Université de Strasbourg

IRMA Institut de Recherche Mathématique Avancée

RAPPORT DE STAGE

Modélisation 2D de l'équation du transfert radiatif et reconstruction de la densité par un réseau de neurones

Auteur
Roussel Desmond NZOYEM

Maitres de stage
Emmanuel FRANCK
Laurent NAVORET
Vincent VIGON

Enseignant référent
Christophe PRUD'HOMME

Stage réalisé dans le cadre du Master CSMI
du 15 juin 2020 au 15 août 2020
à l'initiative de l'équipe MOCO
au sein de l'UFR de mathématiques et d'informatique

Année académique 2019 - 2020

19 août 2020

Remerciements

Je tiens à remercier dans un premier temps mes maîtres de stages MM. Emmanuel FRANCK, Laurent NAVORET, et Vincent VIGON de m'avoir permis d'effectuer un stage scientifique très enrichissant dans les meilleurs conditions possibles, compte tenu de la situation sanitaires liée au COVID-19.

Je remercie aussi mes camarades de CSMI Guillaume STEIMER et Léo BOIS pour leurs conseils qui ont été instrumental dans l'amélioration de mes résultats.

Table des matières

Liste des symboles

Symbole	Définition	Unité
x	Abscisse	m
y	Ordonnée	m
$\boldsymbol{x}$	Vecteur position	m
$\boldsymbol{\Omega}$	Vecteur direction de propagation des photons	m^2/m^2
ν	Fréquence	Hz
p	Fonction de distribution angulaire de « scattering »	
I	Intensité spécifique de radiation	$W/m^2/sr/Hz$
B	Fonction de Planck	$W/m^2/sr/Hz$
ρ	Densité du milieu	$g\,cm^{-3}$
σ_a	Opacité d'absorption	cm^{-1}
σ_c	Opacité de dispersion (de « scattering »)	cm^{-1}
t	Temps	sh
C_v	Capacité thermique du milieu	Jerk/g/keV
a	Constante de radiation ($E = aT^4$)	$g/cm/sh^2/keV$
c	Vitesse de la lumière	cm/sh
T	Température matière	keV
E	Energie des photons	$g/cm/sh^2$
F	Flux de photons	g/sh^2

Chapitre 1

Introduction

En 2015, le réseau de neurones vainqueur de l'ILSVRC [1] obtient une précision de 97.3 % ce qui conduit les chercheurs à postuler que les machines peuvent identifier les objets dans des images mieux que les humains (**Reference1**). Depuis lors, le domaine du Machine Learning a continué à prendre de l'ampleur. Aujourd'hui ses applications se multiplient dans plusieurs secteurs d'activité parmi lesquelles l'automobile, la finance, le divertissement, et plus important, celui de la santé à travers l'imagerie médicale.

Les tumeurs ont des propriétés de propagation différentes des tissus qui les entourent [2]. Étant donné un domaine avec une onde qui s'y propage, reconstruire sa densité à l'aide du signal temporel mesuré sur ses bords constitue un problème inverse. Les problèmes inverses sont très importants en sciences mathématiques et ont des applications variées en imagerie médicale, radar, vision, etc. Ils sont malheureusement très difficiles à résoudre car ils nécessitent l'utilisation d'algorithme d'optimisation avancés. Les réseaux de neurones artificiels se présente comme une méthode potentiellement moins couteuse mais plus rapide.

Grace à son unité mixte de recherche IRMA, l'UFR de mathématique et d'informatique de l'Université de Strasbourg est un pôle de recherche en mathématiques appliquées. À travers ses équipes MOCO et Probabilités, l'IRMA s'intéresse aux problématiques de modélisation des EDP et de Machine Learning, raison pour laquelle j'ai choisi d'y effectuer mon stage de master 1 CSMI [3]. Au cours de ce stage (du 15 juin au 15 août 2020), j'ai pu m'intéresser au problème inverse de reconstruction de la densité d'un domaine par un réseau de neurones convolutif (CNN).

Ce stage a été suivi par les enseignants-chercheurs MM. Emmanuel FRANCK, Laurent NAVORET, et Vincent VIGON et s'inscrit dans la continuation d'un projet (encadré par la même équipe) qui s'est déroulé du 19 mars au 28 mai 2020. Le projet consistait en la simulation 1D d'un schéma de « splitting » pour le modèle P1 de l'équation du transfert radiatif couplé avec la matière. Le stage quant à lui a essentiellement consisté en la simulation du même schéma en 2D, et en la reconstruction de la densité par un CNN. Plus généralement, ce stage a été l'opportunité pour moi d'apprendre sur les EDP et l'apprentissage profond tout en me familiarisant avec l'interface de programmation de la librairie de réseaux de neurones Keras.

En vue de rendre compte de manière fidèle des deux mois passés au sein de l'IRMA, il apparait logique de présenter en titre de préambule le cadre du stage et son environnement technique. Ensuite il s'agira de présenter les différentes missions et tâches qui j'ai pu effectuer. Enfin je présenterais un bilan du stage, en incluant les différents apports et enseignements que j'ai pu en tirer.

1. ImageNet Large Scale Visual Recognition Challenge
2. Les tissus cancéreux sont généralement plus denses que les tissus sains.
3. Calcul Scientifique et Mathématiques de l'Information

Chapitre 2

Présentation de l'IRMA

Les informations présentées dans cette section sont entièrement issues du site web de l'IRMA.

Créée en 1966 par Jean FRENJEL et Gearges REEEB, l'IRMA [1] est une unité mixte de recherche (UMR 7501) sous la double tutelle du CNRS (à travers l'INSMI [2]) et de l'Université de Strasbourg (à travers l'UFR de Mathématique et d'Informatique).

Dirigée par le professeur Philippe HELLUY, l'IRMA comporte environ 130 membres. On y compte environ 87 chercheurs et enseignants-chercheurs permanents et une quarantaine de membres non permanents repartis en 7 équipes de recherche.

Les activités majeures de l'entreprise sont l'organisation des séminaires, des journées, des colloques et des conférences. Ces activités sont renforcées par les nombreux partenariats qu'elle maintient dans les secteurs académique (Cemosis, Labex IRMIA, etc.) et indutriel (AxesSIM, Electis, etc.).

2.1 Structure

L'organigramme de l'entreprise est représenté à la figure **??**.

2.2 Les équipes

L'équipe MOCO [3] se compose de spécialistes des EDP, de la théorie du contrôle, du calcul scientifique, du calcul haute performance et des statistiques. Les enseignants-chercheurs MM. Emmanuel FRANCK et en Laurent NAVORET y sont responsables des séminaires en équations aux dérivées partielles (EDP).

