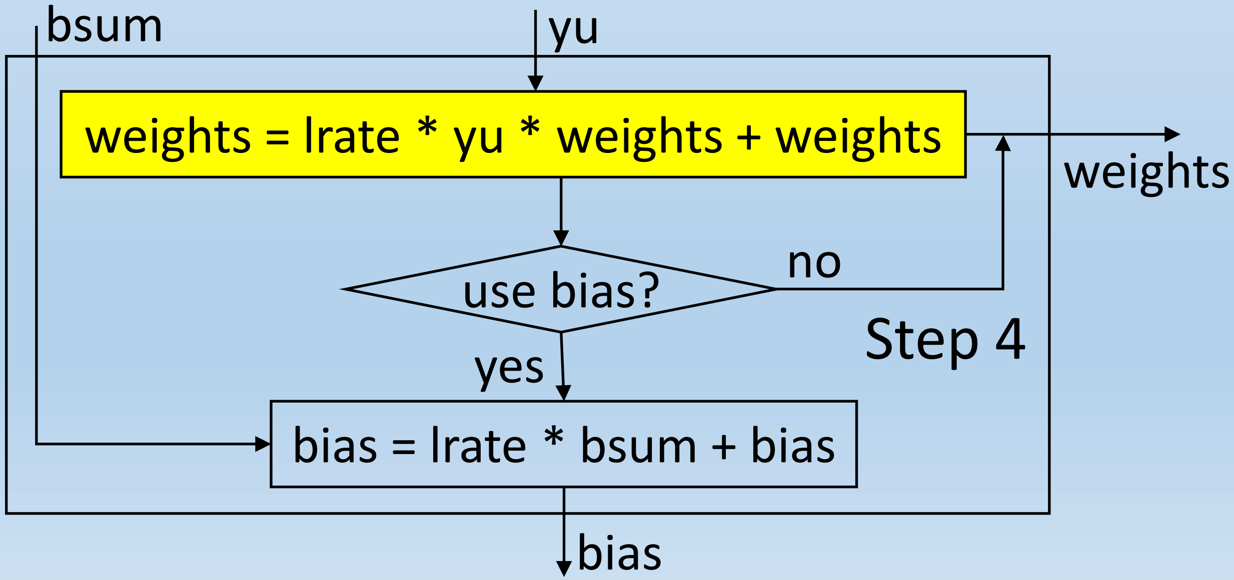


3.1 Locating matrix multiplication





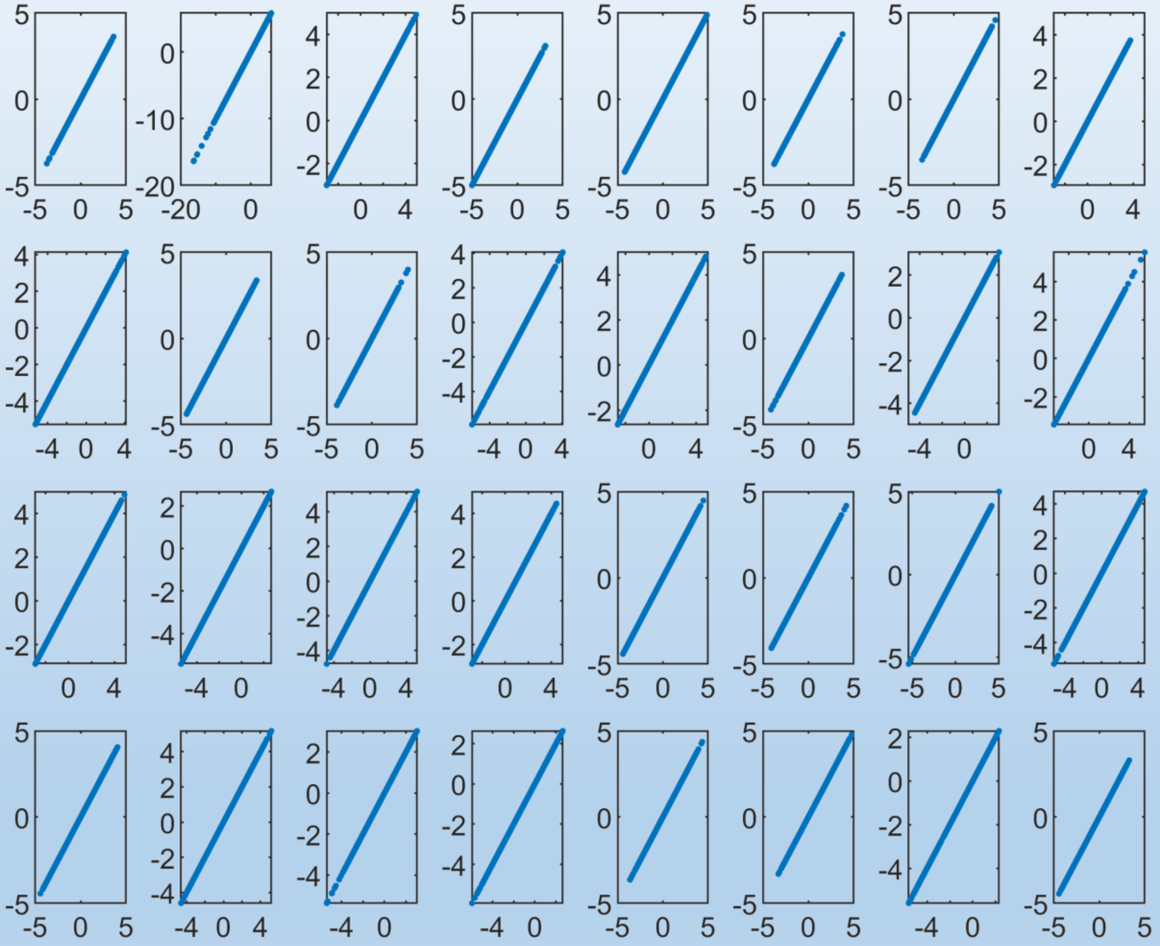




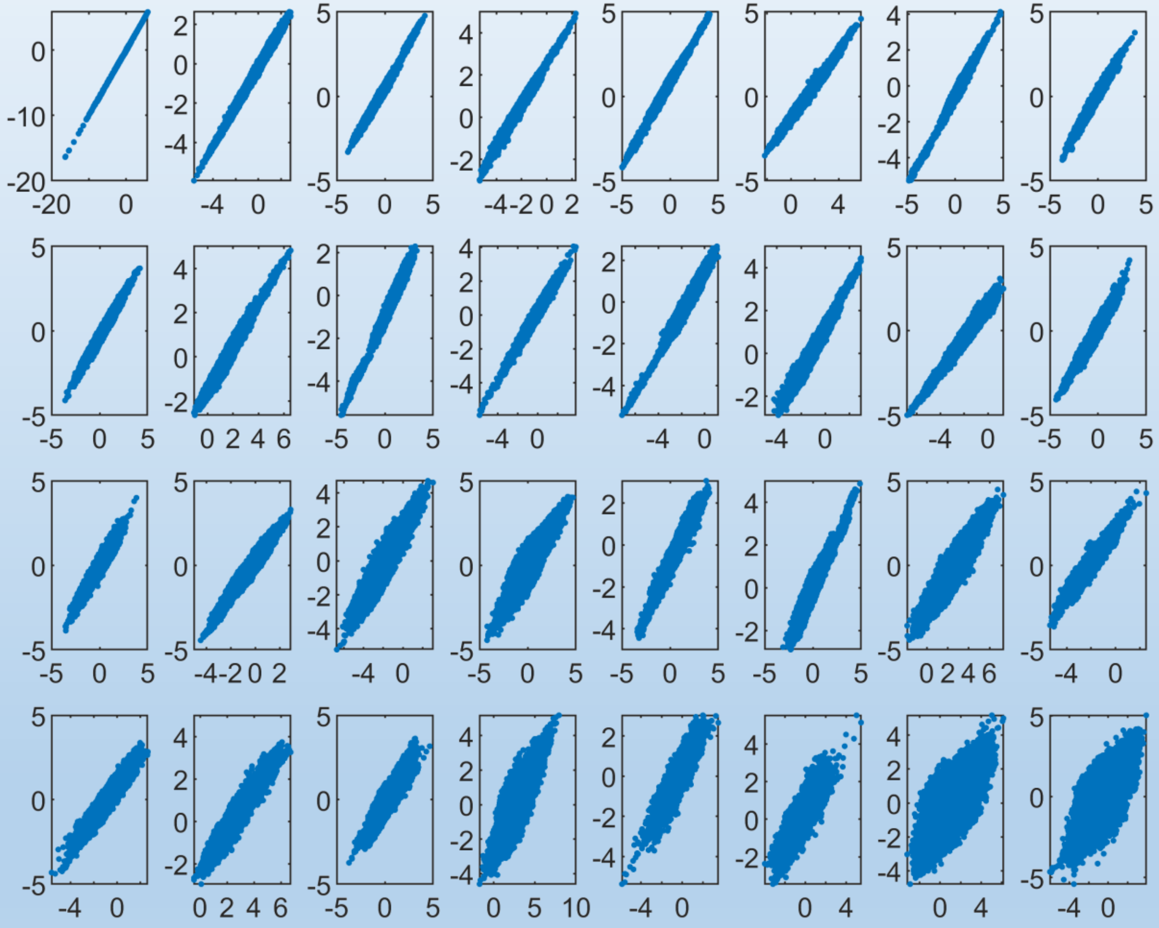
1. Initialize $W(0)$
2. **while** (not converged and $t <$ limit)
   2.1 $\mathbf{Y}$ = permutate($\mathbf{X}$)
   2.2 **for** each block $\mathbf{B}$ in signal $\mathbf{Y}$
      2.2.1 $\mathbf{U} = \mathbf{WY} + bias$
      2.2.2 $\mathbf{Y} = tanh(\mathbf{U})$
      2.2.3 $\mathbf{W}^* =$
      $\mathbf{W} + \eta(\mathbf{I} - \mathbf{YU}^\mathbf{T} - \mathbf{UU}^\mathbf{T})\mathbf{W}$
      2.2.4 update bias
   end for
end while

