NEMESIS

New generation
methods for numerical
simulations

# MACHINE LEARNING - ENHANCED POLYTOPAL FINITE ELEMENT METHODS

**Paola F. Antonietti**

**MOX-Dipartimento di Matematica**
**Politecnico di Milano**

NEMESIS KickOff Workshop - Montpellier - June 2024

MODELLISTICA E CALCOLO SCIENTIFICO
MOX
MODELING AND SCIENTIFIC COMPUTING

POLITECNICO
MILANO 1863
DIPARTIMENTO DI MATEMATICA

lymph

Funded by
the European Union

erc
European Research Council
Established by the European Commission

# Structure of RC 2: efficiency boost

Competitive numerical methods for challenging multiphysics/multiscale PDE models must deliver the desired accuracy at a manageable computational cost.

The great flexibility provided by polytopal methods paves the way for innovative (possibly ML-enanched) acceleration strategies:

- Efficient agglomeration-based multilevel strategies
- Automatic adaptive strategies
- Fast assembly strategies
- Efficient mesh quality indicators

# A (far to be complete) polytopal literature……

- Efficient (agglomeration-based) multilevel strategies:

  [Antonietti, Bassi, Berrone, Busetto, Bertoluzza, Botti, Colombo, Di Pietro, Houston, Matalon, Mascotto, Pennacchio, Pennesi, Prada, Rude, Scacchi, Suli, Verani, Zong,….]

- Adaptive strategies

  [Antonietti, Beirao da Veiga, Berrone, BorioCanuto, D'Auria, Dassi, Manuzzi, Manzini, Mascotto, Nochetto, Russo, Vacca, Verani, ….]

- Fast assembly strategies

  [Antonietti, Di Pietro, Houston, Mousavi, Pennesi, Prada, Sukumar, ….]

- Efficient mesh quality indicators

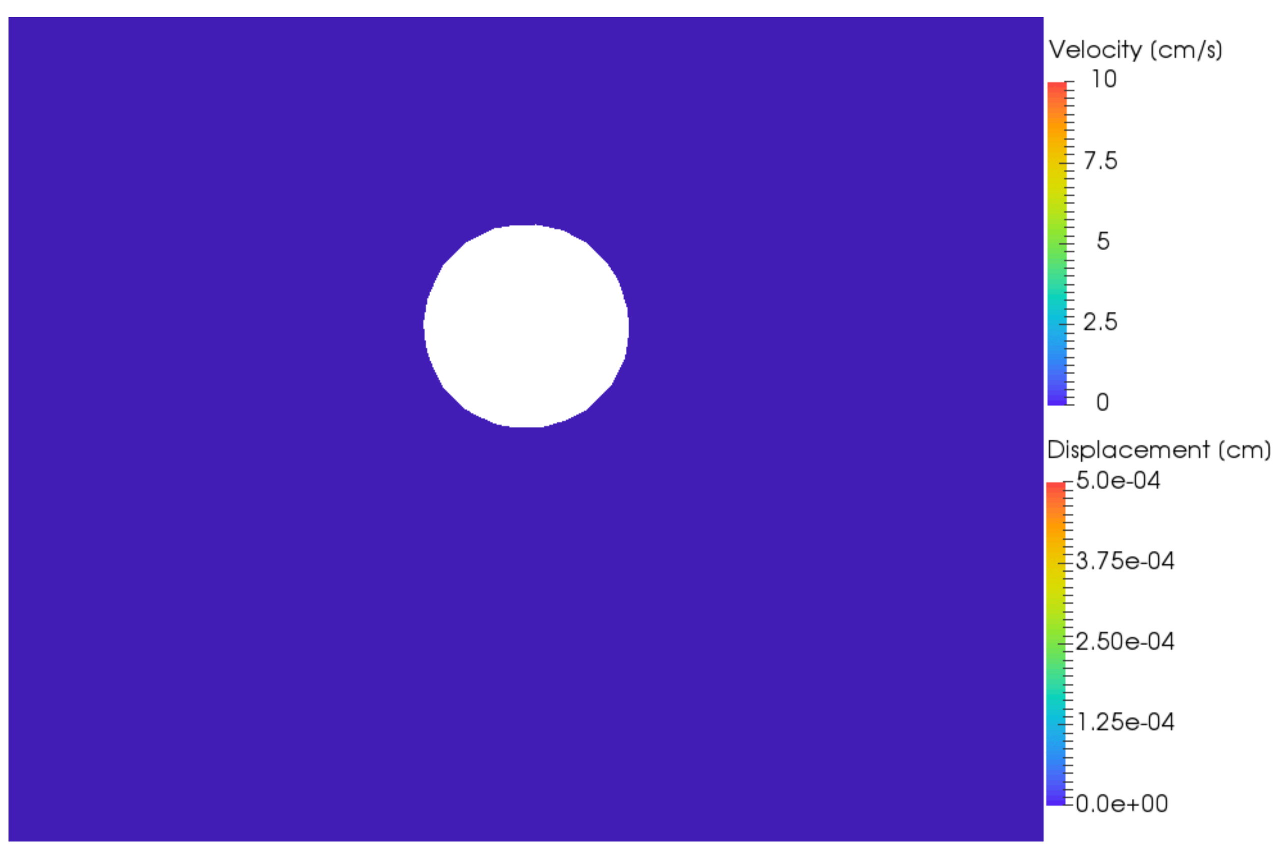
  [Berrone, Bertoluzza, Cabiddu, Manzini, Patane, Spagnuolo, ….]

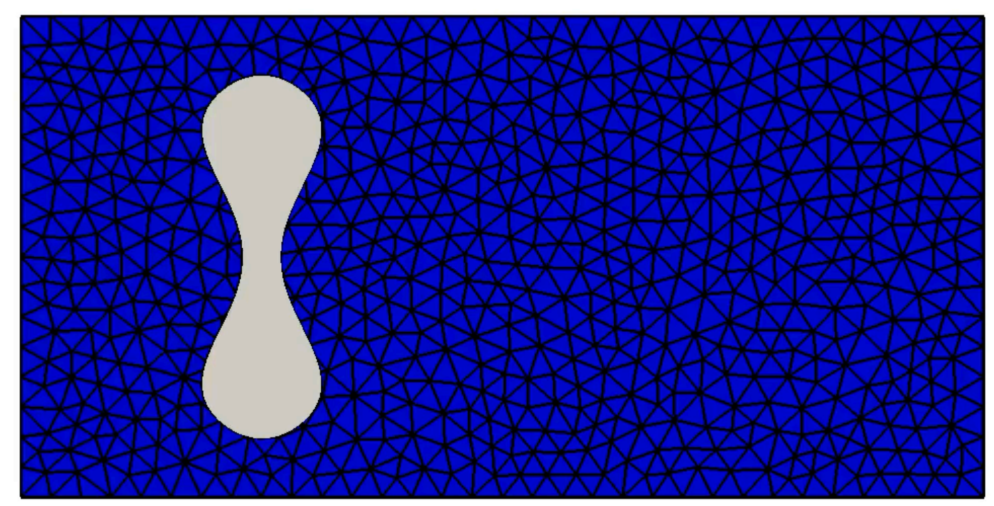
- Improve efficiency through ML-enhancement

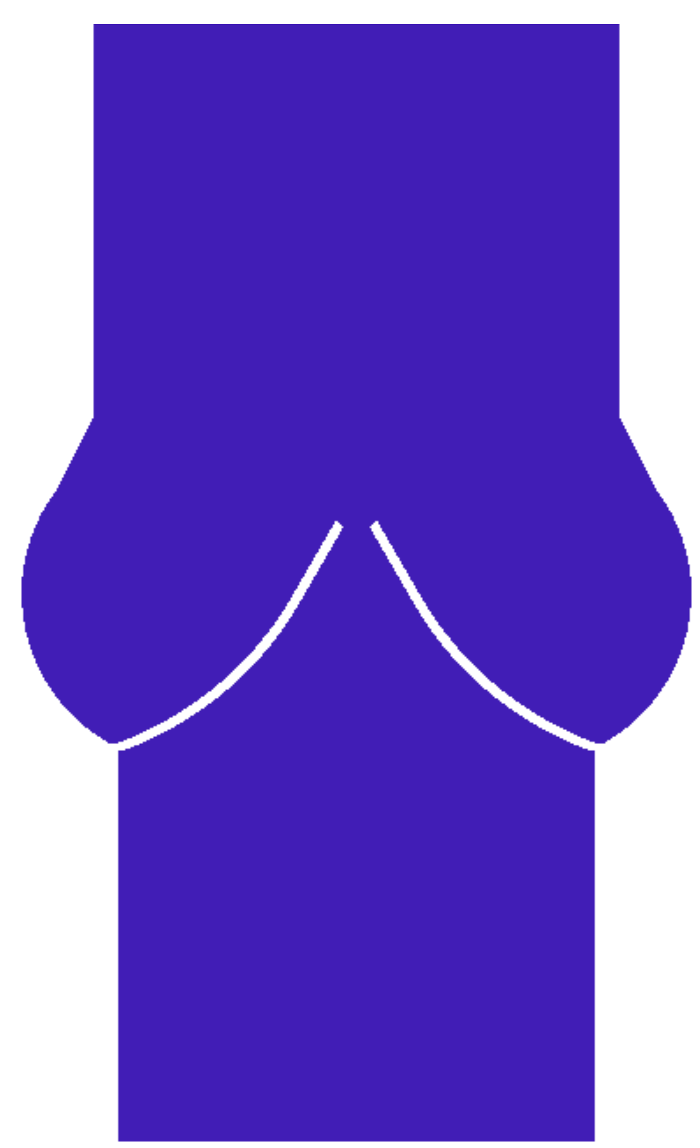
  [Antonietti, Dassi, Caldana, Manuzzi,….]

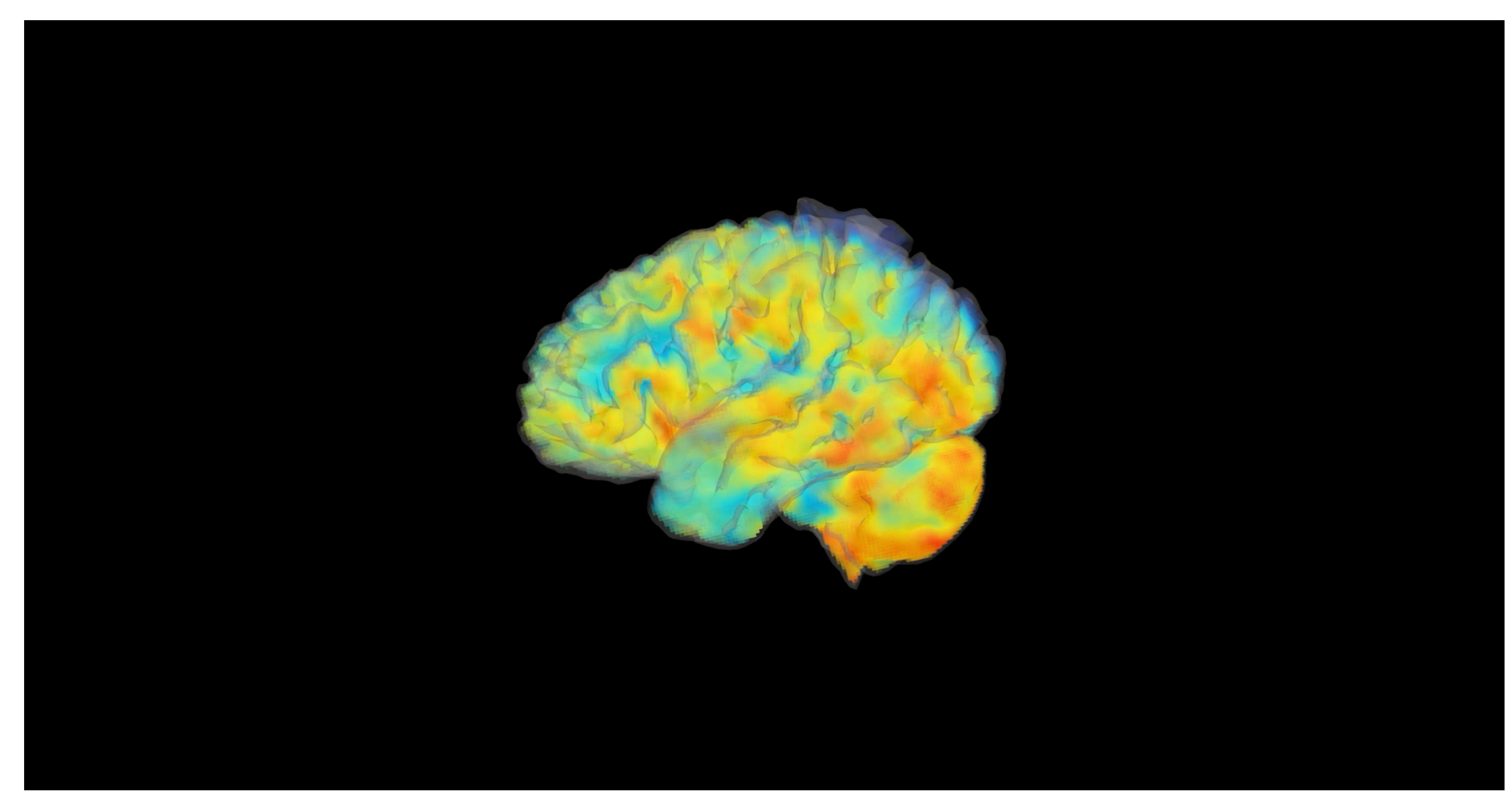
# Motivations

Many engineering and applied science applications are posed on complex (possibly moving) physical domains (e.g., fluid-structure interaction, crack propagation, flow in fractured porous media).



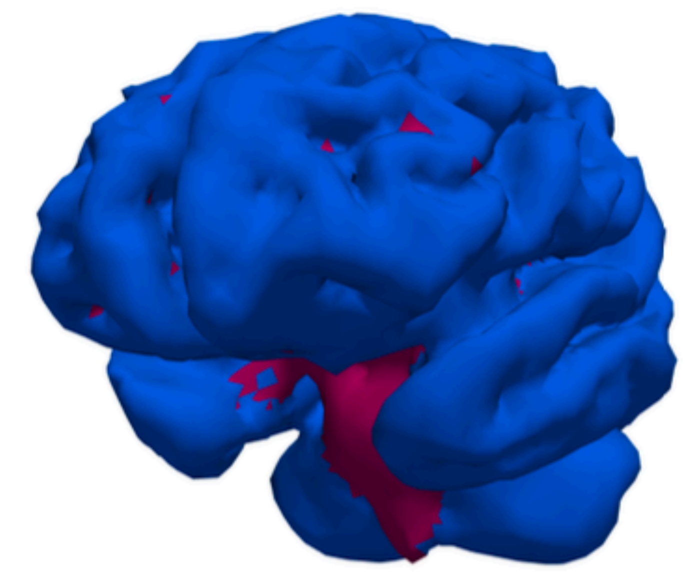



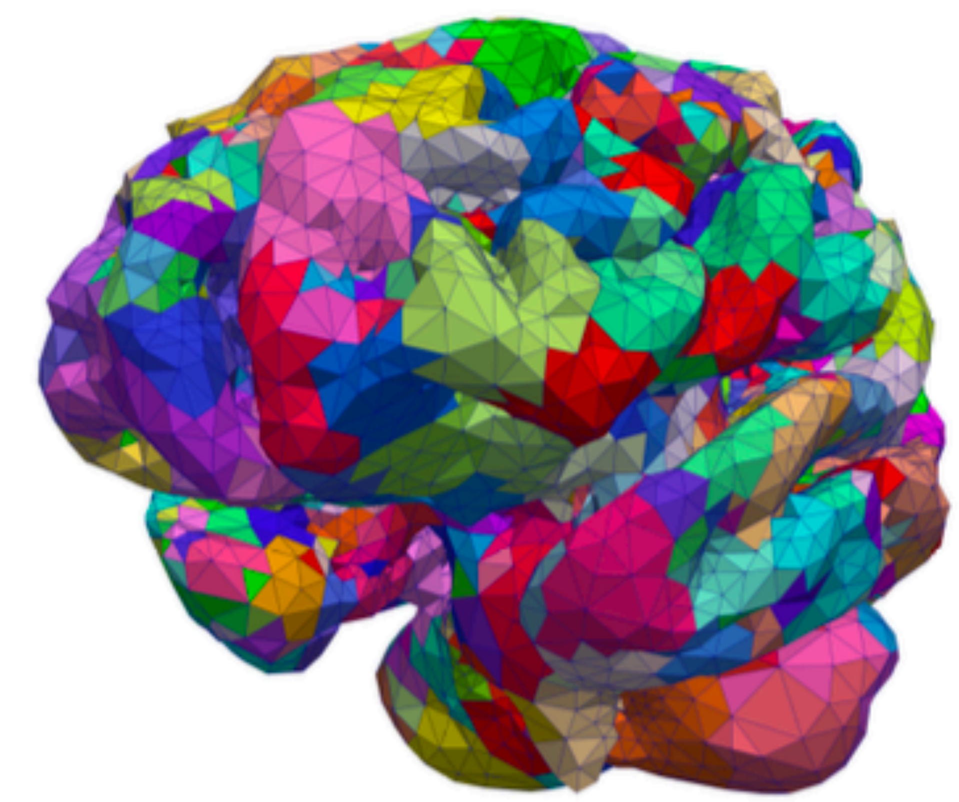


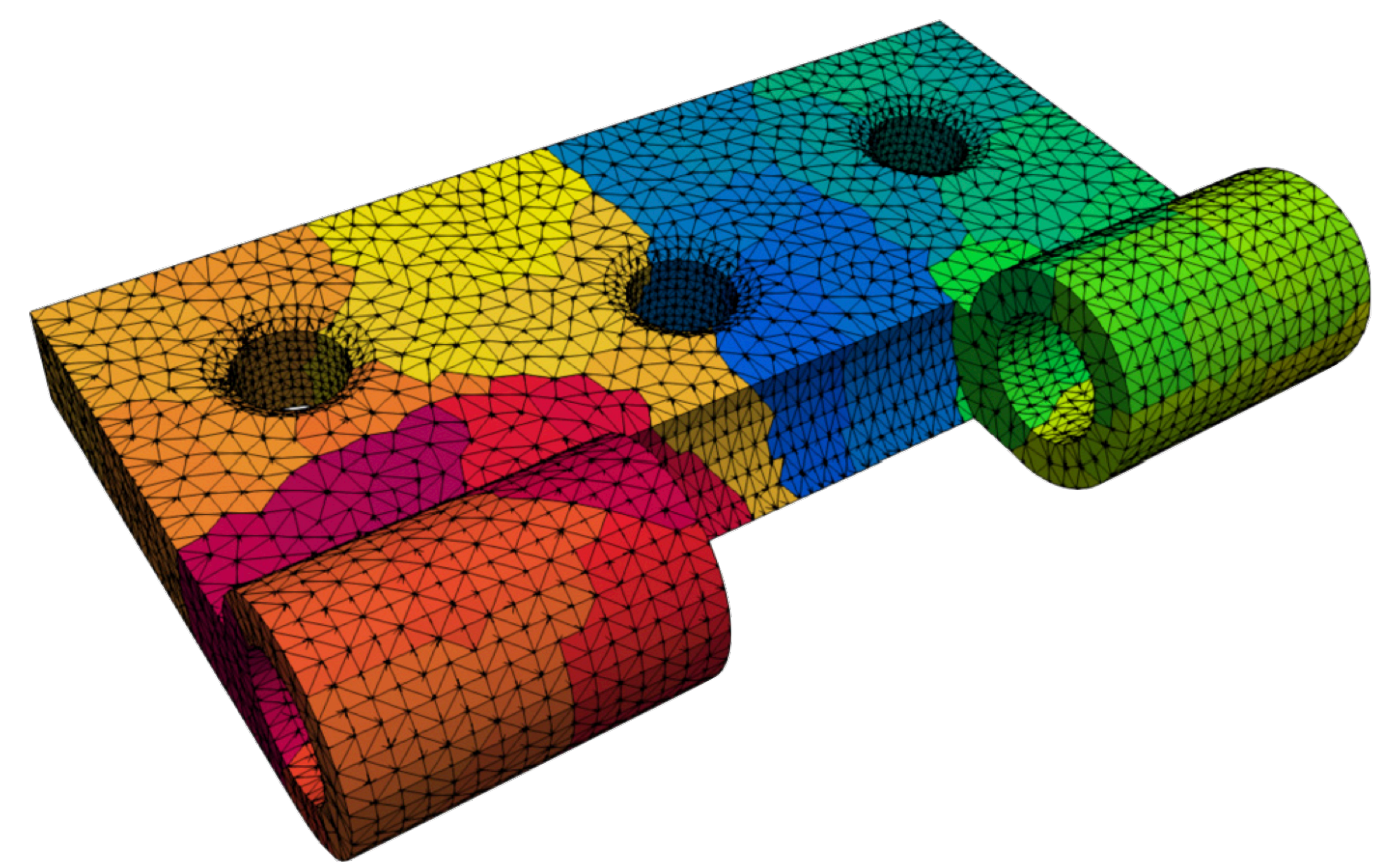


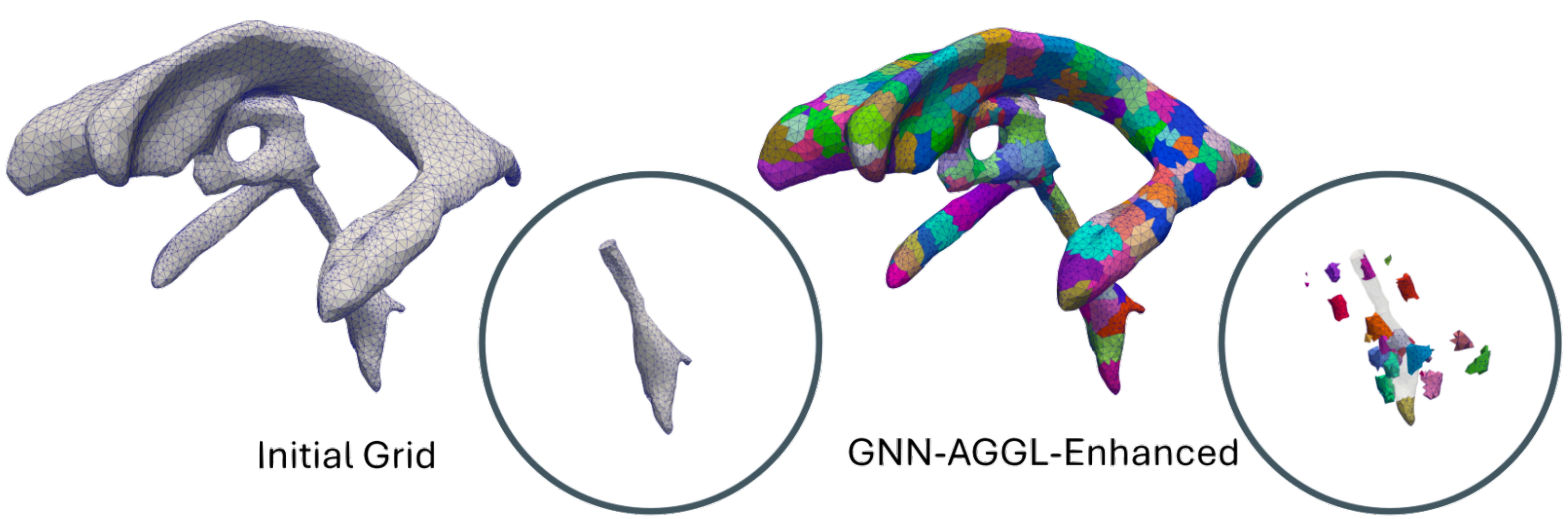
# Objective

- Develop effective algorithms to handle polygonal and polyhedral grids: agglomeration and mesh refinement algorithms based on employing geometrical deep learning
- Enhance the performance and accuracy of Polytopal Finite Element methods based on employing ML-aided acceleration strategies





