CSE Semester Project

École Polytechnique Fédérale de Lausanne

Section de Mathématiques

Numerical Simulation of the Transport Equation with Neural Networks

Author:
Eliott Zemour

Supervisor:
Prof. Marco Picasso

June 10, 2021

Abstract

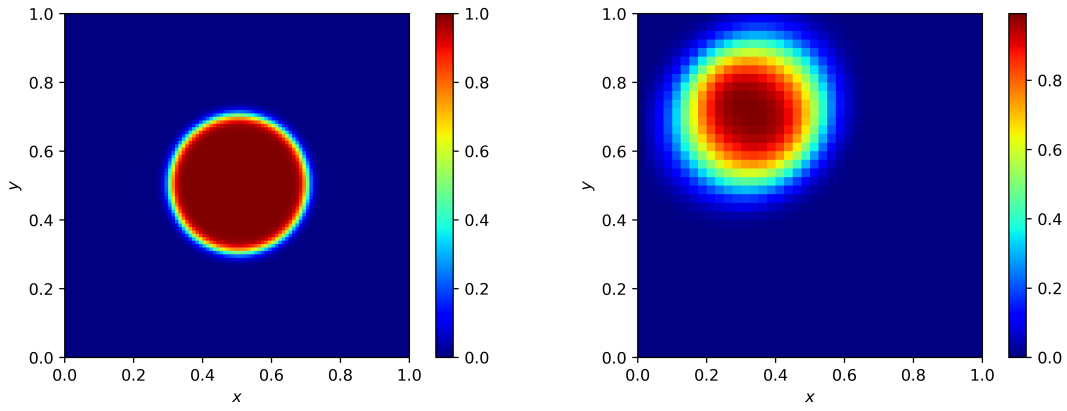
In this report, we implemented numerical schemes for solving the one and two dimensional transport problems. Numerical solutions are obtained with rather coarse grids and are passed to a neural network which aims to improve the quality of such solutions by mapping them into finer grids. We found strong performance as the networks implemented demonstrated the capability to correct numerical diffusion artefacts and refine the solution at key points of the grid. When the the network is trained on a given type of functions, it appears to replicate very well the targeted outputs. However, this approach does not seem to be very promising if the goal is to train a model over basis functions, and use it for enhancing transported functions that consist of linear combination of these.

Contents

1 Introduction

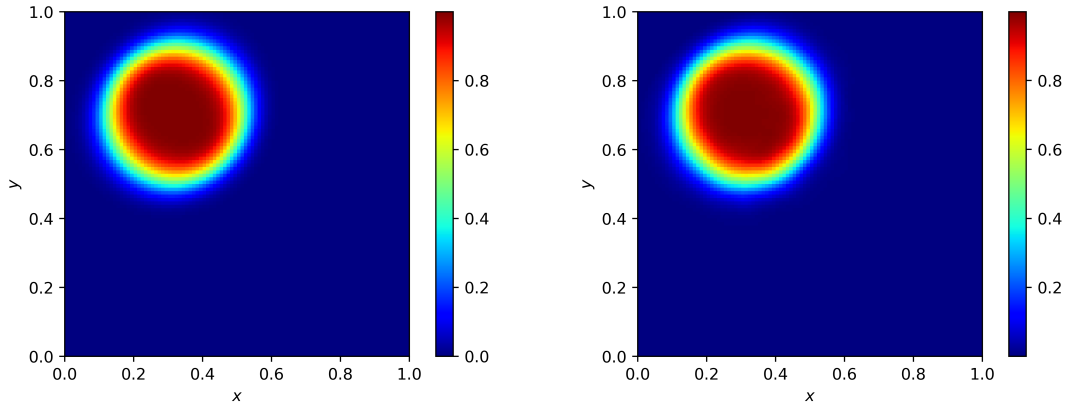
Over the past 10 years, progresses in Deep Learning (DL) demonstrated that artificial neural networks were capable of mind blowing achievements that no other Machine learning or classical numerical methods could hope to match. These exiting results have led to breakthroughs in numerous fields such as speech and image recognition [1], finance, recommender systems and medical applications [2, 3]. Therefore, it was only a question of time before neural networks were applied to the numerical resolution of partial differential equations (PDEs). Many engineering problems require a massive amount of computational power to solve complex PDEs repeatedly using finite difference methods. Recently, some deep learning techniques were deployed with the aim to outperform conventional solvers in terms of computational time: Finite-dimensional operators, Neural-FEM and Neural Operators [4].

Furthermore, another achievement of DL and artificial neural networks can be thought as a potential application to numerical solutions of PDEs: Image Super-Resolution (SR) [5]. Indeed, many deployed algorithms demonstrated outstanding performance in image quality enhancement: Letsenhance.io, Deep Image, etc... Concerning numerical methods for PDE's, obtaining a fine approximation of the solution can be an extremely computational-intensive task in high dimensions but cheaply enhancing coarse approximation could save orders of magnitude of computational time. In 2-dimensional space, the relationship with Image Super-Resolution is straightforward but it can easily be generalized to other dimensions. Figure 1 shows a motivational example of how a neural network achieves to "enhance" an initially coarse approximation of the solution of a 2D transport problem.



(a) Initial solution u_0 to be transported: $N = 100$.

(b) Approximation of transported solution: $N = 40$.



(c) Approximation of transported solution: $N = 100$.

(d) Output of neural network: $N = 100$.

Figure 1: Motivational example: coarse approximation and output of neural network for the solution of 2D transport problem. $N \times N$ grid. This numerical experiment is described in Section 3.2.

This semester project aims to investigate how such computationally cheap (coarse) approximations can be improved by the use of neural networks, with a focus on the transport problem in 1 and 2 dimensions. First, we will introduce the PDE associated with the transport problem, as well as the numerical scheme implemented based on finite difference methods (Sects. 2.1, 2.2). After having visualized the numerical diffusion artefacts at sharp transitions (Sect. 2.3.1), we will perform numerical experiments with neural networks in 1 and 2 dimensions for addressing this problem while keeping the computation relatively cheap (Sect. 3).

2 Statement of the problem

In this project, the focus is put on the transport equation, which is one of the simplest PDEs with lots of application in fluid dynamics. A scalar quantity u, which can be thought as some substance concentration, is transported across space and time by a speed field a. The rate of change $\partial_t u$ of concentration is proportional to the gradient ∇u multiplied by the speed field at a given point.

2.1 The 1-dimensional transport problem

The formulation of the 1D transport problem is as following: $u_0 : [0,1] \to \mathbb{R}$ satisfying $u_0(0) = 0$. Find $u :]0,1[\times[0,\infty[\to \mathbb{R}$ such that:

$$\begin{cases} \partial u/\partial t + a(x,t) \cdot \partial u/\partial x = 0 & \\ u(x,0) = u_0(x), & 0 < x < 1 \\ u(0,t) = 0, & t > 0 \end{cases} \tag{1}$$

To solve it numerically, a discretization of $[0,1] \times [0,T]$ is performed in the following way: let $N, M > 0$ be integers, T the final time, $h := 1/N$ (space step) and $\tau = T/M$ (time step) such that $\tau \leq h/a$. We obtain:

$$x_i = i \cdot h, \quad i = 0, 1, \ldots, N \qquad\qquad t_n = n \cdot \tau, \quad n = 0, 1, \ldots, M.$$

Let us write $a_i = a(x_i)$ and define $a_i^+ = \max(0, a_i)$, $a_i^- = \max(-a_i, 0)$. We denote u_i^n the approximation of $u(x_i, t_n)$ obtained using the following upwind scheme:

$$\frac{u_i^{n+1} - u_i^n}{\tau} + a_i^- \cdot \frac{u_i^n - u_{i+1}^n}{h} + a_i^+ \cdot \frac{u_i^n - u_{i-1}^n}{h} = 0, \qquad u_0^n = 0. \tag{2}$$

for $n = 0, 1, \ldots, M-1$ and $i = 1, 2, \ldots, M$. The explicit update is therefore:

$$u_i^{n+1} = u_i^n - a_i^+ \frac{\tau}{h}\left(u_i^n - u_{i-1}^n\right) - a_i^- \frac{\tau}{h}\left(u_i^n - u_{i+1}^n\right) \tag{3}$$

2.1.1 Convergence study

The scheme in Eq.(2) is subject to a CFL [6] stability condition given by:

$$\text{CFL} := \frac{\max a(x) \cdot \tau}{h} = \frac{\max a(x) \cdot T \cdot N}{M} < 1 \tag{4}$$

In the following, M, N and T are chosen such that CFL is fixed to 0.5.
To study the convergence of the above scheme, we consider the case where $a(x,t) = x/20$. The exact solution at time T can be computed using the method of characteristics and is given by:

$$u_{ex}(x,T) = u_0\left(xe^{-t/20}\right) \tag{5}$$

In Figure 2 is shown the max error plotted against N in log space. The initial condition u_0 is given by:

$$u_0(x) = \exp((x-\mu)^2/(2\sigma^2)), \qquad 0 < x < 1, \mu = 0.3, \sigma = 0.05 \tag{6}$$

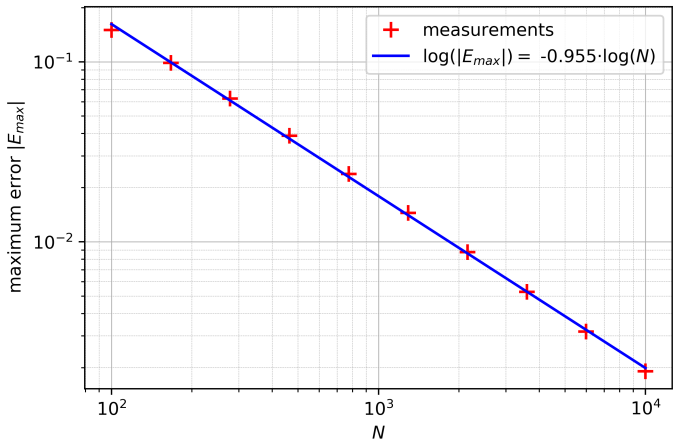


Figure 2: Convergence study for Gaussian u_0 given in Eq.(6) and $a(x) = x/20$. CFL $= 0.5$.

