Accelerating training of Physics Informed Neural Network for 1D PDEs with Hierarchical Matrices

30.01.2025 – AIRA seminar

Mateusz Dobija[1,3], Anna Paszyńska[1], Carlos Uriarte[4], Maciej Paszyński[2]

1) Jagiellonian University, Krakow, Poland

2) AGH University of Krakow, Poland

3) Doctoral School of Exact and Natural Sciences, Jagiellonian University

4) Basque Center for Applied Mathematics, Bilbao, Spain

Plan

- Introduction
- Matrix compression
- The compressed matrix-vector multiplication
- Using the algorithm of hierarchically compressed matrix-vector multiplication to speed up neural network training
- Results
- Conclusions

Introduction

- Physics Informed Neural Networks (PINN)
- Applications:
  - fluid mechanics
  - wave propagation
  - phase-filed modeling
  - Biomechanics
  - inverse problems

Introduction

- Motivation of work
  - Time needed for the training of neural network is crucial
- Idea for improvement
  - Speeding up the process of training the neural network
- Solution
  - Usage of hierarchical matrices

Introduction

- One-dimensional advection-diffusion problem

$u \in C2(0, 1)$

$$\underbrace{-\epsilon\frac{d^2u(x)}{dx^2}}_{\text{diffusion}=\epsilon} \underbrace{+\beta\frac{du(x)}{dx}}_{\text{advection “wind”}=1} = 0, x \in (0,1).$$

Boundary conditions:

$$-\epsilon\frac{du}{dx}(0) + u(0) = 1.0, \; u(1) = 0.$$

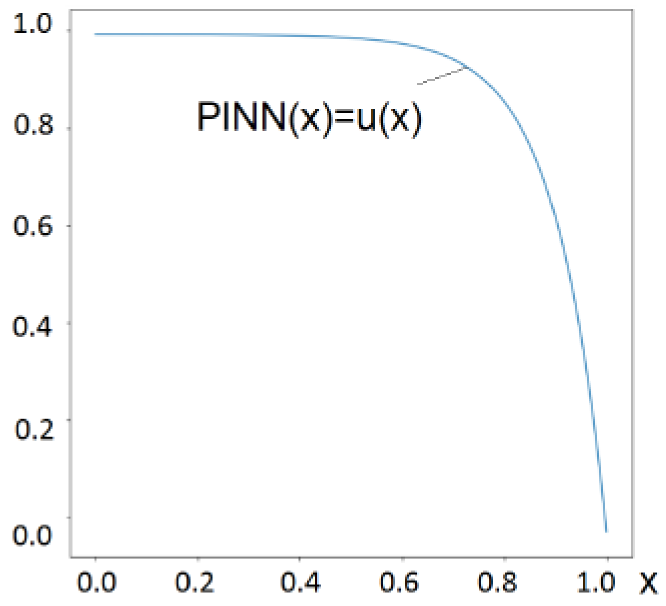


Fig. 1: PINN solution of the advection-diffusion problem for $\epsilon = 0.1$.

Introduction

Following the idea of PINN, we represent the solution as the neural network:

$$u(x) = PINN(x) = A_n\sigma\left(A_{n-1}\sigma(\ldots\sigma(A_1x + B_1)\ldots + B_{n-1}\right) + B_n \quad (3)$$

We define the loss function for the residual of the PDE

$$LOSS_{PDE}(x) = \left(-\epsilon\frac{d^2PINN(x)}{dx^2} + \beta\frac{dPINN(x)}{dx}\right)^2 \quad (4)$$