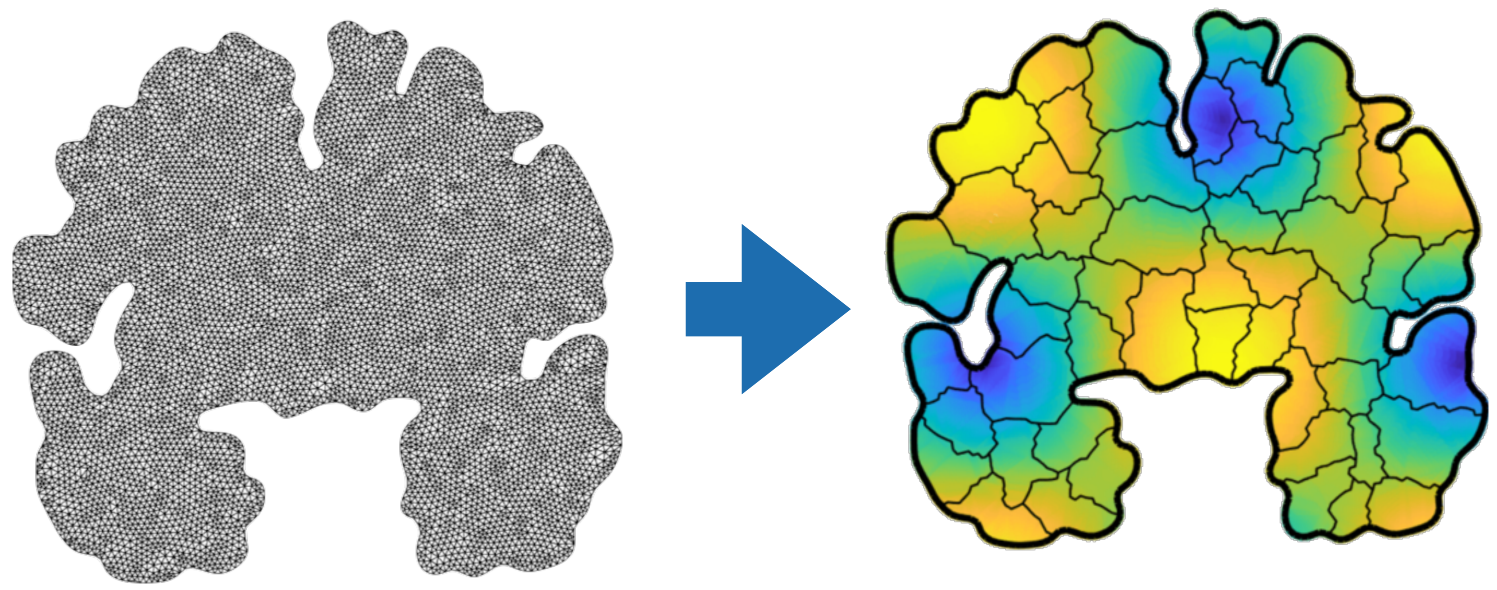


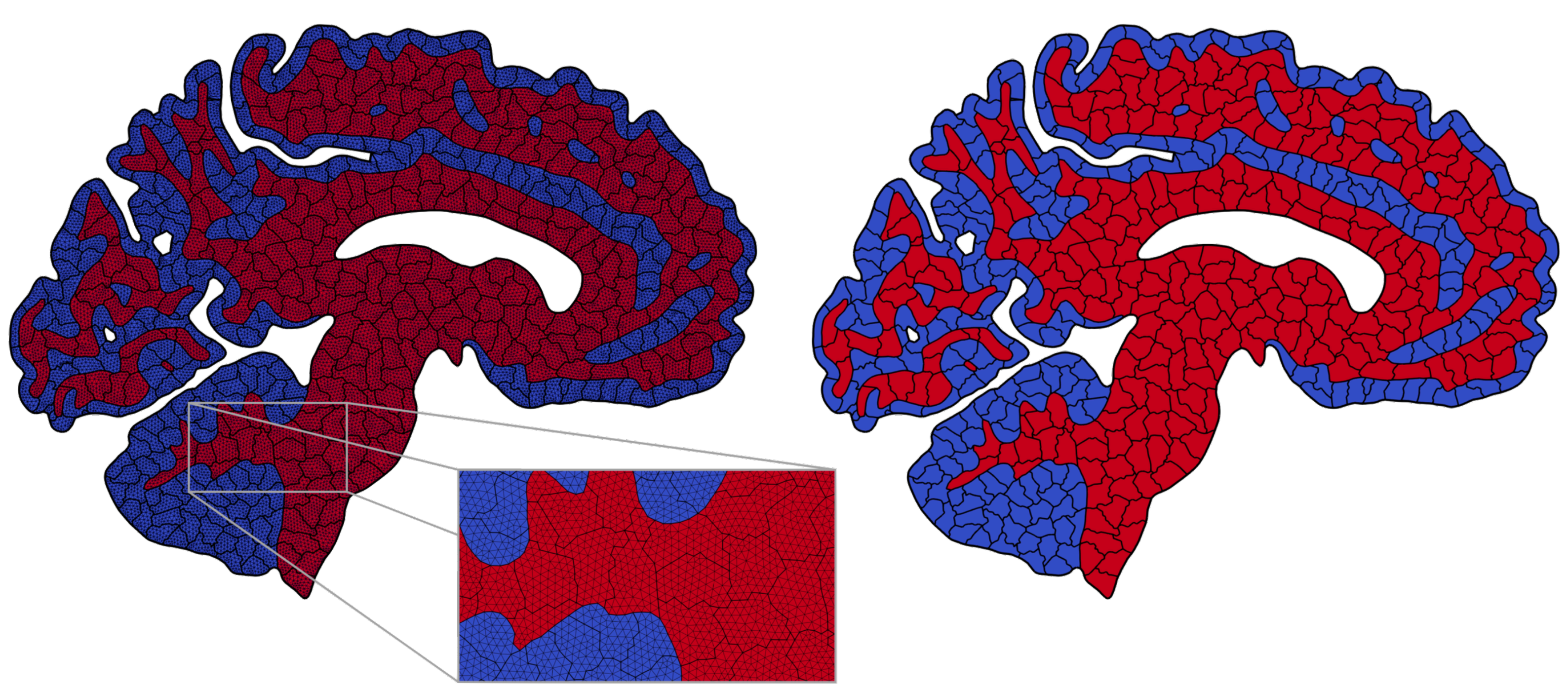


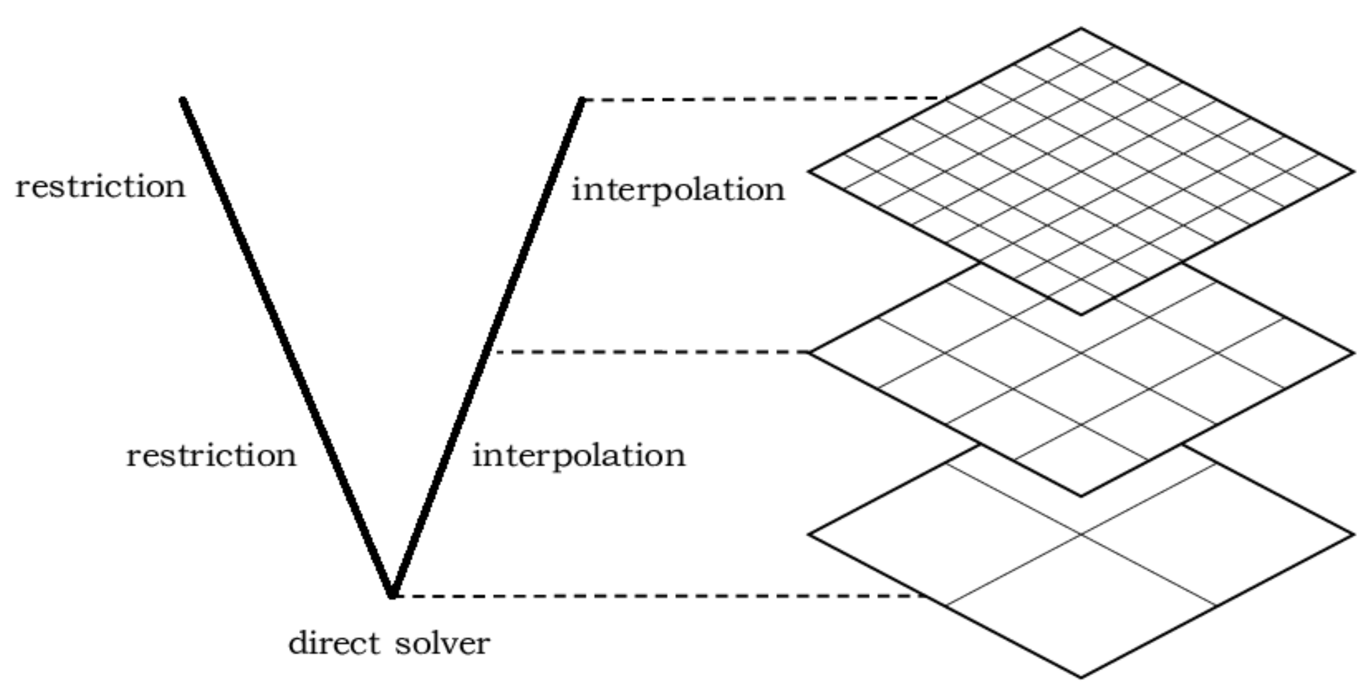
# Agglomeration of polytopal grids through Geometrical Deep Learning

Joint work with M. Corti, N. Farenga, G. Martinelli, Manuzzi, G. Martinelli, L. Saverio

# ML-enhanced agglomeration strategies








Merging neighboring mesh elements to obtain a coarser grid.

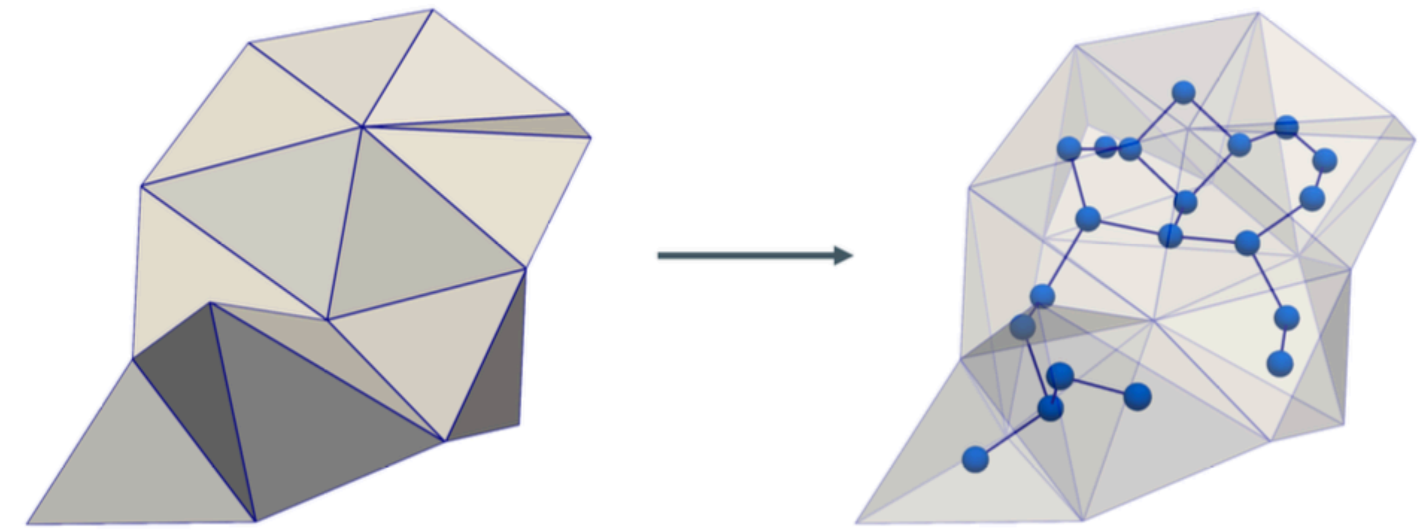
- Properly describe complicated geometries, respecting interior layers, heterogeneous material,…..
- Defeaturing of complicated geometries
- Design of multilevel solvers
- Reduce of the number of degrees of freedom

# Agglomeration via Geometrical Deep Learning

[Antonietti, Farenga, Manuzzi, Martinelli, Saverio, 2024, Antonietti, Corti, Martinelli, 2024]

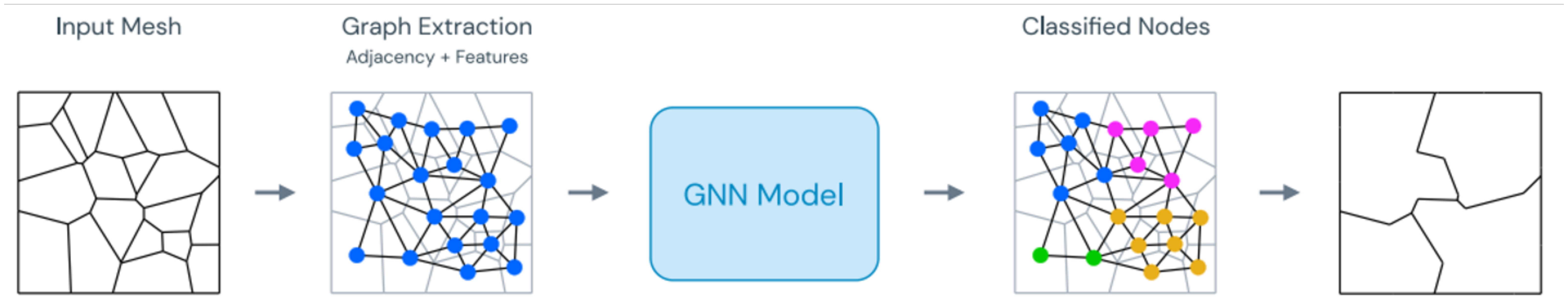
**Objective**

Find a partition with minimal interconnections between sets, while keeping errors (volumes) balanced.




## Advantages

- Process naturally and simultaneously the graph structure of mesh, the geometrical information (areas of the elements, their barycentric coordinates, …), and physical information (e.g. coefficients, layers, barriers)
- PDE independent, method independent, no need to perform agglomeration subdomainwise.
- Fast on line inference



# Agglomeration via Geometrical Deep Learning

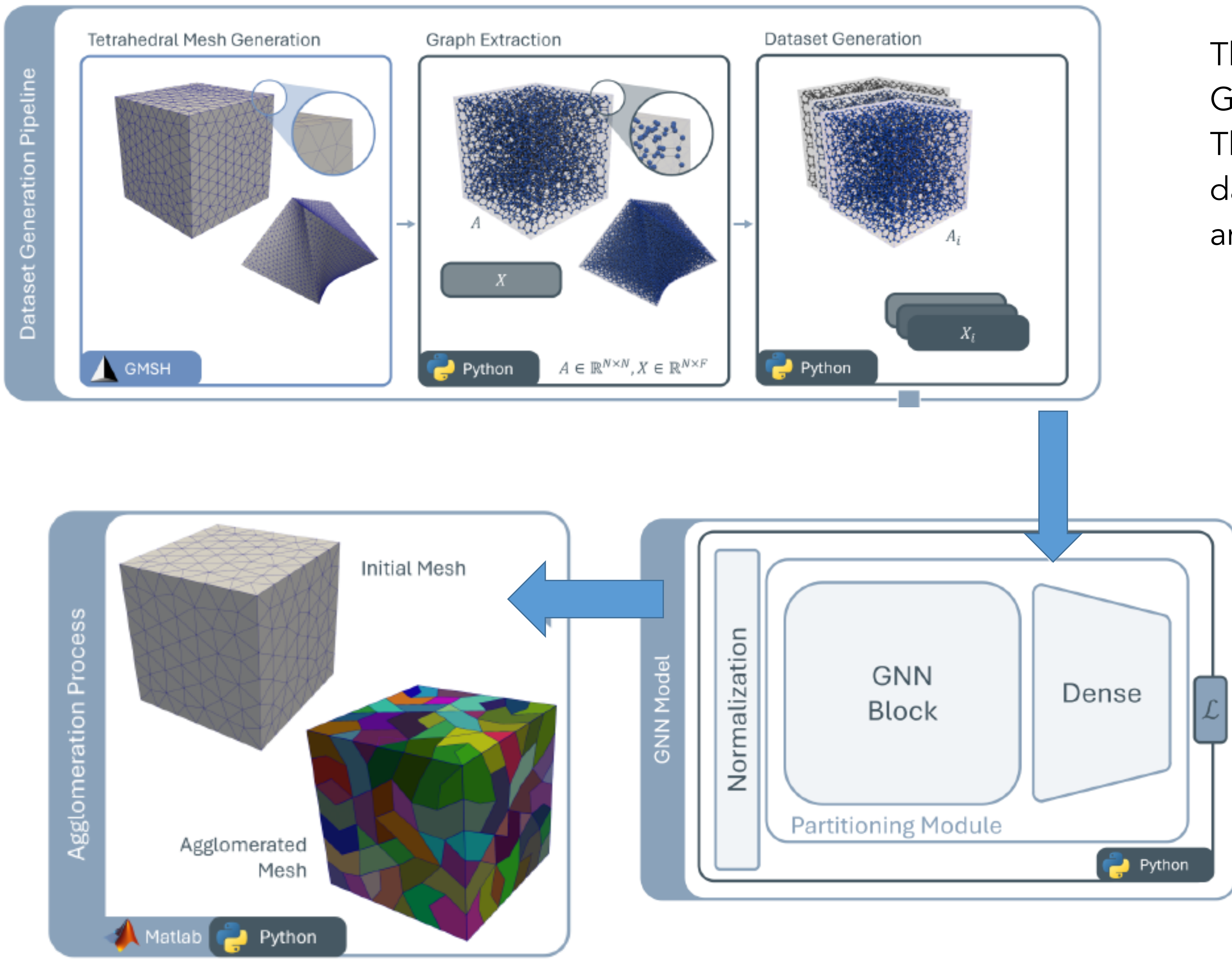
- Most agglomeration algorithms (e.g. METIS) tackle the problem by re-framing it into a graph partitioning problem by exploiting the connectivity structure of the mesh.

**Algorithm 1** General mesh agglomeration strategy AGGLOMERATE($\mathscr{T}_h, h^*$)

**Require:** Mesh $\mathscr{T}_h$, target mesh size $h^*$, and bisection model $\mathscr{M}$.
**Ensure:** Agglomerated mesh $\mathscr{T}_{h^*}$.
  **if** diam($\mathscr{T}_h$) $\leq h^*$ **then return** $\mathscr{T}_h$
  **else**
    Extract the connectivity graph $G = (V, E)$ and features $X$ from $\mathscr{T}_h$.
    $Y \leftarrow \mathscr{M}(G, X)$
    Adjust partition $Y$.
    Partition $\mathscr{T}_h$ into sub-meshes $\mathscr{T}_h^{(1)}, \mathscr{T}_h^{(2)}$ according to $Y$.
    $\mathscr{T}_{h^*}^{(1)} \leftarrow$ AGGLOMERATE($\mathscr{T}_h^{(1)}, h^*$)
    $\mathscr{T}_{h^*}^{(2)} \leftarrow$ AGGLOMERATE($\mathscr{T}_h^{(2)}, h^*$)
    $\mathscr{T}_{h^*} \leftarrow$ merge $\mathscr{T}_{h^*}^{(1)}, \mathscr{T}_{h^*}^{(2)}$
  **end if**

- The algorithm recursively bisects the input mesh's connectivity graph (with associated features) until the agglomerated elements have the desired size $h^*$.
- The GNN-based bisection model $\mathscr{M}$
  - input: mesh graph $\mathscr{G}$ + a set of features $X$ attached to each node (e.g. barycentric coordinates, area/volume, physical parameters)
  - output: vector of probabilities $Y$ of each node to belong to cluster 1 or 2.

# Agglomeration via Geometrical Deep Learning




The dataset of meshes is generated by means of GMSH.
Then, the graph is extracted for each mesh and the dataset is created by bundling graphs adjacencies and features matrices

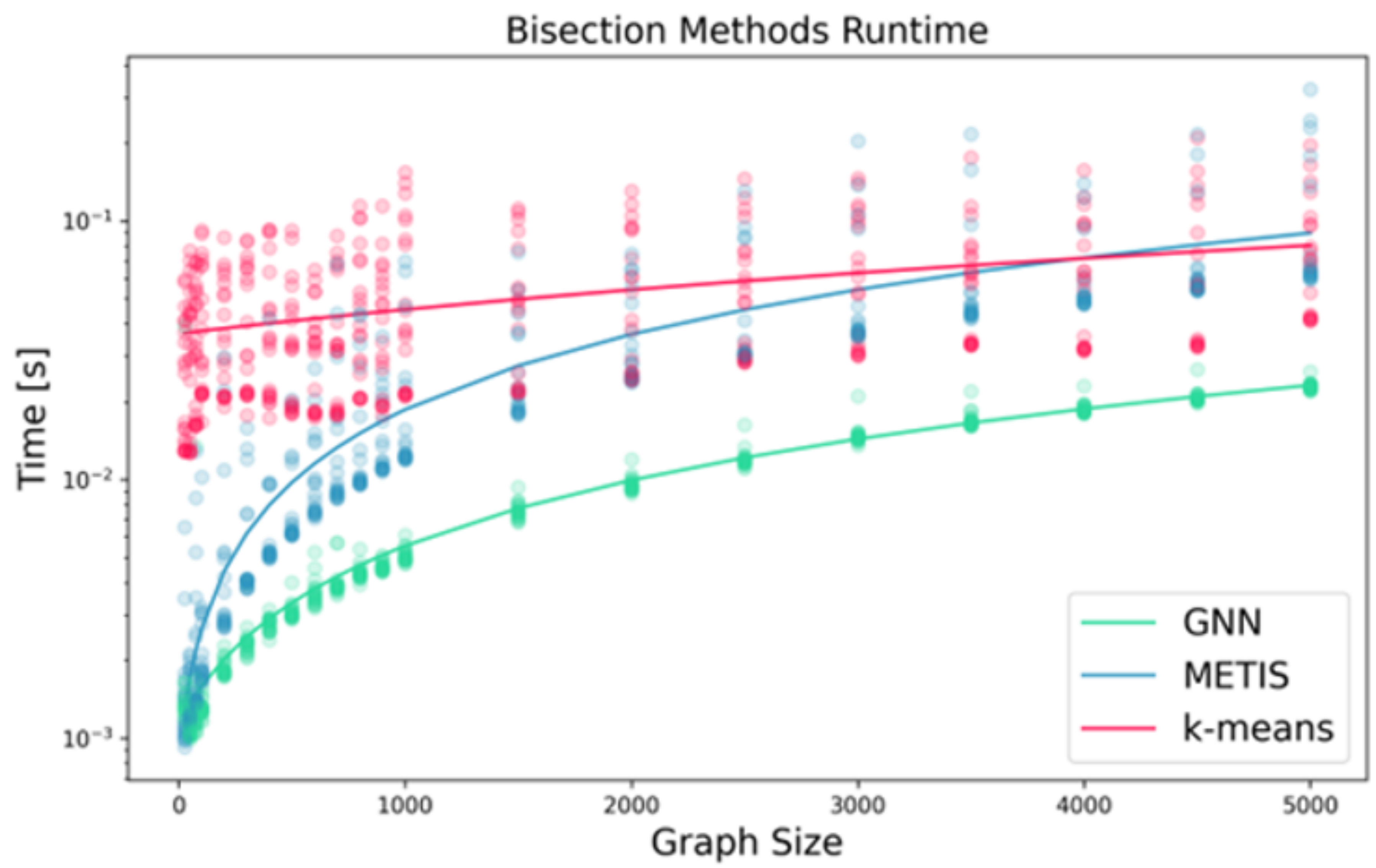
GNN-model is trained in an unsupervised learning setting, using a PyTorch Geometric implementation