2.2 The 2-dimensional transport problem

The problem becomes, with $u = u(x, y, t)$:

$$\begin{cases} \partial u/\partial t + \vec{a}(x,y,t) \cdot \nabla_{x,y}\, u = 0, & \vec{a} = (a_x, a_y)^T \\ u(x,y,0) = u_0(x,y), & 0 < x, y < 1 \\ u(0,t) = 0, & t > 0 \end{cases} \tag{7}$$

2.2.1 Numerical scheme

Similarly to the 1D case, the speed field is given by $\vec{a}(x,y) = (a_x(x,y), a_y(x,y))$ and we write $a_{x,i,j} = a_x(x_i, y_j)$. Let us denote $a^+_{x,i,j} = \max(0, a_{x,i,j})$ and $a^-_{x,i,j} = \max(-a_{x,i,j}, 0)$.
With $u^n_{i,j}$ the approximation of $u(x_i, y_j, t_n)$, the scheme is therefore:

$$\begin{aligned} u^{n+1}_{i,j} = u^n_{i,j} &- a^+_{x,i-1,j} \cdot \frac{\tau}{h} \cdot \left(u^n_{i,j} - u^n_{i-1,j}\right) - a^-_{x,i,j} \cdot \frac{\tau}{h} \cdot \left(u^n_{i,j} - u^n_{i+1,j}\right) \\ &- a^+_{x,i,j-1} \cdot \frac{\tau}{h} \cdot \left(u^n_{i,j} - u^n_{i,j-1}\right) - a^-_{y,i,j} \cdot \frac{\tau}{h} \cdot \left(u^n_{i,j} - u^n_{i,j+1}\right) \end{aligned} \tag{8}$$

where τ is the time step and h the space steps in both directions.

2.2.2 Convergence study

In order to test the convergence of the scheme, we compute a solution to the transport problem for which we know the exact solution. Indeed, using the following initial function:

$$u_0(x,y) = \exp\left(- \frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2} \right) \tag{9}$$

with $(\mu_x, \mu_y) = (0.25, 0.25)$, $(\sigma_x, \sigma_y) = (0.1, 0.1)$ and $(a_x, a_y) = (0.05, 0.05)$, we can easily deduce the solution at $T = 10$, which is the same Gaussian as Eq.(9) with $(\mu_x, \mu_y) = (0.75, 0.75)$. The CFL stability condition can also be extended to the 2-dimensional case and becomes:

$$\text{CFL} = \frac{\max a_x \cdot \tau}{h} + \frac{\max a_y \cdot \tau}{h} < 1 \tag{10}$$

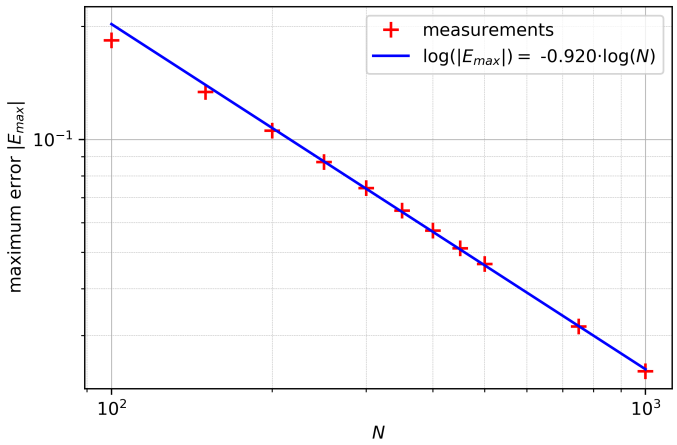


Figure 3: Convergence study for Gaussian u_0 given in Eq.(9) and $\vec{a}(x, y) = (0.05, 0.05)$. CFL $=$ 0.5.

2.3 Transported functions

In this project, a special focus is put on the numerical transportation of functions with sharp transitions between 0 and 1. Indeed, as mentioned above, this physical problem involves quantities that can be interpreted as a concentration or indicator function of some substance.
For the 1-dimensional case, such functions are shown in Figures 4, 5 and their mathematical definitions are given by Eqs.(11) and (12):

$$\text{step:} \quad u_0(x) = \frac{1}{2}\big(1 + \tanh(k(x - x_1))\big) \tag{11}$$

$$\text{wall:} \quad u_0(x) = \frac{1}{2}\big(\tanh(k(x - x_1) - \tanh(k(x - x_2))\big) \tag{12}$$

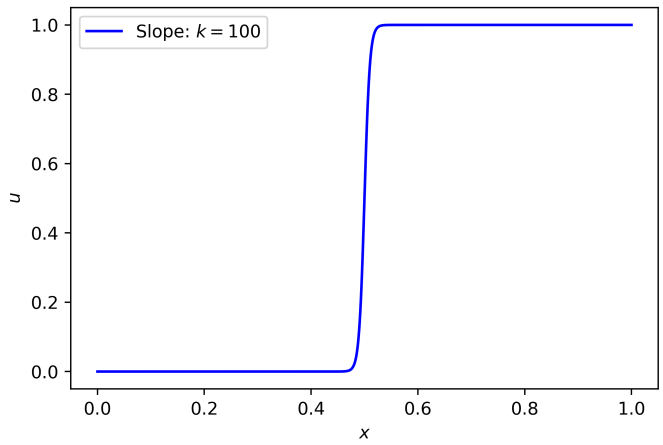


Figure 4: Step function defined in Eq.(11) with $k = 100$, $x_1 = 0.5$.

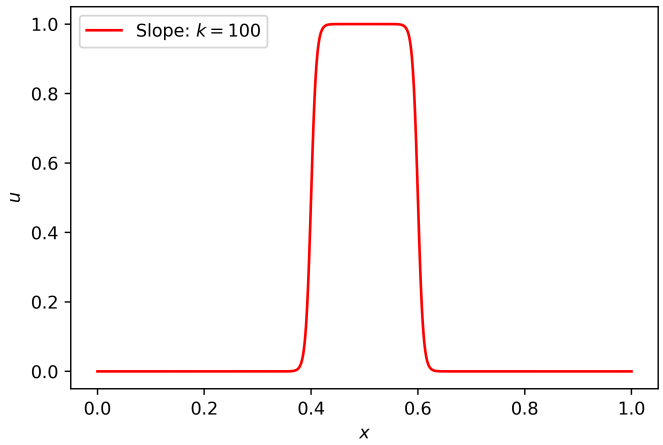


Figure 5: Wall function defined in Eq.(12) with $k = 100$, $x_1 = 0.4$, $x_2 = 0.6$.

Moreover, an initial function in the 2-dimensional transport problem is directly inspired by [7] and is given by:

$$u_0(x, y) = \frac{1}{2}\left(1 + \tanh\left(-C(\sqrt{(x - x_1)^2 + (y - y_1)^2} - r)\right)\right) \tag{13}$$

It represents a function whose value is one inside a circle of radius r and center (x_1, y_1) and zero elsewhere: Figure 6. In the following, we will denote this function as the 'circle function'.
Finally, we introduce a similar function whose value is one inside a square defined by the conditions $\{x_1 \leq x \leq x_2,\ y_1 \leq y \leq y_2\}$. It is defined by:

$$u_0(x, y) = \frac{1}{4}\big[\tanh\big(C(x - x_1)\big) - \tanh\big(C(x - x_2)\big)\big] \cdot \big[\tanh\big(C(y - y_1)\big) - \tanh\big(C(y - y_2)\big)\big] \tag{14}$$

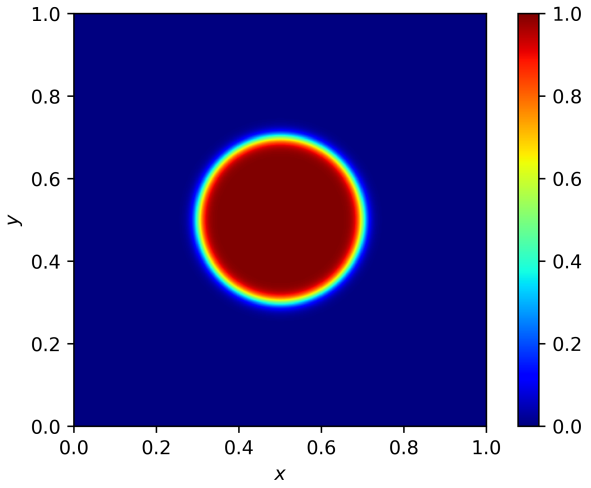


Figure 6: Initial circle function u_0 defined in Eq.(13) with $C = 60$, $r = 0.2$, $x_1 = y_1 = 0.5$.

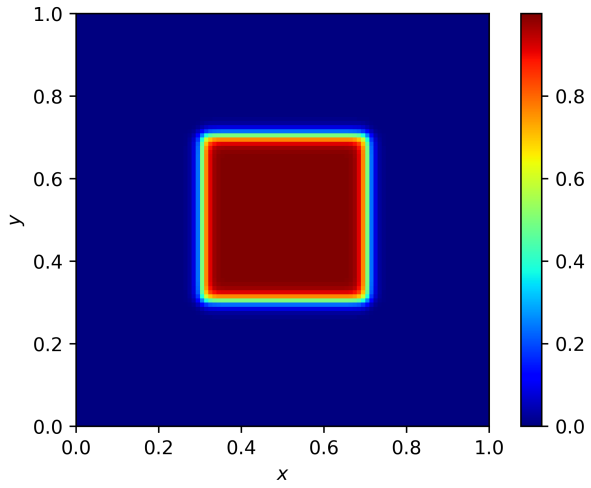


Figure 7: Initial square function u_0 defined in Eq.(14) with $C = 60$, $x_1 = y_1 = 0.3$, $x_2 = y_2 = 0.7$.

