Graph Convolutional Networks and some applications

Corentin MENGEL,
under the supervision of Vincent VIGON, Emmanuel FRANCK,
Laurent NAVORET and Laurène HUME

August 24, 2021

Introduction

continuation of the previous project

Introduction

continuation of the previous project

GCNs achieve good results even after modifications of the graph

Introduction

continuation of the previous project

GCNs achieve good results even after modifications of the graph

develop a new model and use it in two problems:

Introduction

continuation of the previous project

GCNs achieve good results even after modifications of the graph

develop a new model and use it in two problems:

- discontinuities detection and Burgers' equation

Introduction

continuation of the previous project

GCNs achieve good results even after modifications of the graph

develop a new model and use it in two problems:

- discontinuities detection and Burgers' equation
- interpolation problem and linear transport equation

Definitions

Definitions: Graph Convolutional Networks

Definitions: Graph Convolutional Networks

GCN: sequence of layers put one after the other

Definitions: Graph Convolutional Networks

GCN: sequence of layers put one after the other

graph convolutional layer:

- takes d dimensional node features as input

Definitions: Graph Convolutional Networks

GCN: sequence of layers put one after the other

graph convolutional layer:

- takes d dimensional node features as input
- computes d' dimensional representations of the nodes

Definitions: Graph Convolutional Networks

GCN: sequence of layers put one after the other

graph convolutional layer:

- takes d dimensional node features as input
- computes d' dimensional representations of the nodes
- uses recursive neighborhood diffusion and message passing

Definitions: Graph Convolutional Networks

GCN: sequence of layers put one after the other

graph convolutional layer:

- takes d dimensional node features as input
- computes d' dimensional representations of the nodes
- uses recursive neighborhood diffusion and message passing
- each graph node gathers features from its neighbors