The trained model is loaded in evaluation-mode and used to perform iteratively the graph-bisection, obtaining a clustered graph

lymph

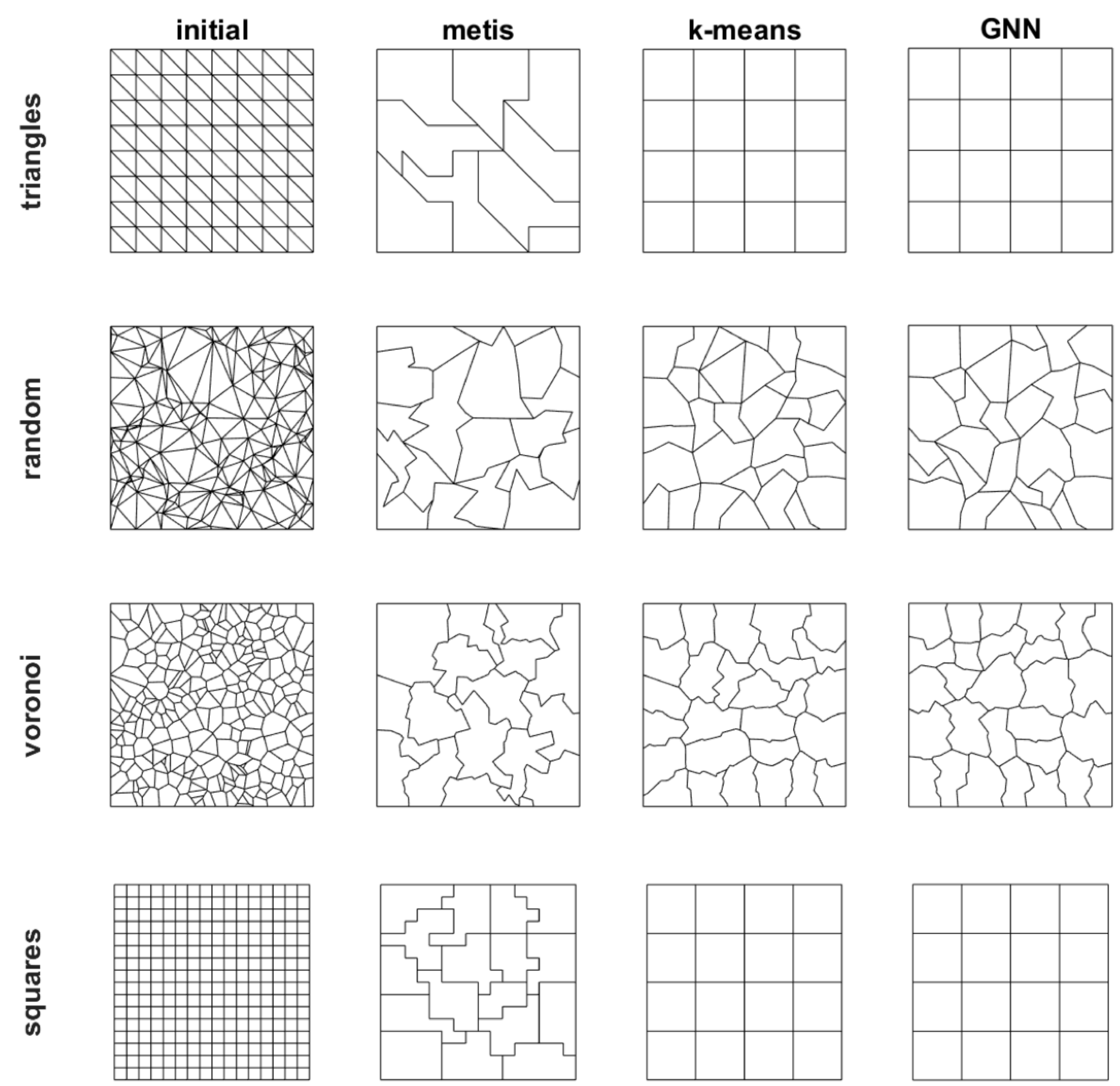
**Soon available in lymph**

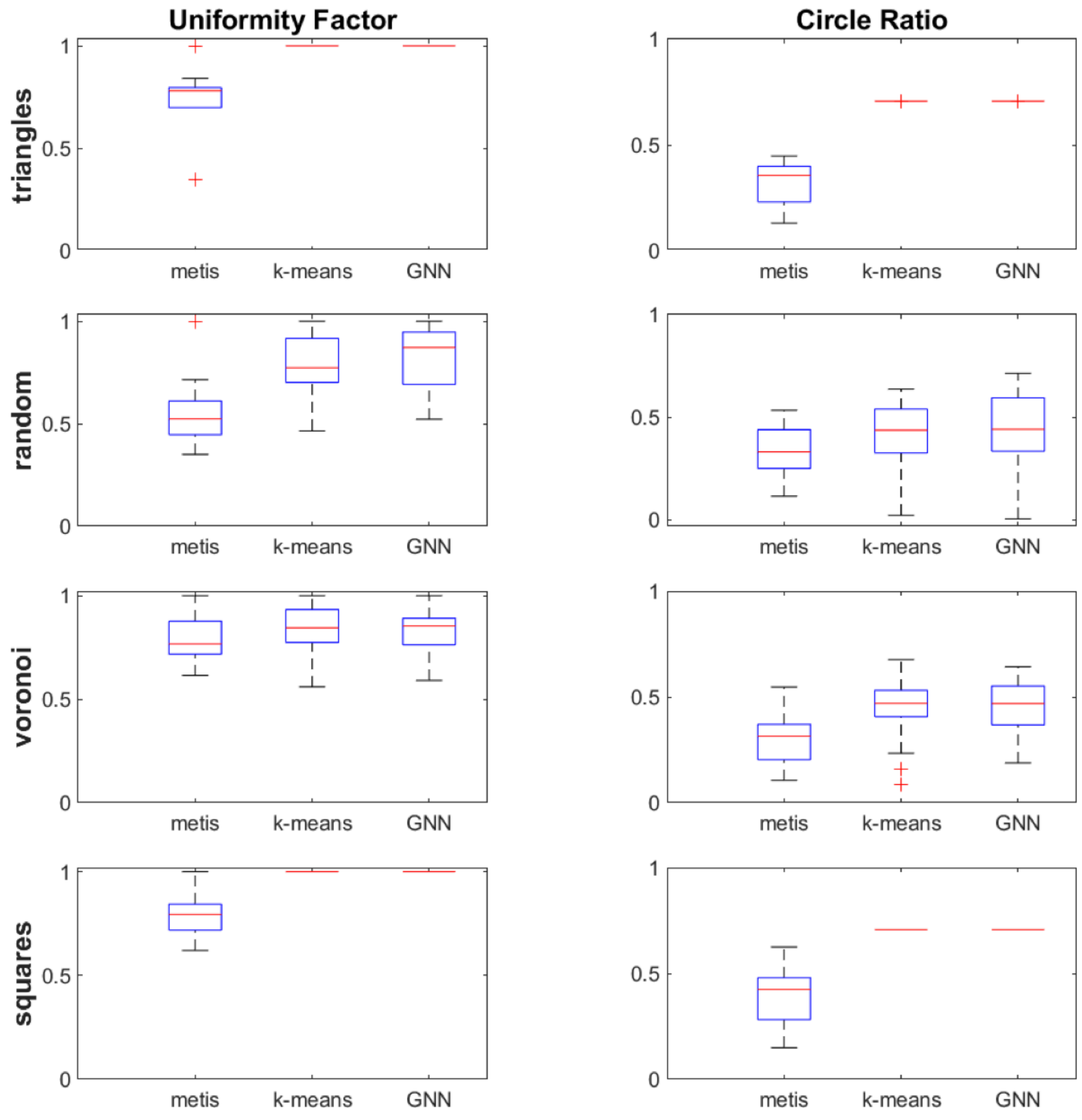
# Computational costs (2D)




Runtime performance for different graph bisection models (METIS, k-means, GNN) as a function of the number of nodes in the connectivity graph of Voronoi meshes. The y-axis is in logarithmic scale

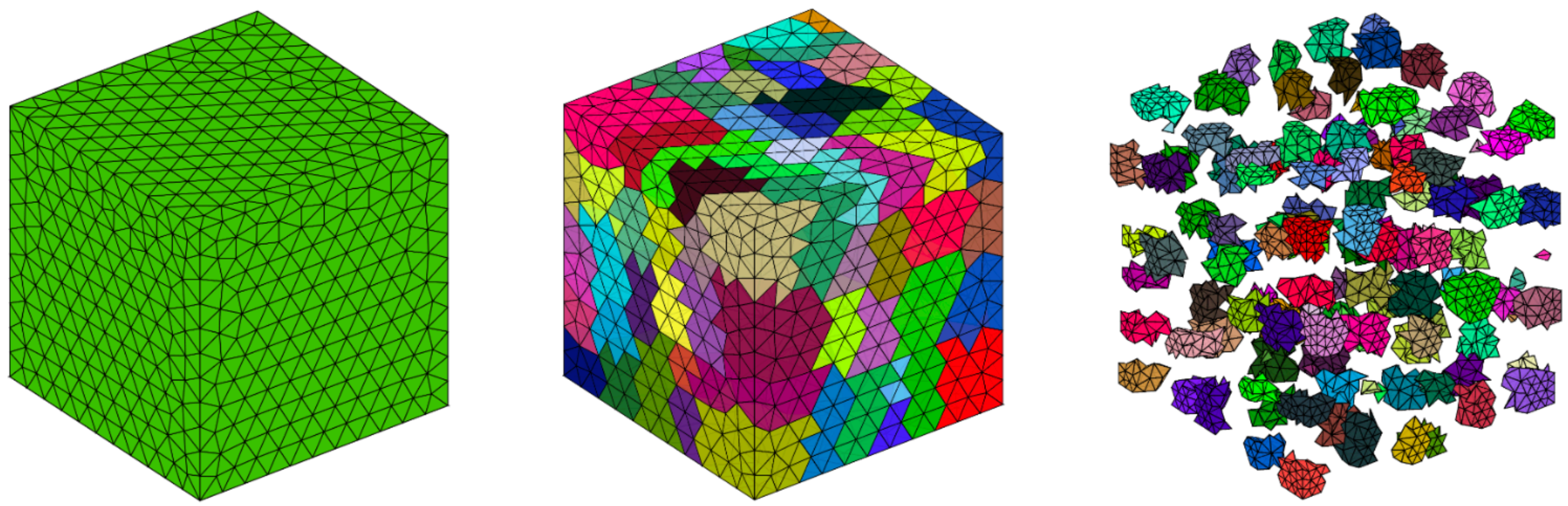
# Quality Metrics (2D)



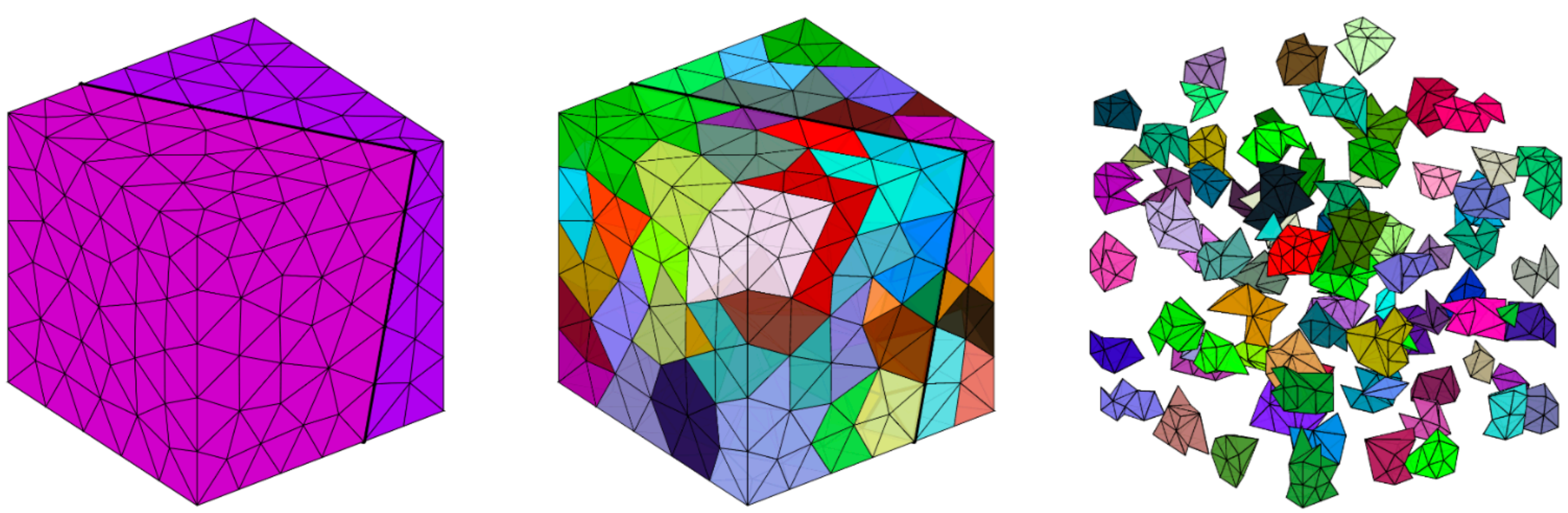



# Agglomeration in 3D

The agglomeration in 3D is much more complicated, suitable modification to the 2D algorithm.

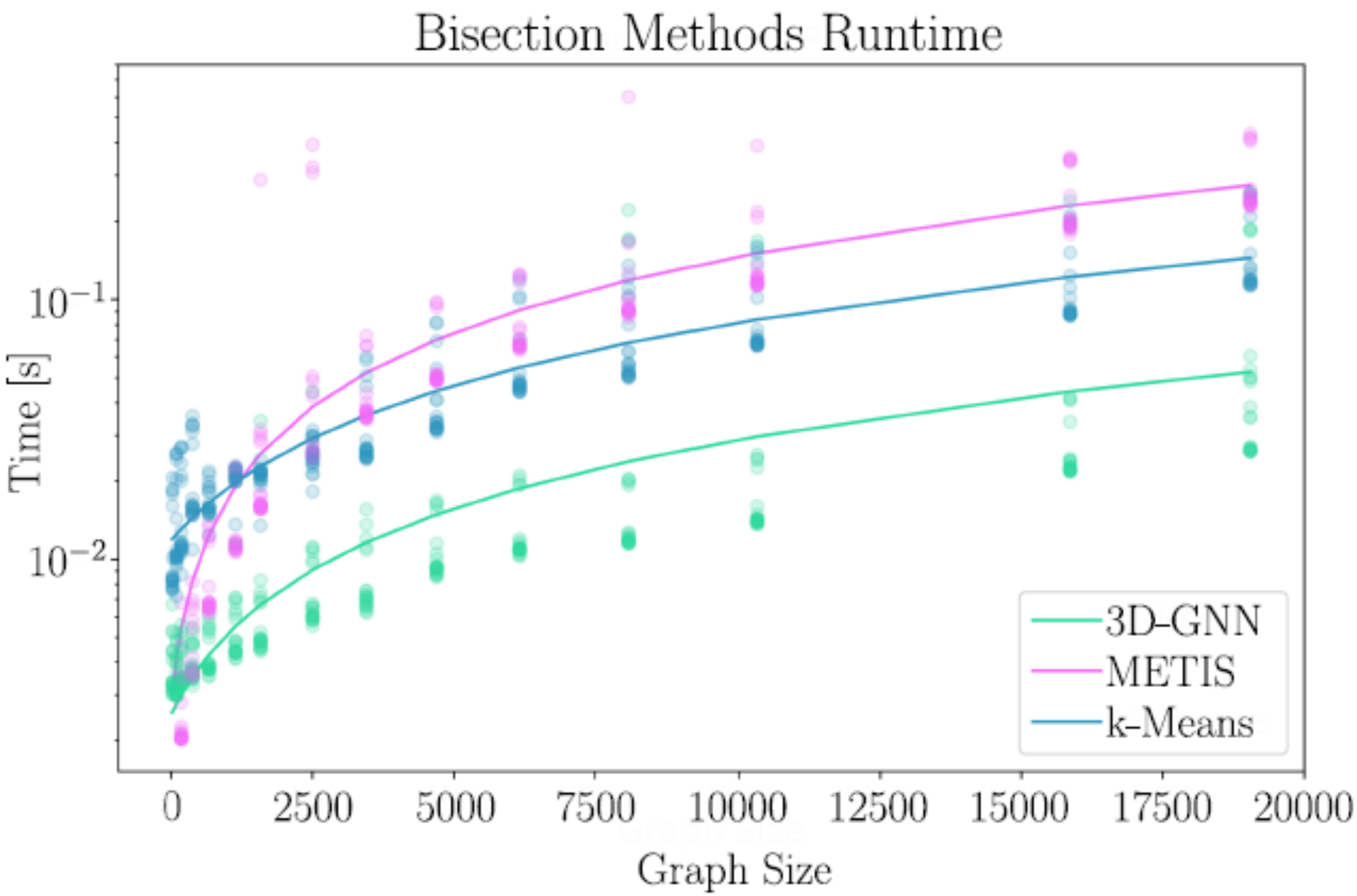


**(a)** original mesh **(b)** agglomerated mesh **(c)** agglomerated mesh (exploded)



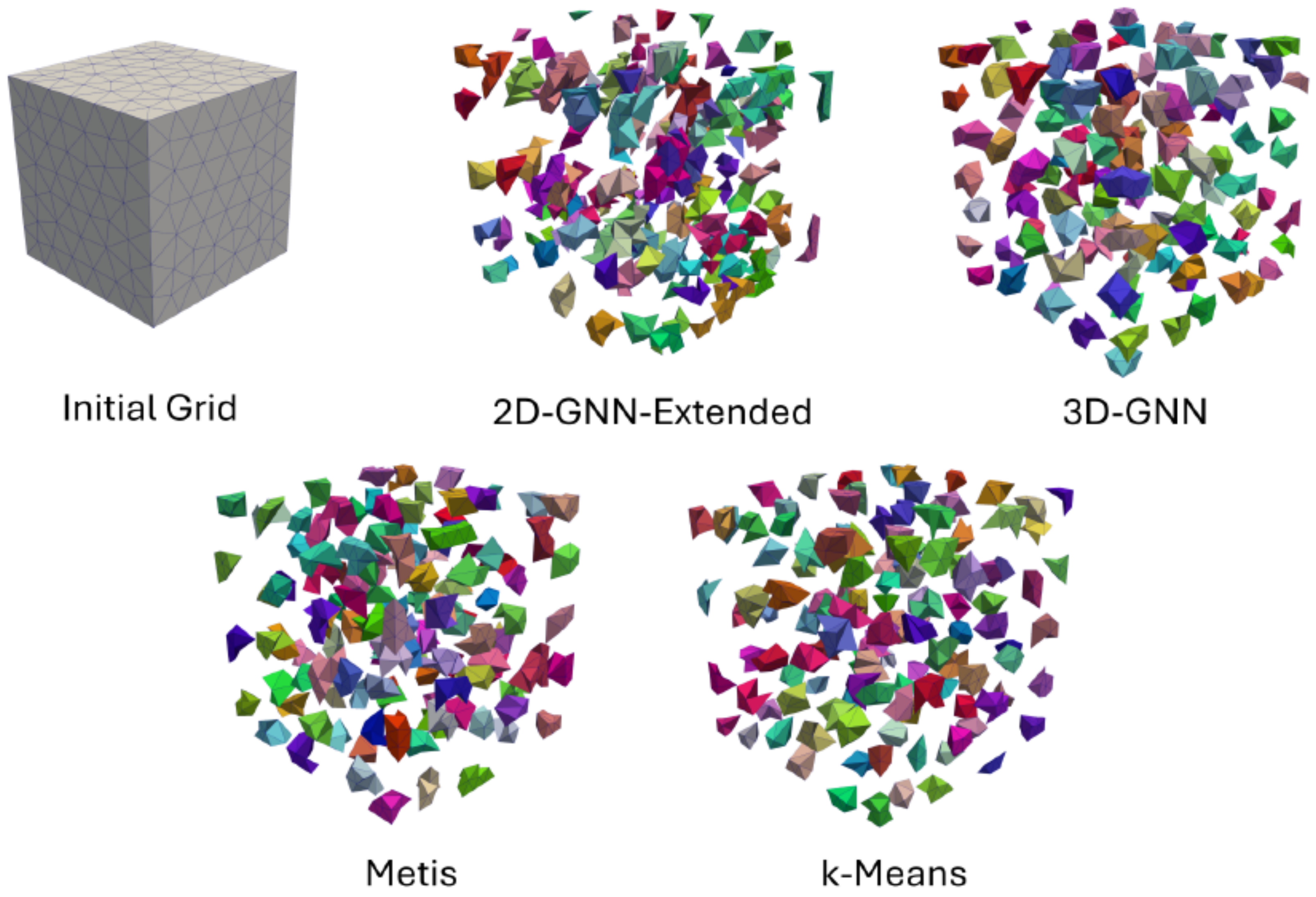
**(a)** original mesh **(b)** agglomerated mesh **(c)** agglomerated mesh (exploded)

# Runtimes (3D)




Runtime performance for different graph bisection models (METIS, k-Means, and 3D-GNN) as a function of the number of nodes in the connectivity graph of tetrahedral meshes.

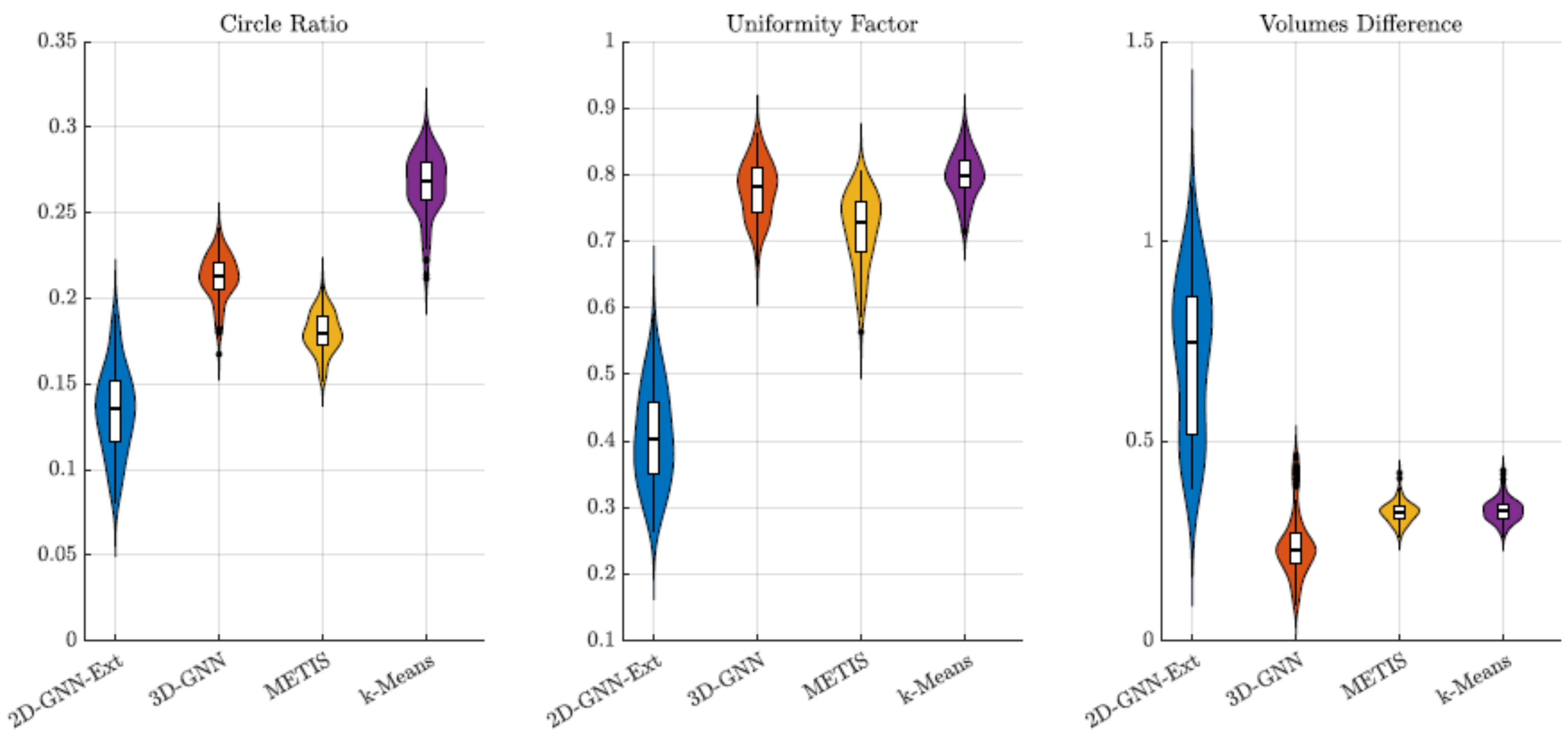
# Agglomeration capabilities assessment of a test dataset (3D)



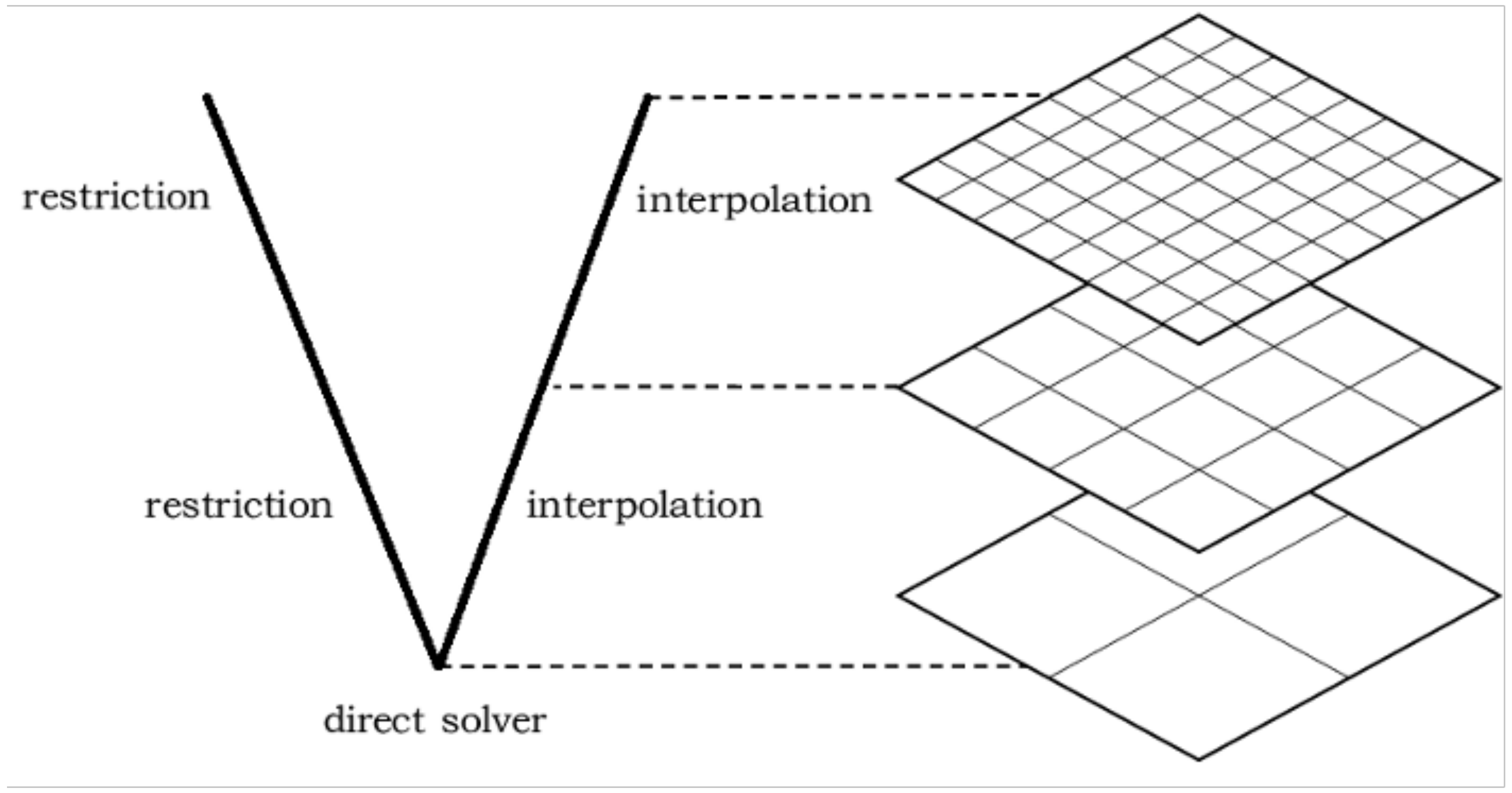

| Agglomeration Method | Circle Ratio (CR) | Uniformity Factor (UF) | Volume Difference (VD) |
|---|---|---|---|
| **2D-GNN-Extended** | 0.1342 | 0.4053 | 0.7166 |
| **3D-GNN** | 0.1946 | 0.7676 | 0.2355 |
| **METIS** | 0.1799 | 0.7183 | 0.3517 |
| **k-Means** | 0.2662 | 0.7982 | 0.3270 |

Average values of the computed quality metrics for the agglomerated grids obtained with different agglomeration strategies (2D-GNN-Extended, 3D-GNN, METIS, k-Means)

Distributions of numeric data (violin plots) of the computed quality metrics obtained with different agglomeration strategies (2D-GNN-Extended, 3D-GNN, METIS, k-Means).