2.3.1 Numerical diffusion phenomena

The numerical transportation of such functions presented above (Figures 4, 5, 6) by order 1 schemes (Eqs.(2),(8)) is often subject to numerical diffusion [8, 9]. It is an artefact which causes a flattening at the transition $0 \to 1$ of such solutions. Figures 8 and 9 allow to visualize this phenomenon.

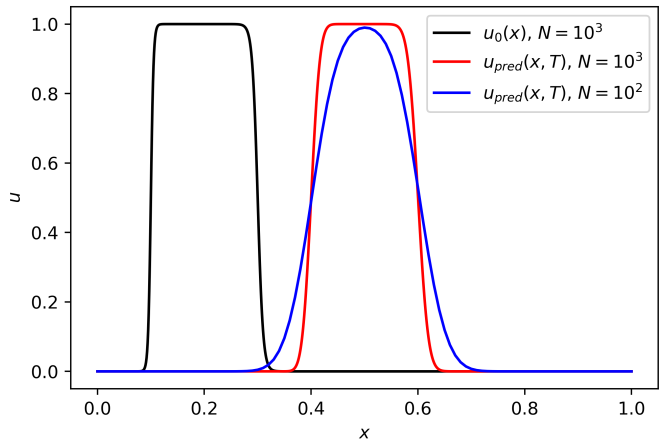
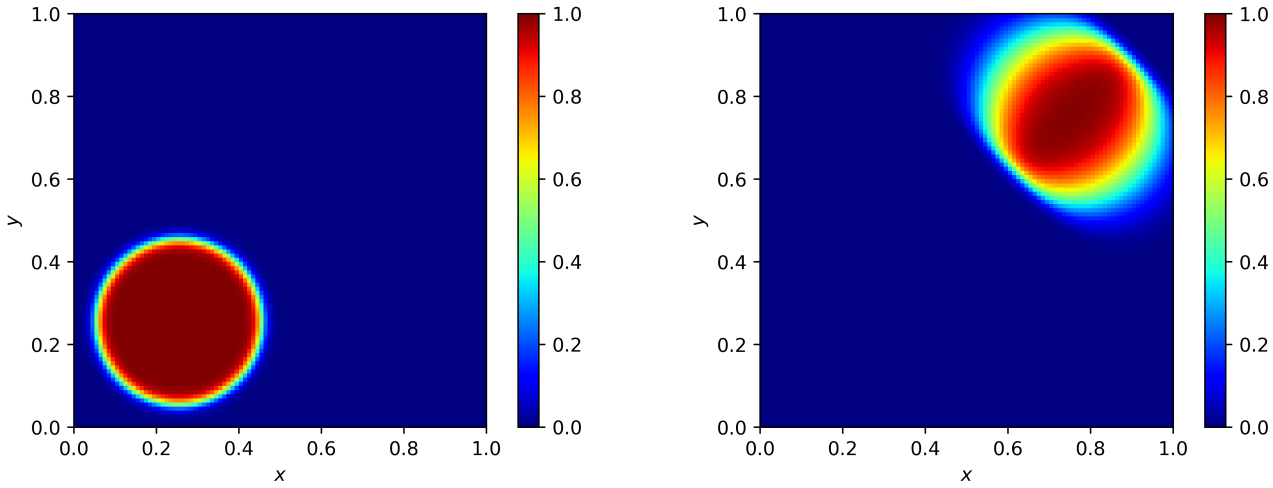


Figure 8: Visualisation of numerical diffusion in the 1D case. u_0 is defined in Eq.(12) with $k = 200$. $T = 10$, CFL=0.5.



(a) Initial solution u_0 to be transported: $N = 100$.

(b) Approximation of transported solution: $N = 100$.

Figure 9: Visualisation of numerical diffusion in the 2D case: $N = 100$ in each dimension leads to poor conservation of the initial shape.

2.4 Performance metrics of the network

In order to evaluate properly the performances of the neural networks presented in Section 3, we introduce the following metric measuring how well the output of the network reproduces the targeted output (which is the fine approximation of the solution obtained with a numerical scheme):

- The mean squared error (MSE)
- The maximum absolute error (MaAE)

The MSE and MaAE metrics are defined in Eqs.(15) and (16), where u_{pred} is the output of the network and u_{true} is the targeted output:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(u^i_{pred} - u^i_{true}\right)^2 \tag{15}$$

$$\text{MaAE} = \max_{i=1,\dots,N}\left|u^i_{pred} - u^i_{true}\right| \tag{16}$$

Note that the MSE will be used as the loss function for the training of the neural networks. In the 2-dimensional case, the above metrics are computed over indices i, j. The motivation behind the use of these two metrics is the following: (i) MSE is used as a loss to update the parameters of the NN and gives a global insight about the performance over the range $[0, 1]$. As a consequence, the MSE will take into account the very good accuracy of the network over the constant regions of the functions (where $u \equiv 0$), which are not the main points of interest. (ii) The MaAE provides a very localized feedback concerning the performance of the model at the key points of the grid, that is, at the transitions $0 \rightleftarrows 1$. Indeed, regarding the transported functions considered, and the numerical diffusion phenomena, these transition regions are exactly what the network aims to improve.

3 Numerical experiments and results

3.1 The 1-dimensional case

3.1.1 Dataset

The training process of a neural network requires a large amount of training data consisting of input/output pairs. For this purpose, the dataset in 1-dimension is composed of the so-called *step* and *wall* functions with several degrees of freedom. Indeed, for such functions, an approximation of the solution at time $T = 10$ is computed with both coarse and fine grids. **The coarse approximation is then considered as the input of the neural network and the targeted output is the fine one**. These transported initial functions are defined in Eqs.(11) and (12) and the following parameters are subject to variation:

- x_1, the x at which the transition $0 \to 1$ occurs,
- x_2, the x at which the transition $1 \to 0$ occurs (for the wall function exclusively).

For the step function, x_1 varies from 0.1 to 0.7 with a `linspace` of size 2500. Meanwhile, for the wall function, x_1 varies from 0.2 to 0.4 (size 50) and x_2 from 0.6 to 0.8 (size 50) creating 2500 combinations (x_1, x_2). Overall, a dataset of size 5000 is created, each example being made of a coarse approximation with $N = 50$ and a fine one with $20N = 1000$. The speed field $a = 1 \times 10^{-2}$ is positive and constant and the CFL parameter (Eq.(4)) is also kept constant: CFL $= 0.5$, leading to $M = N/5$ in both cases. In Figure 10 is shown an example of one training example for the neural network: the black line being the coarse approximation meant to be the input of the NN and the red one considered as the targeted output.

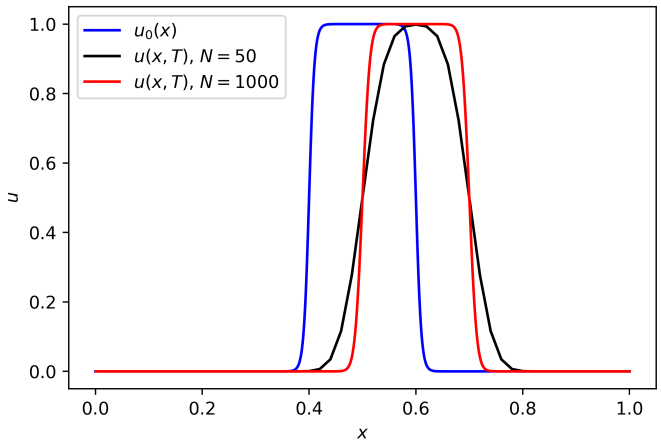


Figure 10: Visualization of one training example: the input of NN is represented by the black line while the objective output is the red line. The initial function u_0 is also shown for the sake of completeness. Parameters of the simulation: $T = 10$, $a = 1 \times 10^{-2}$, CFL $= 0.5$.

3.1.2 Neural network

The neural network used for this task is a feedforward neural network called multilayer perceptrons (MLPs). MLPs are the most typical deep learning models where each neuron at layer (l) is connected to all other neurons at layers $(l \pm 1)$ [10]. The architecture of the MLP implemented is shown in Figure 11 and described as following:

- The input layer is of size $N = 50$ (the size of the coarse grid), with ReLU[1] activation function [11, 12].
- There are 2 hidden layers: the first one with 64 neurons and the second one with 1024 neurons. Both have ReLU activation function.

[1] ReLU stands for Rectified Linear Unit, it is defined as $f(z) = \max\{0, z\}$ for $z \in \mathbb{R}$.

– The output layer is of size $20N = 1000$ (the size of the fine grid), with sigmoid[2] activation function.

The motivation behind the use of a sigmoid activation function is that it ensures the output to be between 0 and 1. Overall, a total of 1 094 824 parameters are trained using the Adam optimizer (learning rate 0.001) [13]. The loss function to be minimized by the network is the mean-squared error (MSE).