Definitions: pooling layers

inconvenient: convolutional layers do not change the mesh structure

Definitions: pooling layers

inconvenient: convolutional layers do not change the mesh structure

new layers reducing the graph resolutions

Definitions: pooling layers

inconvenient: convolutional layers do not change the mesh structure

new layers reducing the graph resolutions

enlarge receptive field for better performance and generalization

Definitions: pooling layers

inconvenient: convolutional layers do not change the mesh structure

new layers reducing the graph resolutions

enlarge receptive field for better performance and generalization

$\Rightarrow$ pooling layers

Definitions: Top-k pooling

inputs: mesh Ω, nodes features X, integer k,
output: new mesh with k nodes

Definitions: Top-k pooling

inputs: mesh Ω, nodes features X, integer k,
output: new mesh with k nodes

selects subset of nodes to form a smaller graph

Definitions: Top-k pooling

inputs: mesh Ω, nodes features X, integer k,
output: new mesh with k nodes

selects subset of nodes to form a smaller graph

a score $y_i \in \mathbb{R}$ is associated to each node n_i of Ω

Definitions: Top-k pooling

inputs: mesh Ω, nodes features X, integer k,
output: new mesh with k nodes

selects subset of nodes to form a smaller graph

a score $y_i \in \mathbb{R}$ is associated to each node n_i of Ω

$$y_i = X_i \cdot p / \|p\|$$

p trainable vector

Definitions: Top-k pooling

inputs: mesh Ω, nodes features X, integer k,
output: new mesh with k nodes

selects subset of nodes to form a smaller graph

a score $y_i \in \mathbb{R}$ is associated to each node n_i of Ω

$$y_i = X_i \cdot p / \|p\|$$

p trainable vector

the new mesh has the k nodes with the highest score

Top-k pooling example

X_i = node position, $p = (1, 1)$

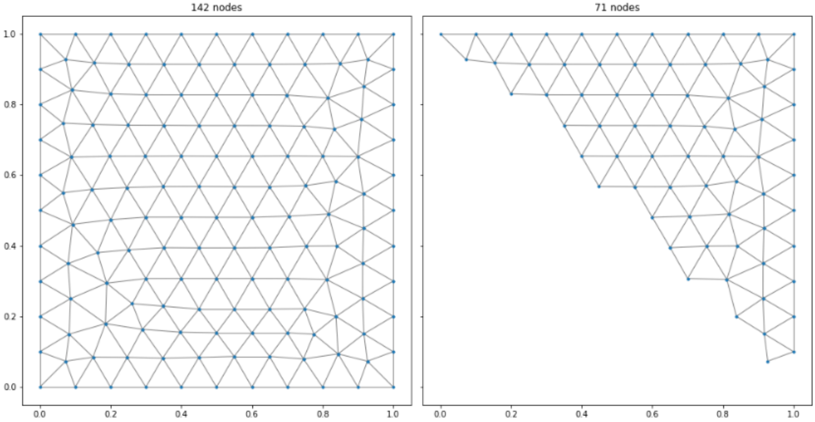


Figure: Initial mesh (left) and pooled mesh (right).

Definitions: k-Means pooling

pooling based on the k-Means clustering algorithm

Definitions: k-Means pooling

pooling based on the k-Means clustering algorithm

inputs: mesh Ω, integer k, output: new mesh with k nodes

Definitions: k-Means pooling

pooling based on the k-Means clustering algorithm

inputs: mesh Ω, integer k, output: new mesh with k nodes

- compute k clusters of the nodes of Ω

Definitions: k-Means pooling

pooling based on the k-Means clustering algorithm

inputs: mesh Ω, integer k, output: new mesh with k nodes

- compute k clusters of the nodes of Ω
- center of the k clusters $\rightarrow$ nodes of the new mesh

Definitions: k-Means pooling

pooling based on the k-Means clustering algorithm

inputs: mesh Ω, integer k, output: new mesh with k nodes

- compute k clusters of the nodes of Ω
- center of the k clusters $\rightarrow$ nodes of the new mesh
- new node features $=$ average of the node features in the clusters

k-Means example

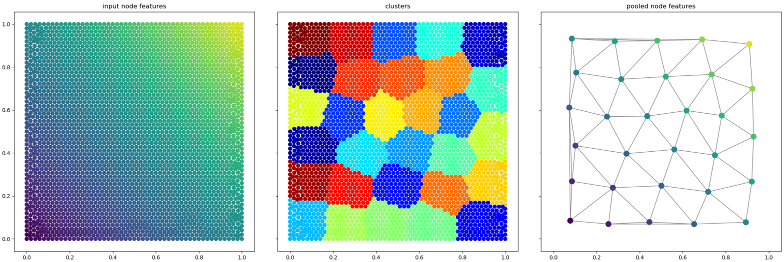


Figure: Initial mesh (left), clusters (center), and pooled mesh (right).

Frontier detection problem

Frontier detection: dataset

problem: detect the frontier between to areas on a mesh

Frontier detection: dataset

problem: detect the frontier between to areas on a mesh

3 types of areas:

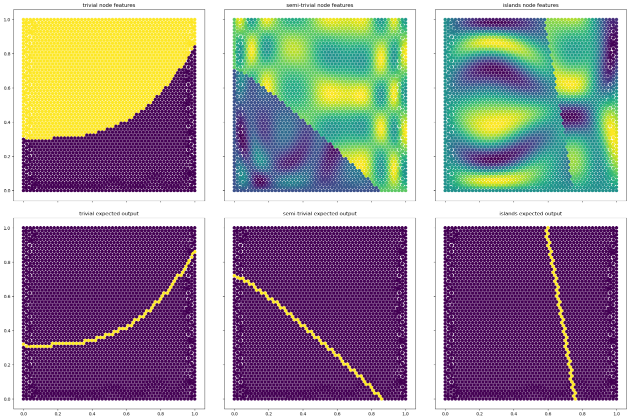


Figure: Trivial dataset (left), semi-trivial dataset (center) and islands dataset (right).