L'équipe Probabilités est composée d'experts en statistique et calcul de probabilité. Ses membres se retrouvent régulièrement lors du Séminaire Stochastique. C'est à cette équipe qu'appartient M. Vincent VIGON.

Je tiens une fois de plus à remercier les trois chercheurs mentionnés ci-hauts qui ont encadré ce stage. La combinaison des deux équipes dont ils font partie a permis de faire face aux deux aspects de ce stage ; premièrement la simulation d'EDP et deuxièmement l'utilisation des réseaux de neurones.

1. Institut de Recherche Mathématique Avancée
2. Institut National des Sciences Mathématiques et de leurs Interactions
3. MOdélisation et COntrôle

FIGURE 2.1 – Organigramme représentant l'organisation de l'IRMA au mois de mars 2020 (**Reference7**)

Chapitre 3

Simulation 2D

Ayant simulé le schéma de « splitting » en 1D avant le stage, on s'intéresse dans cette partie à sa simulation en 2D. Il s'agit donc de résoudre le problème direct du transfert radiatif avant de passer au problème inverse dans le chapitre suivant. On rappelle brièvement l'équation du transfert radiatif (ETR), le modèle P1 couplé avec la matière, le schéma de « splitting », avant de présenter l'implémentation en C++.

3.1 Le transfert radiatif

On considère un rayonnement transporté par des particules de masse nulle appelés photons. Lorsqu'ils se trouvent en présence de la matière, les photons interagissent avec les atomes. Trois phénomènes sont prépondérants (figure **??**) :

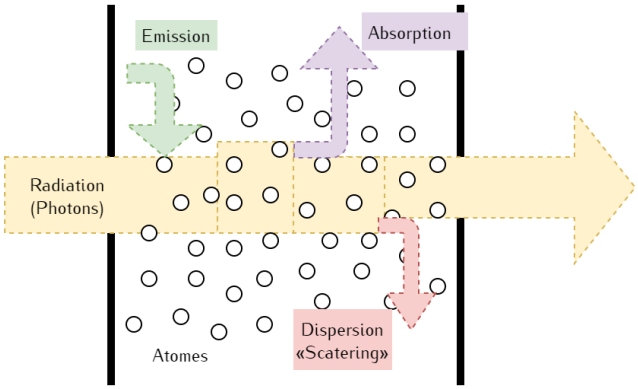


FIGURE 3.1 – Illustration des interactions entre radiation et matière.

— L'**émission** : des photons sont émis en réponse aux électrons excités descendants à des niveaux d'énergie plus bas. Ce phénomène est caractérisé par l'opacité d'émission σ_e. Il s'agit de l'inverse du libre parcours moyen d'émission [1]. Plus la température matière est élevée, plus ce phénomène est important.
— L'**absorption** : à l'inverse, certains photons sont absorbés, les électrons deviennent plus excités (ou se libèrent complètement de leurs atomes), et la matière se réchauffe. Ce phénomène se caractérise par l'opacité d'absorption σ_a. Lorsqu'on est à l'équilibre thermique, $\sigma_a = \sigma_e$.
— La **dispersion** (ou « **scattering** » ou parfois **diffusion**) : certains photons sont déviés de leur trajectoire originale par la matière. Ce phénomène se caractérise non seulement par son

1. Le libre pacours moyen d'émission represente la distance moyenne entre deux émissions consecutives de photons. Les libres parcours moyen d'absorption et de dispersion sont définis de maniere similaire.

opacité de « scattering » σ_c [2], mais aussi par une fonction de distribution angulaire décrivant la manière dont les photons sont déviés (**Reference3**).

L'équation du transfert radiatif (ETR) (équation **??**) représente un bilan d'énergie lié au rayonnement au niveau microscopique. Nous nous placerons dans le cas particulier d'équilibre thermodynamique local (ETL) [3]. L'équilibre radiatif [4] quant à lui sera considéré plus tard comme condition initiale pour les simulations.

$$\begin{aligned}\frac{1}{c}\frac{\partial}{\partial t}I(t,\mathbf{x},\mathbf{\Omega},\nu)+\mathbf{\Omega}\cdot\nabla_{\mathbf{x}}I(t,\mathbf{x},\mathbf{\Omega},\nu)&=\sigma_a(\rho,\mathbf{\Omega},\nu)\left(B(\nu,T)-I(t,\mathbf{x},\mathbf{\Omega},\nu)\right)\\&+\frac{1}{4\pi}\int_0^\infty\int_{S^2}\sigma_c(\rho,\mathbf{\Omega},\nu)p(\mathbf{\Omega}'\to\mathbf{\Omega})\left(I(t,\mathbf{x},\mathbf{\Omega}',\nu)-I(t,\mathbf{x},\mathbf{\Omega},\nu)\right)\,d\mathbf{\Omega}'\,d\nu\end{aligned}\tag{3.1}$$

Où I est l'intensité spécifique de radiation et p est la fonction de distribution angulaire de « scattering » (telle que $\oint p(\mathbf{\Omega}'\to\mathbf{\Omega})\,d\mathbf{\Omega}'=1$). Les autres termes sont définis dans la liste des symboles de la page **??**.

Plusieurs modèles ont été créés pour modéliser l'ETR avec différents niveaux de précision. Le modèle P1 est un modèle macroscopique [5] aux moments (d'ordre 2), linéaire et hyperbolique. Vu que l'énergie du rayonnement n'est pas conservée durant son interaction avec la matière, il faut coupler le modèle P1 avec une équation régissant l'énergie de la matière. On utilisera une équation d'énergie matière simplifiée qui ne tient compte que des termes d'échange avec le rayonnement. Le modèle P1 couplé avec la matière est présenté ci-bas (**Reference2**) :

$$\begin{cases}\partial_t E+c\ \operatorname{div}\mathbf{F}=c\sigma_a\left(aT^4-E\right)\\ \partial_t\boldsymbol{F}+c\ \nabla E=-c\sigma_c\mathbf{F}\\ \rho C_v\partial_t T=c\sigma_a\left(E-aT^4\right)\end{cases}\tag{3.2}$$

Dans l'équation **??**, σ_a et σ_c sont écrits sans indiquer leurs arguments [6] afin de faciliter la lisibilité. E, F, et T représentent l'énergie des photons, le flux de photons, et la température matière. Partant de l'équation **??**, E et T sont définis de la manière suivante :

$$\begin{aligned}E(t,\mathbf{x})&=\frac{4\pi}{c}\int_0^\infty\int_{S^2}I(t,\mathbf{x},\mathbf{\Omega},\nu)\,d\mathbf{\Omega}\,d\nu\\ \mathbf{F}(t,\mathbf{x})&=\frac{4\pi}{c}\int_0^\infty\int_{S^2}\mathbf{\Omega}I(t,\mathbf{x},\mathbf{\Omega},\nu)\,d\mathbf{\Omega}\,d\nu\end{aligned}$$

Comme on peut le voir à travers la définition de E et F, notre modèle est dit "gris" car nous l'intégrons sur tout le spectre de fréquence. En effet, nous nous intéressons au rayonnement à travers son bilan d'énergie transporté par le flux radiatif. Sur ce point, la version du modèle P1 que nous avons utilisé est mois précise qu'un modèle microscopique basé soit sur une méthode de Monte-Carlo ou une méthode des ordonnées discrètes. Néanmoins notre modèle présente l'avantage d'être très peu couteux et relativement facile à implémenter (**Reference3**).