: FMA operation

3.2 Numerical correctness



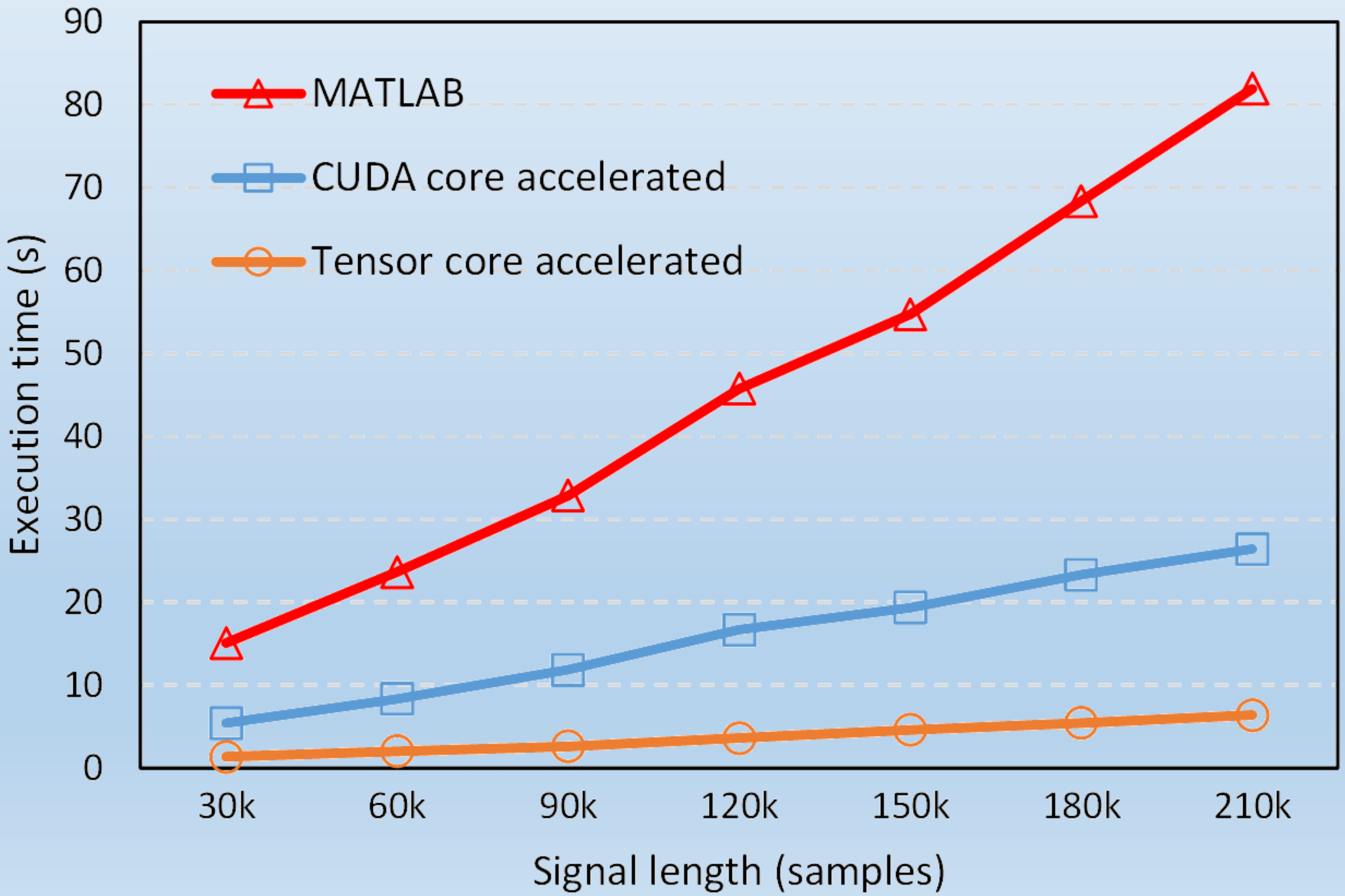
The diagonal lines produced by IC pairs from MATLAB-ICA and CUDAICA