Frontier detection: previous results

simple sequential model: GCN layers put one after the other

Frontier detection: previous results

simple sequential model: GCN layers put one after the other

worked only on the trivial dataset

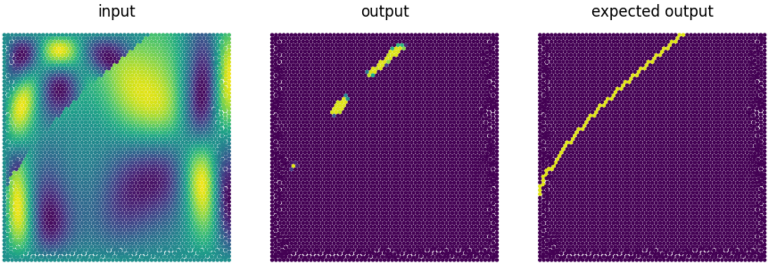


Figure: Old model results on the islands dataset.

Frontier detection: U-Net architecture

more complex architecture

Frontier detection: U-Net architecture

more complex architecture

model: contractive + expansive path

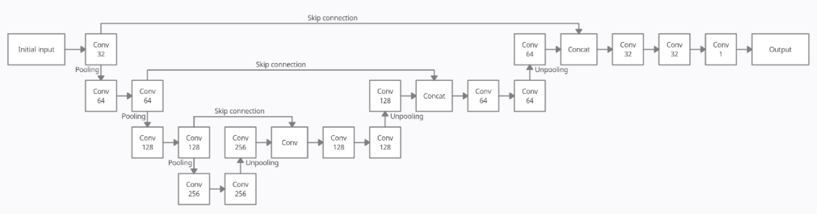


Figure: Architecture of the model used.

Frontier detection: U-Net architecture

more complex architecture

model: contractive + expansive path

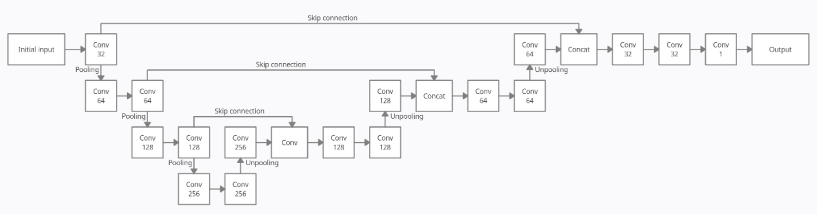


Figure: Architecture of the model used.

3 pooling layers and 3 unpooling layers

Frontier detection: results

first model: U-Net with Vanilla GCN layers and Top-k pooling layers

Frontier detection: results

first model: U-Net with Vanilla GCN layers and Top-k pooling layers
$\Rightarrow$ bad results: Top-k pooling discard big portions of the graph

Frontier detection: results

first model: U-Net with Vanilla GCN layers and Top-k pooling layers
$\Rightarrow$ bad results: Top-k pooling discard big portions of the graph

second model: replace Top-k pooling by k-Means pooling

Frontier detection: results

first model: U-Net with Vanilla GCN layers and Top-k pooling layers
⇒ bad results: Top-k pooling discard big portions of the graph

second model: replace Top-k pooling by k-Means pooling
⇒ good results on trivial/semi-trivial dataset

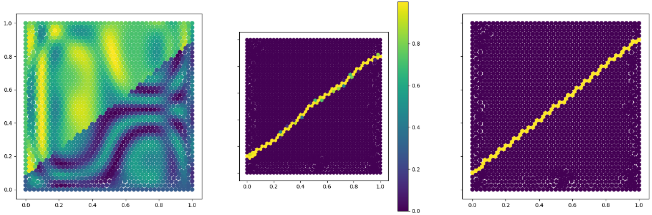


Figure: Input (left), model prediction (middle), expected output (right).