2. σ_a et σ_c sont definis de manière similaire à σ_e
3. État dans lequel on peut définir une température pour chaque point du domaine, et l'émission est décrite par la fonction de Planck (**Reference3**).
4. Il se produit si la matière est à l'équilibre avec le rayonnement. Si on est dans l'ETL, les photons sont émis suivant la fonction de Planck à la température de la manière.
5. Il ne prend en compte que les variables d'espace et de temps et est obtenu par intégration de termes microscopique tels que I par rapport à la fréquence et la direction.
6. Juste ρ et T si on se place dans l'ETL.

3.2 Schéma de splitting

Le modèle P1 tend vers une équation de diffusion lorsque les opacités d'absorption (σ_a) et de dispersion (σ_c) sont élevées (de façon à ce que $c/\sigma_a \approx 1$). Ce cas limite est appelé **approximation de diffusion** et est indiqué à l'équation **??**.

$$\partial_t \left(aT^4 + \rho C_v T \right) - \text{div} \left(\frac{acT^3}{\sigma_a} \nabla T \right) = O\left(\frac{1}{c} \right) \tag{3.3}$$

Les schémas classiques tels que le schéma de Rusanov ne sont pas assez précis pour capturer cette propriété. Le schéma en deux étapes (ou de « splitting ») proposé par M. FRANCK permet de pallier ce problème. Ses deux étapes sont résumées ci-bas (**Reference4**).

3.2.1 Etape 1

La première étape (dite étape de couplage ou d'équilibre, ou de relaxation de la température) permet de régler la température sur chaque maille (indépendamment des autres mailles). On ne considère que les équations où la température est impliquée (EDP 1 et 3 du système d'équation **??**), en fixant la valeur du flux sur chaque maille. Il s'agit d'une méthode de point fixe qui est toujours définie (**Reference2**).

Le domaine rectangulaire est supposé discrétisé en $N \times M$ mailles uniformes ; on se trouve sur la maille j (voir figure **??** pour les détails de la discrétisation) à l'étape d'itération n. On pose donc $\Theta = aT^4$ et on obtient le système :

$$\begin{cases} \dfrac{E_j^{q+1} - E_j^n}{\Delta t} = c\sigma_a(\Theta_j^{q+1} - E_j^{q+1}) \\ \rho_j C_v \mu_q \dfrac{\Theta_j^{q+1} - \Theta_j^n}{\Delta t} = c\sigma_a(E_j^{q+1} - \Theta_j^{q+1}) \end{cases}$$

Où $\mu_q = \dfrac{1}{T^{3,n} + T^n T^{2,q} + T^q T^{2,n} + T^{3,q}}$.

L'étape revient à résoudre un système de Cramer. On obtient donc :

$$\begin{cases} E_j^{q+1} = \dfrac{\alpha E_j^n + \beta\gamma\Theta_j^n}{1 - \beta\delta} \\ \Theta_j^{q+1} = \dfrac{\gamma\Theta_j^n + \alpha\delta E_j^n}{1 - \beta\delta} \end{cases} \tag{3.4}$$

Avec $\alpha = \dfrac{1}{\Delta t \left(\frac{1}{\Delta t} + c\sigma_a \right)}$, $\beta = \dfrac{c\sigma_a}{\frac{1}{\Delta t} + c\sigma_a}$, $\gamma = \dfrac{\rho_j C_v \mu_q}{\Delta t \left(\frac{\rho_j C_v \mu_q}{\Delta t} + c\sigma_a \right)}$ et $\delta = \dfrac{c\sigma_a}{\frac{\rho_j C_v \mu_q}{\Delta t} + c\sigma_a}$.

On itère ainsi sur q jusqu'à ce que E et Θ convergent respectivement vers E^* et Θ^*. F reste inchangé durant cette étape.

3.2.2 Etape 2

Il s'agit ici de résoudre les deux premières EDP du système d'équation **??**. Avant d'attaquer le schéma de « splitting », on note que les équations à résoudre sont hyperboliques et que la méthode des volumes finis est donc la mieux adaptée. On se place sur une maille j caractérisée par son

volume Ω_j [7].

$$\begin{cases} \partial_t \int_{\Omega_j} E + c \int_{\Omega_j} \operatorname{div} \mathbf{F} = 0 \\ \partial_t \int_{\Omega_j} \mathbf{F} + c \int_{\Omega_j} \nabla E = -c\sigma_c \int_{\Omega_j} \mathbf{F} \end{cases}$$

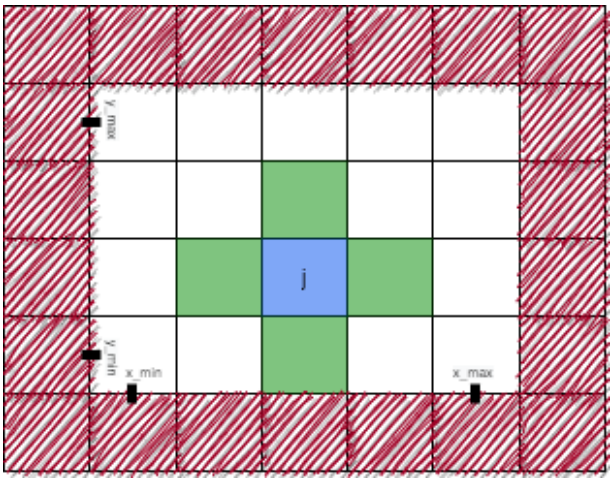
On définit une normale $\mathbf{n}_j$ à la surface Ω_j [8] et on applique le théorème de la divergence. On moyenne les intégrales sur chaque maille pour obtenir :

$$\begin{cases} \partial_t E_j + \dfrac{c}{|\Omega_j|} \displaystyle\int_{\partial\Omega_j} (\mathbf{F}, \mathbf{n}_j) = 0 \\ \partial_t \mathbf{F}_j + \dfrac{c}{|\Omega_j|} \displaystyle\int_{\partial\Omega_j} (E, \mathbf{n}_j) = \dfrac{c\sigma_c}{|\Omega_j|} \displaystyle\int_{\Omega_j} \mathbf{F} \end{cases} \tag{3.5}$$

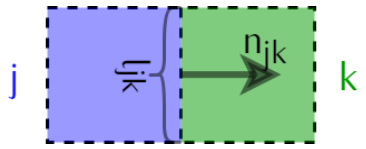
Avec

$$E_j(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} E(t, \mathbf{x}) \quad \text{et} \quad \mathbf{F}_j(t) = \frac{1}{|\Omega_j|} \int_{\Omega_j} \mathbf{F}(t, \mathbf{x})$$

Nous devons maintenant retourner sur le maillage en définissant les différents flux numériques impliqués. Durant cette étape, il faut considérer l'ajout de mailles fantômes (figure **??**), ce qui porte le nombre total de mailles à $(N+2) \times (M+2)$.