The diagonal lines produced by IC pairs from tensor core ICA and CUDAICA

3.3 Performance preview

32-channel EEG data, up to 210k time points, CUDA core and tensor core execution time are produced by A100 GPU on Komondor supercomputer. The MATLAB execution time is produce by a 8-core CPU (i7-9700k)



3x speedup

4x speedup

4 Future works

- Numerical correctness issue
- Tensor core + shared memory strategy
- Further performance testing
  - New strategy and larger dataset
  - EEG processing pipeline integration
- Impact of mixed precision computing
  - FP32, TF32, FP16, BF16 ……
- **Other potential algorithms that tensor cores can optimize**
  - **Are there a lot of matrix multiplication operations in the algorithm? Why not consider using tensor cores to speed it up?**

This research is supported by
Új Nemzeti Kiválóság Program (UNKP)

Decorrelation

Samples

Channels

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \end{bmatrix}$$

Data matrix

$$Cov(X,Y) = \frac{1}{n}\sum_{i=n}^{n}(x_i - E(x))(y_i - E(Y))$$

Covariance

$$\begin{bmatrix} Var(CH1) & Cov(CH1,CH2) & Cov(CH1,CH3) \\ Cov(CH2,CH1) & Var(CH2) & Cov(CH2,CH3) \\ Cov(CH3,CH1) & Cov(CH3,CH2) & Var(CH3) \end{bmatrix}$$

Covariance matrix

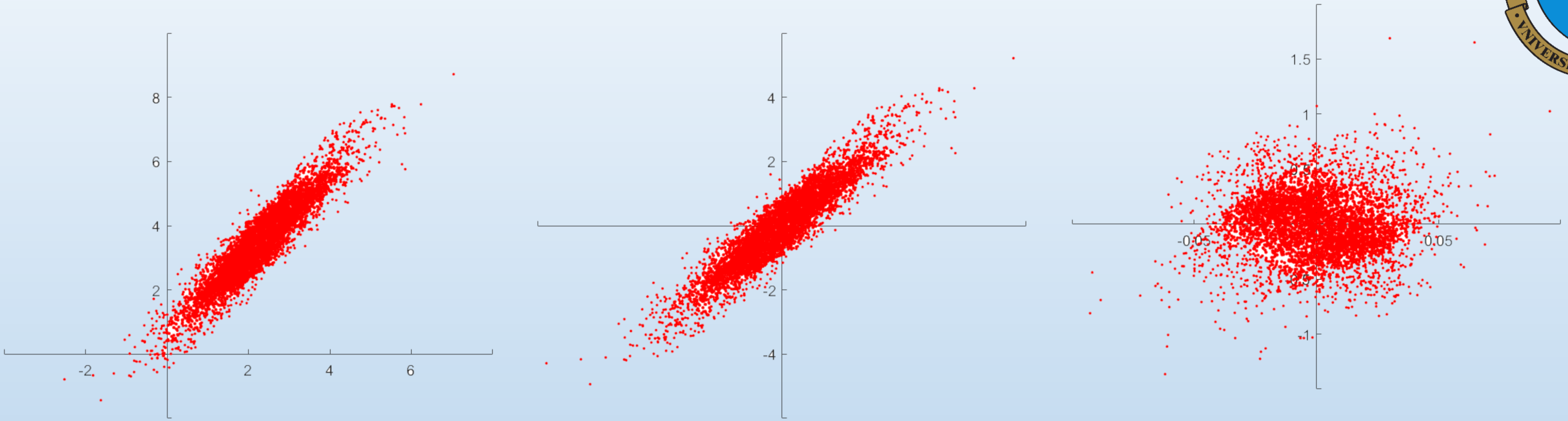
Diagonalization

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Diagonal matrix

$$Cov(X,Y)\begin{cases} > 0: & \textit{Positive correlation} \\ < 0: & \textit{Negative correlation} \\ = 0: & \textit{No correlation} \end{cases}$$