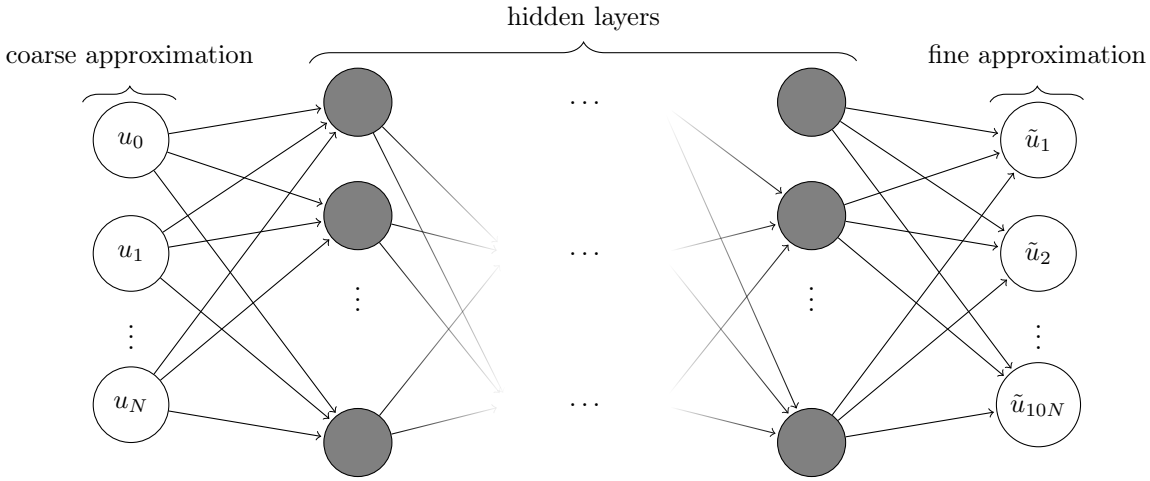


Figure 11: Network graph of the multi-layer perceptron with N input units and $20N$ output units. The coarse approximation of transported solution is taken as input and written $(u_1, \dots, u_N)^T$. The targeted output is a fine approximation of solution of size $20N$: $(\tilde{u}_1, \dots \tilde{u}_{20N})^T$

3.1.3 Results

A *testing* set of size 500 (10%) is separated from the dataset and the remaining examples are split into a *training* and *validation set*. According to DL good practices[3], the use of validation datasets is recommended for hyperparameters tuning such as the number of hidden units/layers in a neural network. Indeed, it aims to provide an unbiased evaluation of a model performance (the NN is not trained over such example) but becomes more and more biased as the as the feedback from the validation dataset is incorporated into the model configuration.
The main performance results of this experiment are summarized in Table 1. The "(train)" and "(test)" labels indicate that the metric is computed over the training and the testing sample, respectively. Figure 12 shows the MSE metric as a function of training epochs for both training and validation datasets.
Furthermore, Figures 13 and 14 allow to visualize the distribution of the performance metrics over the testing set. As a result, we observe that the networks performs well at improving coarse approximations of transported solutions in the 1-dimensional case. On Figure 15 is shown the output of the network for the "worst" testing example, that is the example for which the largest MSE is reached. We can still see that the performance of the network is very good as it manages to accurately identify the areas of transitions $0 \leftrightarrows 1$.

av. MSE (train)	av. MSE (test)	max MSE (test)	av. MaAE (test)	max MaAE (test)
3.016×10^{-6}	3.059×10^{-6}	3.514×10^{-5}	1.243×10^{-2}	4.133×10^{-2}

Table 1: Performance of the neural network over the training and testing set. Two measures are provided: MSE and MaAE. "av." means than the metric is averaged over the whole sample. "max" indicates that the maximum value over the sample is provided.

[2]The sigmoid function is defined as $\sigma(z) = (1 + e^{-z})^{-1}$ for $z \in \mathbb{R}$.
[3]https://machinelearningmastery.com/difference-test-validation-datasets/

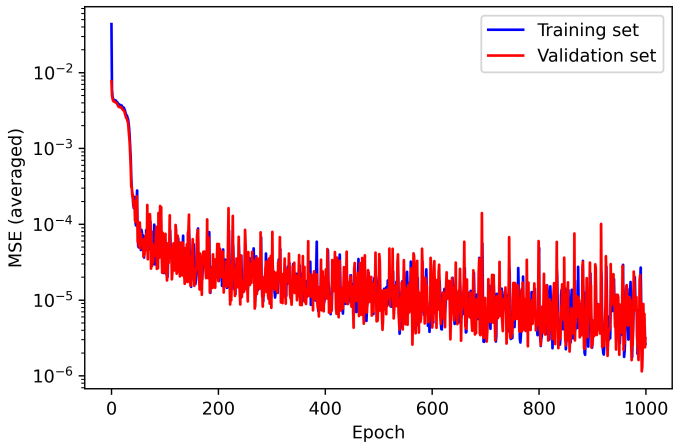


Figure 12: Evolution of the MSE (averaged over samples) through training epochs

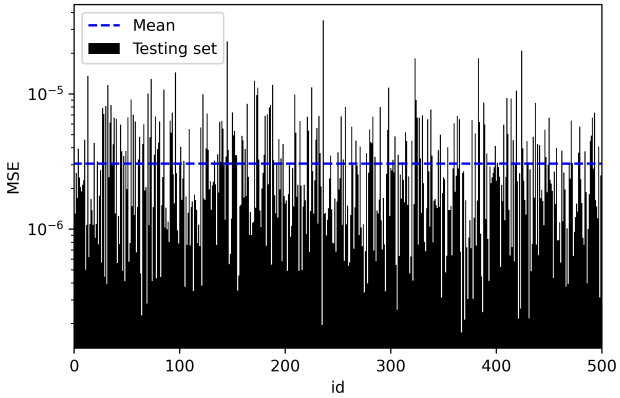


Figure 13: MSE for each of the 500 example of the testing set

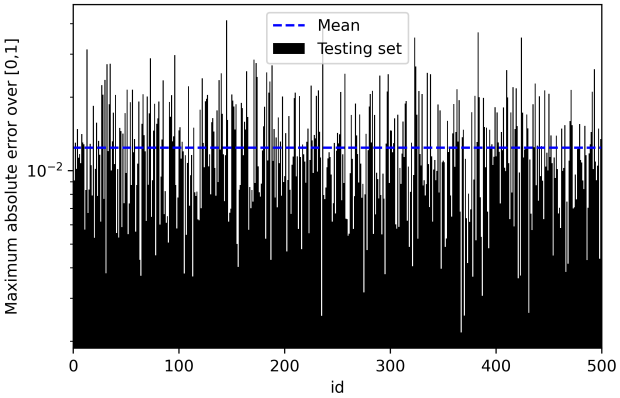
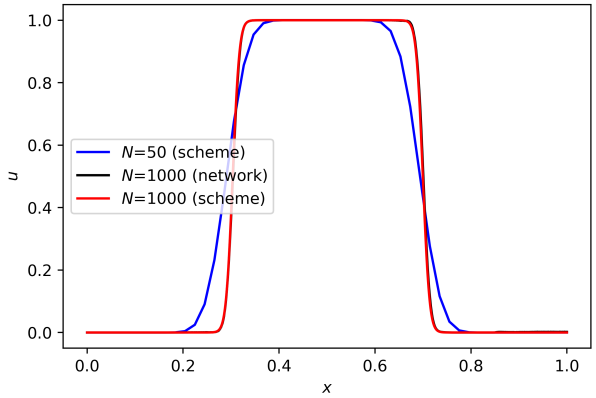
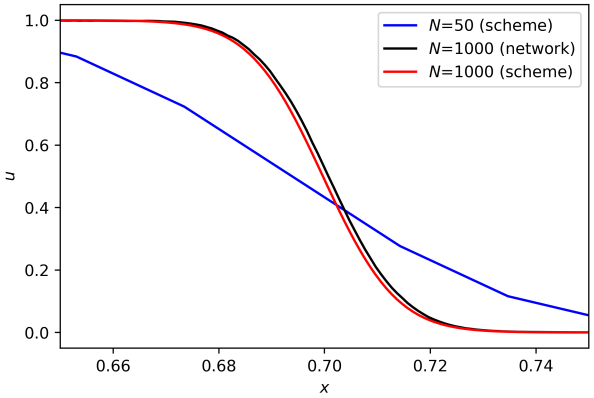


Figure 14: MaAE for each of the 500 example of the testing set



(a) Input of network (blue line), output of network (black line) and targeted output (red line) for the worst testing example



(b) zoom over the range $[0.65, 0.75]$

Figure 15: Worst testing example and a zoom into the region where the MaAE is reached. MSE $= 3.514 \times 10^{-5}$, MaAE $= 0.0385$.

3.1.4 Generalization performance on unknown data: multiple walls

Introducing multiple walls functions

To go further in the 1 dimensional case, we now consider multiple walls initial conditions made of the linear combination of u_0 functions such as the one defined in Eq.(12). The number of wall functions N_{wall} is limited to 5 otherwise the transitions $0 \to 1$ will almost be superposed and a very fine grid would be needed to distinguish between the different walls. The mathematical definition of the u_0 considered in this part can be found in Eq.(17). Figure 16 shows an example of one such function with $N_{\text{wall}} = 3$.

$$u_0(x) = \frac{1}{2} \sum_{n=1}^{N_{\text{wall}}} \left(\tanh(k(x - x_1^n) - \tanh(k(x - x_2^n)) \right) \tag{17}$$

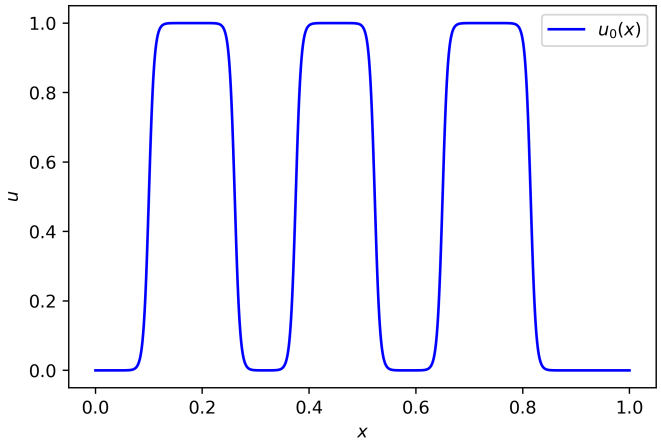


Figure 16: Example of initial function $u_0(x)$ defined in Eq.(17) for the multiple walls case. $N_{\text{wall}} = 3$, $k = 100$.