(A) Maillage 2D



(B) Interaction entre deux mailles j et k

FIGURE 3.2 – Discrétisation du maillage 2D. Sur la figure (A), on peut observer les mailles dites "fantômes" hachurées en rouge. Les quatre mailles voisines d'une maille j sont indiquées en vert. Les nombres de mailles suivant la verticale N et suivant l'horizontale M sont choisis telle que le maillage soit uniforme i.e $\Delta x = \frac{x_{max}-x_{min}}{N} = \frac{y_{max}-y_{min}}{M} = \Delta y$. Sur la figure (B), on observe la représentation d'une des normales n_{jk} de la mailles j. On peut aussi observer la longueur caractéristique l_{jk}.

7. On utilisera généralement le terme "maille" vu qu'en 2D, le "volume" Ω_j est en réalité une surface.
8. Il s'agit en réalité d'une série de quatre normales définies par rapport aux mailles voisines de j (voir figure **??**).

Les flux numériques entre une maille j et sa maille voisine k à l'étape d'itération n sont définis comme suit (**Reference4**) :

$$\begin{aligned}\left(\mathbf{F}_{jk}, \mathbf{n}_{jk}\right) &= l_{jk} M_{jk} \left(\frac{\mathbf{F}_j^n \cdot \mathbf{n}_{jk} + \mathbf{F}_k^n \cdot \mathbf{n}_{jk}}{2} - \frac{E_k^n - E_j^n}{2} \right) \\ \left(E_{jk}, \mathbf{n}_{jk}\right) &= l_{jk} M_{jk} \left(\frac{E_j^n + E_k^n}{2} - \frac{\mathbf{F}_k^n \cdot \mathbf{n}_{jk} - \mathbf{F}_j^n \cdot \mathbf{n}_{jk}}{2} \right) \mathbf{n}_{jk}\end{aligned}$$

En posant :

$$\begin{aligned}\mathbf{S}_j &= -\left(\sum_k M_{jk} \sigma_{jk} \right) \mathbf{F}_j^{n+1} \\ \mathbf{S}'_j &= \frac{1}{|\Omega_j|} \left(\sum_k l_{jk} M_{jk} \mathbf{n}_{jk} \right) E_j^n \\ M_{jk} &= \frac{2}{2 + \Delta x \sigma_{jk}} \\ \sigma_{jk} &= \frac{1}{2} \left(\sigma_c(\rho_j, T_j^n) + \sigma_c(\rho_k, T_k^n) \right)\end{aligned}$$

Cette étape du schémas s'écrit donc sous la forme :

$$\begin{cases} \dfrac{E_j^{n+1} - E_j^*}{\Delta t} + \dfrac{c}{|\Omega_j|} \displaystyle\sum_k \left(\mathbf{F}_{jk}, \mathbf{n}_{jk}\right) = 0 \\ \dfrac{\mathbf{F}_j^{n+1} - \mathbf{F}_j^*}{\Delta t} + \dfrac{c}{|\Omega_j|} \displaystyle\sum_k \left(E_{jk}, \mathbf{n}_{jk}\right) - c\mathbf{S}'_j = c\mathbf{S}_j \end{cases}$$

Qui se réécrit comme suit :

$$\begin{cases} E_j^{n+1} = E_j^* + \alpha \displaystyle\sum_k \left(\mathbf{F}_{jk}, \mathbf{n}_{jk}\right) \\ \mathbf{F}_j^{n+1} = \beta \mathbf{F}_j^* + \gamma E_j^n + \delta \displaystyle\sum_k \left(E_{jk}, \mathbf{n}_{jk}\right) \end{cases} \tag{3.6}$$

Avec :

$$\alpha = -\frac{c\Delta t}{|\Omega_j|}, \quad \beta = \frac{1}{\Delta t} \left(\frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1}, \quad \gamma = \frac{c}{|\Omega_j|} \left(\frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1} \left(\sum_k l_{jk} M_{jk} \mathbf{n}_{jk} \right)$$
$$\text{et} \quad \delta = -\frac{c}{|\Omega_j|} \left(\frac{1}{\Delta t} + c \sum_k M_{jk} \sigma_{jk} \right)^{-1}$$

La condition de CFL $\Delta t < \dfrac{\Delta x}{c}$ est nécessaire pour assurer la stabilité du schéma. Lors de l'implémentation en C++, on remarquera qu'en pratique, il faut prendre $\Delta t < 0.5 \times \dfrac{\Delta x}{c}$.

3.3 Implémentation en C++

Le code de calcul 2D a été développé durant la cinquième semaine du stage. En fournissant des paramètres (physiques, géométriques, etc.) au programme, l'on peut récupérer les signaux temporels E, F et T sur les quatre bords du domaine. Il permet aussi d'exporter les signaux sur l'entièreté du domaine en tout temps. Ces signaux peuvent ensuite être visualisés sous forme d'animation à l'aide d'un notebook Jupyter construit à cet effet. L'exécutable pour effectuer des simulations se nomme `transfer` et est disponible avec le reste du code sur le dépôt Github projet-inverse-2d.

3.3.1 Configuration du modèle

L'exécutable nécessite un fichier de configuration pour s'exécuter (.cfg, .txt, etc.). Les paramètres obligatoires à définir sont indiqués à la figure **??**. Les unités sont celles de la liste des symboles de la page **??**. Des détails supplémentaires sont donnés en annexe **??**.

```
x_min 0
x_max 1
y_min 0
y_max 1
N 90

c 299
a 0.01372
C_v 0.14361

CFL 0.5
precision 1e-6
t_0 0
t_f 0.01

rho crenau(0.5,0.5,0.1,10)
sigma_a rho*T
sigma_c rho*T

E_0 0.01372*(5^4)
F_0_x 0
F_0_y 0
T_0 5
```

```
E_u neumann
F_u_x neumann
F_u_y neumann
T_u neumann

E_d neumann
F_d_x neumann
F_d_y neumann
T_d neumann

E_l ponctuel(0.4,0.6)
F_l_x 0
F_l_y 0
T_l 5

E_r neumann
F_r_x neumann
F_r_y neumann
T_r neumann

export_file data/df_simu.csv
export_mode dataframe
write_mode truncate
```

FIGURE 3.3 – Exemple d'un fichier de configuration. Ici ne sont représentés que les 38 paramètres obligatoires pour faire tourner une simulation et l'exporter. On remarque que le nombre de mailles en horizontale M n'est pas inclu ; il est calculé automatique afin d'obtenir un maillage uniforme. Le résultat produit par ce fichier est présenté aux figures **??** et **??**.