# MG solvers




| grids | $\ell$ | Agglomeration-based MG | | | CG |
|---|---|---|---|---|---|
| | | metis | k-means | GNN | |
| triangles | 2 | 21 | 9 | 9 | 114 |
| | 3 | 21 | 9 | 9 | |
| | 4 | 21 | 9 | 9 | |
| random | 2 | 46 | 41 | 32 | 655 |
| | 3 | 46 | 41 | 32 | |
| | 4 | 46 | 41 | 32 | |

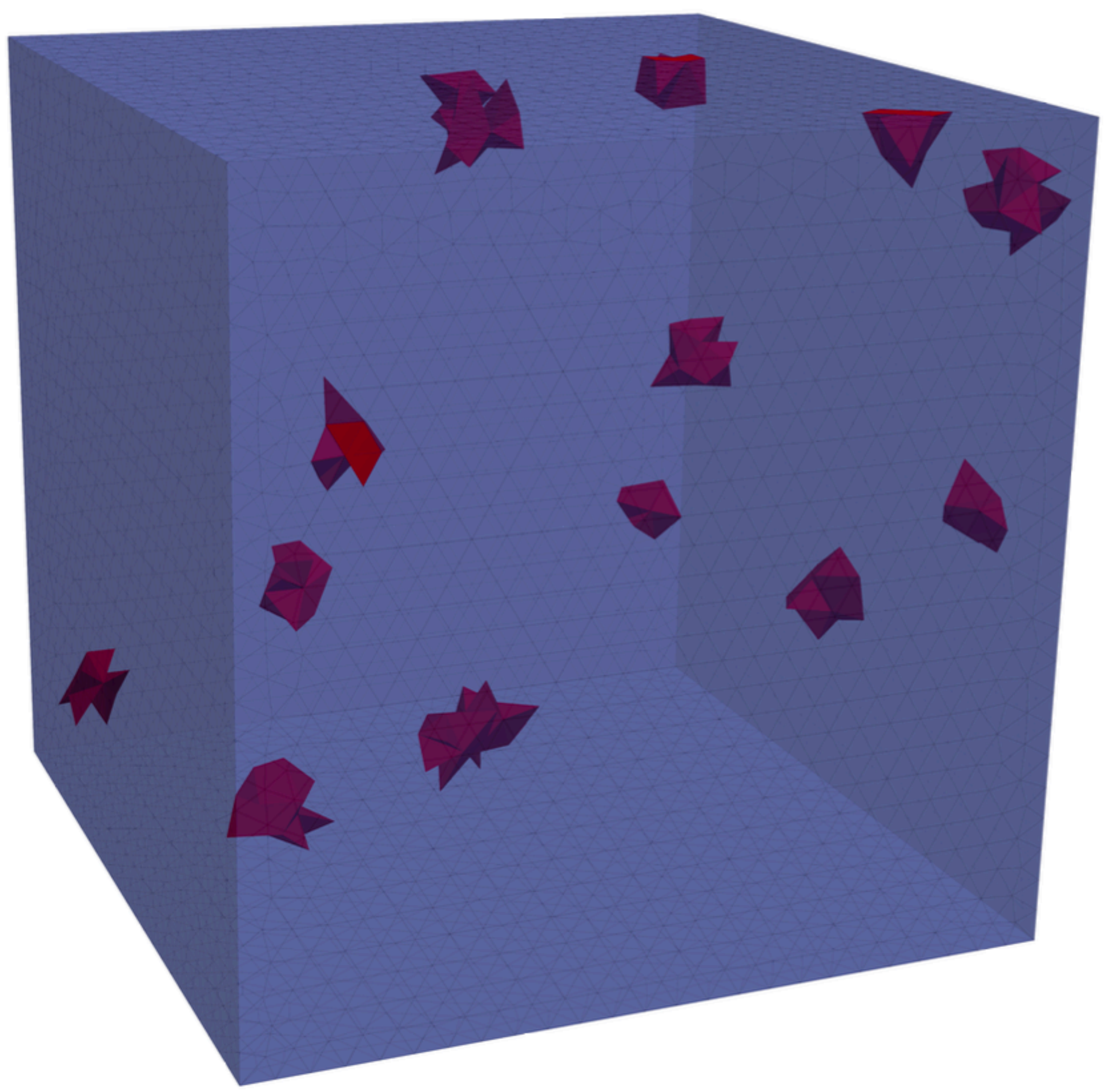
Iteration counts of the MG algorithm. Different initial grids agglomerated with different strategies (METIS, k-means, GNN) and different levels.

As a comparison, the iteration count of the Conjugate Gradient (CG) method are reported in the last column.

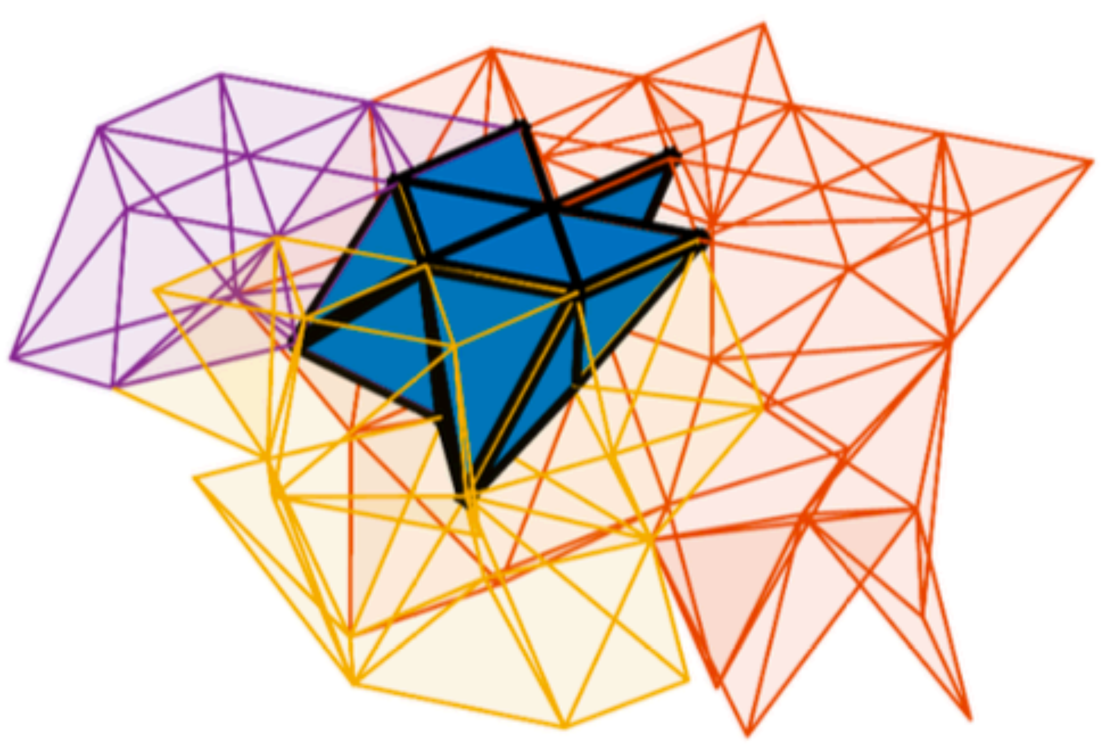


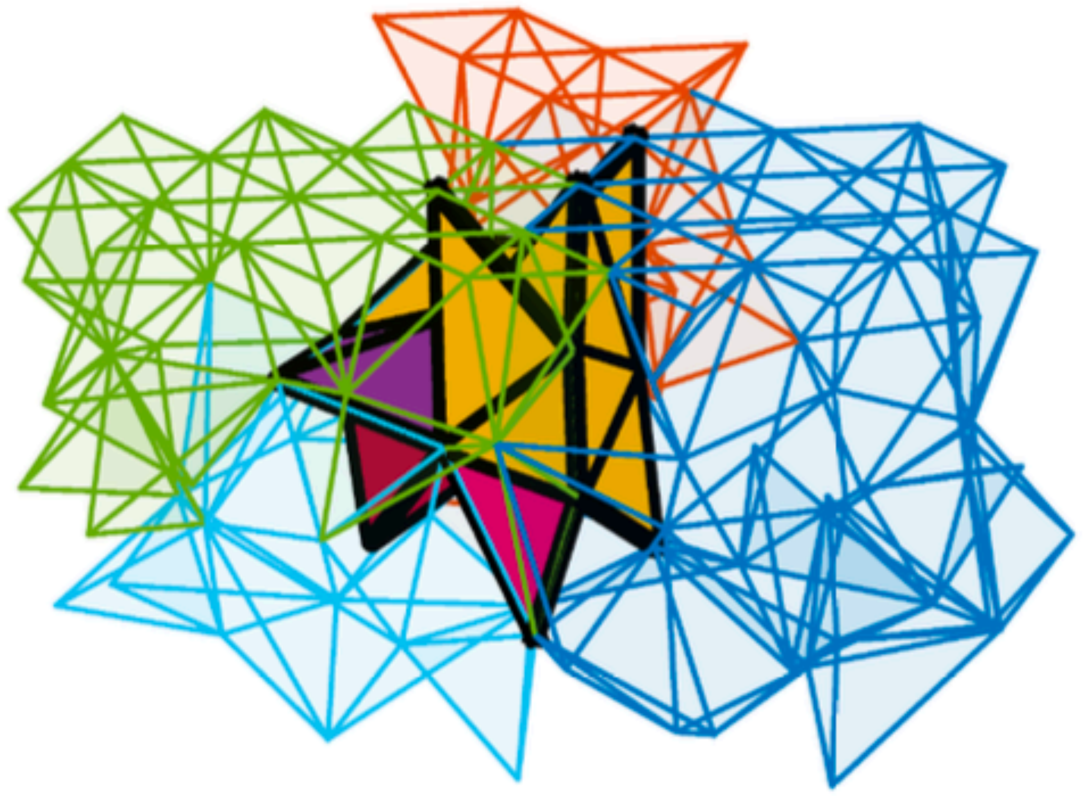
[Antonietti, Pennesi, JSC, 2019], [Antonietti, Houston, Hu, Sarti, Verani, 2017]

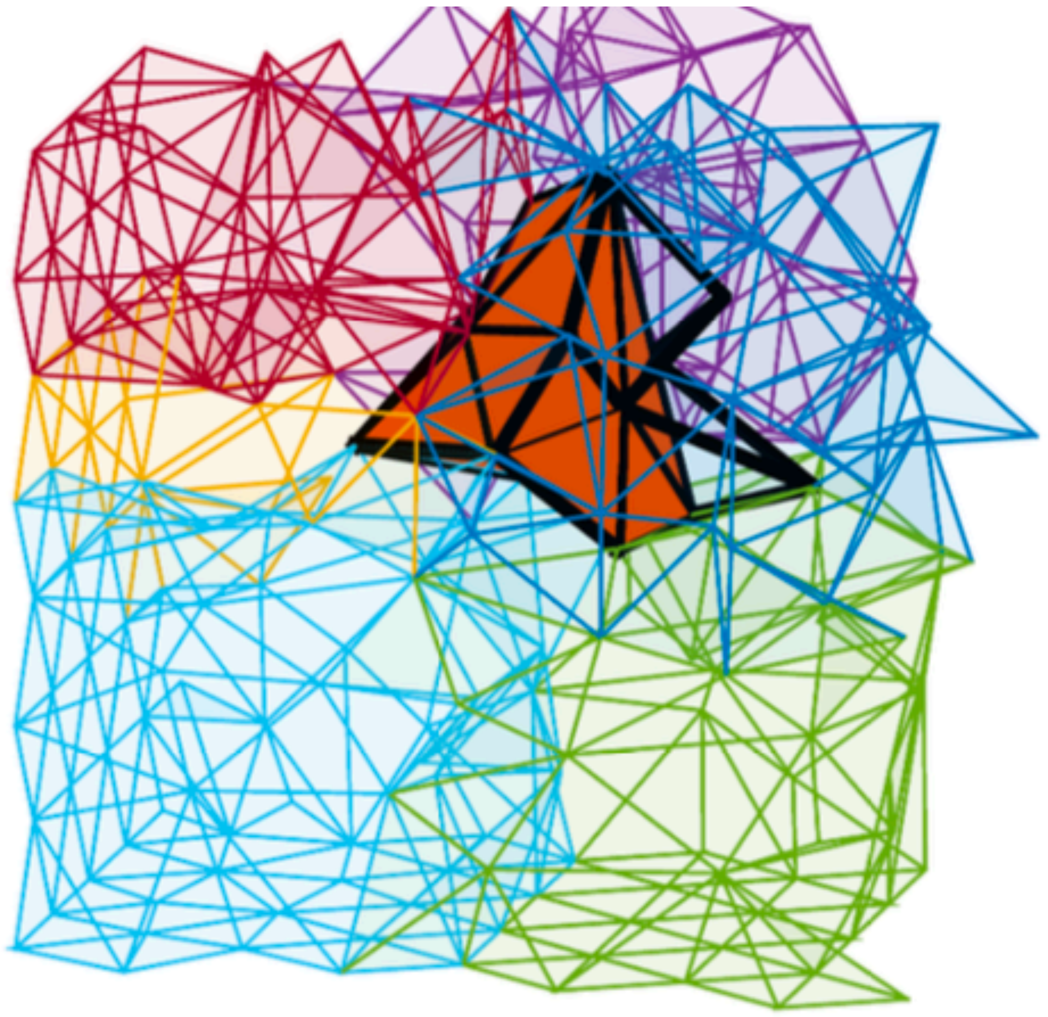
# “Simple” domain with microstructures



Domain with inclusions

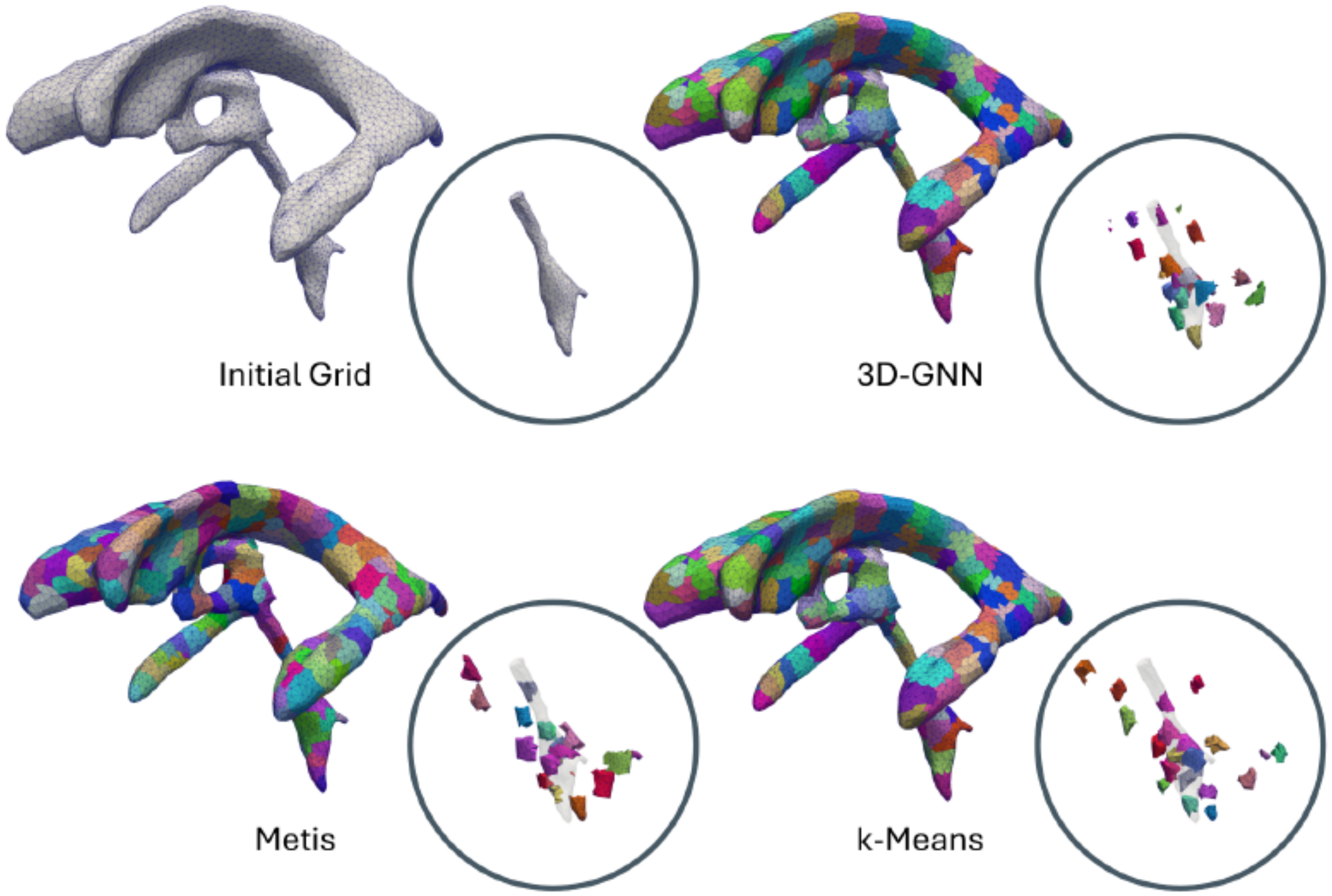






# Complicated geometries

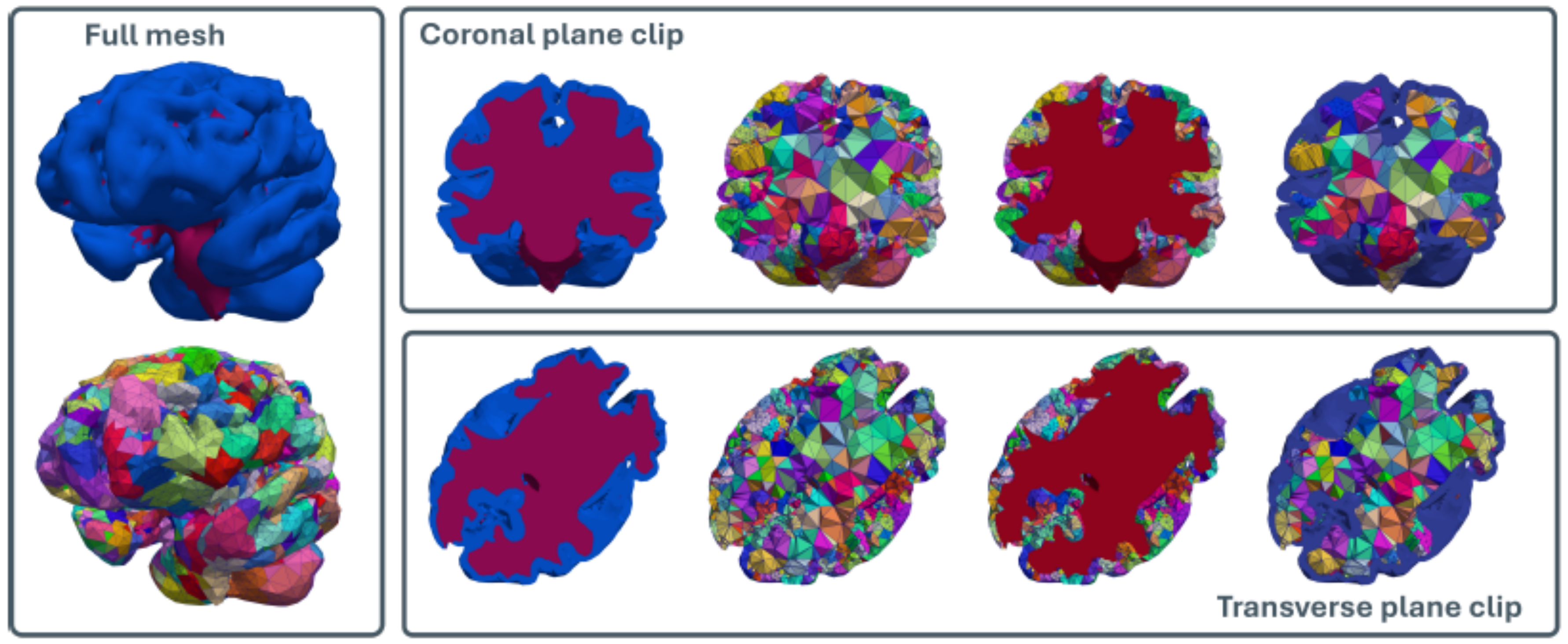
**Agglomeration of a mesh of human brain ventricles**




homogeneous coefficients

**Agglomeration of a mesh of human brain**

heterogeneous coefficients



# Testing the GNN-enhanced algorithm in practice

*Find* $(\boldsymbol{u}, p_A, p_C, p_V, p_E)$ *such that in* $\Omega \times (0, T_f]$ *it holds:*

**PARENCHYMAL TISSUE** $\rho \dfrac{\partial^2 \boldsymbol{u}}{\partial t^2} - \nabla \cdot (\mu \boldsymbol{\varepsilon}(\boldsymbol{u}) + \lambda (\nabla \cdot \boldsymbol{u}) \mathbf{I}) + \alpha_A \nabla p_A + \alpha_C \nabla p_C + \alpha_V \nabla p_V + \alpha_E \nabla p_E = \boldsymbol{f}$

**ARTERIES (A)** $c_A \dfrac{\partial p_A}{\partial t} + \nabla \cdot \left( \alpha_A \dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{\mathbf{K}_A}{\mu_A} \nabla p_A \right) + \beta_{AC}(p_A - p_C) = g_A$

**CAPILLARIES (C)** $c_C \dfrac{\partial p_C}{\partial t} + \nabla \cdot \left( \alpha_C \dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{\mathbf{K}_C}{\mu_C} \nabla p_C \right) + \beta_{AC}(p_C - p_A) + \beta_{CV}(p_C - p_V) + \beta_{CE}(p_C - p_E) = g_C$

**VEINS (V)** $c_V \dfrac{\partial p_V}{\partial t} + \nabla \cdot \left( \alpha_V \dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{\mathbf{K}_V}{\mu_V} \nabla p_V \right) + \beta_{CV}(p_V - p_C) + \beta_{VE}(p_V - p_E) + \beta_V^e (p_V - \tilde{p}_{Vein}) = g_V$

**CSF-ISF (E)** $c_E \dfrac{\partial p_E}{\partial t} + \nabla \cdot \left( \alpha_E \dfrac{\partial \boldsymbol{u}}{\partial t} - \dfrac{\mathbf{K}_E}{\mu_E} \nabla p_E \right) + \beta_{CE}(p_E - p_C) + \beta_{VE}(p_E - p_V) = g_E$

PolyDG semi-discretisation + analysis



[M. Corti, P. F. Antonietti, L. Dede', A. Quarteroni, M3AS, 2023]

# Testing the GNN-enhanced algorithm in practice

| Parameter | Value | | Parameter | Value | |
|---|---|---|---|---|---|
| $\rho$ | 1000.00 | [Kg/m$^3$] | $k_1 = k_2$ | $3.50 \times 10^{-11}$ | [m$^2$] |
| $\lambda$ | 505.00 | [Pa] | $\mu$ | 216.00 | [Pa] |
| $\mu_1 = \mu_2$ | $3.50 \times 10^{-3}$ | [Pa $\cdot$ s] | $\beta_{12}$ | $10^{-7}$ | [m$^2$/(N $\cdot$ s)] |
| $\alpha_1$ | 0.49 | [−] | $\alpha_2$ | 0.51 | [−] |
| $c_1 = c_2$ | $10^{-6}$ | [m$^2$/N] | $\beta_1^e = \beta_2^e$ | 0.00 | [m$^2$/(N $\cdot$ s)] |

Table 1: Physical parameter values used in the 2D brain simulation.