Frontier detection: results

second model still unable to detect the border on the islands dataset

Frontier detection: results

second model still unable to detect the border on the islands dataset

third model: replace VanillaGCN with ChebConv layers

Frontier detection: results

second model still unable to detect the border on the islands dataset

third model: replace VanillaGCN with ChebConv layers
$\Rightarrow$ model more complex/more trainable weights

Frontier detection: results

second model still unable to detect the border on the islands dataset

third model: replace VanillaGCN with ChebConv layers
$\Rightarrow$ model more complex/more trainable weights

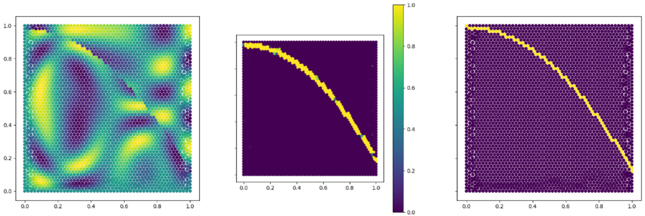


Figure: Input (left), model prediction (middle), expected output (right).

Burgers' equation and dynamic refining

Burgers' equation and dynamic refining

PDE used for example in fluid mechanics or traffic flow

Burgers' equation and dynamic refining

PDE used for example in fluid mechanics or traffic flow

$$\partial_t \rho(t, x) + \nabla \cdot \left(a \frac{\rho(t, x)^2}{2} \right) = 0 \quad (1)$$

with $\rho : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}$, $a \in \mathbb{R}^2$ and $t \in [0, T]$.

Burgers' equation and dynamic refining

PDE used for example in fluid mechanics or traffic flow

$$\partial_t \rho(t, x) + \nabla \cdot \left(a \frac{\rho(t, x)^2}{2} \right) = 0 \tag{1}$$

with $\rho : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}$, $a \in \mathbb{R}^2$ and $t \in [0, T]$.

multiple methods: kinetic relaxation or finite volume method

Burgers' equation and dynamic refining

PDE used for example in fluid mechanics or traffic flow

$$\partial_t \rho(t, x) + \nabla \cdot \left(a \frac{\rho(t, x)^2}{2} \right) = 0 \quad (1)$$

with $\rho : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}$, $a \in \mathbb{R}^2$ and $t \in [0, T]$.

multiple methods: kinetic relaxation or finite volume method

transforms PDE into algebraic equations

Finite Volume Method

mesh Ω, triangles Ω_j, t_n discretization of $[0, T]$

Finite Volume Method

mesh Ω, triangles Ω_j, t_n discretization of $[0,\ T]$

final scheme:

$$\rho_j^{n+1} = \rho_j^n - \frac{\Delta t}{|\Omega_j|} \sum_{k \in E_j} d_{jk} F(\rho_j^n,\ \rho_k^n).$$

where:

$$F(\rho_j^n,\ \rho_k^n) = \frac{1}{2}\Big[a \cdot n_{jk}\left(\rho_j^{n2} + \rho_j^{n2}\right) + \max\left(|a \cdot n_{jk}\rho_j^n|,\ |a \cdot n_{jk}\rho_k^n|\right)(\rho_j^n - \rho_k^n)\Big]$$

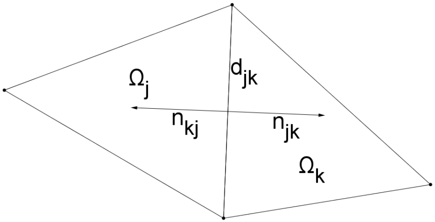


Figure: Notations.