We also define the loss function for the left boundary condition

$$LOSS_{BC0} = \left(-\epsilon\frac{dPINN}{dx}(0) + PINN(0) - 1.0\right)^2, \quad (5)$$

and the loss function for the right boundary condition

$$LOSS_{BC1} = (PINN(1))^2, \quad (6)$$

The total loss function is defined by combining a weighted sum

$$LOSS = w_{PDE}\sum_{x\in(0,1)}(LOSS_{PDE}(x))^2 \quad (7)$$

$$+w_{BC0}\,(LOSS_{BC0}(0))^2 \quad (8)$$

$$+w_{BC1}\,(LOSS_{BC1}(1))^2. \quad (9)$$

Introduction

- Neural network with hierarchical matrices:
- $y = PINN(x) = Hn\sigma(Hn-1...H2\sigma(H1+b1) + b2) + ... + bn-1) + bn$,
- H_i - hierarchical matrices,
- b_i - vectors

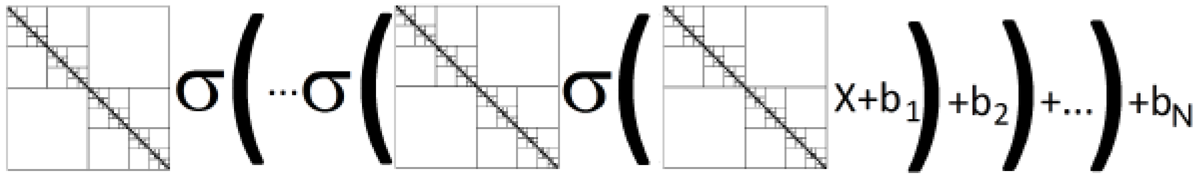
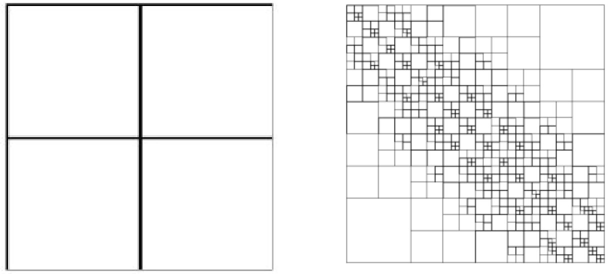
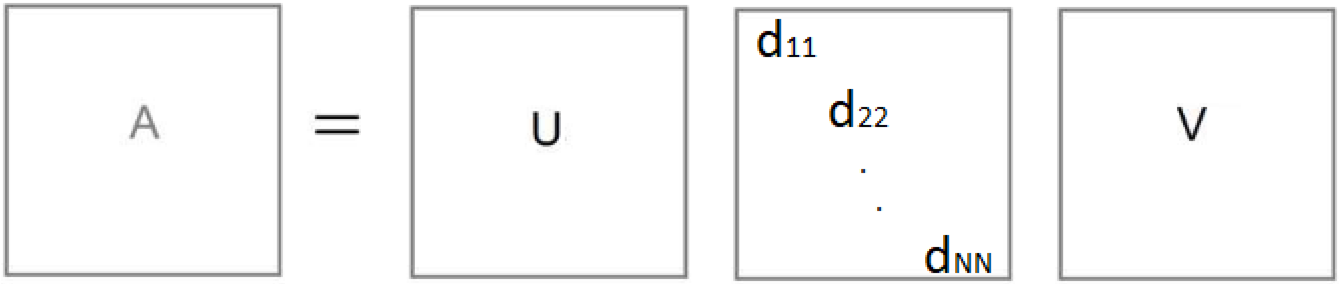


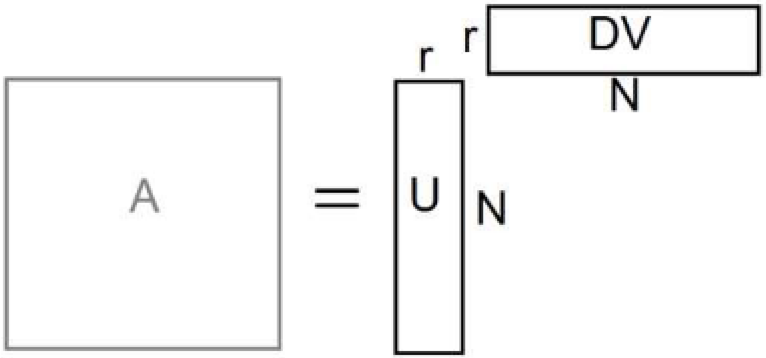
Fig. 2: Neural network with hierarchical matrices.