## Analytical Solution

$$\boldsymbol{u}(x,y,t) = \sin(\pi t)\begin{bmatrix} -\cos(\pi x)\cos(\pi y) \\ \sin(\pi x)\sin(\pi y)\end{bmatrix},$$

$$p_1(x,y,t) = 10^4\pi\sin(\pi t)(\cos(\pi y)\sin(\pi x) + \cos(\pi x)\sin(\pi y)),$$

$$p_2(x,y,t) = 10^4\pi\sin(\pi t)(\cos(\pi y)\sin(\pi x) - \cos(\pi x)\sin(\pi y)).$$

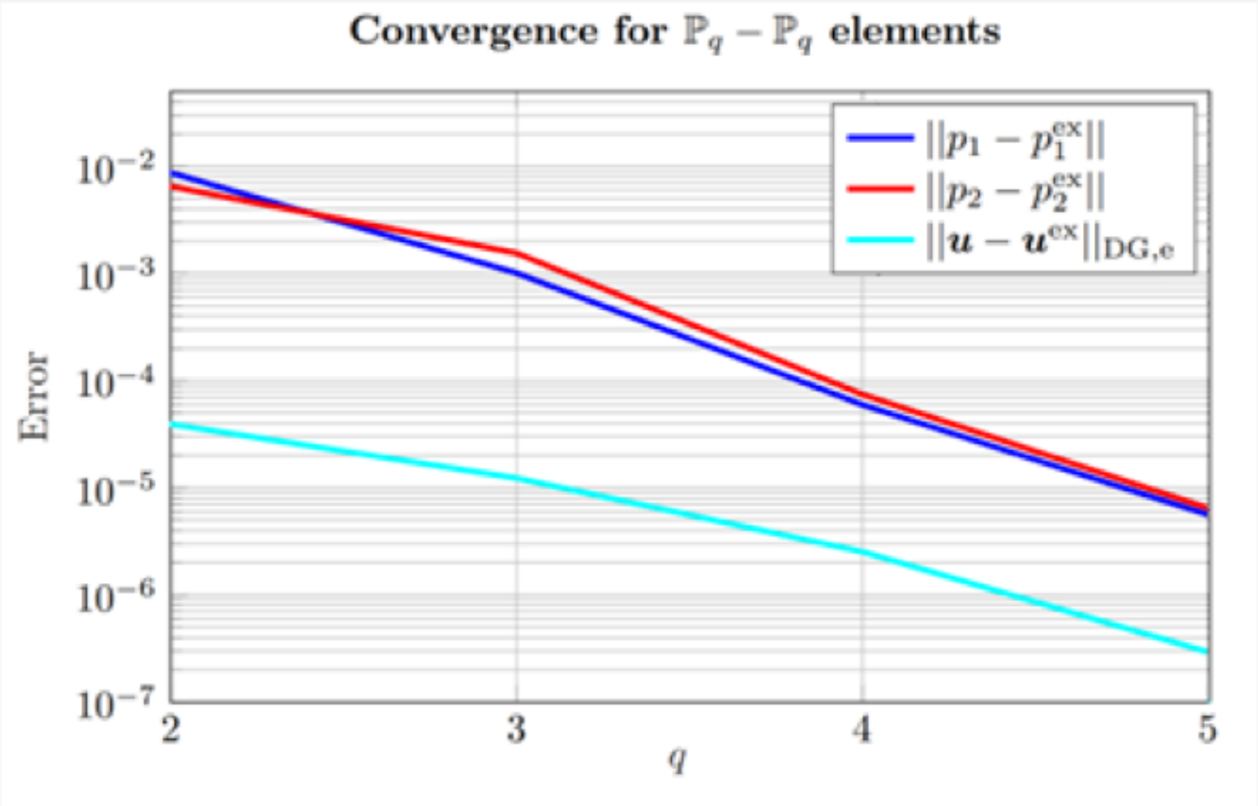



**Figure:** Errors against the PolyDG polynomial order

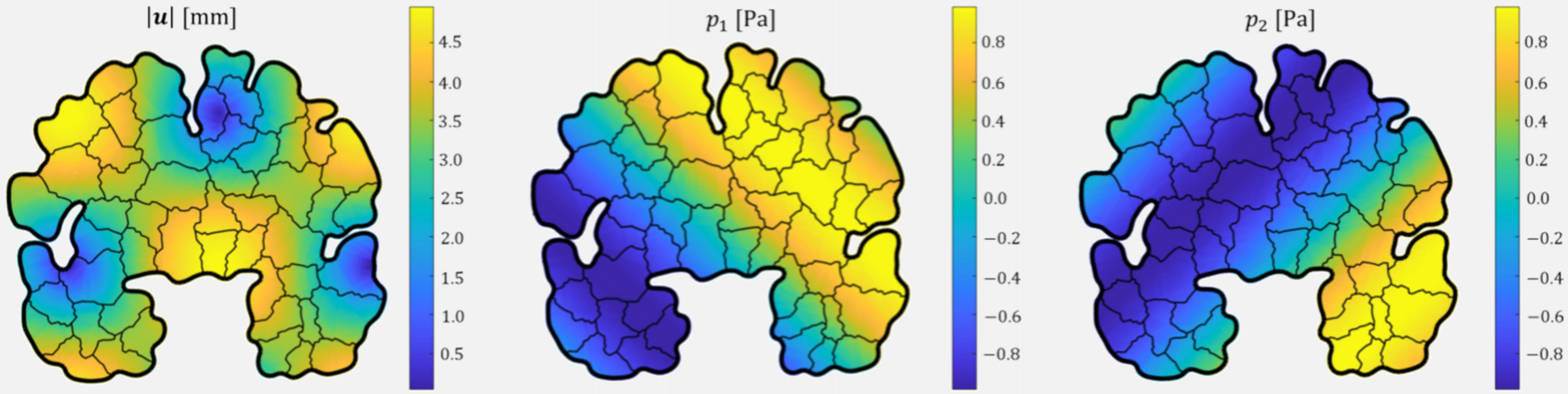
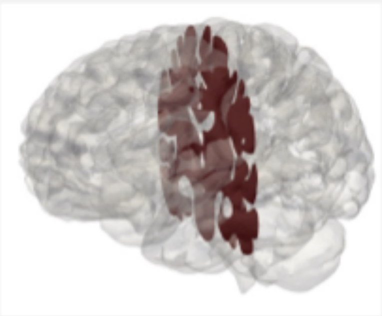



**Figure:** Agglomerated mesh of brain slice (with GNN)

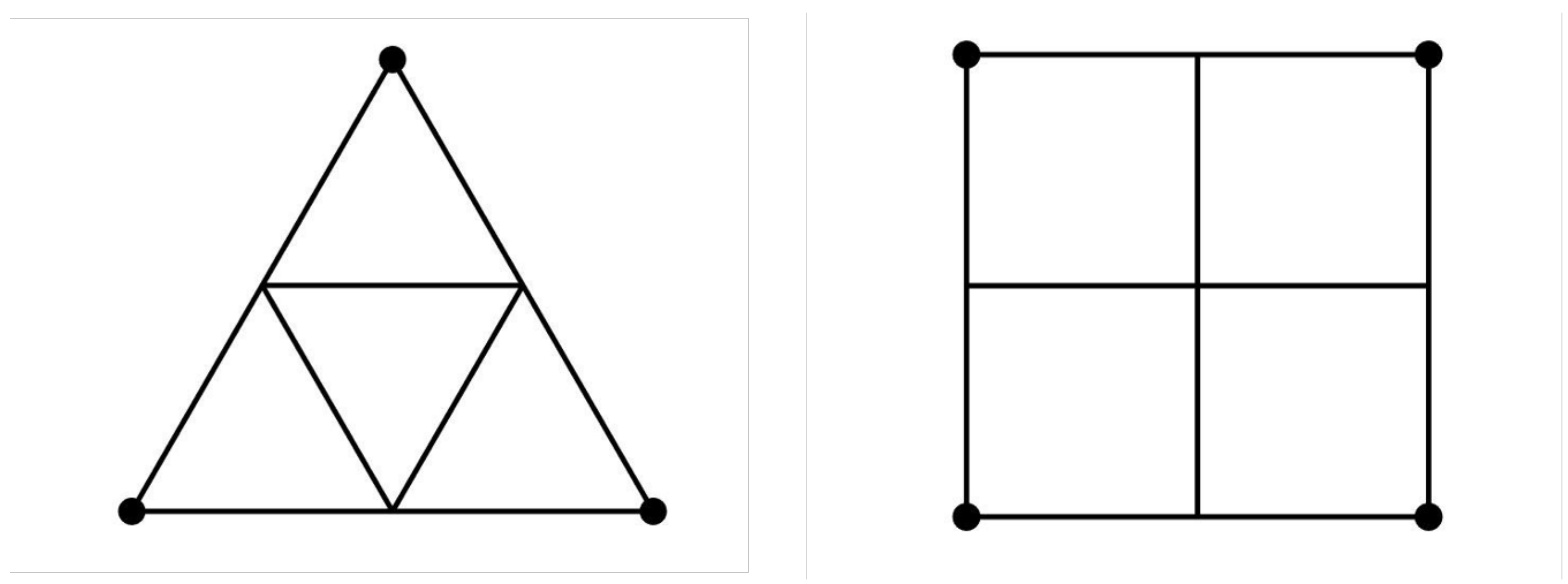


# ML-driven refinement strategies

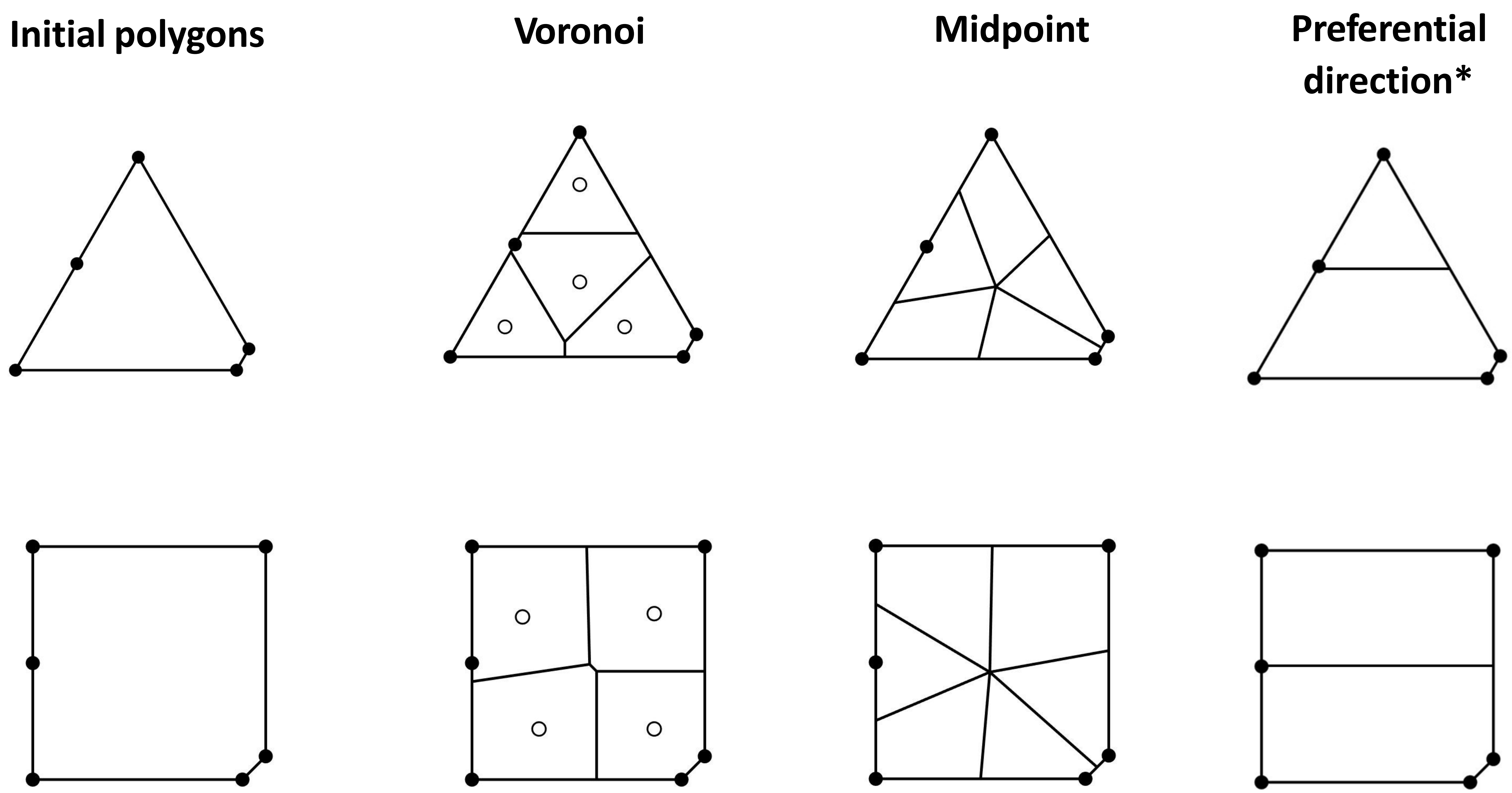
Joint work with F. Dassi, E.Manuzzi

# ML-enhanced mesh refinement (2D)

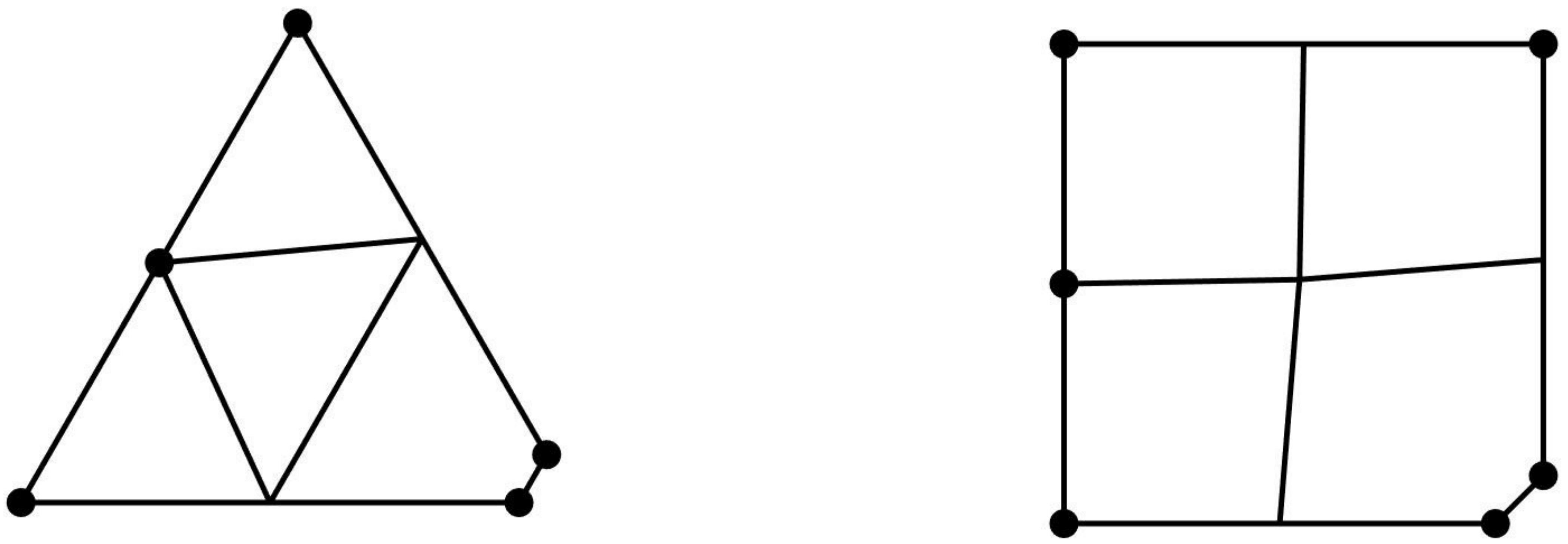
Refinement strategies for triangles and quadrilaterals.



# Refinement strategies for general polygons



# “Ideal” strategy

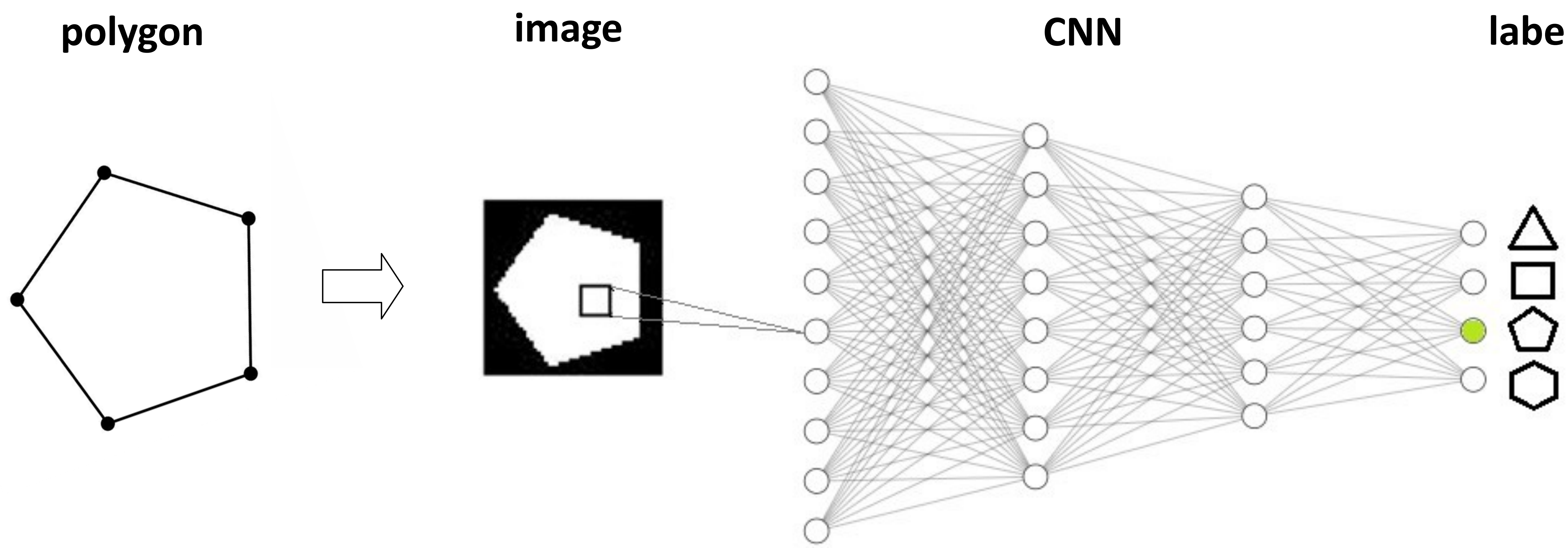


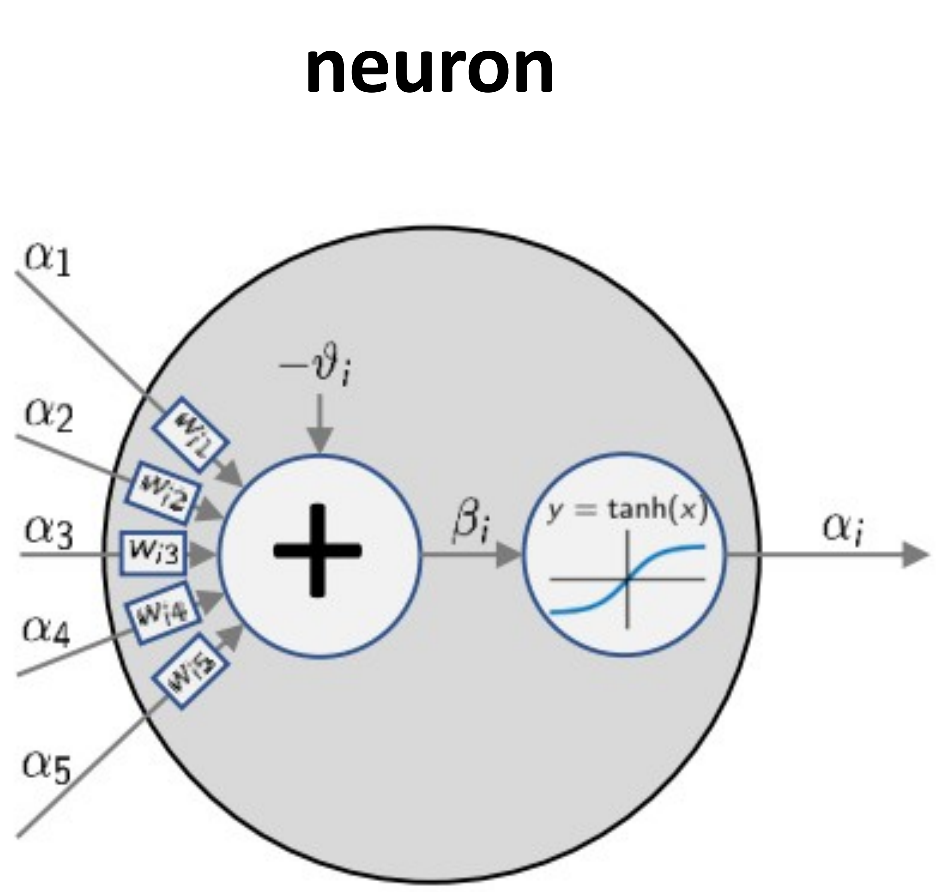
1. Classify the "**shape**" of a polygon.
2. Apply a suitable refinement for that specific shape.

Step 1 can be learned from a database of examples using Machine Learning (ML).

[Antonietti, Manuzzi, JCP, 2022]

# Image classification using Convolutional Neural Networks (CNNs)



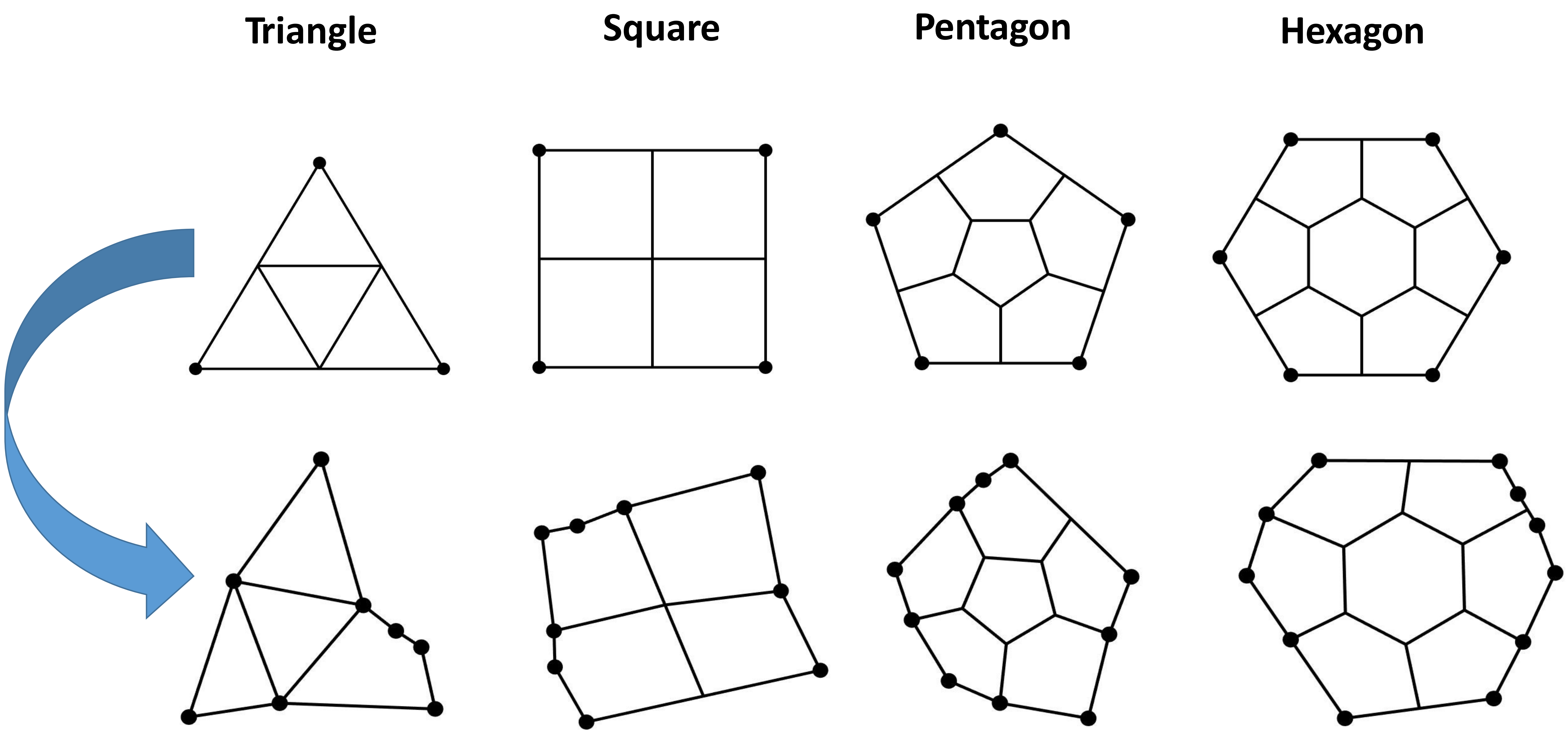




Network training is the process of tuning the neurons parameters, in order to correctly classify a given database of samples.

It is expensive but it can be done offline once and for all, while online classification is very fast.