3.3.2 Sauvegarde des données

Comme mentionné ci-haut, on dispose de deux options pour sauvegarder les résultats de la simulation :

— Sous le forme **CSV** : ce mode permet une visualisation facile des résultats à l'aide de notebook. Il est très couteux en espace mémoire et nécessite la librairie Pandas pour la lecture sous forme de dataframe. Cette opération est lente et prend une quantité non négligeable de RAM, ce qui peut nuire à l'usage qu'on veut faire des données.

— Sous le format **SDS**[9] : ce format binaire ne sauvegarde que les informations les plus importantes de la simulation. En l'occurrence la source utilisée, la densité du domaine, et les différents signaux sur les bords du domaine. Il est particulièrement intéressant pour générer les données nécessaires à l'apprentissage. Les détails concernant ce format sont donnés en annexe **??**.

3.4 Résultats

Quelques résultats obtenus sont présentés ici. Le premier résultat (figures **??** et **??**) est obtenu avec le fichier de configuration de la figure **??**. La source est une onde sinusoïdale placée en E sur une portion de la gauche. La densité a la forme d'un signal en créneau égale à 10 sur le créneau et 0.1 en dehors. Les opacités d'absorption et de dispersion sont proportionnelles à la densité.

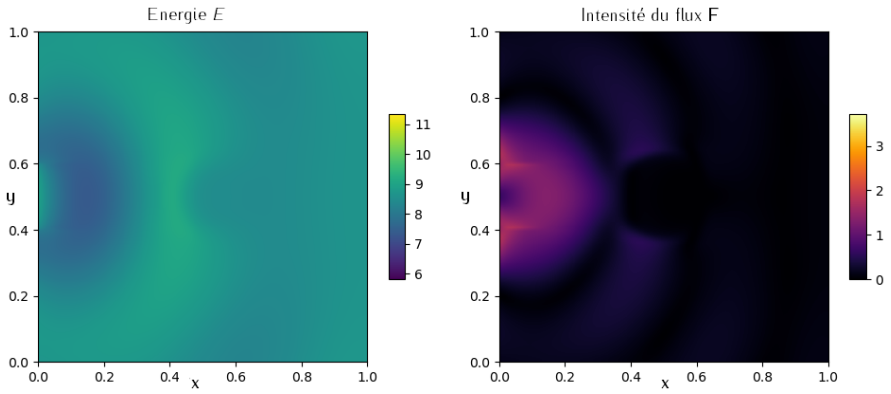


FIGURE 3.4 – Visualisation de l'énergie et de l'intensité du flux des photons au temps final pour un domaine avec une densité en forme de créneau circulaire (vu du haut). Cette figure correspond au résultat obtenus avec la configuration **??**.

La figure **??** permet d'observer une absorption presque totale du signal au niveau du créneau due à la forte valeur des opacités d'absorption et de dispersion. En ce sens, le créneau (saut de densité) très opaque agit comme un obstacle à la propagation du signal. L'évolution de l'énergie sur les bords du domaine (figure **??**) traduit aussi bien l'effet qu'a la densité sur la propagation du signal.

9. Source - Densité - Signal

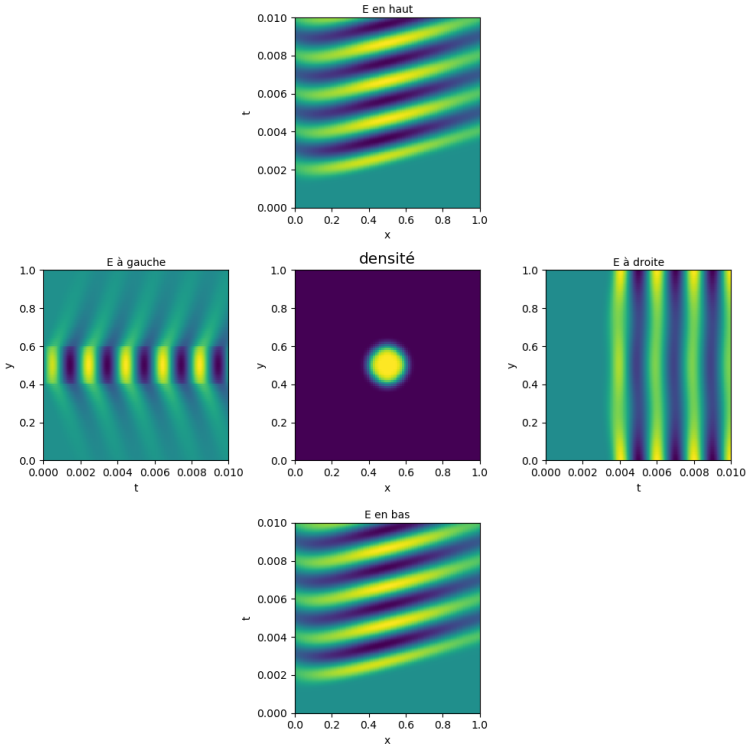


FIGURE 3.5 – Évolution de l'énergie sur les bords (vue du haut). La cause du problème direct (le saut de densité) est illustrée au milieu de l'image, et son effet sur l'énergie E est présentée aux alentours. Les indices x, y, et t représentent respectivement l'abscisse, l'ordonnée et le temps. Les autres figures associées à ce cas sont **??** et **??**.

Nous testons ensuite notre modèle sur le cas très particulier de l'approximation de diffusion (figures **??**, **??** et **??**). Le bord gauche est continûment chauffé ($E_{gauche} = a * ((T_{gauche} + 1)^4)$). La densité est un signal en forme de créneau rectangulaire (vu du haut) valant 10. En dehors de ce créneau, la densité vaut 0.1. Théoriquement, la limite de diffusion s'observe pour $c/\sigma_c \approx 1$ (voir équation **??**), mais cela pose des problèmes de visualisation en utilisant les coefficients que nous avons choisis. On prend donc $\sigma_a = \sigma_c = 100 \times \rho$ ce qui donne $c/\sigma_c \approx 30$ en dehors de l'obstacle.