Matrix compression



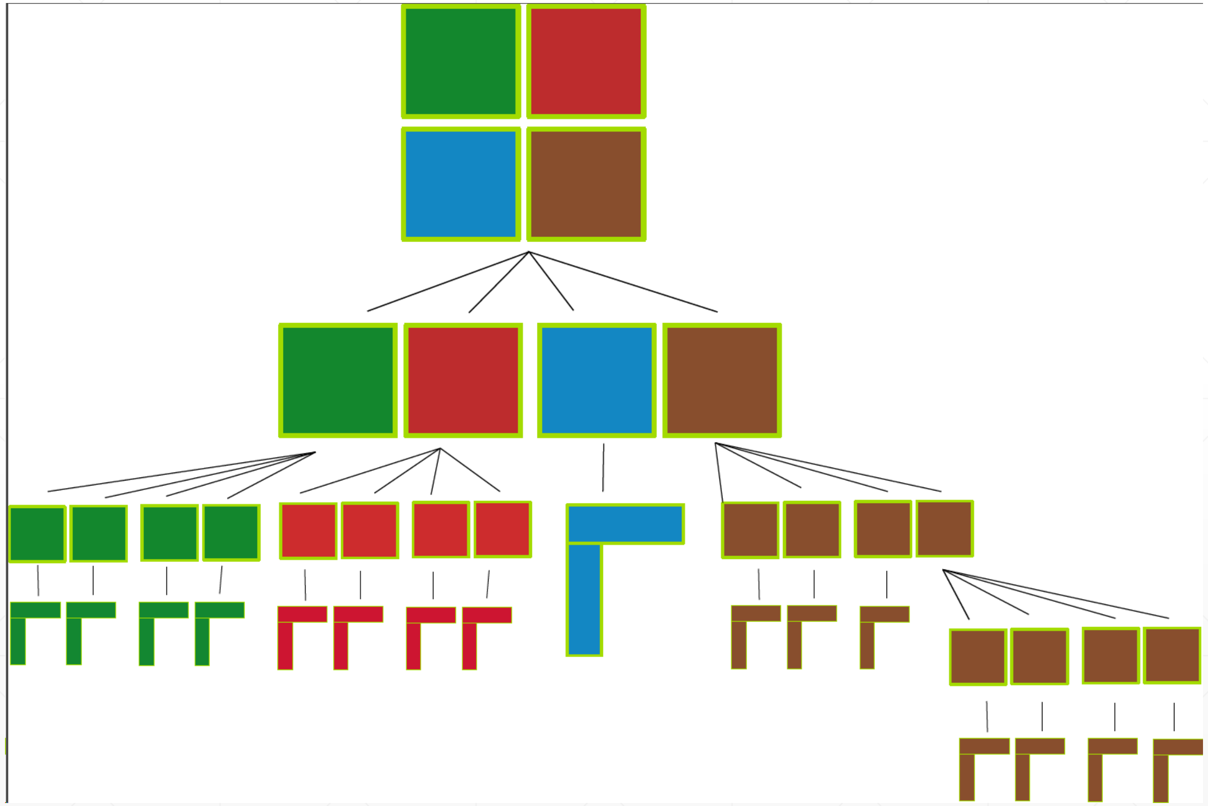
- Idea
  - Sparse matrix occurring in simulations contains low-rank blocks
  - Sparse matrix can be divided into smaller blocks – submatrices
  - For each block– submatrix:
  - run the SVD algorithm, which shows them as a product of
    - some number of rows, columns,
    - singular values
  - rows and columns related to small singular values can be zeroed.