Given a fixed N_{wall}, specifying a set $\{(x_1^n, x_2^n)\}_{n=1,\dots,N_{\text{wall}}}$ of transition abscissas requires making sure that the walls do not overlap. The following algorithm shows how it is handled, plus how some randomness in the width is added, for the purpose of creating a robust dataset. As a result, each wall pattern forming a given initial condition u_0 have roughly the same width.

Algorithm 1: Generating transition abscissas $\{(x_1^n, x_2^n)\}_{n=1,\dots,N_{\text{wall}}}$ for a given N_{wall}

Input : Number of wall functions N_{wall}
Bounds $[x_{\min}, x_{\max}]$ for the abscissa of the transition $0 \to 1$.
Output: $\{(x_1^n, x_2^n)\}_{n=1,\dots,N_{\text{wall}}}$.

 # find x_1^n for each n and store them in array X_1 s.t $X_{1,n} = x_1^n$
$X_1 = \text{linspace}\,(x_{\min}, x_{\max}, N_{\text{wall}})$
$h = (x_{\max} - x_{\min})/(N_{\text{wall}} - 1)$

 # select x_2^n for each n with added randomness
for $n = 1, \dots N_{\text{wall}}$ **do**
  sample $p \in \mathbb{R}$ from uniform distribution $[0, 1]$
  $x_2^n = x_1^n + (h/2) \cdot (1 + 0.1p)$
end

Training dataset and testing procedures

This part aims to investigate the generalization performance of neural networks over this type of function. the strategy is to (i) train the NN on a dataset containing only 1 and 2 walls (ii) test the trained model over u_0 functions with 1 to 5 walls and visualize its performance on unseen data. For this purpose, a dataset of size 5000 for a neural network is generated by varying the number

of walls N_{wall} (between 1 and 2), as well as their position and width (randomness in Algorithm 1). Keeping the same practices as Section 3.1.3, the training examples are split into a *training* and *validation set*. The MSE metric during training is shown in Figure 17 for both such datasets. The average MSE after 1000 training epochs is of the order $10^{-4}/10^{-5}$, which is not as good as the performance obtained during the training over the dataset containing step and single-wall function only. This could be explained by the presence of functions with twice as many $0 \leftrightarrows 1$ transitions which are the main contributions to the total MSE over $[0, 1]$.

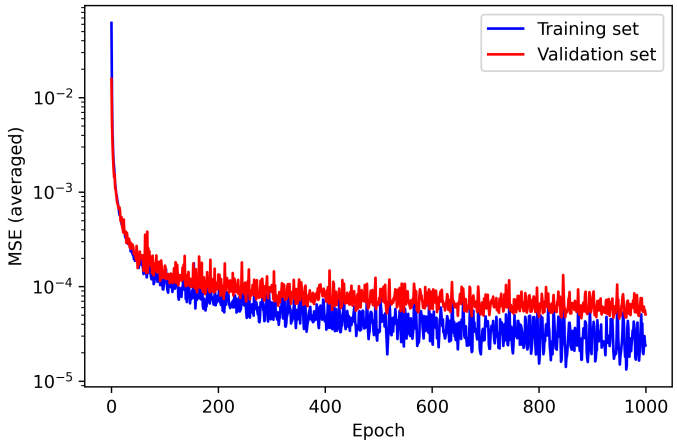


Figure 17: Evolution of the MSE (averaged over samples) through training epochs using the 1 and 2 walls train sets

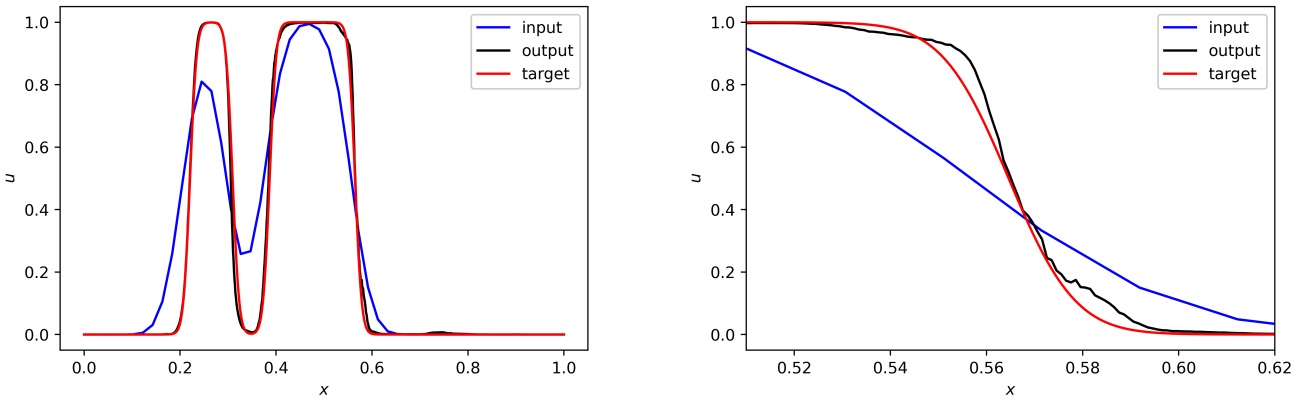
After having trained the model, the following paragraphs will assess its performance over two different test sets.

a) Case 1: $N_{\text{wall}} = 1, 2$
Here, we present the generalization performance of the network over a test set (size 500) made of single and double wall patterns only. Therefore, the testing examples are similar to the ones used for training and one could expect a similar performance to Section 3.1.3. The performance metrics for this experiment are summarized in Table 2, and the worst testing example in terms of MaAe can be visualized in Figure 18.

av. MSE (train)	av. MSE (test)	max MSE (test)	av. MaAE (test)	max MaAE (test)
2.475×10^{-5}	5.466×10^{-5}	3.514×10^{-5}	3.873×10^{-2}	0.201

Table 2: Performance of the neural network over the training and testing ($N_{\text{wall}} = 1, 2$) sets. Two measures are provided: MSE and MaAE. "av." means than the metric is averaged over the whole sample. "max" indicates that the maximum value over the sample is provided.

As a result, the model demonstrate good levels of performance over this test set, with an average MaAE of order 10^{-2}, suggesting that the $0 \leftrightarrows 1$ transitions are predicted accurately. To go further, we observe that the region between the two walls is well handled by the neural network, despite that the coarse approximation stays above 0.2 in Figure 18.



(a) MaAE = 0.201.

(b) zoom over the range $[0.52, 0.62]$

Figure 18: Worst example in the test set for the MaAE metric. The test set is made of single and double wall patterns.

a) Case 2: $N_{\text{wall}} = 3, 4, 5$ (unseen data)

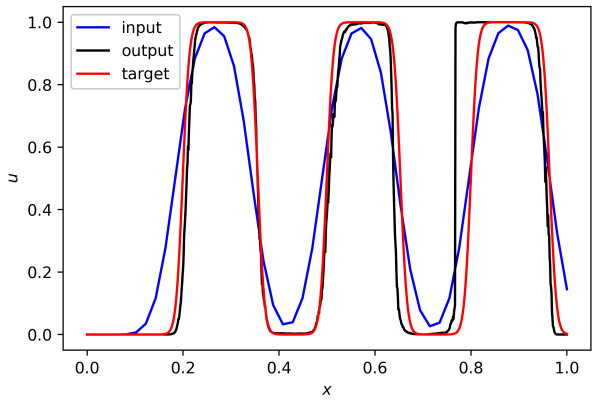
In this part, we aim to investigate how well the trained model can adapt to unseen data ($N_{\text{wall}} = 3, 4, 5$), which consists of linear combination of functions it has been trained on ($N_{\text{wall}} = 1, 2$). The performance metrics for this experiment are summarized in Table 3, and the some testing examples can be visualized in Figure 19 (best and worst for the MSE metric).

As a result, this numerical experiment led to poor performance for both the test MSE and MaAE: an average of 10^{-2} for the MSE is quite high compared to the metrics reported in Tables 1 and 2, and the MaAE is close to its maximum value (1) for every testing example.

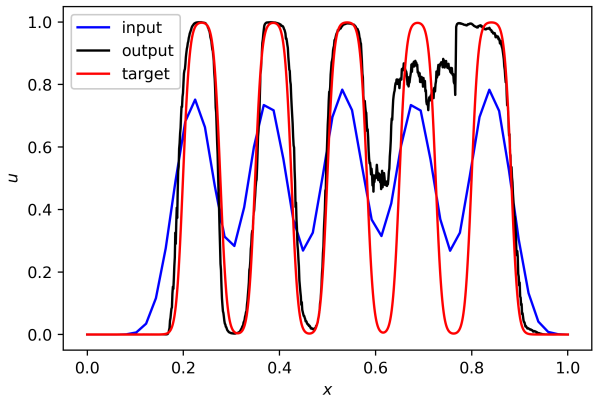
av. MSE (train)	av. MSE (test)	max MSE (test)	av. MaAE (test)	max MaAE (test)
2.475×10^{-5}	4.960×10^{-2}	0.071	0.987	0.992

Table 3: Performance of the neural network over the training and testing ($N_{\text{wall}} = 3, 4, 5$) sets. Two measures are provided: MSE and MaAE. "av." means than the metric is averaged over the whole sample. "max" indicates that the maximum value over the sample is provided.

In Figure 19a, the trained model seems to accurately detect the presence of 3 such wall patterns but struggles with predicting the abscissas of transition. As a result, the maximum absolute error is very high due to the bad performance in the transition regions. When the number of walls is even larger ($N_{\text{wall}} = 5$), the coarse approximation is bound between 0.2 and 0.8 and the output of the neural network is quite far from the ground truth (Figure 19b).



(a) MSE $= 3.3 \times 10^{-2}$, MaAE $= 0.990$.



(b) MSE $= 8.6 \times 10^{-2}$, MaAE $= 0.988$.

Figure 19: Best (a) and worst (b) testing example for the MSE metric.