On confirme effectivement l'effet de diffusion du signal dans le domaine. Nous pouvons à présent passer à l'apprentissage. Nous nous placerons dans des conditions proches de celles de la configuration **??**. L'apprentissage utilisera aussi les données de simulation 1D non présentées ici. Les résultats 1D ont été présentés et analysés dans le rapport de fin de projet de CSMI.

```
x_min 0
x_max 1
y_min 0
y_max 1
N 90

c 299
a 0.01372
C_v 0.14361

CFL 0.5
precision 1e-6
t_0 0
t_f 0.01

rho crenau(0.5,0.5,0.1,10)
sigma_a rho*100
sigma_c rho*100

E_0 0.01372*(5^4)
F_0_x 0
F_0_y 0
T_0 5
```

```
E_u neumann
F_u_x neumann
F_u_y neumann
T_u neumann

E_d neumann
F_d_x neumann
F_d_y neumann
T_d neumann

E_l 0.01372*(6^4)
F_l_x 0
F_l_y 0
T_l 5

E_r neumann
F_r_x neumann
F_r_y neumann
T_r neumann

export_file data/df_simu.csv
export_mode dataframe
write_mode truncate
```

FIGURE 3.6 – Configuration utilisée pour illustrer la limite de diffusion. L'option pour obtenir un obstacle en forme de rectangle n'est pas insérable dans le fichier de configuration, cela se fait directement dans le code de calcul.

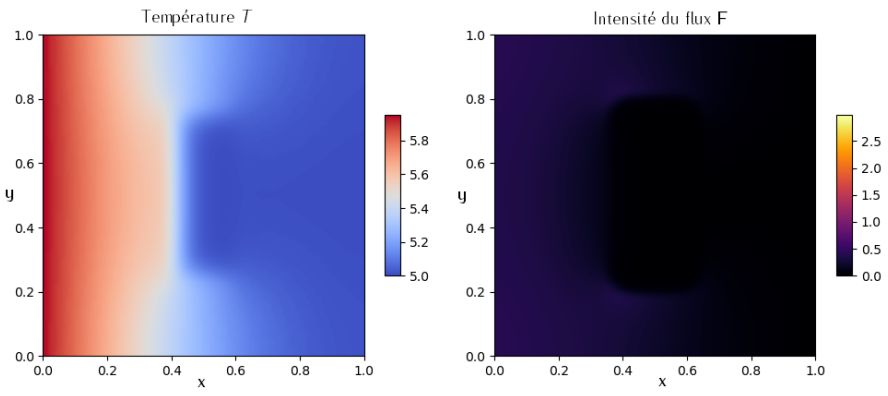


FIGURE 3.7 – Visualisation de la température du domaine et de l'intensité du flux des photons au temps final pour la limite de diffusion. La densité à une forme de créneau rectangulaire comme représenté sur l'image du milieu de la figure **??**.

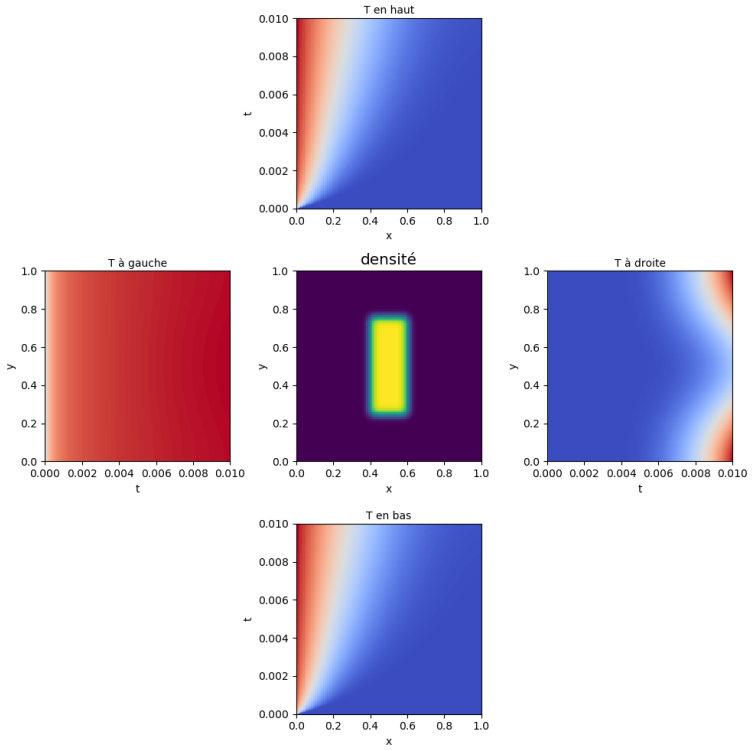


FIGURE 3.8 – Évolution de la température sur les bords illustrant l'effet de diffusion. Tout comme à la figure **??**, l'expression des opacités pour cette simulation est $\sigma_a = \sigma_c = 100 \times \rho$ afin d'obtenir un maximum de diffusion en dehors de l'obstacle mais une absorption totale sur l'obstacle.

Chapitre 4

Apprentissage

L'objectif de cette section est de reconstruire la densité en connaissant l'énergie E, le flux F, et la température T sur les bords d'un domaine aux cours du temps. Il s'agit donc d'un problème inverse de régression [1] ; la densité faisant partie de la cause du problème direct et les signaux temporels de l'effet. Nous ferons une simplification de taille : **la densité est un signal en créneau lissé** [2] (on utilisera un créneau cylindrique i.e ayant la forme d'un cercle vu du haut). Ainsi, reconstruire la densité revient juste à prédire la position et la hauteur du créneau. La valeur de la densité en dehors du créneau sera connue.

Nous recherchons donc une fonction f^{-1} (inverse de la fonction f définissant le problème direct) telle que $y = f^{-1}(X)$. Où y représente la densité(plus précisément les attributs de son créneau), et X les signaux sur les bords. Mais le caractère naturellement mal posé des problèmes inverses rend difficile la détermination de f^{-1}. On procède donc à une approximation de f^{-1} notée $\hat{f}^{-1}$, à l'aide d'un ANN [3]. En notant θ les paramètres de l'ANN, on cherche $\hat{y}$ telle que $\hat{y} = \hat{f}^{-1}(X, \theta)$.

4.1 Description des entrées/sorties

4.1.1 En 1D

L'aspect d'une entrée et d'une sortie 1D est représentée à la figure **??**. Les entrées sont composées des signaux temporels E, F, et T. Suivant chacun de ces canaux, il faut normaliser les entrées avant de les nourrir au réseau de neurones. Vu que le signal provient de la gauche, les entrées ne sont constituées que du signal récupéré sur le bord droit du domaine. La talle d'un exemple est présentée à la figure **??**.