$d_{1,1} \geq d_{2,2} \geq \ldots d_{r,r} \geq \mathrm{epsilon} > d_{r+1,r+1},\ldots,d_{N,N}$

Matrix compression



The compressed matrix-vector multiplication

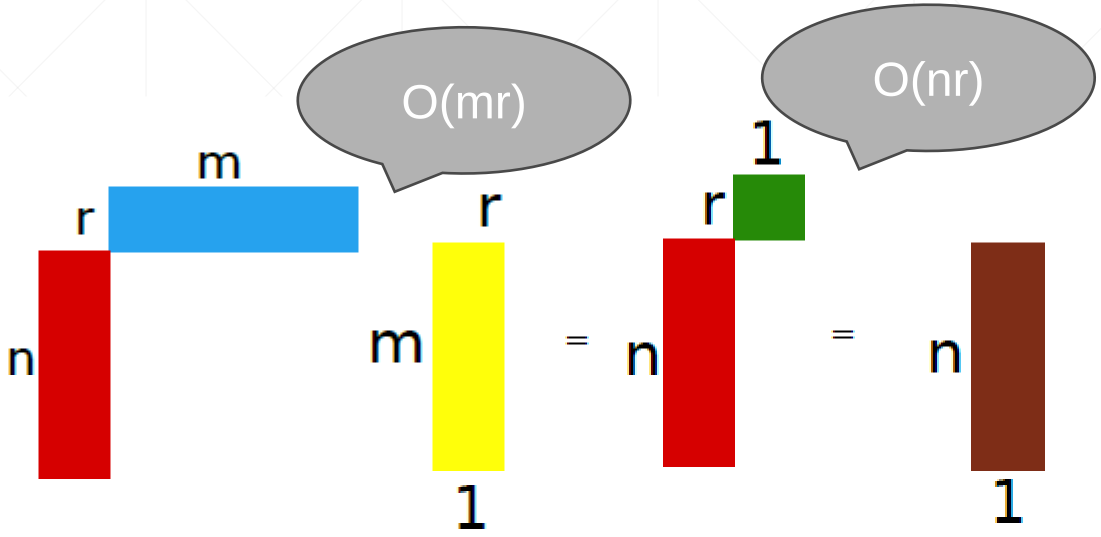


Fig. 4: The idea of SVD compressed matrix by vector multiplication.

The compressed matrix-vector multiplication

Algorithm 1 MultiplyMatrixByVector

Require: node T, vector to multiply v

    if $T.sons = \emptyset$ **then**

        return $T.U * (T.V * v)$;

    end if

    $numRows$ =number of rows of vector v;

    $v_1 = v(1 : floor(numRows/2), :)$ //first part of vector v

    $v2 = v(floor(numRows/2 + 1) : numRows, :)$ //second part of vector v

    res1=MultiplyMatrixByVector(T.children(1),v1)

    res2=MultiplyMatrixByVector(T.children(2),v2)

    res3=MultiplyMatrixByVector(T.children(3),v1)

    res4=MultiplyMatrixByVector(T.children(4),v2)

    //calculate the final result of multiplication

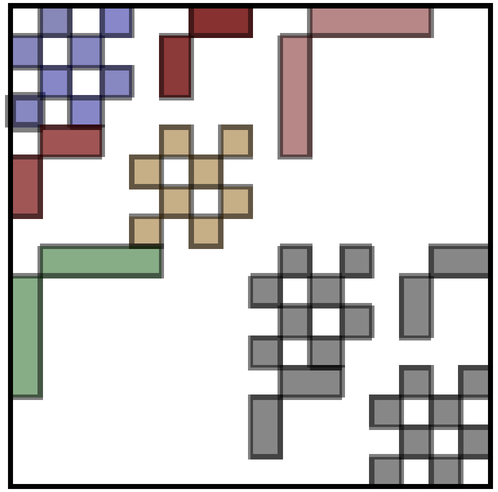
    res1res2=res1+res2

    res3res4=res3+res4

    return result=[res1res2;res3res4]