Example of solutions

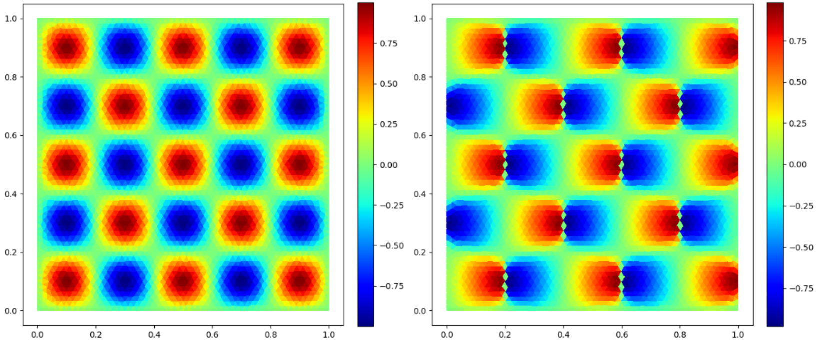


Figure: Initial solution (left) and final solution (right) at $t = 0.05s$, $a = (1, 0)$.

Example of solutions

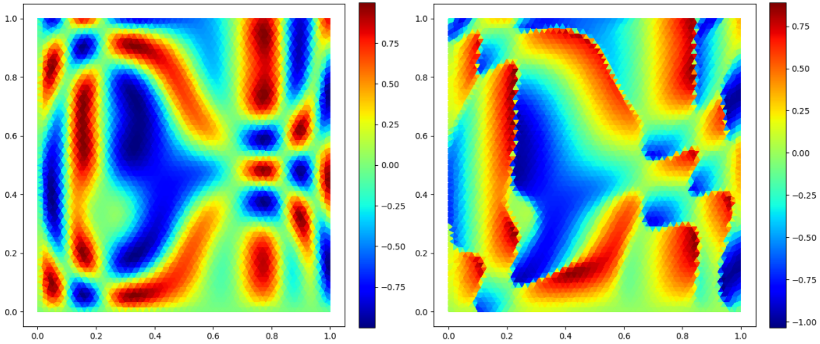


Figure: Initial solution (left) and final solution (right) at $t = 0.05s$, $a = (1, 1)$.

Dynamic refining

refine the mesh while we are computing the final solution

Dynamic refining

refine the mesh while we are computing the final solution

use the border detection model to detect discontinuities

Dynamic refining

refine the mesh while we are computing the final solution

use the border detection model to detect discontinuities

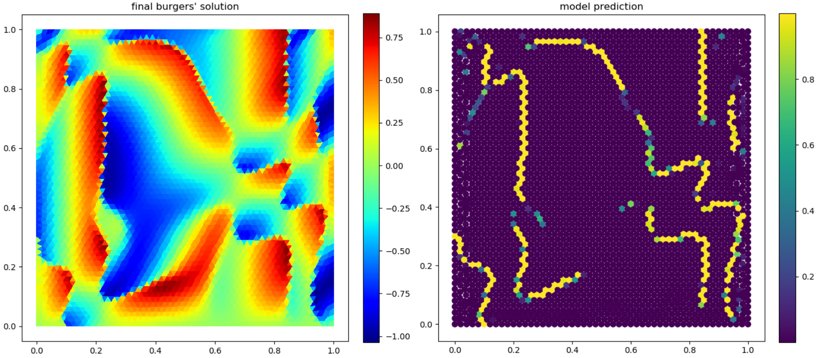


Figure: Final Burgers' solutions (left), and model predictions (right).

Dynamic refining

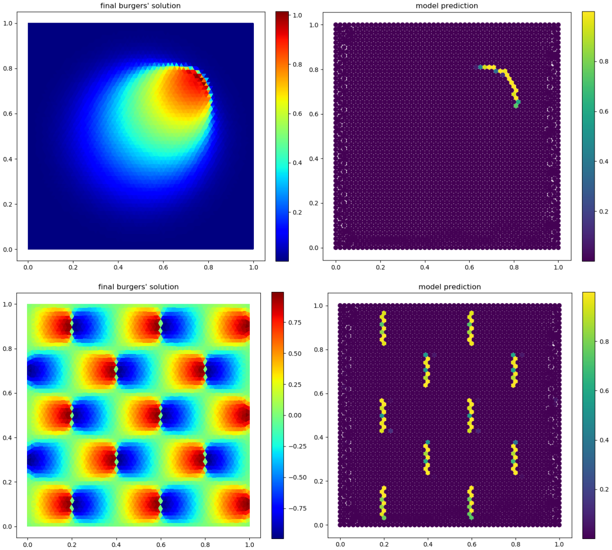


Figure: Final Burgers' solutions (left), and model predictions (right).