Comme mentionné plus haut, nous avons fait quelques simplifications sur la nature de la sortie. Il s'agit d'un vecteur de seulement deux scalaires représentant la position et la hauteur du saut de densité :

— **Position** : il s'agit de l'abscisse de l'obstacle choisi de façon à ce que le créneau se situe entièrement dans le domaine (position de son centre comprise dans l'intervalle $[0.07, 0.92]$).
— **Hauteur** : comprise dans l'intervalle $[1.05, 9.99]$. La valeur de la densité en dehors du créneau est de 1.

1. Une classification peut tout aussi être implémentée (voir paragraphe **??**). Sauf mention du contraire, lorsqu'on parlera d'apprentissage, il s'agira du problème de régression.
2. Un créneau sur la densité sera aussi appelé saut de densité, ou pic de densité, ou obstacle sur la densité
3. Réseau de neurones artificiel

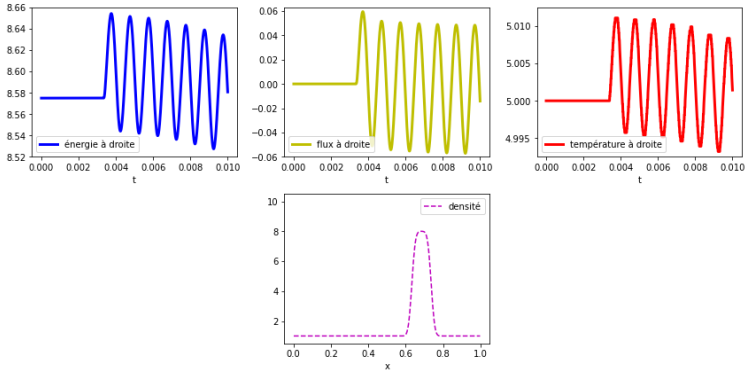


FIGURE 4.1 – Visualisation d'une entrée de l'ANN (en haut) et d'une sortie (en bas) en 1D. Seul le signal sur la droite du domaine est utilisé. Tout les 3 canaux E, F et T sont représentés ici.

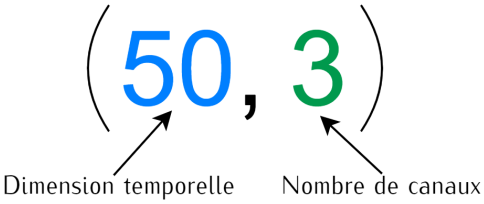


FIGURE 4.2 – Taille d'une entrée en 1D. La dimension spatiale a été échantillonnée de 907 à 50 itérations. Les 3 canaux désignent les signaux E, F et T.

4.1.2 En 2D

Une entrée 2D contient considérablement plus d'information. Les 3 signaux E, F et T sur les 4 bords y sont inclus. On y inclut aussi chacun des groupes de signaux correspondants aux 4 positions de la source sur la gauche. En effet, une entrée correspond à 4 simulations effectuées chacune avec la source à une position différente comme on peut le voire à la figure **??**. La taille d'une entrée est donnée à la figure **??**.

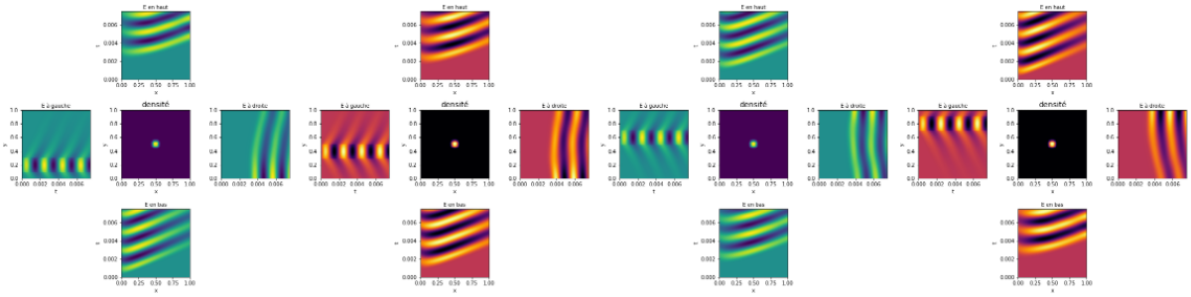


FIGURE 4.3 – Visualisation d'une entrée (aux alentours des quatre images) et d'une sortie (aux milieux) en 2D. On peut voire les postions des 4 sources utilisées à tour de rôle pour former une seule entrée. Ici n'est représentée que l'énergie (qui constitue 1/3 des canaux) sur les 4 bords. Les images des densités ci-contre ont été obtenues par interpolation bilinéaire d'une image initiale très grossière (28×28).

Compare à la 1D, il faut rajouter l'ordonnée du saut de densité à la liste des scalaires prédits. Une sortie est un vecteur de taille 3. Afin d'éviter des cas trop extrêmes, les valeurs observées

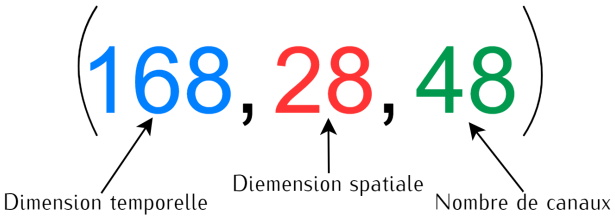


FIGURE 4.4 – Taille d'une entrée en 2D. Contrairement à la 1D, il faut tenir compte de la dimension spatiale qui correspond au nombre de mailles sur chaque bord du domaine rectangulaire ($N = M = 28$). Le nombre de canaux passe à 48 car il faut déjà tenir compte des 3 canaux originaux (E, F, et T), ensuite de chacun de 4 bords du domaine, et enfin des 4 positions de la source.

(ou labels pour l'apprentissage) sont convenablement choisies au moment de la génération des données :

— **Abscisse** : comprise dans l'intervalle $[0.2, 0.8]$ afin de récupérer une réponse non aberrante sur chacun des 4 bords du domaine.
— **Ordonnée** : comprise aussi dans l'intervalle $[0.2, 0.8]$.
— **Hauteur** : comprise dans l'intervalle $[0.1, 10]$. La valeur 0.1 est aussi la valeur de la densité en dehors de son créneau ; sa faible valeur permet d'éviter l'absorption complète de l'onde sur le domaine.

Des jeux de données complets 1D/2D ont été sauvegardés sur Google Drive ; et les détails pour les récupérer et les traiter sont donnés en Annexe **??**.

4.2 Architecture générale

Un réseau de neurones artificiel[4] (ANN) est un système computationnel base sur le réseau de neurones biologique. L'apprentissage profond[5] (DNN) permet de résoudre des problèmes en Machine Learning que les méthodes telles que la régression linéaire, SVM, etc. ne peuvent pas. Il réussit cela en introduisant des représentations des données qui s'expriment sous forme d'autres représentations, plus simples cette fois.