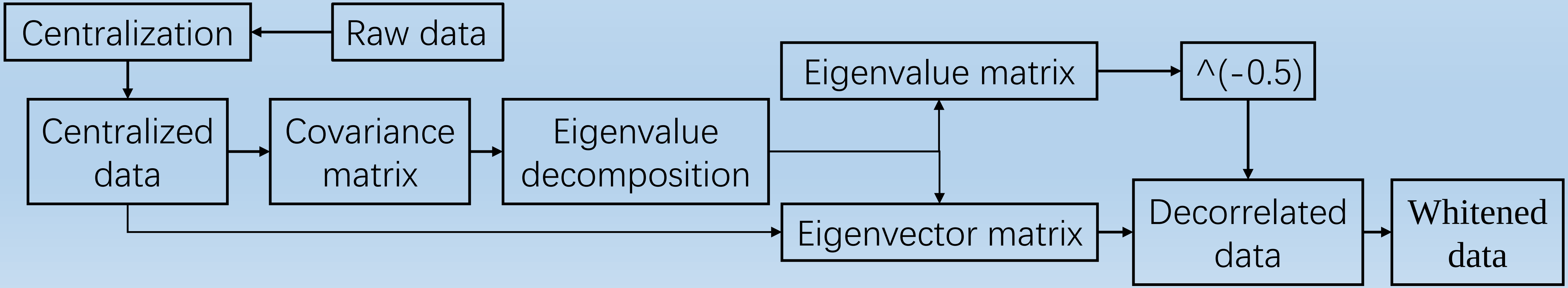
Decorrelation



Raw data

Centralized data

Whitened data



$$X = U\Sigma V^T$$

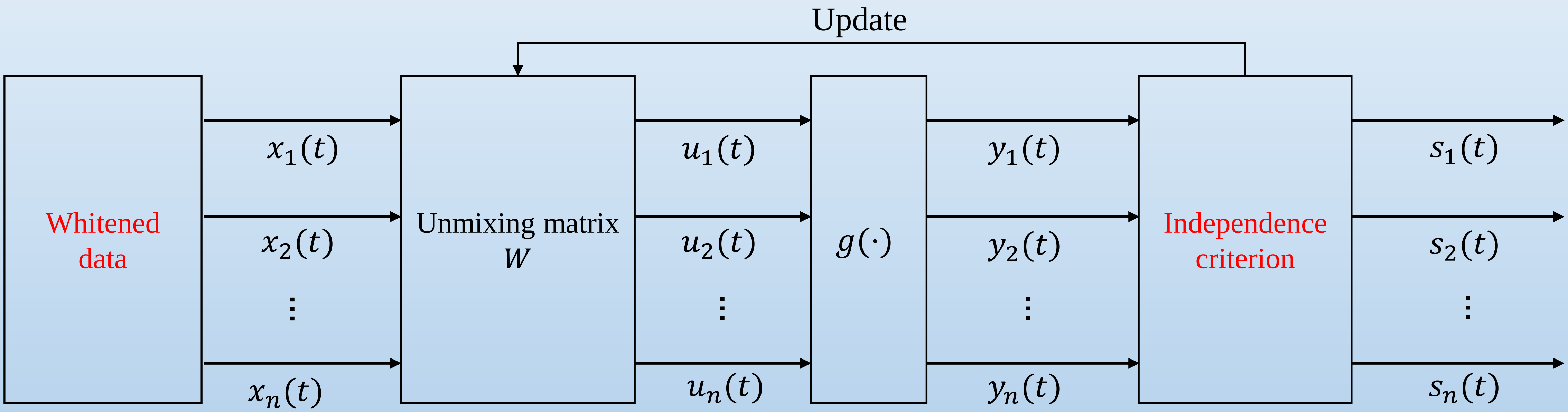
$$UU^T = I$$
$$V^T V = I$$

$$XX^T = U\Sigma V^T * (U\Sigma V^T)^T$$
$$XX^T = U\Sigma V^T * V\Sigma U^T$$
$$XX^T = U\Sigma U^T$$
$$XX^T U = U\Sigma^2$$
$$U^T XX^T U = U^T U\Sigma^2$$

Set $Y = U^T X$

$$YY^T = \Sigma^2$$

Remove dependency



The process of solving the unmixing matrix