Results

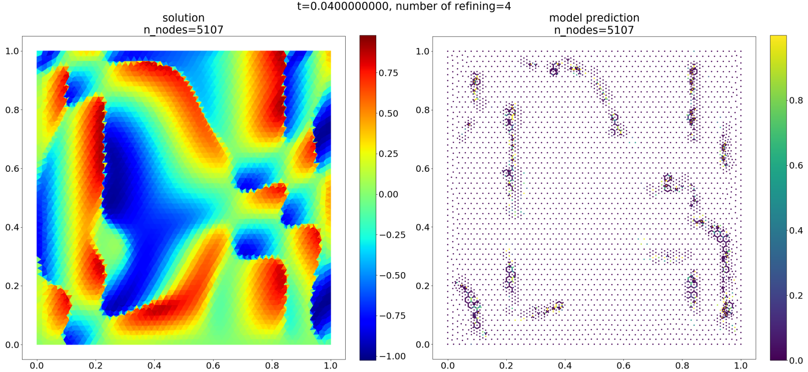


Figure: Solution with refinements (left), and model prediction (right).

Results

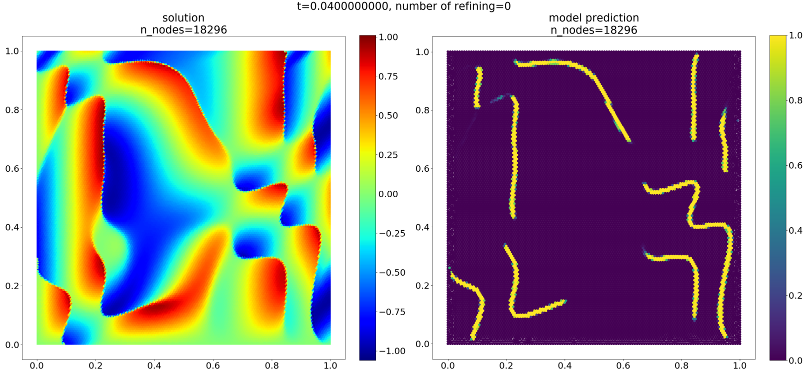


Figure: Finer solution (left), and model prediction (right).

Results

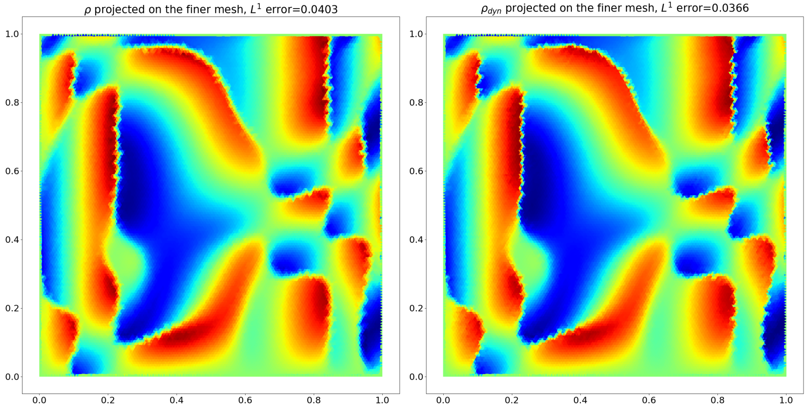


Figure: Projections and errors on the finer mesh.

Transport equation and interpolation problem

Transport equation and interpolation problem

equation describing the displacement of some quantity

Transport equation and interpolation problem

equation describing the displacement of some quantity

$$\partial_t u + a(x) \cdot \nabla_x u = 0 \tag{2}$$

with $u : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}$, $a : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}^2$ the direction.