Using the algorithm of hierarchically compressed matrix-vector multiplication to speed up neural network training

- Assumption
  - The matrix of size n×n
  - Form of representation:
    - hierarchically compressed,
    - off-diagonal blocks on each level of hierarchy
      - represented by SVD compressed blocks
      - Rank = 1
    - remaining blocks
      - Divided smaller blocks



Using the algorithm of hierarchically compressed matrix-vector multiplication to speed up neural network training

Number of entries:
$2 * n * \log2(2 * n)$

Vector's size:
$2 * n * \log2(2 * n)$

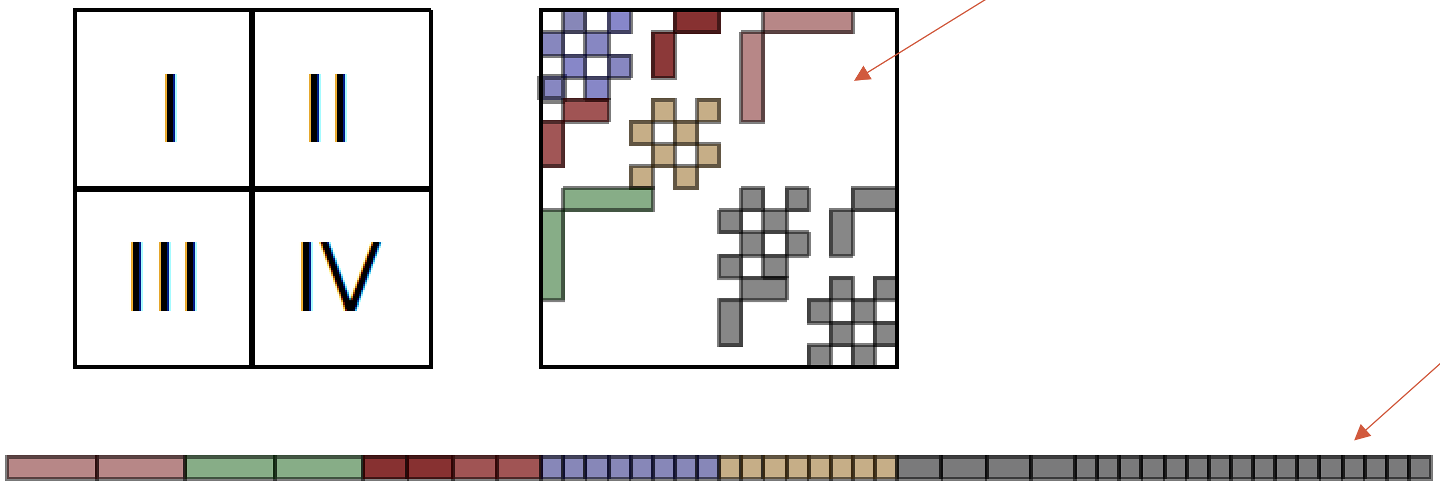


Fig. 5: Compressed matrix and its vector representation.

Using the algorithm of hierarchically compressed matrix-vector multiplication to speed up neural network training

- acceptable solutions:
  - LOSS of the order of 0.001.
- Test's settings:
  - No. of internal neural network layers:
    - 2
  - Matrices' size n (nxn):
    - 32, 64, 128, 256, 512.
  - Learning rate:
    - 0,02
  - Number of epochs:
    - 1000