3.1.5 Variable speed field

In this section, we aim to investigate the case of a variable speed field in 1 dimension. The speed fields $a(x,t)$ considered will vary as a function of x only. Through transportation, in addition to numerical diffusion, this will cause a deformation of the initial shape of the functions u_0. The initial function to be transported is a single-wall function defined in Eq.(12) with $k = 100$ and transition $0 \leftrightarrows 1$ abscissas at 0.1 and 0.3, respectively. The speed field $a(x)$ is such that it is a linear combination of the first 5 functions of a periodic Fourier series. That is:

$$a(x) = \sum_{k=0}^{4} a_k \sin(k\pi x) \tag{18}$$

Where $a_0 = 0$ is fixed. Then, the speed field is normalized so that its maximum is $a_{max} = 0.05$ to ensure that the step function stays in the interval $[0, 1]$. Figures 20 and 21 show examples of variable speed fields and the initial function considered, respectively. Overall, up to 5000 examples are generated by randomly initializing the a_k between 0 and 1. Here again, a *testing* set of size 500 (10%) is separated from the dataset and the remaining examples are split into a *training* and *validation set*.

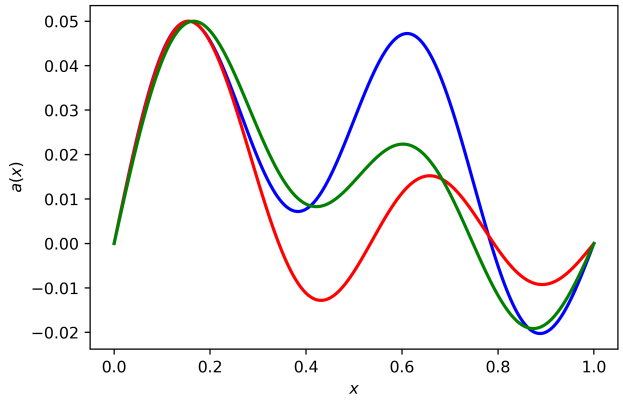


Figure 20: Example of 3 randomly generated speed fields

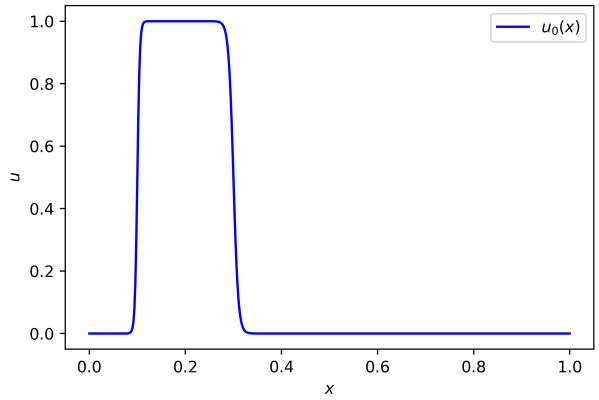


Figure 21: Initial function to be transported at variable speed

Results:
The network, with the exact same architecture as presented in Section 3.1.2, have been trained over the training set for 1000 epochs. Figure 22 shows the evolution of the MSE metric as a function

of training epochs. As a result, the performance metrics for this experiment are summarized in Table 4, and Figures 23, 24 allow to visualize the distribution of MSE and MaAE in the test set, respectively. Finally, some testing examples are shown in Figures 25 and 26.

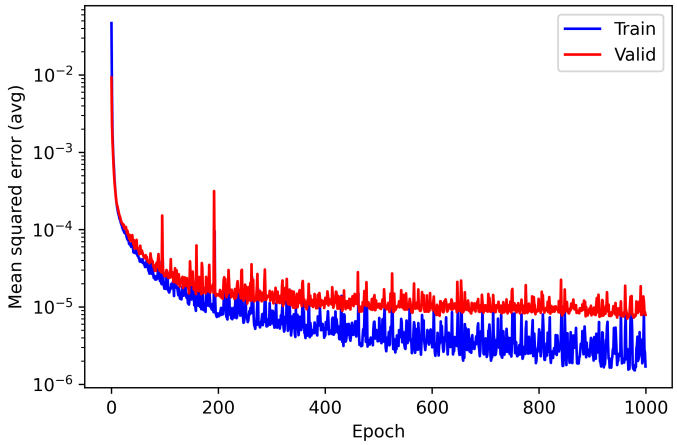


Figure 22: Evolution of the MSE (averaged over samples) through training epochs in the variable speed case

av. MSE (train)	av. MSE (test)	max MSE (test)	av. MaAE (test)	max MaAE (test)
2.793×10^{-6}	6.912×10^{-6}	1.910×10^{-3}	1.139×10^{-2}	0.5044

Table 4: Performance of the neural network over the training and testing sets. Two measures are provided: MSE and MaAE. "av." means than the metric is averaged over the whole sample. "max" indicates that the maximum value over the sample is provided.

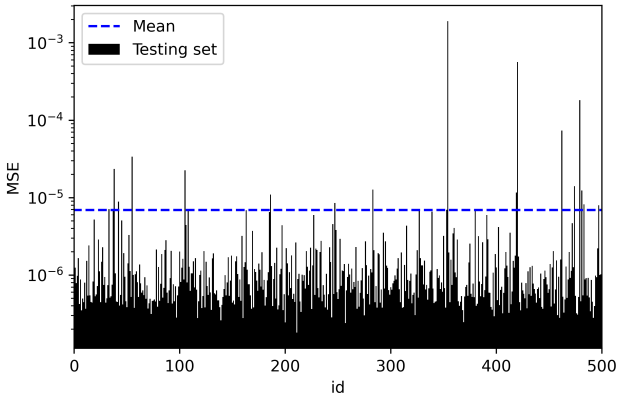


Figure 23: MSE for each of the 500 example of the testing set

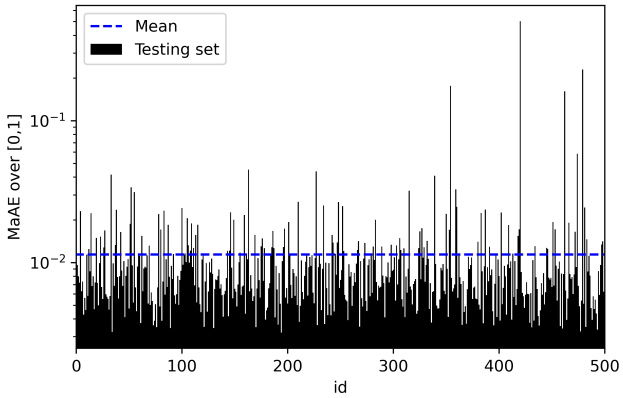


Figure 24: MaAE for each of the 500 example of the testing set

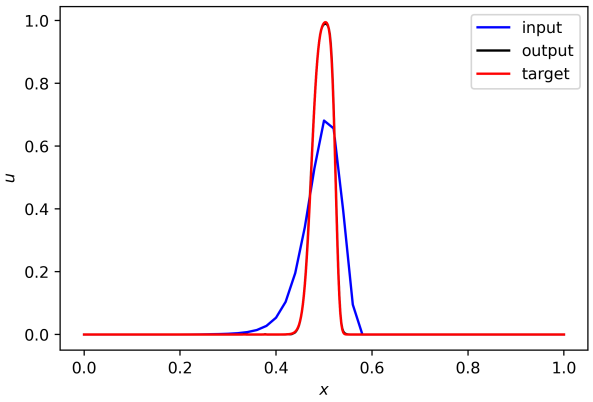


Figure 25: Example taken from the testing set: MSE $= 2.68 \times 10^{-6}$, MaAE $= 0.011$.

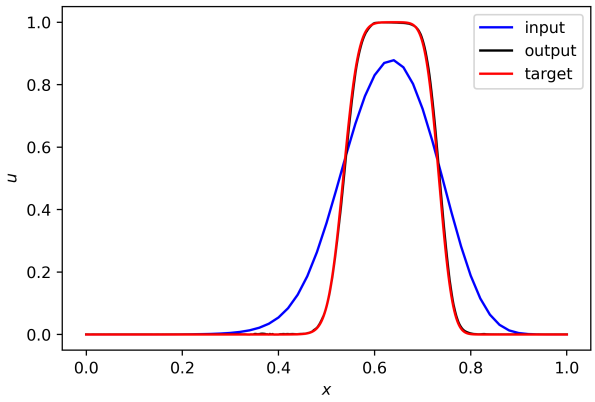


Figure 26: Example taken from the testing set: MSE $= 3.77 \times 10^{-5}$, MaAE $= 0.025$.

3.2 The 2-dimensional case

In this part, numerical approximations of the solution of the transport problem in 2D are considered. The computational complexity grows as N^2 and obtaining numerical solutions (using Eq.(8)) with a fine grid becomes very costly. This is why, in relatively high dimensional spaces, the use of a neural network can be very interesting for saving computational power and time.

3.2.1 Dataset

For the purpose of creating a dataset, we choose the coarse grid to be 40×40 and the fine one 100×100. Overall, a dataset of size 1250 is created, which required a large amount of computational time. The initial functions $u_0(x, y)$ to be transported are the so-called circle ($r = 0.2$, $C = 60$ $x_1 = y_1 = 0.5$) and square ($x_1 = y_1 = 0.4$, $x_2 = y_2 = 0.6$, $C = 60$) functions in 2D defined in Eq.(13) and Eq.(14), respectively. The numerical solutions differ by the speed field $\vec{a}(x, y, t)$, which is constant over time but varies in the range $[-0.02, 0.02] \times [-0.02, 0.02]$. The CFL parameter (Eq.(10)) is also kept constant: CFL $= 0.5$, leading to $M = 32$ and $M = 80$ for the coarse and fine grids, respectively. The final time of the simulation is taken at $T = 10$.