Transport equation and interpolation problem

equation describing the displacement of some quantity

$$\partial_t u + a(x) \cdot \nabla_x u = 0 \tag{2}$$

with $u : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}$, $a : \mathbb{R}_+ \times \mathbb{R}^2 \to \mathbb{R}^2$ the direction.

semi-Lagrangian method

Semi-Lagrangian method

characteristic curve $X_{s,\,y} : \mathbb{R}_+ \to \mathbb{R}^2$

Semi-Lagrangian method

characteristic curve $X_{s,y} : \mathbb{R}_+ \to \mathbb{R}^2$

$$\begin{cases} X'(t) & = a(t, X(t)), \\ X(s) & = y. \end{cases}$$

Semi-Lagrangian method

characteristic curve $X_{s,\,y} : \mathbb{R}_+ \to \mathbb{R}^2$

$$\begin{cases} X'(t) & = a(t,\, X(t)), \\ X(s) & = y. \end{cases}$$

the solution u can be computed as

$$u_j^{n+1} = u(t_{n+1},\, x_j) = u(t_n,\, X_{t_{n+1},x_j}(t_n)).$$

Semi-Lagrangian method

characteristic curve $X_{s,\,y} : \mathbb{R}_+ \to \mathbb{R}^2$

$$\begin{cases} X'(t) & = a(t,\, X(t)), \\ X(s) & = y. \end{cases}$$

the solution u can be computed as

$$u_j^{n+1} = u(t_{n+1},\, x_j) = u(t_n,\, X_{t_{n+1},x_j}(t_n)).$$

$X_{t_{n+1},x_j}(t_n) \in \mathbb{R}^2$ is not necessarily a node of the mesh

Semi-Lagrangian method

characteristic curve $X_{s,\,y} : \mathbb{R}_+ \to \mathbb{R}^2$

$$\begin{cases} X'(t) & = a(t,\, X(t)), \\ X(s) & = y. \end{cases}$$

the solution u can be computed as

$$u_j^{n+1} = u(t_{n+1},\, x_j) = u(t_n,\, X_{t_{n+1},x_j}(t_n)).$$

$X_{t_{n+1},x_j}(t_n) \in \mathbb{R}^2$ is not necessarily a node of the mesh

$$u(t^n,\, X_{t_{n+1},x_j}(t_n)) \simeq (\Pi u^n) \left(X_{t_{n+1},x_j}(t_n)\right),$$

Π an interpolation operator

Semi-Lagrangian method

characteristic curve $X_{s,\,y} : \mathbb{R}_+ \to \mathbb{R}^2$

$$\begin{cases} X'(t) & = a(t,\, X(t)), \\ X(s) & = y. \end{cases}$$

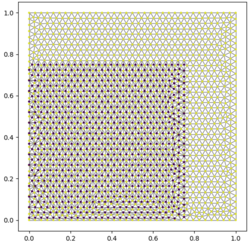
the solution u can be computed as

$$u_j^{n+1} = u(t_{n+1},\, x_j) = u(t_n,\, X_{t_{n+1},x_j}(t_n)).$$

$X_{t_{n+1},x_j}(t_n) \in \mathbb{R}^2$ is not necessarily a node of the mesh

$$u(t^n,\, X_{t_{n+1},x_j}(t_n)) \simeq (\Pi u^n)\left(X_{t_{n+1},x_j}(t_n)\right),$$