Results

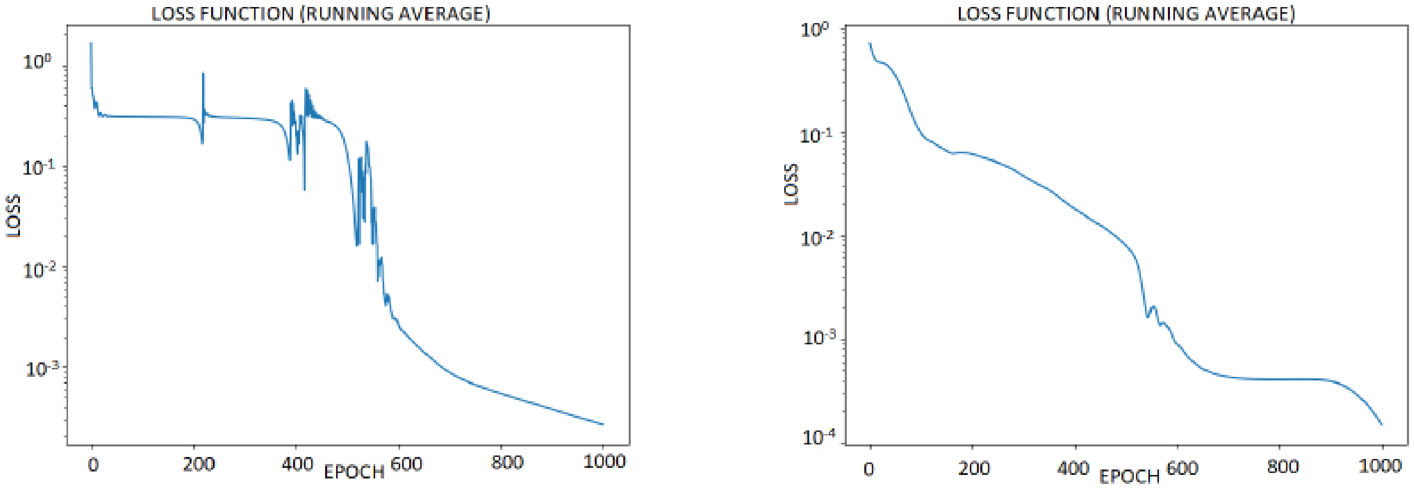
Table 1: Number of epochs and number of FLOPs of classic multiplication, learning rate 0.02, LOSS 0.001

matrix size	number of epochs - classic multiplication	number of FLOPs - classic multiplication
32	620	257,761,280
64	252	419,069,952
128	246	1,629,716,480
256	-	-
512	-	-

Table 2: Number of epochs and number of FLOPs of hierarchical multiplication, learning rate 0.02, LOSS 0.001

matrix size	number of epochs hierarchical multiplication	number of FLOOPs hierarchical multiplication
32	539	77,172,480
64	658	222,604,928
128	427	333,634,560
256	836	1,478,905,344
512	382	1,512,684,544

Results



(a) (b)

Fig. 6: Left panel: convergence of training of the fully connected neural network with 2 layers, 32 neurons per layer. Right panel: convergence of training of the fully connected neural network with 2 layers, 32 neurons per layer using compressed matrix.