# Algorithm 1

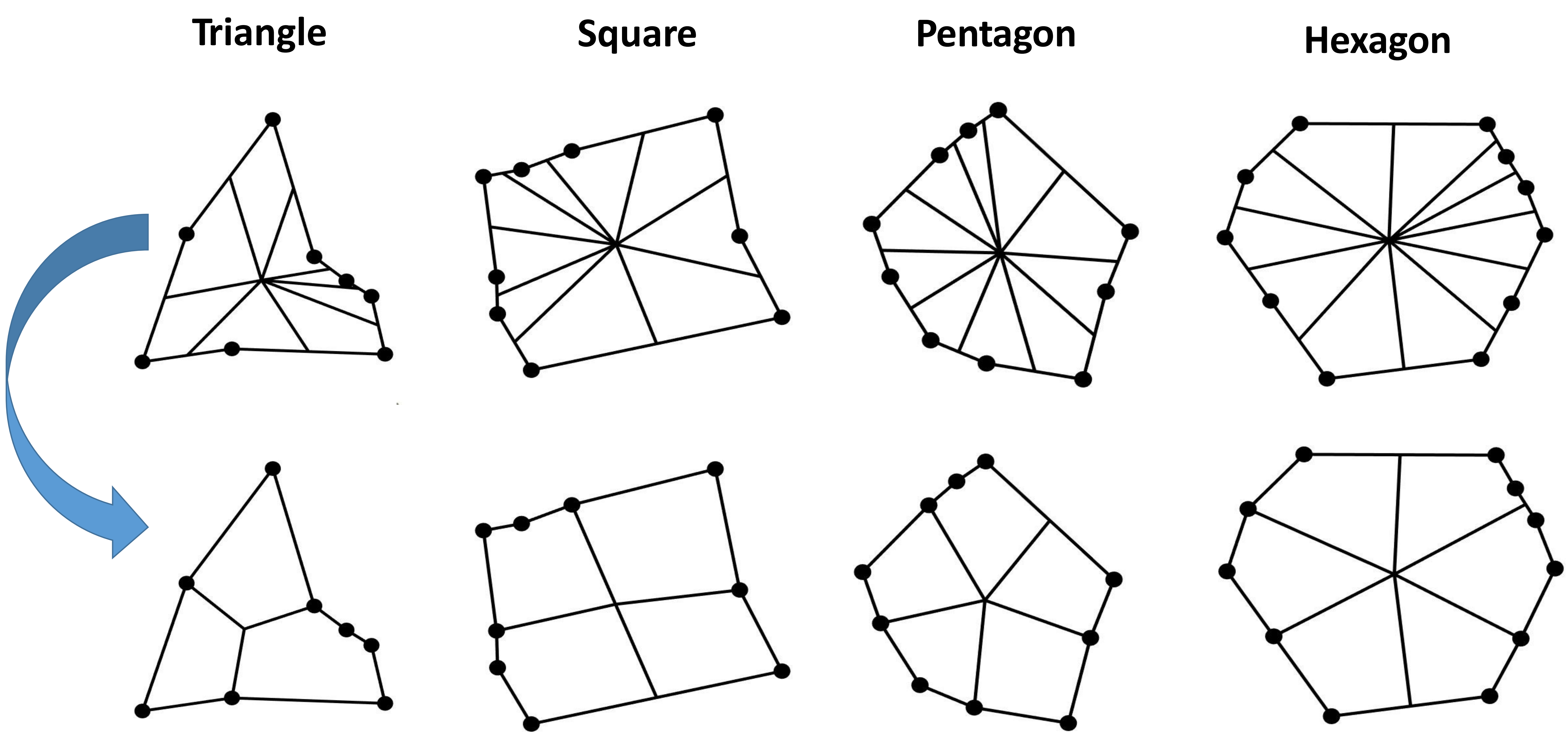
CNN-enhanced Reference Polygon (CNN-RP) strategy




Strategies for regular polygons are extended to arbitrary polygons by exploiting the CNN information about the "shape".

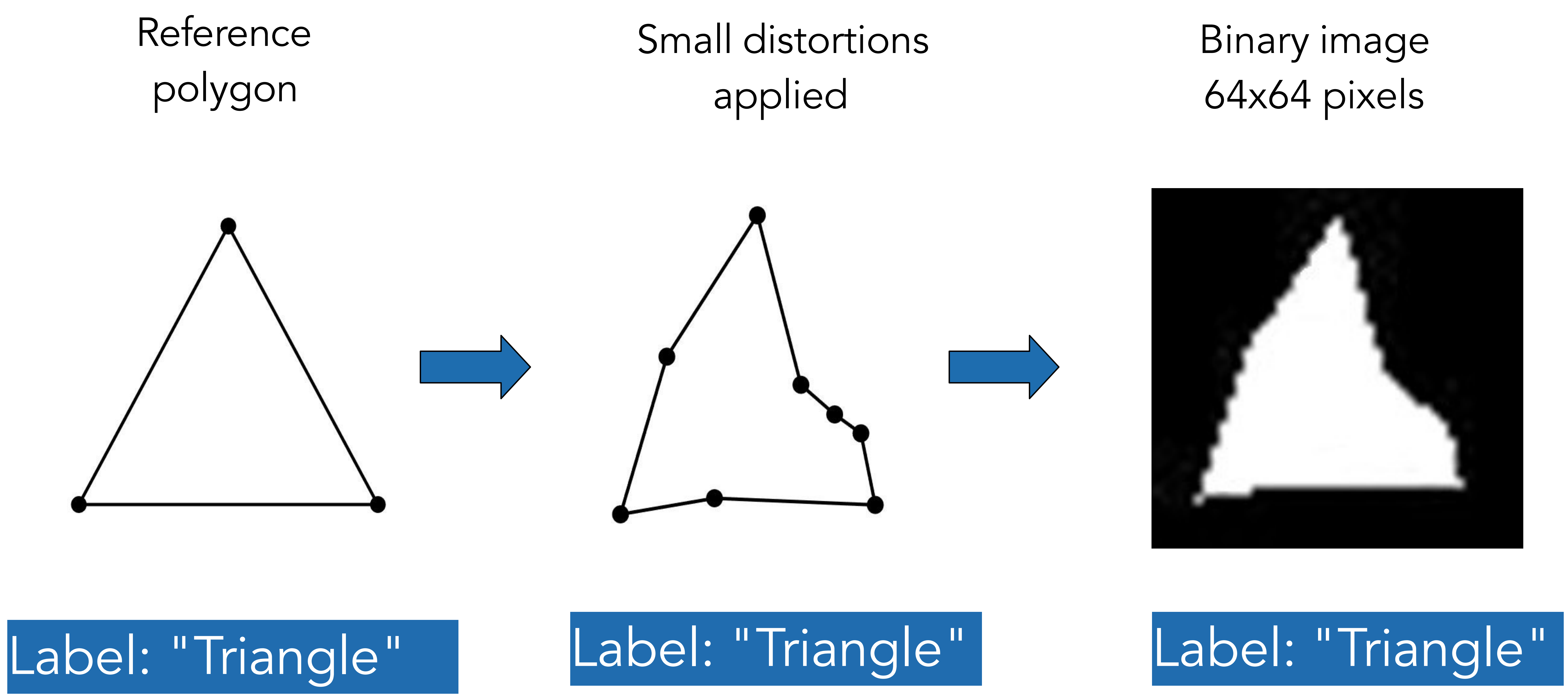
# Algorithm 2

CNN-enhanced Mid-Point (CNN-MP) strategy



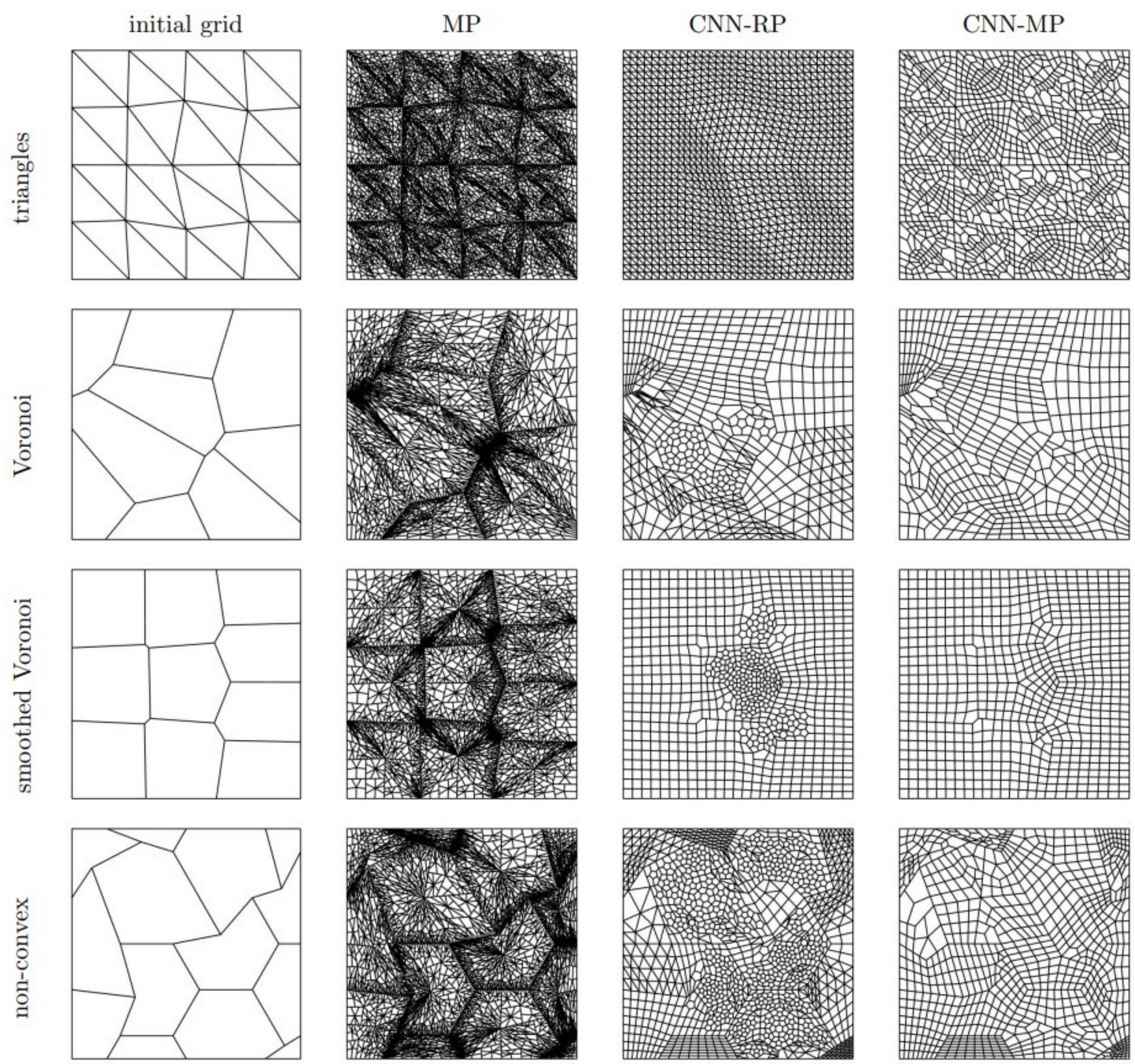

The MP strategy can be enhanced by classifying polygons using a CNN and choosing refinement connections according to the label.

# Automatic dataset generation



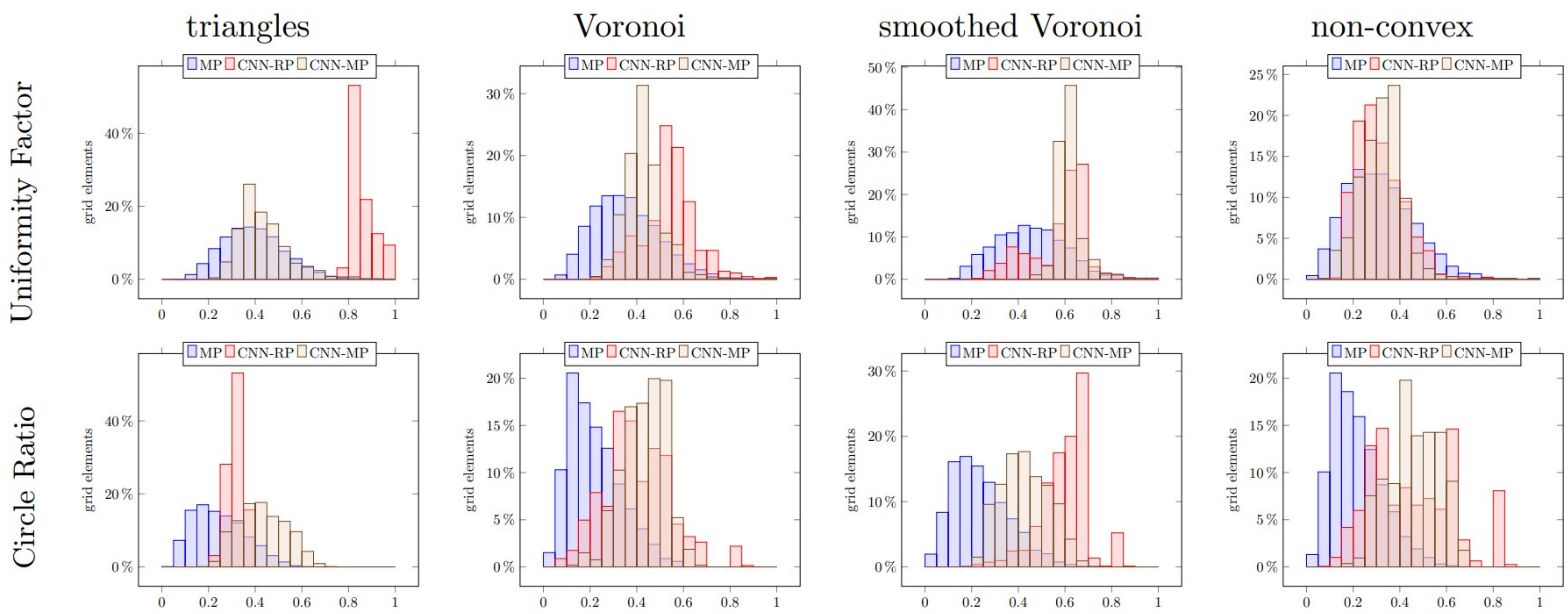
# A preliminary example

Refined grids obtained after three steps of uniform refinement based on employing the MP, the CNN-RP and the CNN-MP strategies.



# Effects on quality metrics

$$\textbf{Uniformity factor} = \frac{\text{Element size}}{\text{Mesh size}}$$



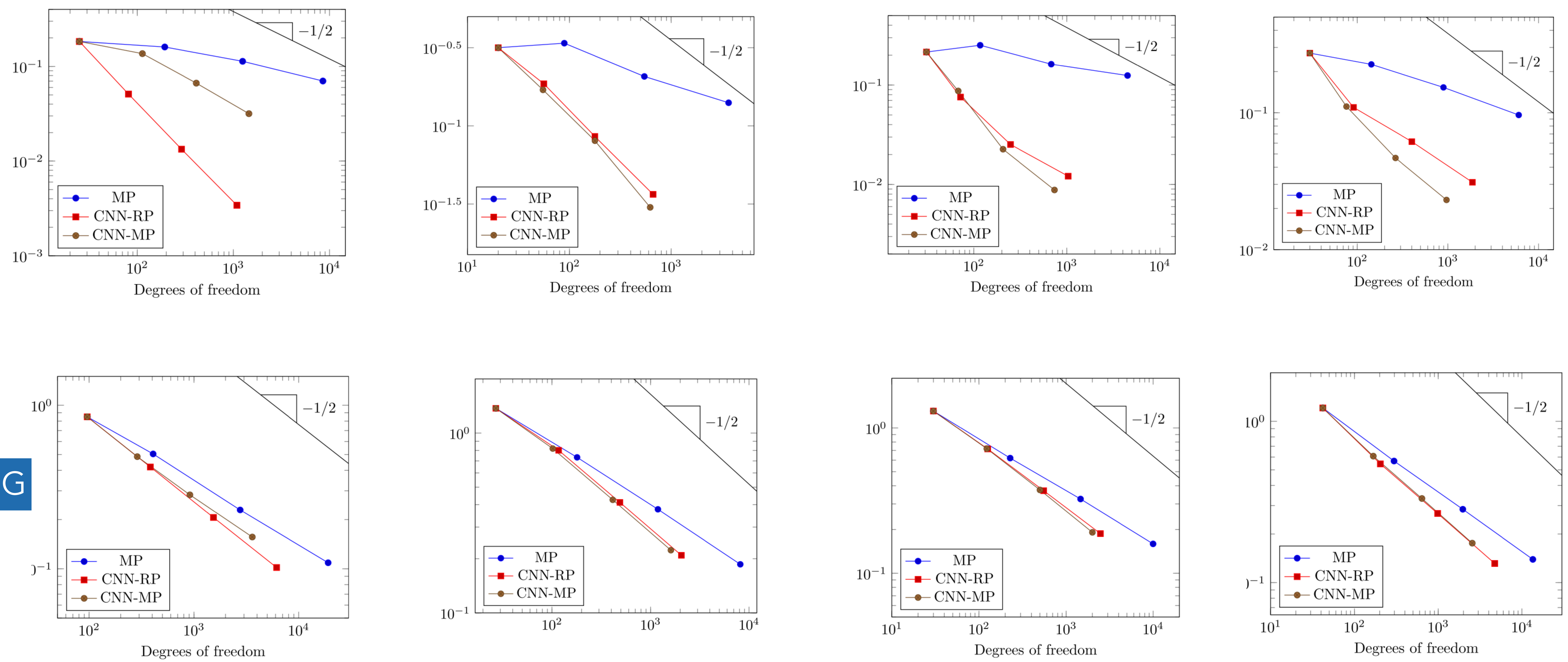

$$\textbf{Circle Ratio} = \frac{\text{Inscribed circle radius}}{\text{Circumscribed circle rario}}$$

# Solving the Poisson problem: uniform refinement

triangles | Voronoi | smoothed Voronoi | non-convex

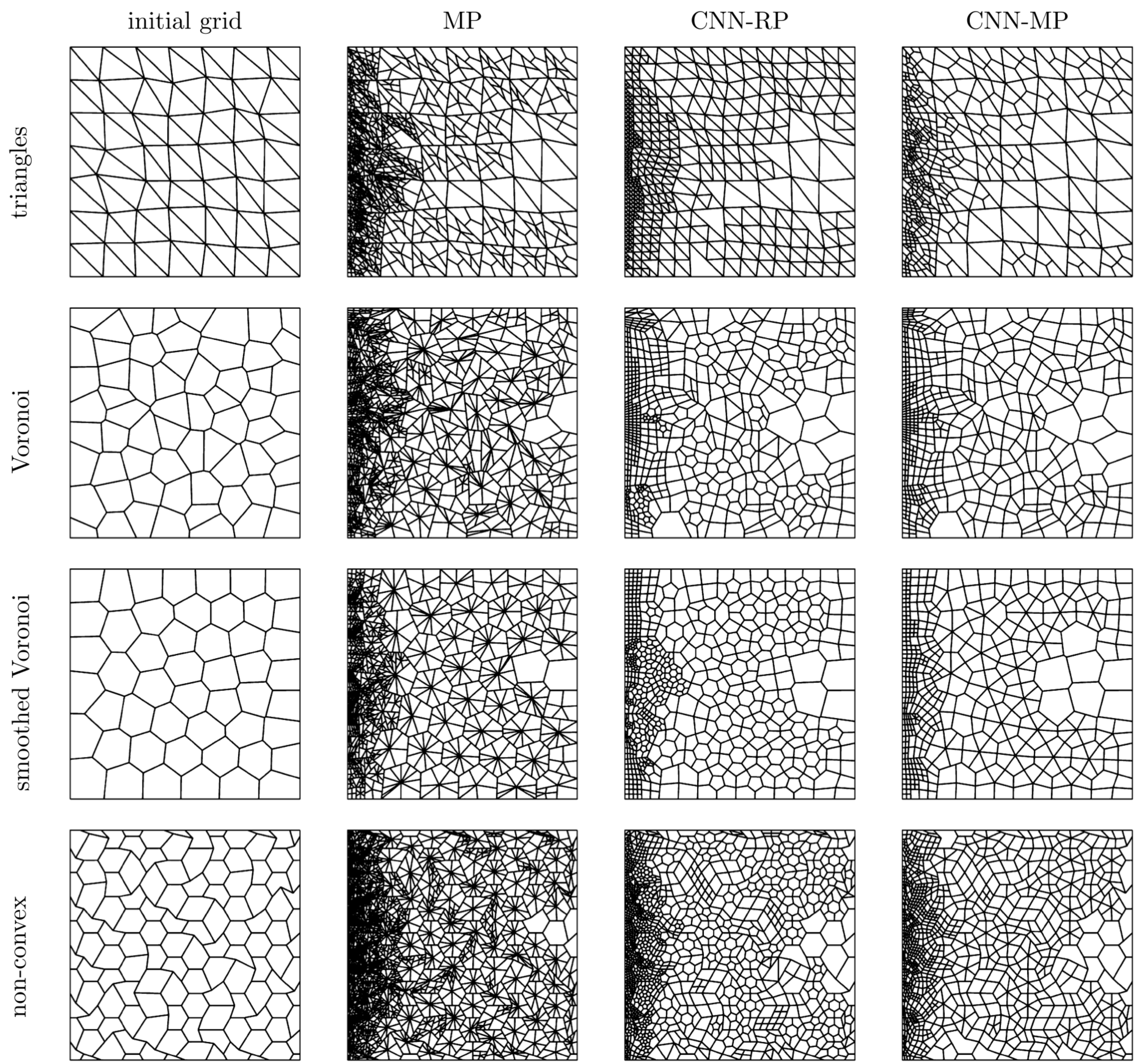
VEM

POLYDG




Analogous results for advection-diffusion and Stokes problems.

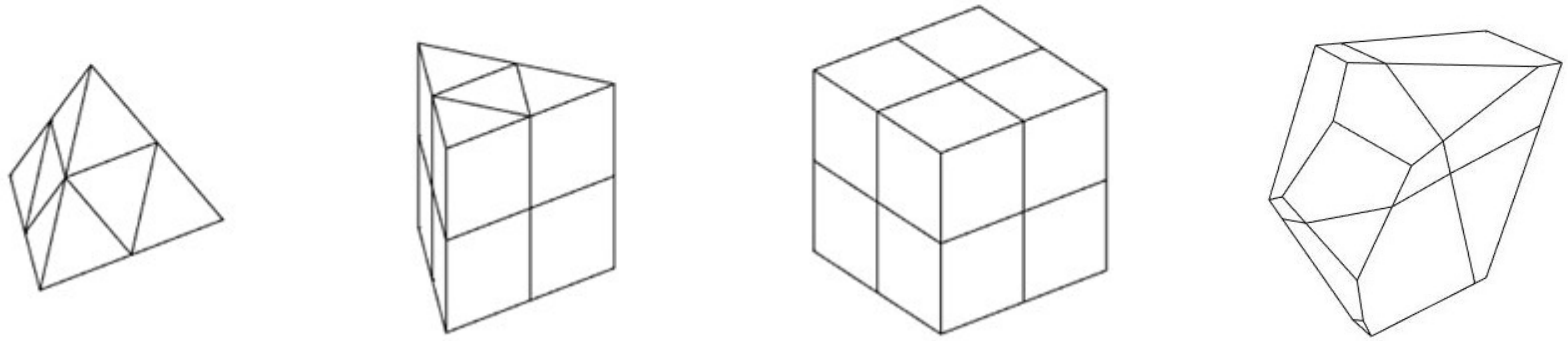
# Adaptively refined grids



# From 2D to 3D

Challenges in 3D:

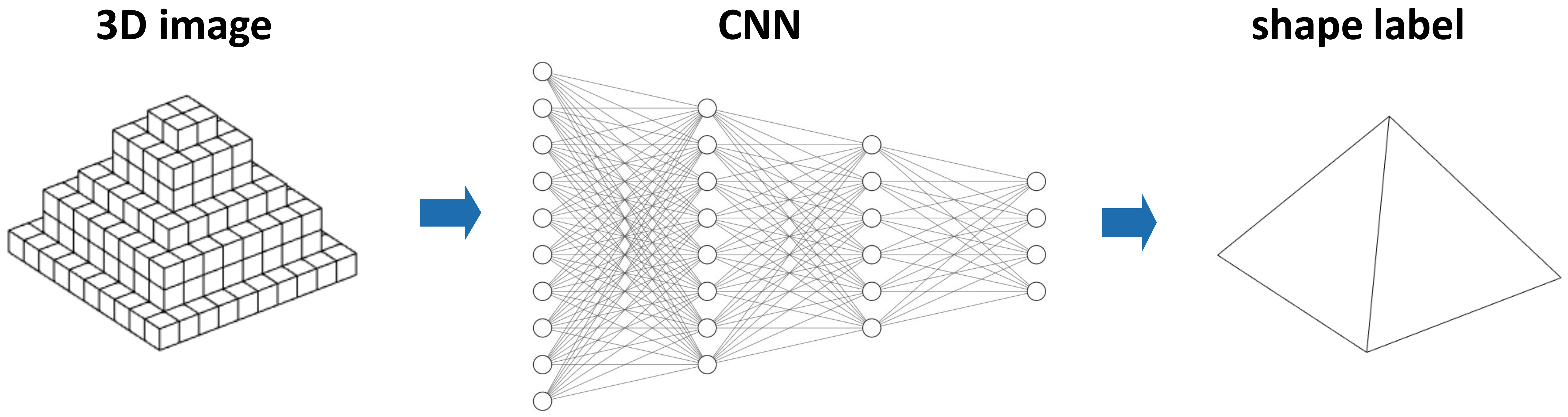
- More geometrical complexity: need to design of simple and robust refinement strategies
- high computational costs: need for fast algorithms (e.g. CNNs)
- high shape variability: need to tackle unknown shapes explicitly