3.2.2 Neural network

The neural network considered for the 2-dimensional case is also a multi-layer perceptron (MLP). The approximations of transported solutions on the 2D grid are flattened for the input layer and unflattened after the output to visualize the prediction of the network. The architecture of the MLP implemented is the following:

- The input layer is of size 1600 (the size of a 40×40 flattened grid), with ReLU activation function.
- There are 4 hidden layers, all with ReLU activation function:
  - The first one is of size 100.
  - The next three have 64 neurons.
- The output layer is of size 10 000 (the size of a 100×100 flattened grid), with sigmoid activation function.

Therefore, a total of 846 049 parameters are trained using the Adam optimizer (learning rate 0.001). The loss function to be minimized by the network is the mean-squared error (MSE).

3.2.3 Results

The main performance results of this experiment are summarized in Table 5, and the MSE metric is shown in Figure 27 as a function of training epochs for both training and validation datasets. Furthermore, Figures 28 and 29 allow to visualize the distribution of the performance metrics over the testing set.
Similarly to the motivational example shown in introduction (Figure 1), we present two testing examples that aim to demonstrate the model capabilities in terms of improving 2D coarse approximations: Figures 30 and 31. As expected, the absolute error plots (Figures 30d, 31d) show that the transition regions are the main contributions to the total error.

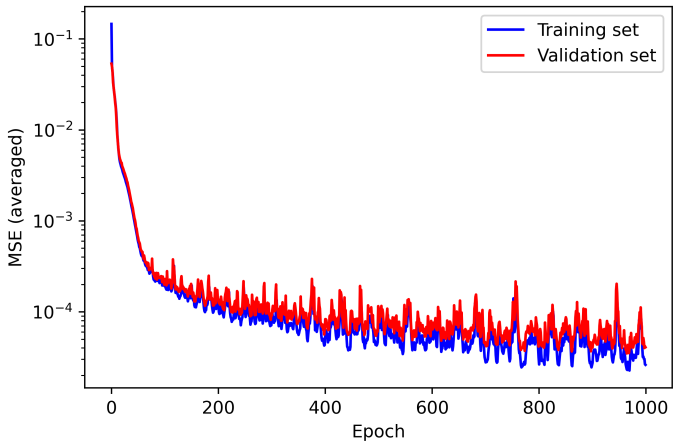


Figure 27: Evolution of the MSE (averaged over samples) through training epochs

av. MSE (train)	av. MSE (test)	max MSE (test)	av. MaAE (test)	max MaAE (test)
2.947×10^{-5}	3.854×10^{-5}	1.711×10^{-4}	5.587×10^{-2}	0.1071

Table 5: Performance of the neural network over the training and testing sets. Two measures are provided: MSE and MaAE. "av." means than the metric is averaged over the whole sample. "max" indicates that the maximum value over the sample is provided.

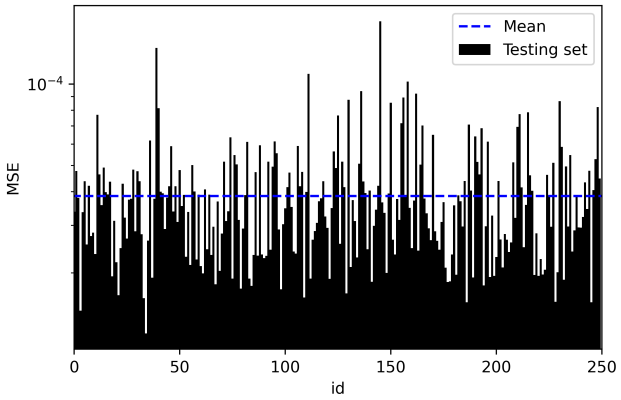


Figure 28: MSE for each of the 250 example of the testing set

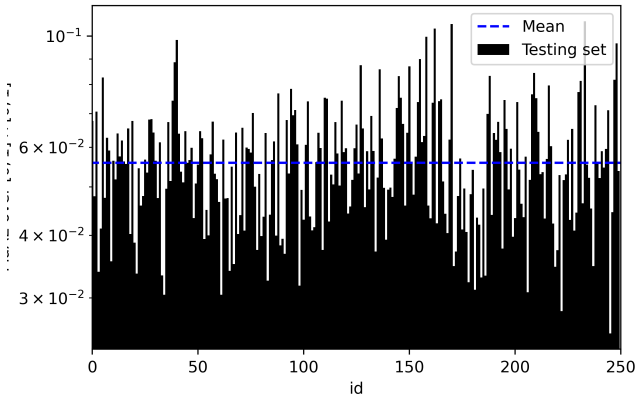
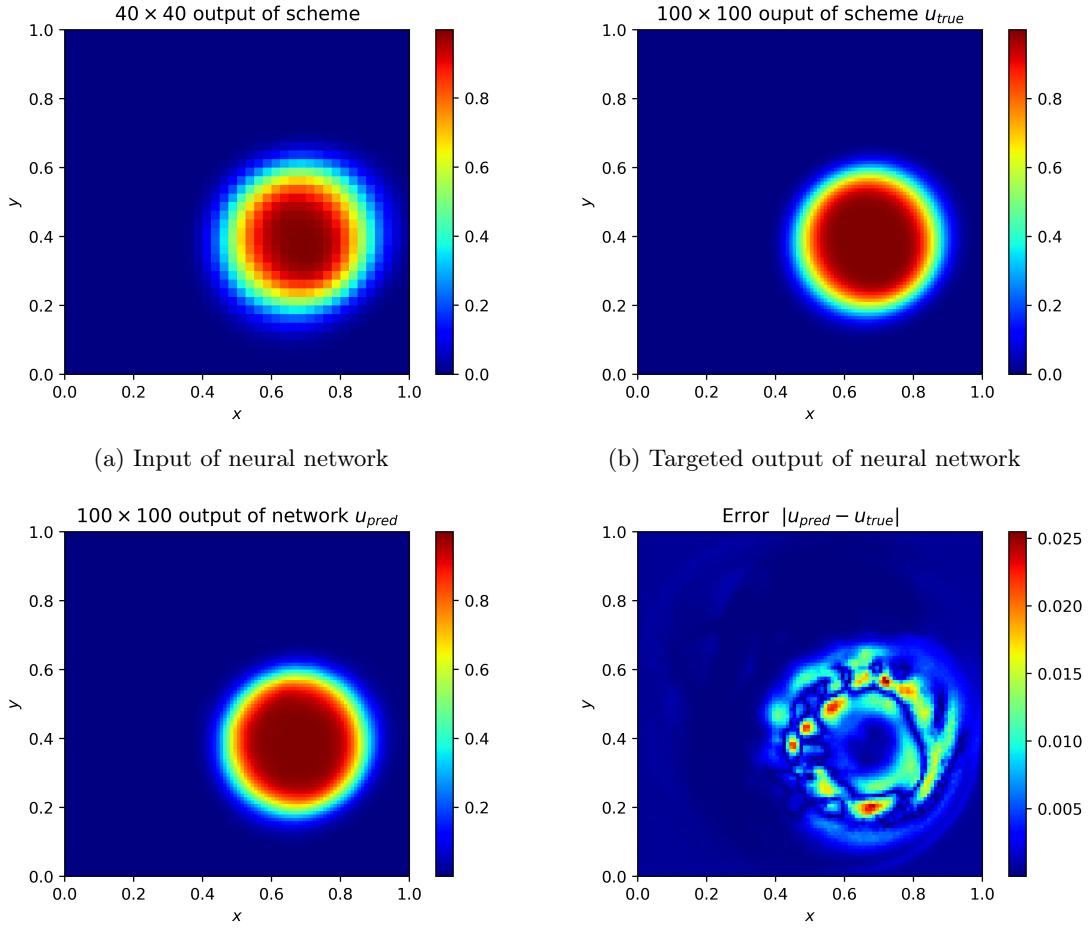