Results

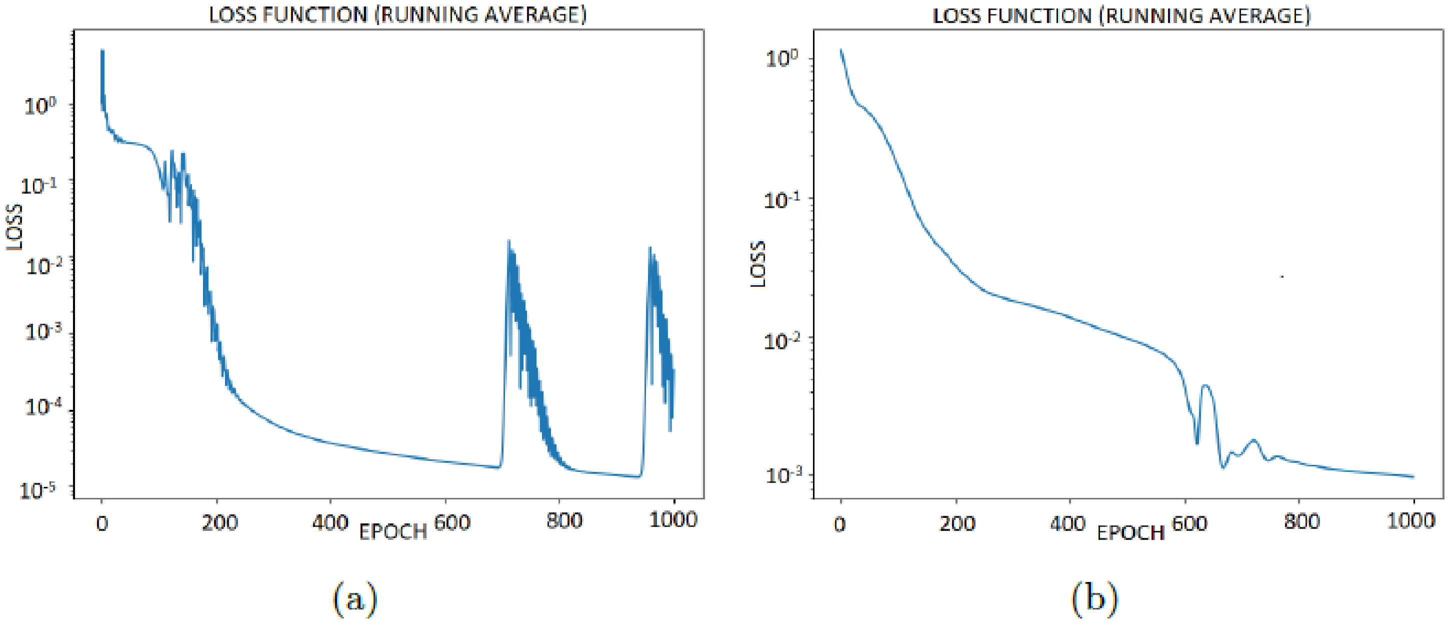


Fig. 7: Left panel: convergence of training of the fully connected neural network with 2 layers, 64 neurons per layer. Right panel: convergence of training of the fully connected neural network with 2 layers, 64 neurons per layer using compressed matrix.