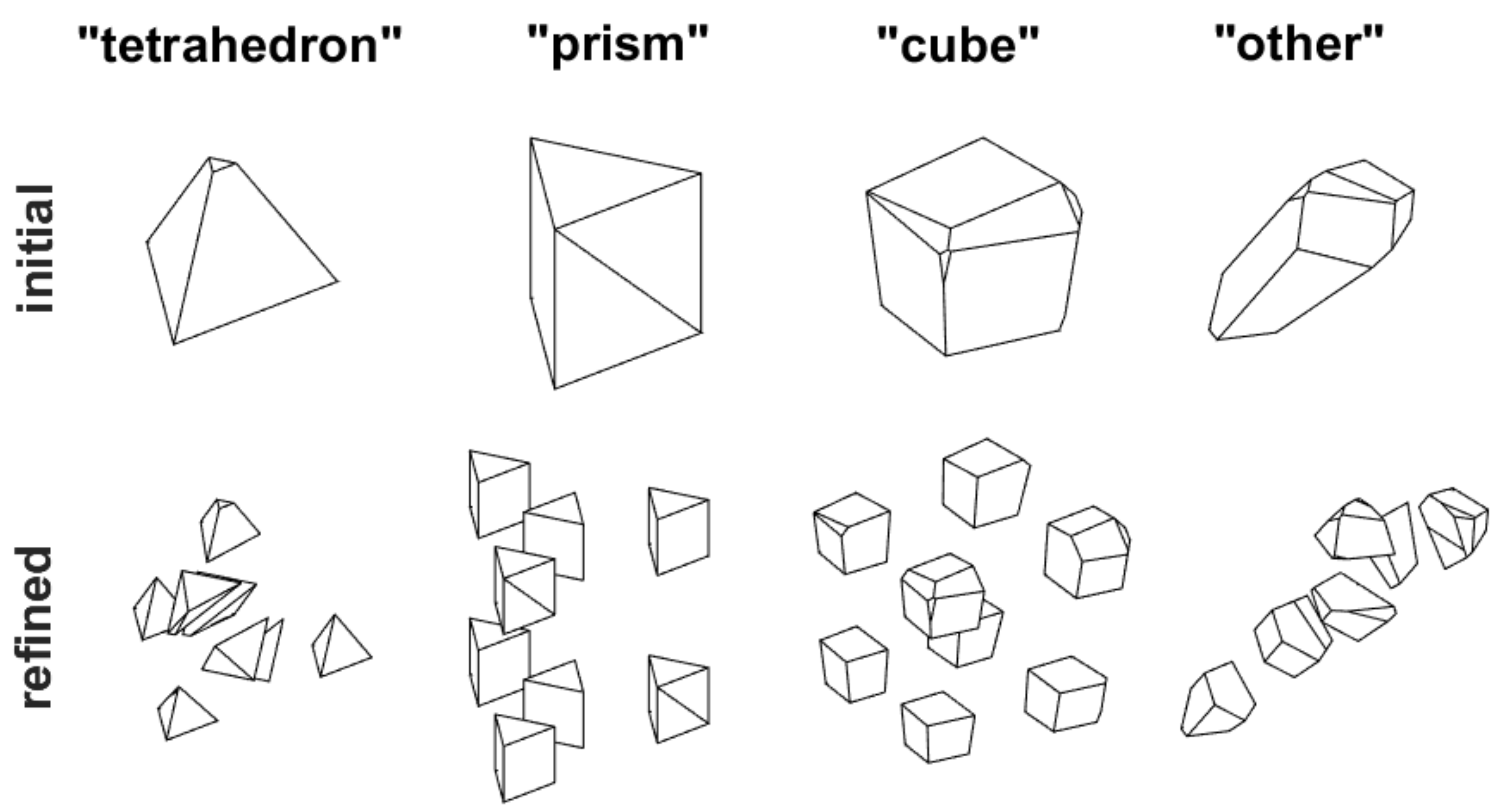


[Antonietti, Dassi, Manuzzi, JCP, 2022]

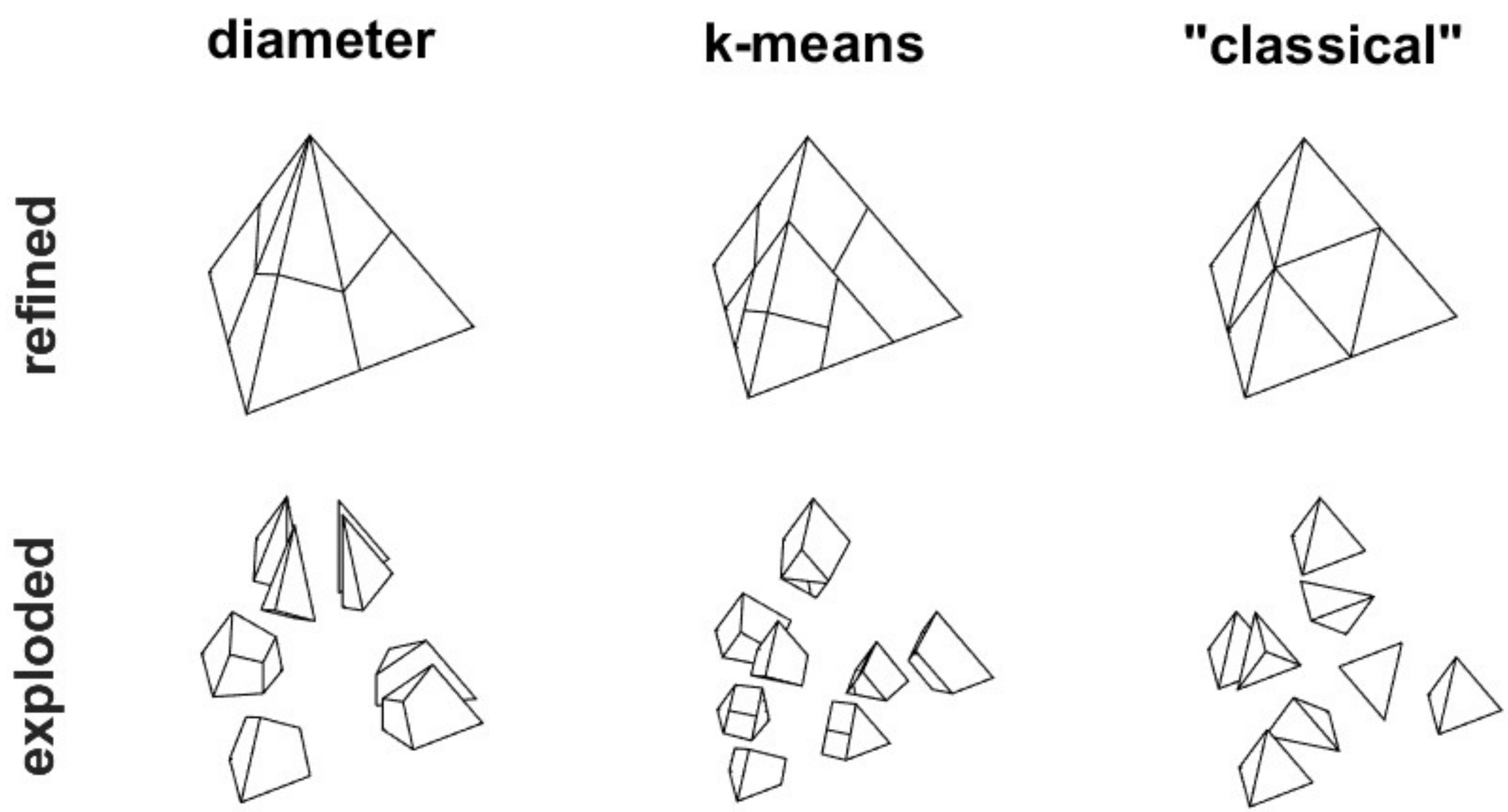
# 3D refinement strategies for general polyhedra

The **CNN** classifies the 3D image of the input polyhedron according to its shape, in order to apply suitable refinement strategies. Elements in class "other" are refined using the k-means strategy.



# ML-enhanced mesh refinement (3D)

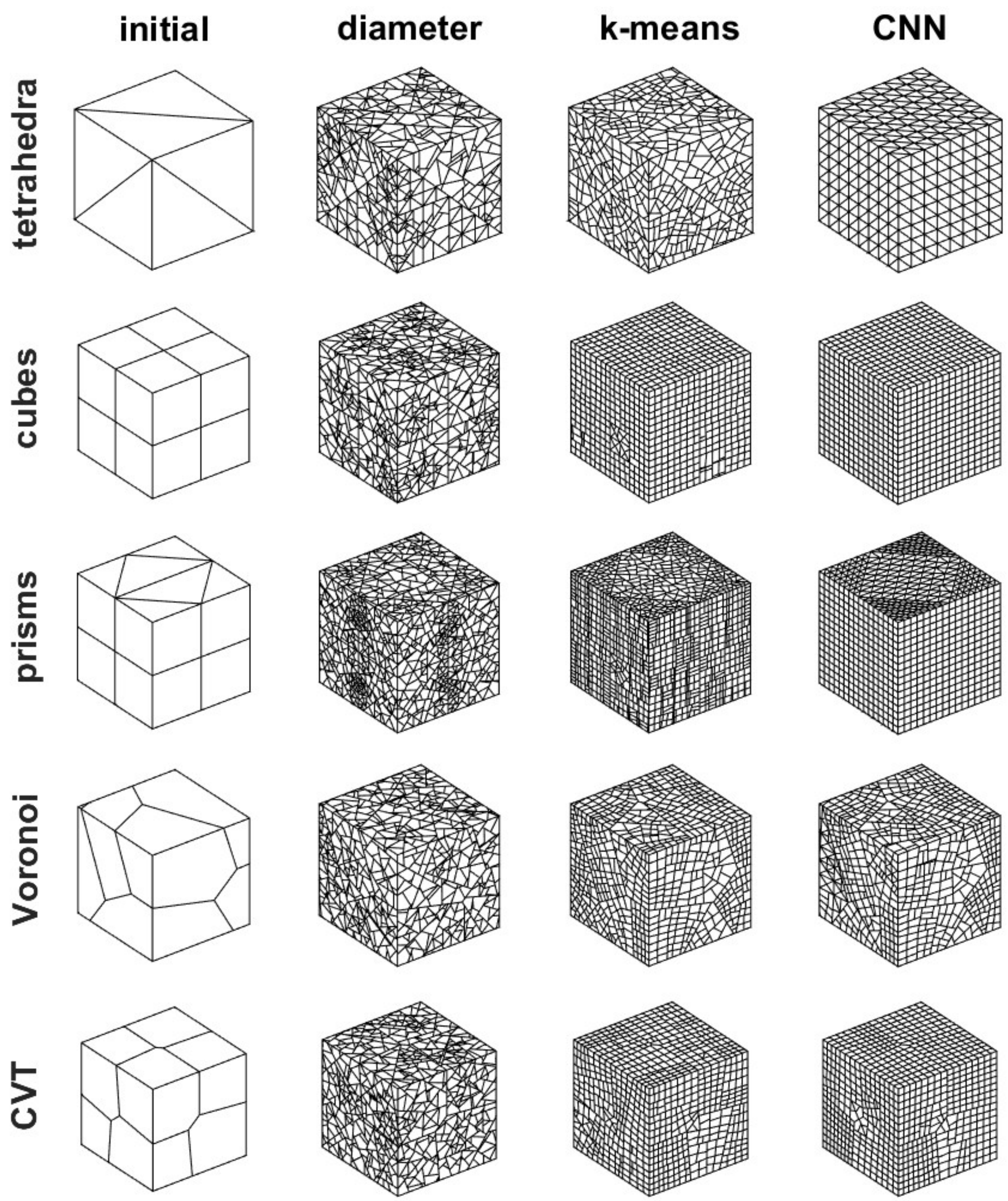



**Strategy**
Cut the element perpendicular to its diameter

**Strategy**
Cut the element balancing the volume distribution

**Strategy**
If the element has a specific shape refine it using a predefined strategy

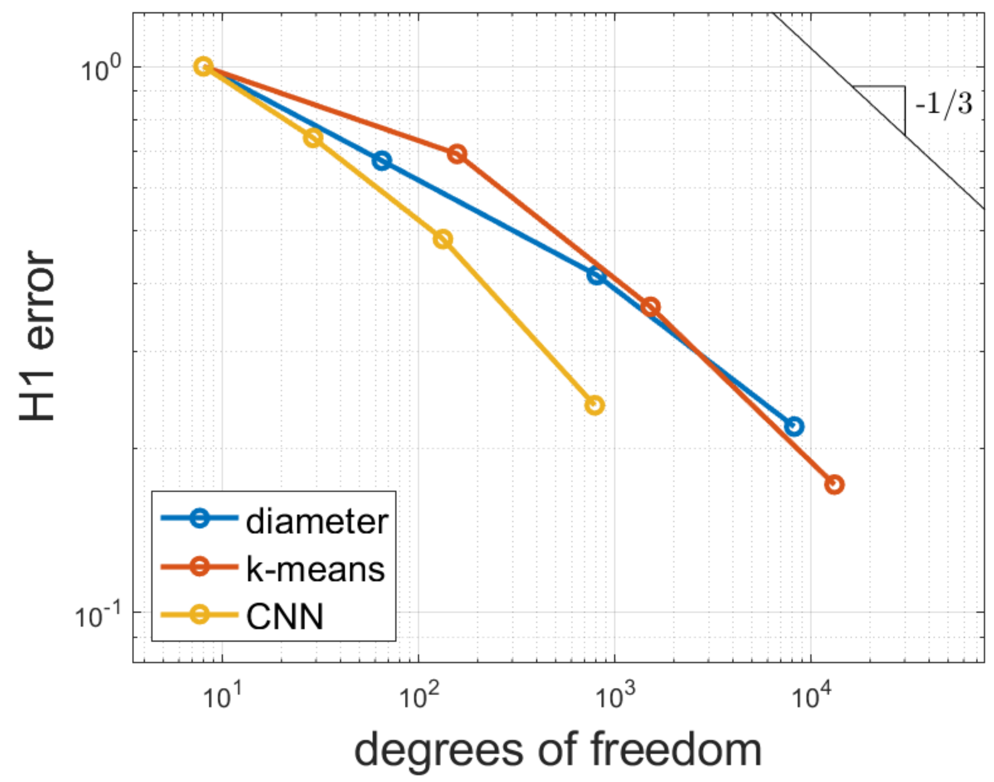
# Some examples



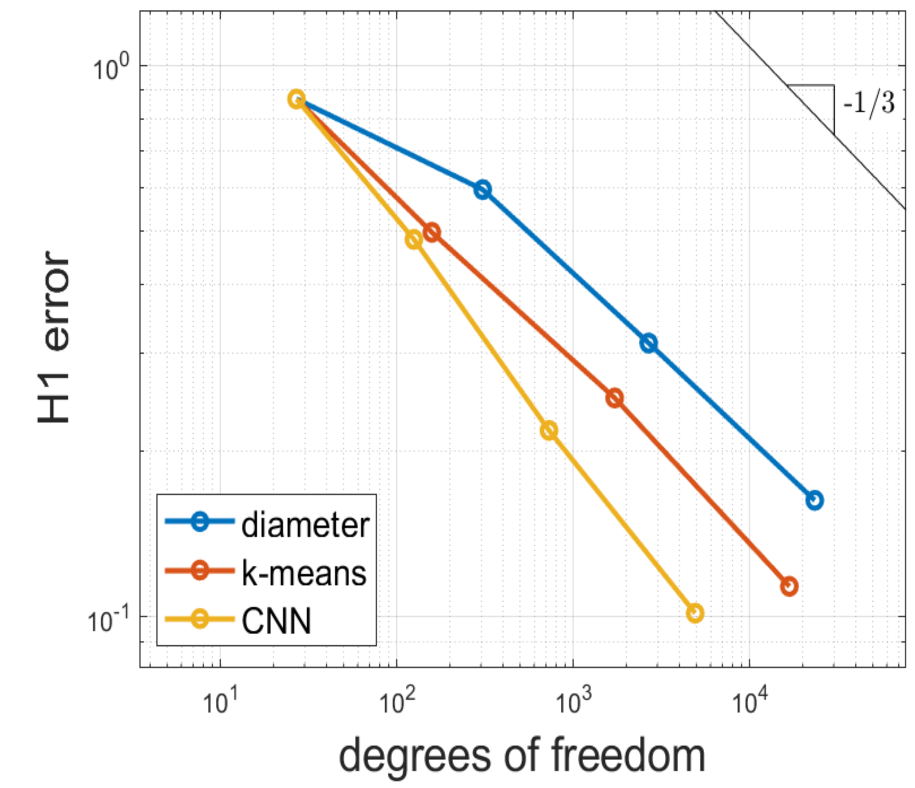

Refined grids obtained after three steps of uniform refinement based on employing the diameter, the k-means and the CNN strategies.

# Solving the Poisson problem with the VEM (3D)

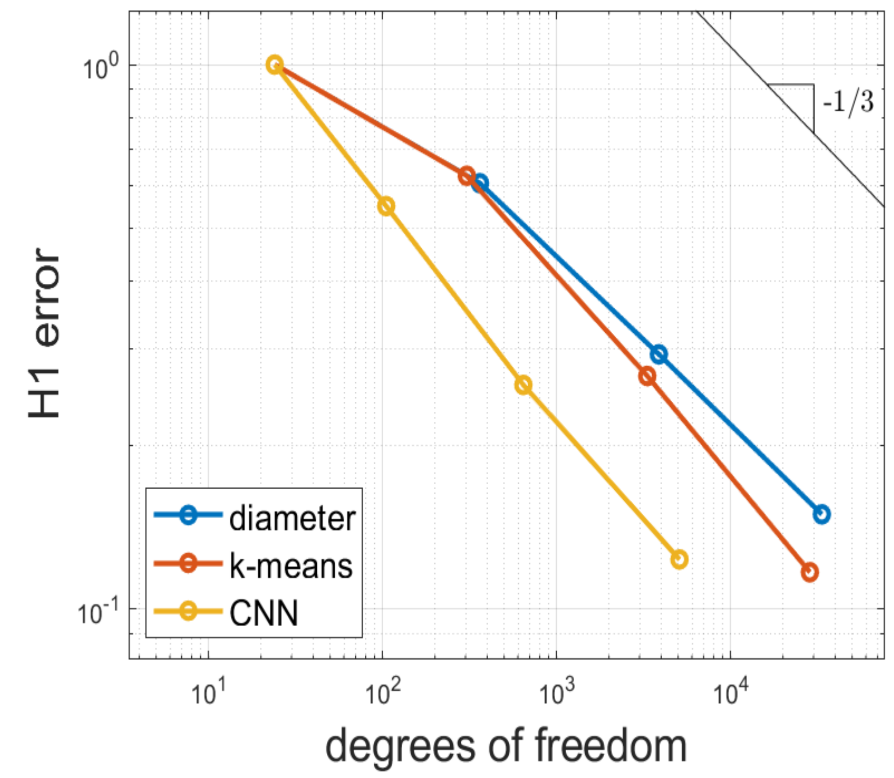
**Tetrahedra**



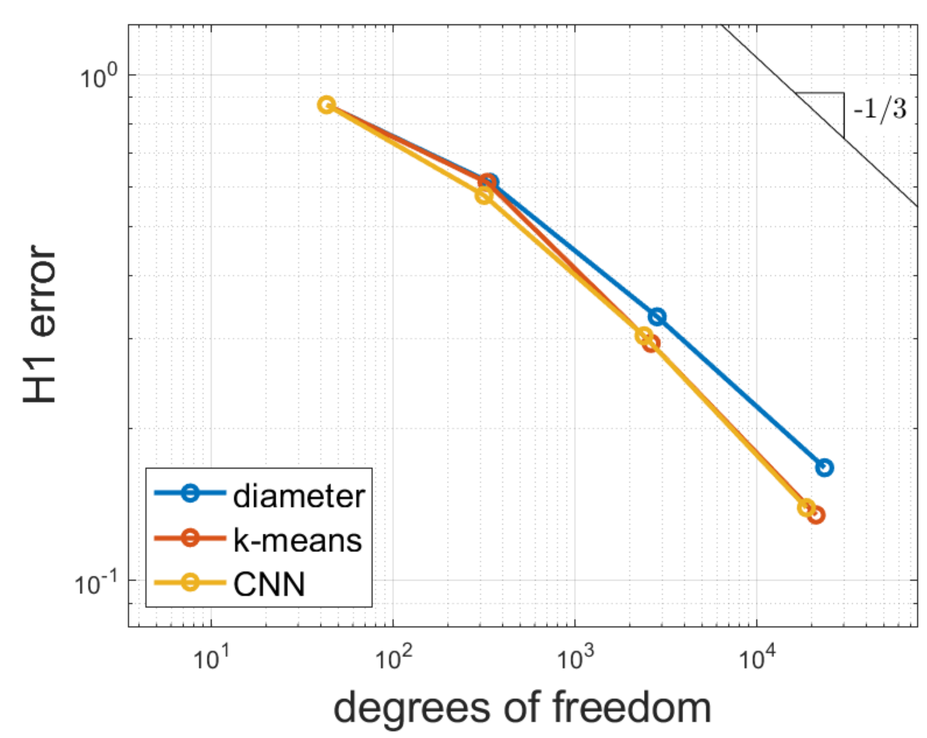

**Cubes**



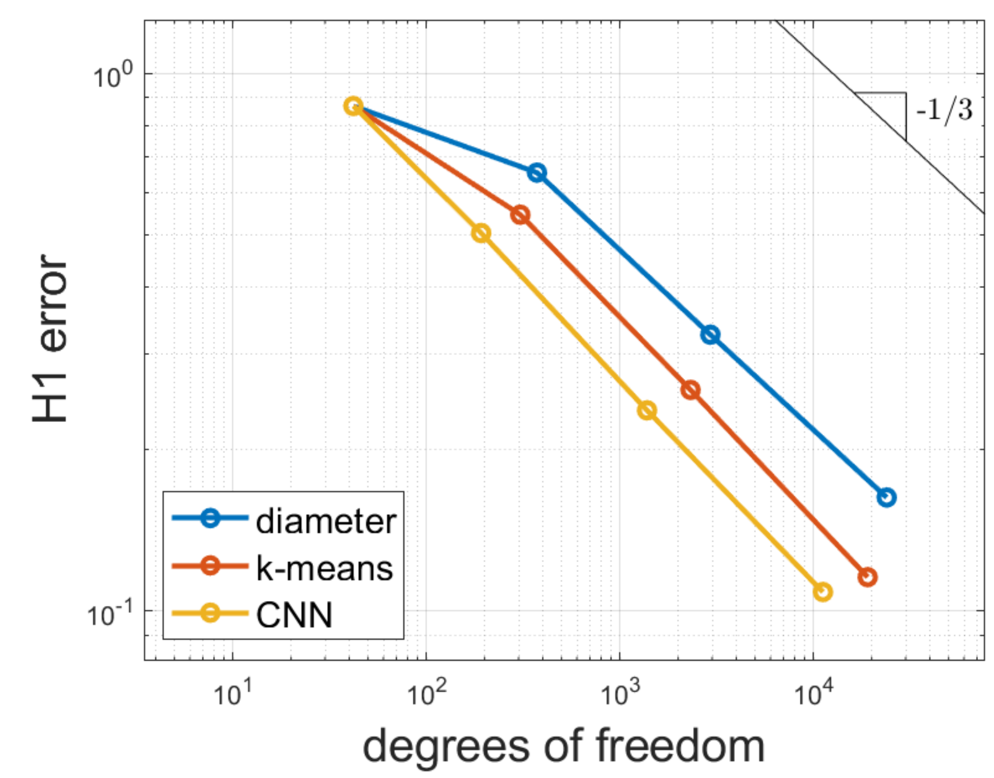

**Prisms**




**Voronoi**

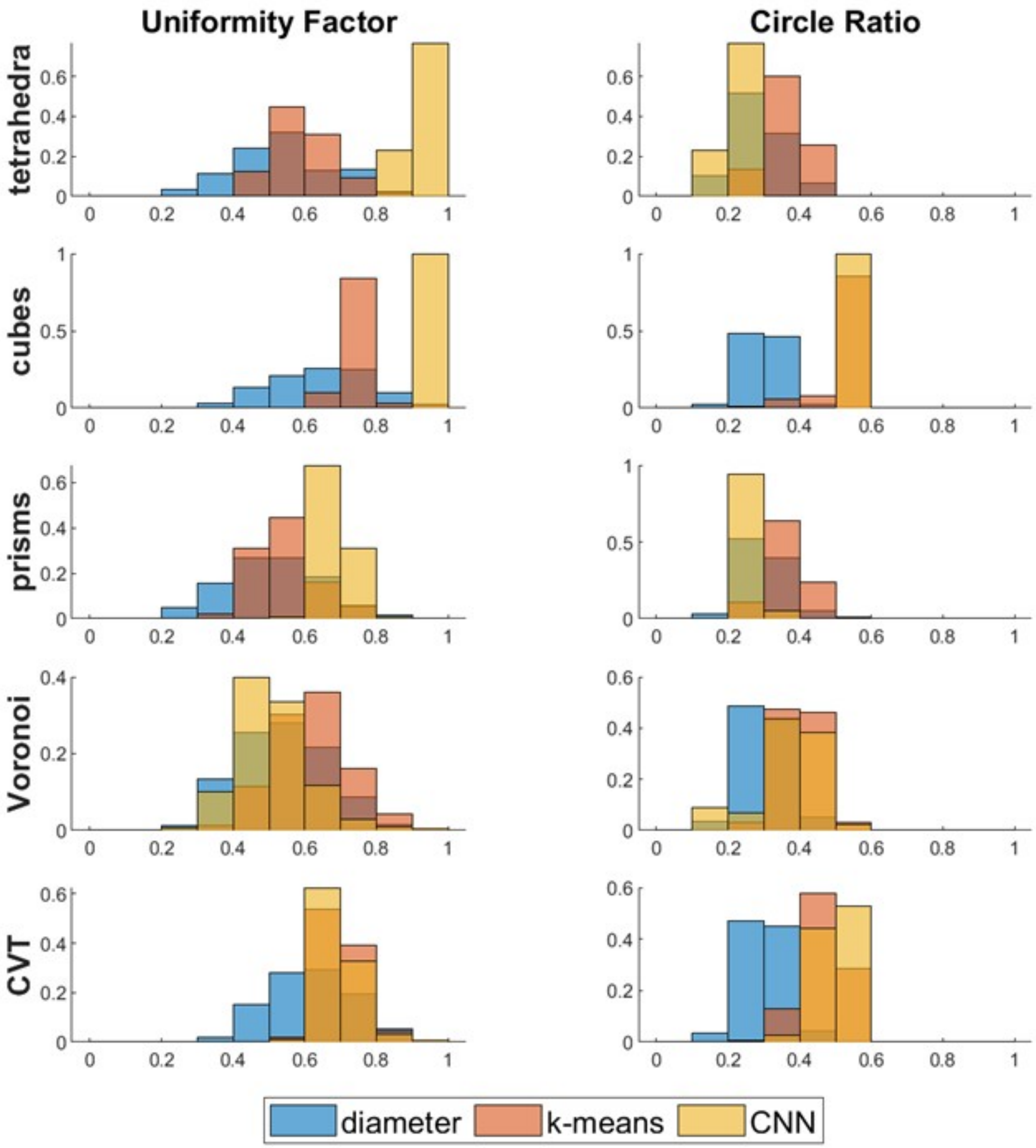



**CVT**




Analogous results for the VEM of order higher than 1.