Figure 29: MaAE for each of the 250 example of the testing set



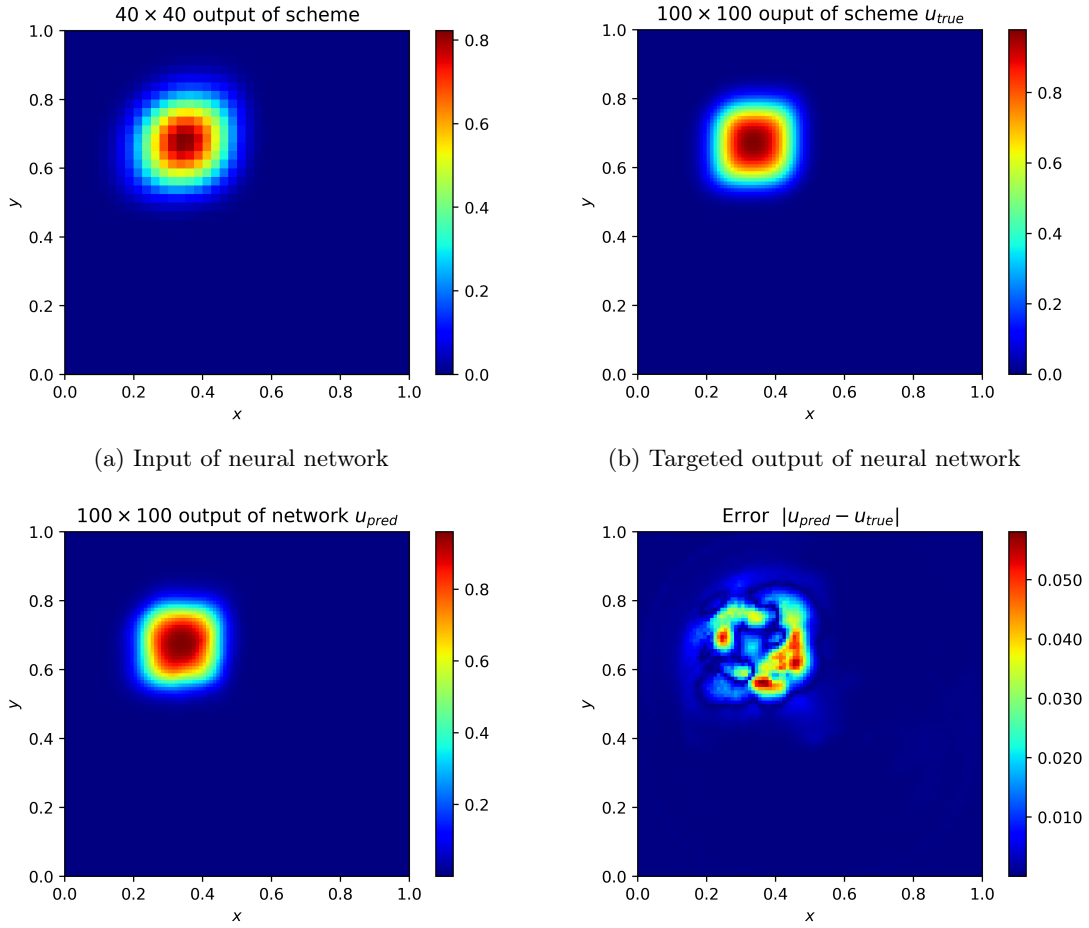
(a) Input of neural network

(b) Targeted output of neural network

(c) Output of neural network

(d) Absolute error between output and target

Figure 30: Visualization of the output of the model and comparison with fine grid solution on a transported circle function at $T = 10$. This example is taken from the testing dataset.



(a) Input of neural network

(b) Targeted output of neural network

(c) Output of neural network

(d) Absolute error between output and target

Figure 31: Visualization of the output of the model and comparison with fine grid solution on a transported square function at $T = 10$. This example is taken from the testing dataset.

4 Discussion

4.1 The 1-dimensional case

In Section 3.1 we focused on numerical experiments for the 1-dimensional transport problem. The neural networks presented for solving multiple tasks demonstrated mixed levels of performance. This part aims to provide a qualitative description and discussion of such results.

4.1.1 Constant speed field

The first model implemented (Section 3.1.3) was trained and tested over step and wall functions only. Table 1 summarizes its good performance in terms of replicating the fine approximation starting from the coarse one, as the maximum absolute error over the test set is of order 10^{-2}. Even though the abscissas of transition are unclear when visualizing the coarse approximation, it appears that the trained model is very accurate when evaluating this quantity: the worst testing example still demonstrate huge improvements at the areas of transition (see Figure 15). This capability is very interesting, especially for such functions with high-slope transitions, and can be thought as an efficient way of correcting numerical diffusion artefacts.
However, it must be kept in mind that there exist high levels of similarity between the train and test sets, since the examples only differ by the initial solution u_0, which is transported with the same speed field and evaluated at the same final time $T = 10$. When the dimensionality of the problem will increase (different u_0, variable speed field, multiple final times), the task of covering the input space to provide a robust training dataset to the model will become very complex.

Furthermore, when the shape of the function is fixed and the solutions are entirely determined by a few sets of parameters (e.g k, x_1, x_2 for the wall function), we could design the network in a way that only these parameters are given as output. This would allow to use the numerical solution with any discretization of space wanted. For example, given a coarse (numerically diffused) approximation of a transported wall function, the network would be able to output good approximations of k, x_1 and x_2 and then plot the solution on any grid size.
In terms of computational complexity, considering an already trained model, the forward pass of the coarse approximation is less costly than the actual computation of the fine solution.

4.1.2 Multiple walls

In Section 3.1.4 we considered initial functions containing multiple wall patterns ($N_{\text{wall}} = 1, \dots, 5$), with the aim to investigate how well the a trained network performs on unseen data. Indeed, it is interesting to assess whether it is possible to train the a model on base functions ($N_{\text{wall}} = 1, 2$) and use it for more complex functions ($N_{\text{wall}} = 3, 4, 5$), which are linear combination of the learned patterns.
In the same line as the constant speed field part, the neural network demonstrated good performance on test examples containing one and two walls. This was expected, since the test and train sets have sufficient levels of similarity. On the other hand, Table 3 showed a poor generalization performance on functions containing 3 to 5 walls. Indeed, the MaAE error is very high (0.987 average) due to the bad handling of transition regions. As shown in Figure 19, the model manages to detect more than three on testing data, but not with good enough accuracy.
Therefore, the results obtained in this experiment are mixed. The generalization performance on unseen data is not very encouraging and seems to suggest that the designed neural networks can not only be trained on base functions. However, the model's capability to detect more wall patterns than what it has been trained on, even with bad accuracy, is an aspect that can be investigated further by changing the NN architecture and designing a better training dataset.

4.1.3 Variable speed field

Furthermore, the 1D transport problem with variable speed field is studied in Section 3.1.5, focusing on wall functions only. The results shown in Table 4 demonstrate interesting levels of performance, despite that the variable speed fields have the capability to deform the initial shape of u_0. The average MaAE on the test set is of order 10^{-2}, which is a similar performance to the constant speed case. The single-peak pattern visible in coarse approximations is really well interpreted by the neural network, which manages to extract the subtleties of the input to reproduce a deformed

output.
Another way of using neural networks for solving this task would be to provide the coefficients of the Fourier decomposition as input, hoping that the model learns to interpret them to predict the deformation of the initial function. Finally, this aspect could also be investigated in the case of a time-dependent speed field, such as the one presented in [7] for the 2D transportation of the circle function. One can imagine a speed field that separates a wall function into two distinct wall shapes, in which case the network should be trained with a good coverage of the possible input space (i.e $N_{\text{wall}} > 1$).

4.2 The 2-dimensional case

The numerical experiments concerning the 2D transport problem are presented in Section 3.2. The performance of the neural network presented is assessed on initial $u_0(x, y)$ such as the circle (Figure 6) and square (Figure 7) functions. As a result, Table 5 summarizes the very good performance results of the model for this task: the average maximum absolute error over the 10^4 grid elements of the 2D space is only of order 10^{-2}. Moreover, visualizing the improvements made by the network between the input and output (Figures 30, 31) confirms the encouraging results of this method. The implemented model seem to have learned the capability to enhance the resolution of coarse approximations by mapping them into fine grids, reminding the recent achievements of DL models in terms of Image Super Resolution (see Introduction).
Note that in this case, a fully connected MLP was implemented but computer vision tasks such as this 2D problem often make use of convolutional neural networks (CNN). A CNN is designed such that it is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters [2]. This possibility have been investigated for the 1-dimensional case (constant speed) and did not improve the (already high) performance of the model. However, it has not been tried for the multiple walls case (where the performance on unseen data is quite poor), neither for the 2-dimensional case. This constitutes a potential direction for further research, in particular for more complex tasks where the examples do not demonstrate such high levels of similarity (the MLP performs already well in the 2D constant speed problem investigated).
Due to the growth of computational complexity in higher dimensions, the dataset is made of so-called "fine approximations" computed on 100×100 grids. However, such solutions still present numerical diffusion artefact and these results should be considered as a motivation to provide better objective outputs to the model so that it learns to generate very fine approximations (e.g 1000×1000 or 10000×10000 grids). Creating a robust training dataset for this problem would require a large amount of computational power, and probably the use of parallel computing techniques.
Finally, as the transport problem has real applications in 3-dimensional spaces, a further research on this topic would be to investigate the potential capabilities of such networks with 3D approximation of solutions.

5 Conclusion

Throughout this semester project, deep learning techniques have been applied to the study of 1 and 2-dimensional transport problem. The main objective was to enhance coarse approximations of numerically transported solution by mapping them into finer grids. The numerical experiments led in Section 3 demonstrated mixed results. First, good performance was found for simple functions transported at constant (3.1, 3.2) and variable speed (3.1.5). However, the multiple-walls experiment (3.1.4) showed that the approach taken might not be suitable for generalization to unseen data, which is less encouraging for the aim of training models on basis functions and expecting them to behave well on linear combinations of those. This work focused on very specific applications to the transport problem and could be generalized by adding many degrees of freedom to the input space: different initial functions, time-dependent speed fields, varying final time. Moreover, the adaptation to other physical problems involving PDEs is straightforward, as the architecture of the network is independent of the problem solved.
On a personal note, this semester project allowed me to learn valuable organizational skills which I am sure will be useful for my future career. I truly enjoyed investigating a subject with very few equivalence in the literature and being able to come up with my own ideas. Through the work provided, I have been able to acquire a decent experience with Keras deep learning API, which have already been of great use in other course projects. Finally, the direct contact with my project supervisor, Prof. Marco Picasso, have been greatly appreciated as it encouraged me to provide a regular work and always put in perspective my progress and findings.

Ressources

The numerical schemes and neural networks implemented in Python code and using Keras API are available at the following public repository:
github.com/EliottZemour/NN-TransportEq-project.

The datasets used for training and testing are made available at:
kaggle.com/eliottzemour/neural-networks-for-tranport-equation.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[3] Matiur Rahman Minar and Jibon Naher. Recent advances in deep learning: An overview. *arXiv preprint arXiv:1807.08169*, 2018.

[4] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[5] Zhihao Wang, Jian Chen, and Steven C.H. Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[6] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, 1967.

[7] Samuel Dubuis and Marco Picasso. An adaptive algorithm for the transport equation with time dependent velocity. *SN Applied Sciences*, 2(9):1581, Aug 2020.

[8] Despoina P. Karadimou and Nikos-Christos Markatos. Study of the numerical diffusion in computational calculations. In Srinivas P. Rao, editor, *Numerical Simulations in Engineering and Science*, chapter 4. IntechOpen, Rijeka, 2018.

[9] RB Lantz. Quantitative evaluation of numerical diffusion (truncation error). *Society of Petroleum Engineers Journal*, 11(03):315–320, 1971.

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[11] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[12] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.