Results

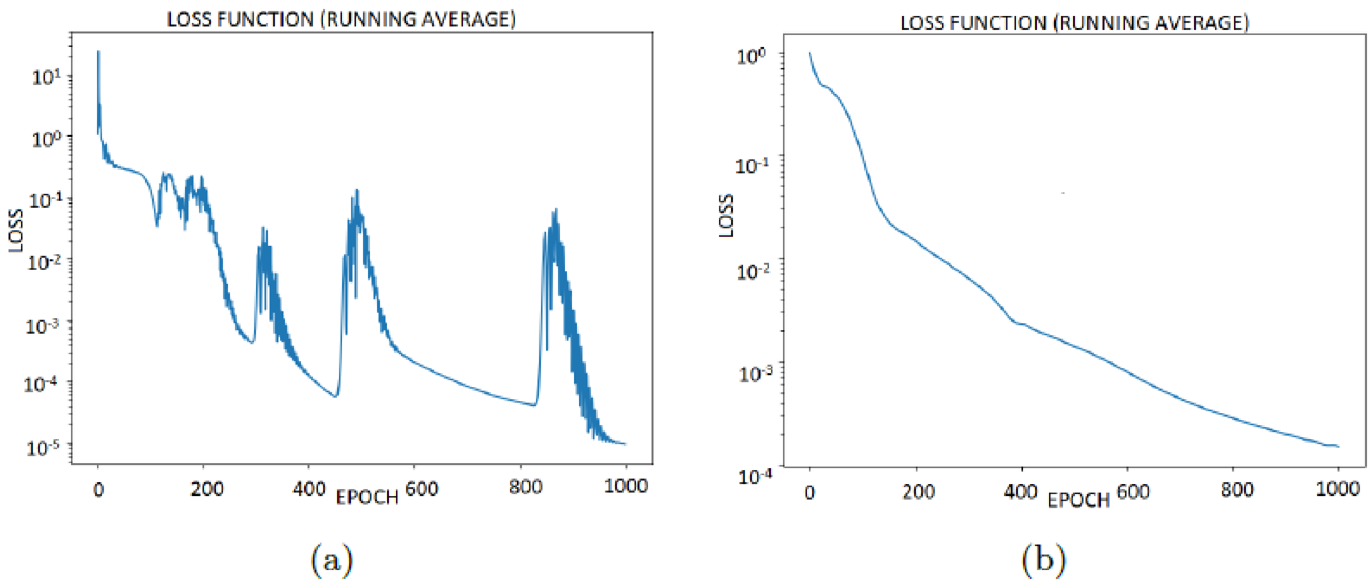
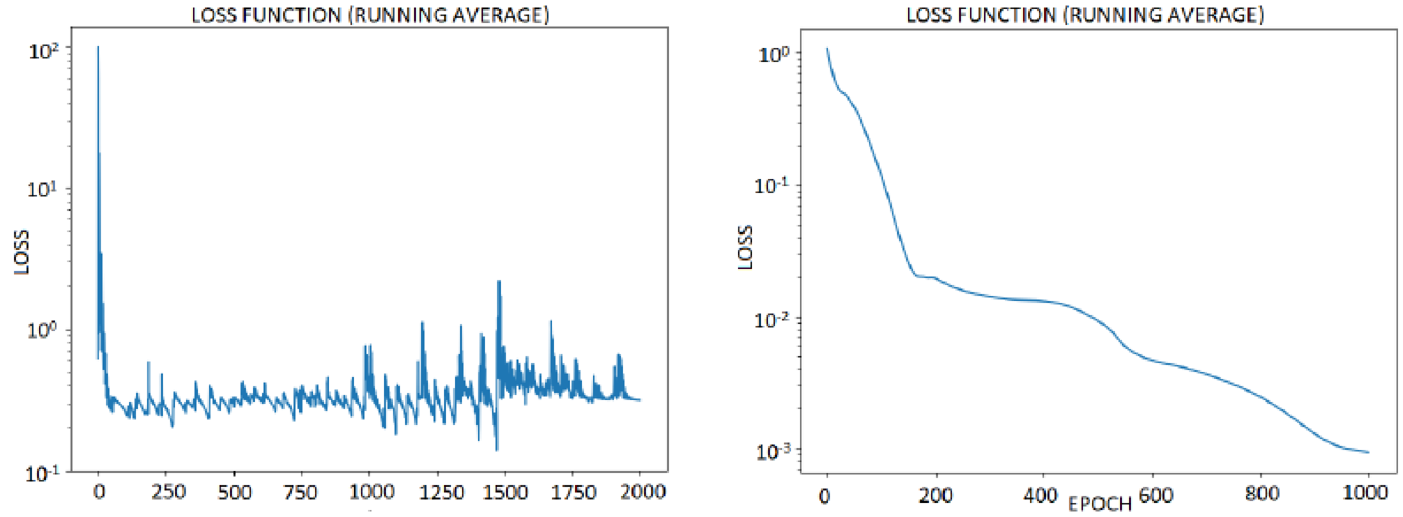


Fig. 8: Left panel: convergence of training of the fully connected neural network with 2 layers, 128 neurons per layer. Right panel: convergence of training of the fully connected neural network with 2 layers, 128 neurons per layer using compressed matrix.

Results



(a) (b)

Fig. 9: Left panel: convergence of training of the fully connected neural network with 2 layers, 256 neurons per layer. Right panel: convergence of training of the fully connected neural network with 2 layers, 256 neurons per layer using compressed matrix.

Conclusions

- Process of training of neural network – speeded up
- Numbers:
  - speed-up rate:
    - 2 - 5 times
  - memory storage reduction rate:
    - 3 - 20 times

Thanks for your attention

Q&A