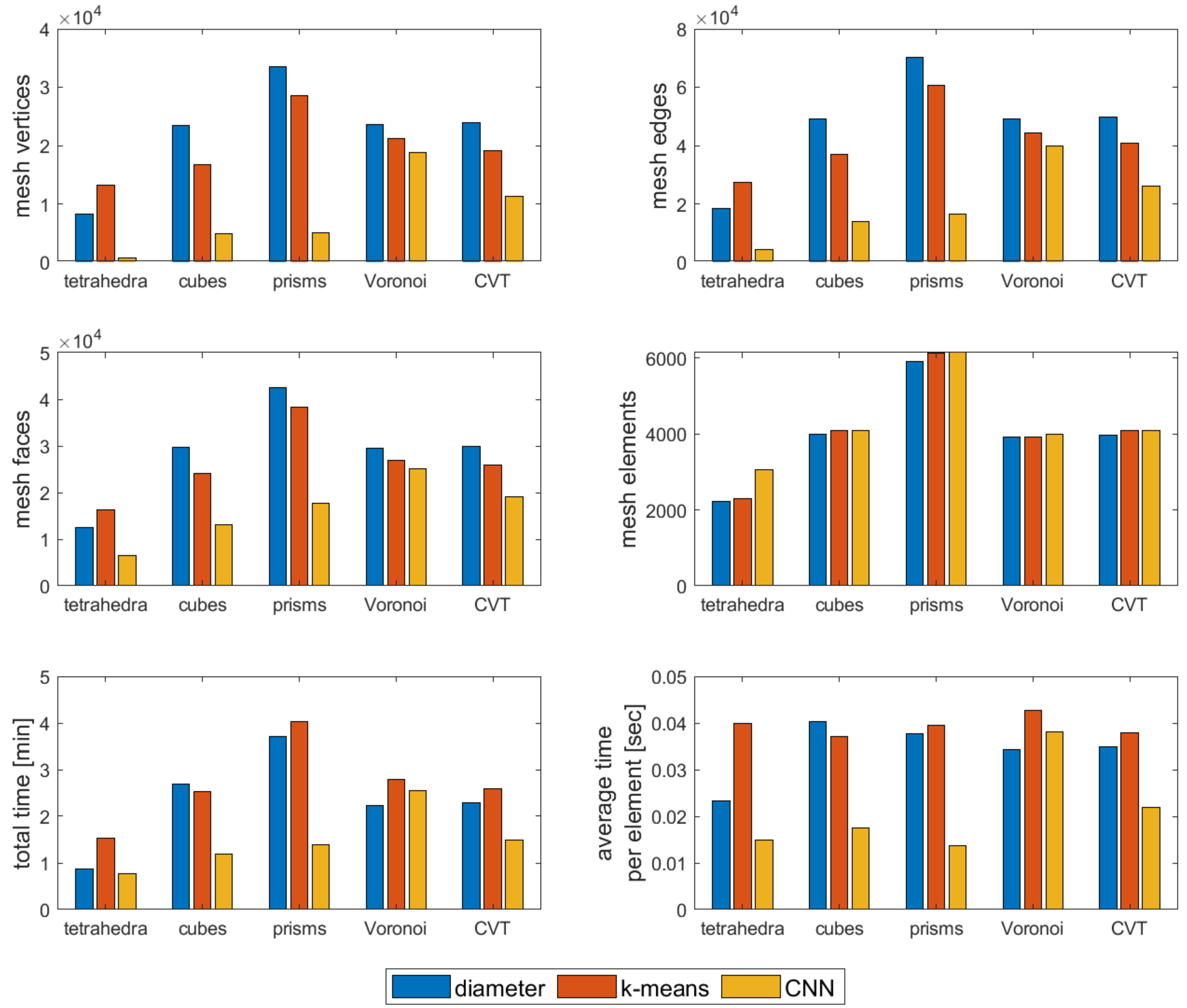
# Quality Metrics




$$\textbf{Uniformity factor} = \frac{\text{Element size}}{\text{Mesh size}}$$

$$\textbf{Circle Ratio} = \frac{\text{Inscribed circle radius}}{\text{Circumscribed circle rario}}$$

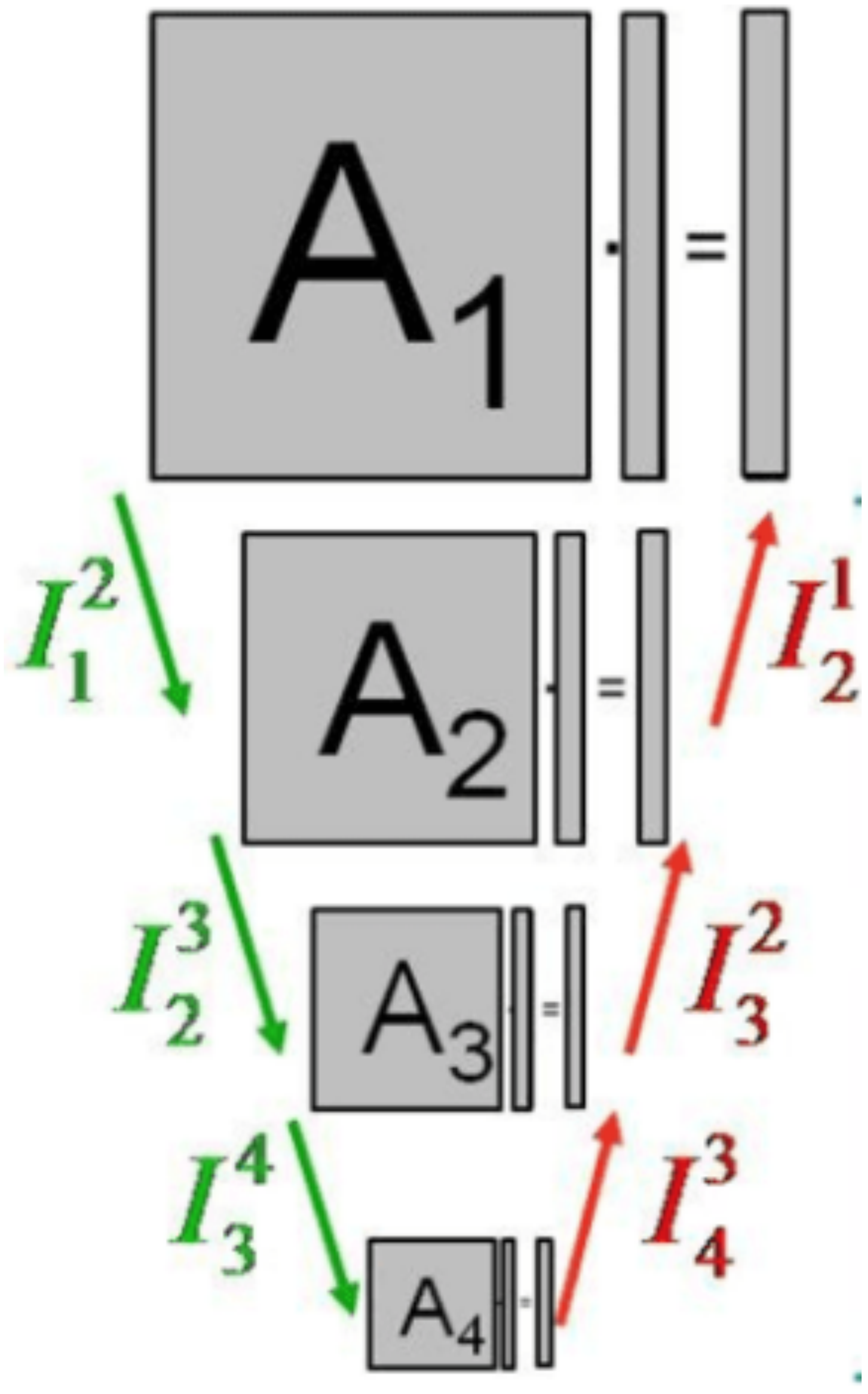
# Effects on statistics of computational complexity

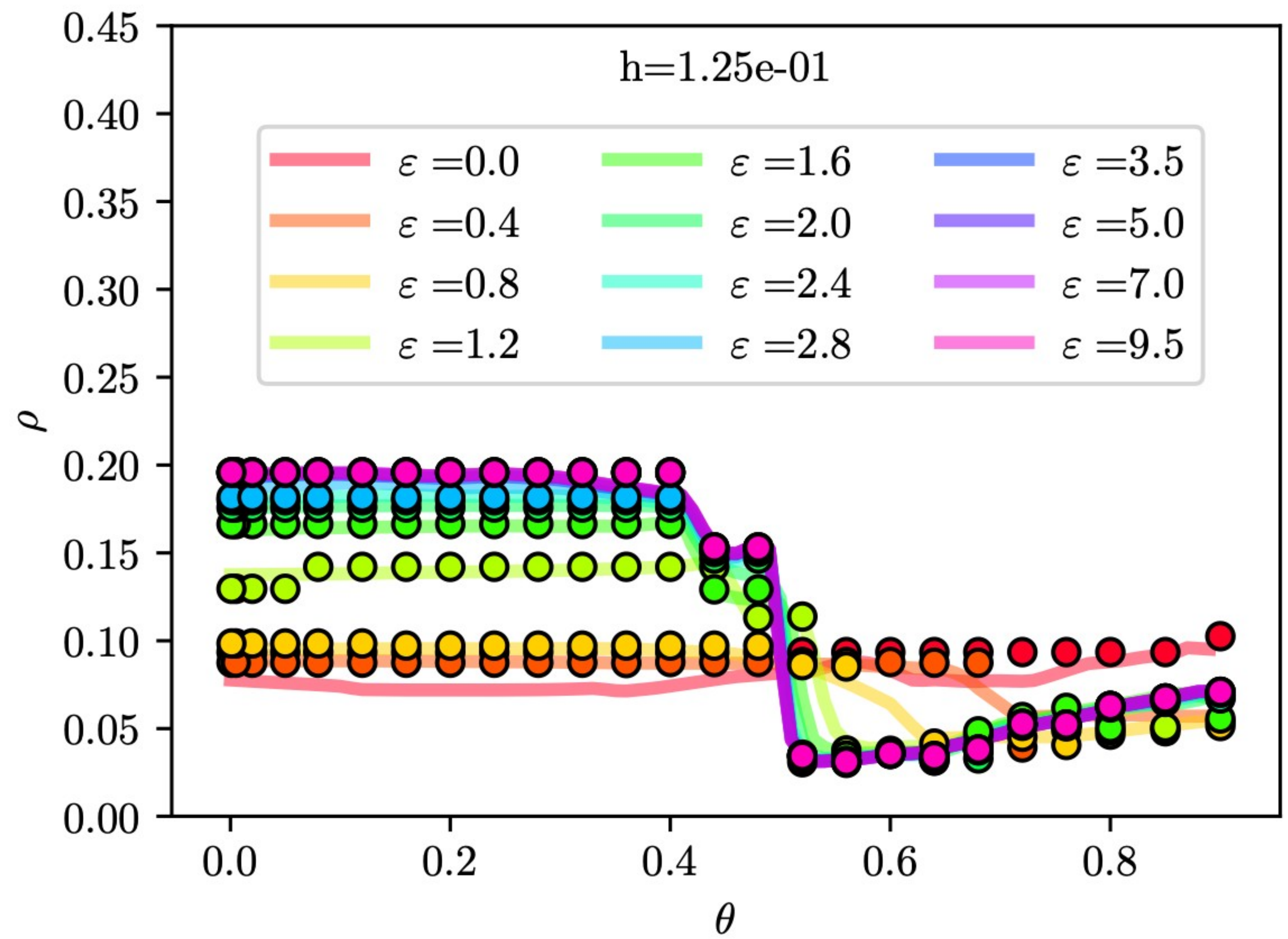


# ML-enached Algebraic Multigrid Methods

Joint work with M. Caldana, L. Dedè

# ML-enhanced Algebraic Multigrid Methods




Improving efficiency of algebraic multigrid methods through artificial neural networks: choosing the strong threshold parameter $\theta$ as the one the ANN predicts to give the best performance (30 % gain)



[P.F. Antonietti, L. Dede', M. Caldana, 2023 & 2024]

# Conclusions

The synergy between machine learning and polytopal methods presents a promising avenue.

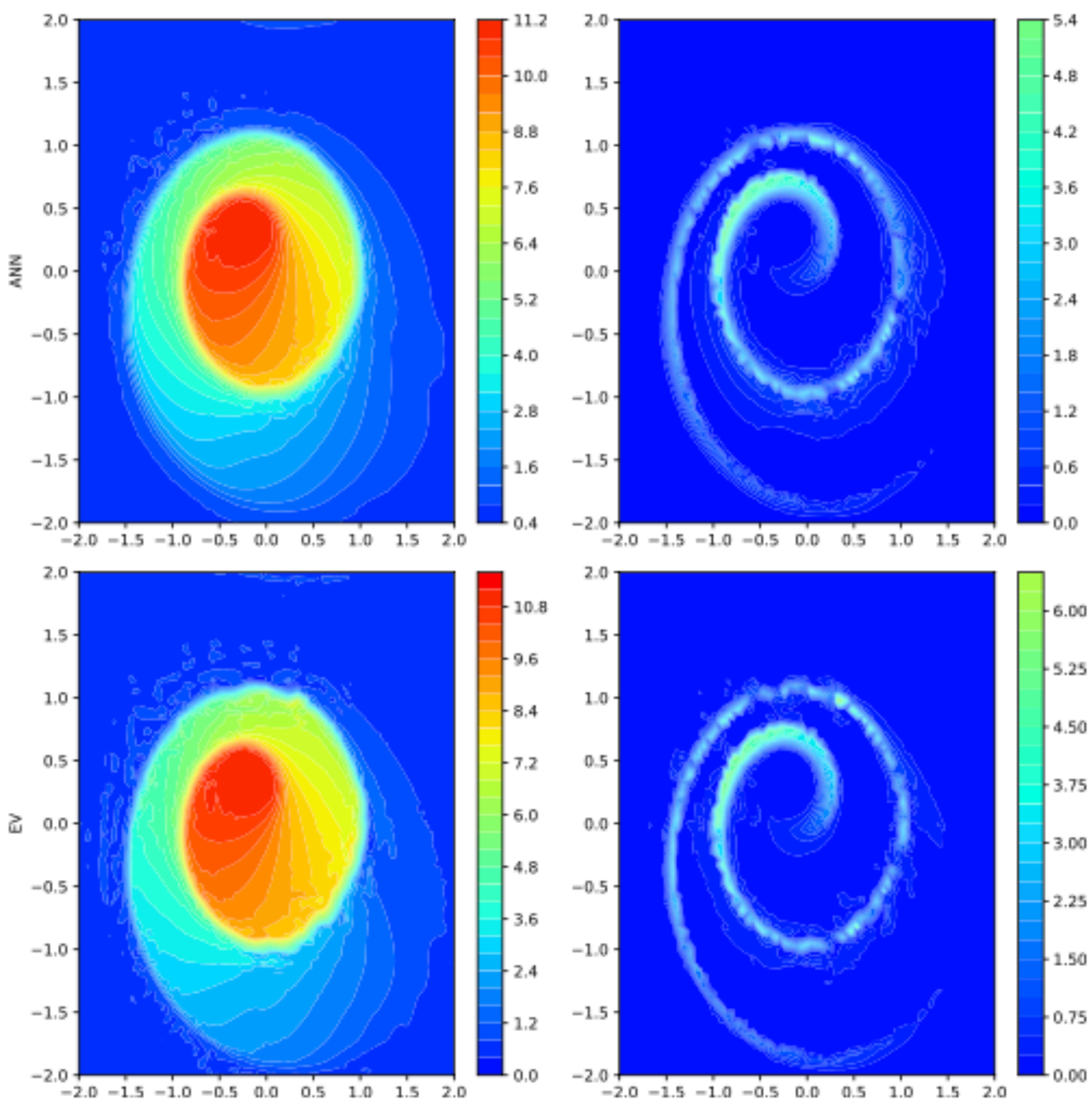
By leveraging the strengths of ML paradigms, we can enhance the capabilities of polytopal methods, leading to more efficient and effective solutions to multiphysics problems in heterogeneous domains.

- Allows to extend or boost existing (refinement/agglomeration) strategies.
- Improves the performance in terms of accuracy.
- It is fully automatic and it has a low (online) computational cost
- It is independent of the underlying differential model and the numerical method used.
- It stands at the basis of the design and acceleration of algebraic solvers
- Defeaturing complex geometries

lymph

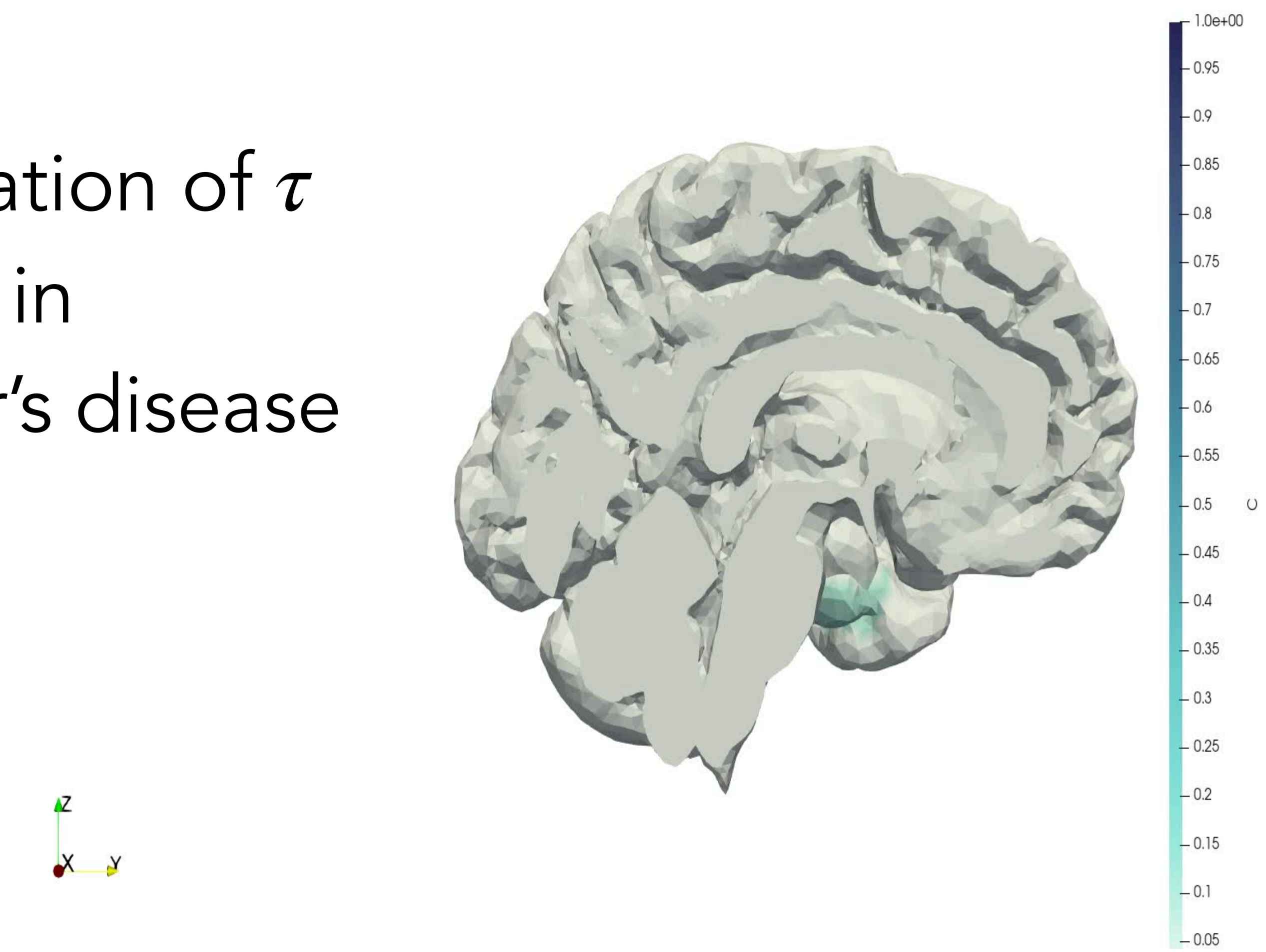
# Ongoing

Learning "the stabilisation"
(e.g., artificial viscosity)

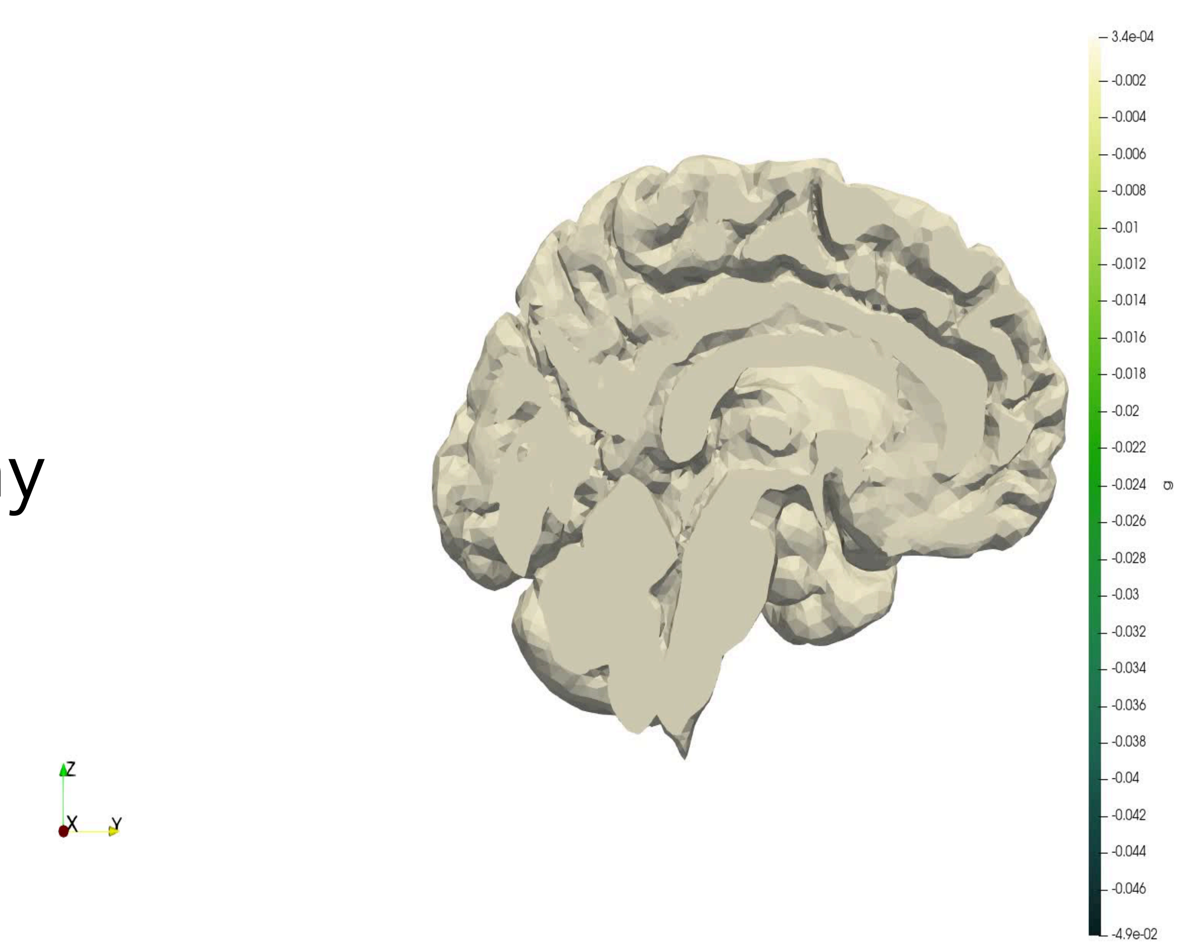



Learning "the physics"
(e.g., constitutive laws)

Concentration of $\tau$ -proteins' in Alzheimer's disease



Atrophy



# Thank you for your attention!

This work received funding by the European Union:
**ERC SyG**, **NEMESIS**, project number 101115663
**https://erc-nemesis.eu/**

Funded by the European Union

European Research Council
Established by the European Commission