Π an interpolation operator



Interpolation problem

constant direction $a \Rightarrow X_{s,y}(t) = y + (t - s)a$, and:

$$u_j^{n+1} = \left(\Pi u^n\right)\left(x_j - a\Delta t\right).$$

Interpolation problem

constant direction $a \Rightarrow X_{s,y}(t) = y + (t - s)a$, and:

$$u_j^{n+1} = \left(\Pi u^n\right)\left(x_j - a\Delta t\right).$$

operator Π: same model than for border detection

Interpolation problem

constant direction $a \Rightarrow X_{s,y}(t) = y + (t-s)a$, and:

$$u_j^{n+1} = (\Pi u^n)(x_j - a\Delta t).$$

operator Π: same model than for border detection

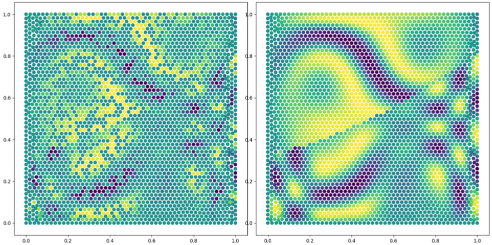


Figure: Input (left) and expected output (right).

Results

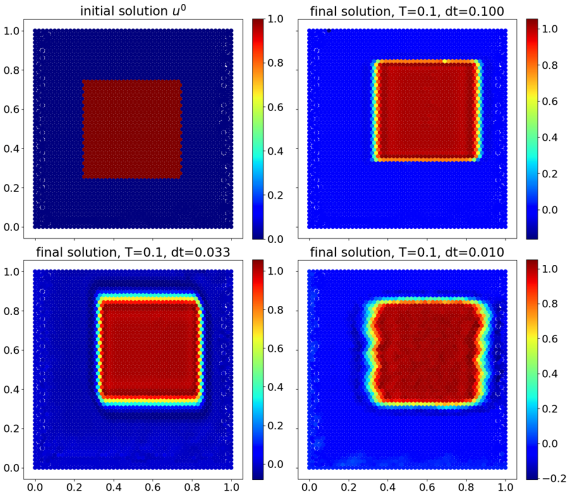


Figure: Solutions computed using the U-Net interpolation model.

Results

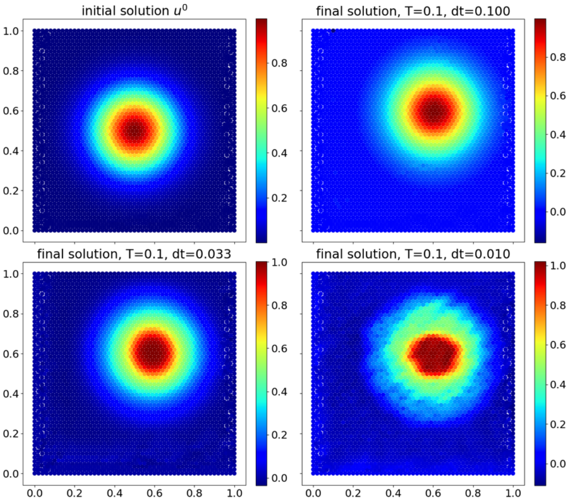


Figure: Solutions computed using the U-Net interpolation model.

Conclusion

U-Net architecture is efficient

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements
- use a solving method without time constraints

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements
- use a solving method without time constraints

we made an interpolation operator, but instable

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements
- use a solving method without time constraints

we made an interpolation operator, but instable

possible corrections:

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements
- use a solving method without time constraints

we made an interpolation operator, but instable

possible corrections:

- modify the training dataset

Conclusion

U-Net architecture is efficient
$\Rightarrow$ allowed us to solve the frontier problem

good results on Burgers' equation but:

- limitation on number of refinements
- use a solving method without time constraints

we made an interpolation operator, but instable

possible corrections:

- modify the training dataset
- modify the training process

Tools used

totality of this project is coded in Python

Tools used

totality of this project is coded in Python

- Tensorflow/Keras (model training)
- Spektral (convolutional layers)
- Github
- PyGMSH (generate meshes)

Tools used

totality of this project is coded in Python

- Tensorflow/Keras (model training)
- Spektral (convolutional layers)
- Github
- PyGMSH (generate meshes)

v100 GPU for training sessions