Les réseaux profonds en aval[6] (ou MLP[7]) constituent l'exemple de choix en apprentissage profond. Il s'agit juste d'une fonction (composition de différentes fonctions) faisant correspondre une série d'entrée à une sérié de sortie ($f^{-1} = f_1 o f_2 o \cdots o f_n(X)$). Un MLP est constitué de plusieurs couches (assimilables aux fonctions f_1, f_2, .., f_n précédentes) apprenant chacune un aspect particulier des données. On distingue une couche d'entrée, une ou plusieurs couches cachées, et une coche de sortie (voir figure **??**).

Les réseaux de neurones convolutifs (CNN) sont une forme de MLP spécialisés dans le traitement des données en forme de grille. Par exemple des séries en temps qui peuvent êtres vues comme des grilles 1D (l'axe de temps) prenant des données (vecteur de données) à intervalle de temps régulier (**Reference5**). Ils sont donc particulièrement adaptés à la reconstruction de la densité partant des signaux temporels E, F, et T.

L'architecture CNN de base pour notre apprentissage a été proposée par M. VIGON. Nous utiliserons deux variantes : DRNN[8] 1 (figure **??**) et DRNN 2 (figure **??**). Les architectures seront

4. Ou juste réseau de neurones dorénavant
5. Réseaux de neurones composé d'un nombre relativement élevé de couches cachées
6. En opposition à un réseau de neurones récurrent qui réutilisent les résultats de son modèle pour s'améliorer.
7. Multi-Layer Perceptron
8. Density Reconstruction Neural Network

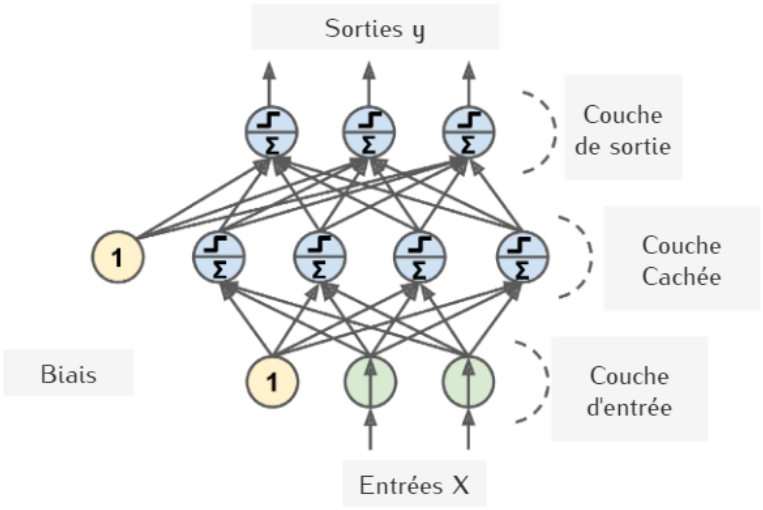


FIGURE 4.5 – Illustration d'un MLP avec une couche dense comme couche cachée. Le nombre de couches cachées peut être élevée ce qui conduit aux réseaux profonds. Le 1 représente le biais (**Reference8**).

implémentées sous la librairie de Machine Learning Keras (avec le « backend » TensorFlow). Les différentes couches présentées seront détaillés dans la suite. Nous indiquerons aussi en quoi elles sont importantes pour notre apprentissage.

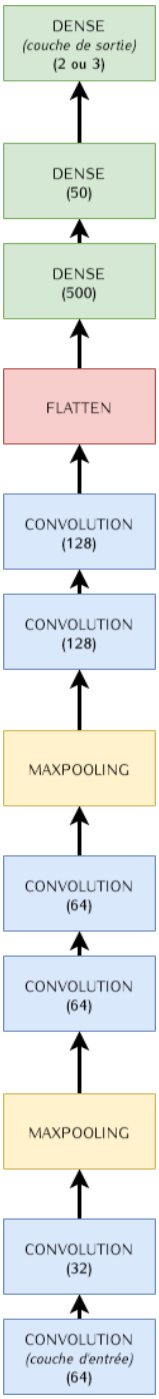


FIGURE 4.6 – Première architecture (nommée DRNN 1). Le nombre de neurones utilisés pour chaque couche est indiqué entre parenthèses. Le nombre de neurones de la couche de sortie dépend qu'on soit en 1D ou en 2D.

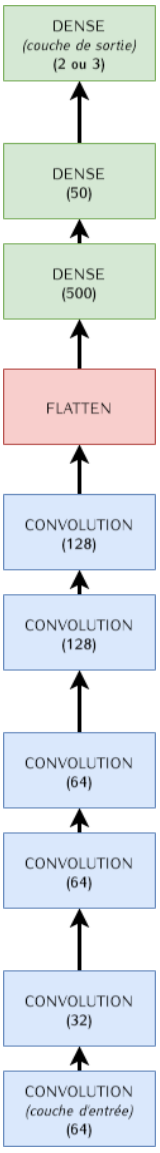


FIGURE 4.7 – Deuxième architecture utilisée (nommée DRNN 2). Ce modèle ne contient pas de couche de « pooling » [9]

6. Les détails concernant l'opération de « pooling » seront donnés à la section **??**

4.3 Les couches utilisées

4.3.1 Les couches de convolution

La convolution est l'opération fondamentale d'un CNN. Il s'agit d'une opération linéaire qui combine deux signaux pour en extraire un troisième. En général, une opération de convolution se définit par la formule suivante (i est le signal d'entrée et k est le noyau de la convolution) :

$$s(t) = (i * k)(t) = \int i(x)k(t-x)\,dx$$

En pratique, les signaux temporels ne sont pas continus, ils sont discrétisés par intervalles de temps Δt. Dans ce contexte, la convolution 1D se définit par la formule :

$$s(t) = \sum_{x=-\infty}^{\infty} i(x)k(t-x) \tag{4.1}$$

Cette formule doit être adaptée en 2D vu qu'on fera aussi un apprentissage 2D. La formule devient donc :

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{4.2}$$

L'opération de convolution est commutative grâce à l'inversion du noyaux relativement par rapport au signal d'entrée. Cette propriété, bien qu'importante d'un point de vue théorique, ne présente pas d'avantages du point de vue computationnel. C'est la raison pour laquelle on dispose de l'opération de cross-corrélation qui est une convolution sans inversion du noyau. En 2D elle se présente comme ceci :

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{4.3}$$

On remarque aussi que le parcours des indices se fait suivant l'input. Il se trouve que c'est plus direct et rapide ainsi, parce qu'il y a moins de variation dans la plage de valeurs valides pour n et m.

Plusieurs librairies de Machine Leanring implémentent la cross-corrélation mais l'appellent convolution. C'est le cas de Keras lorsqu'elle utilise le « backend » TensorFlow [10] (**Reference6**).

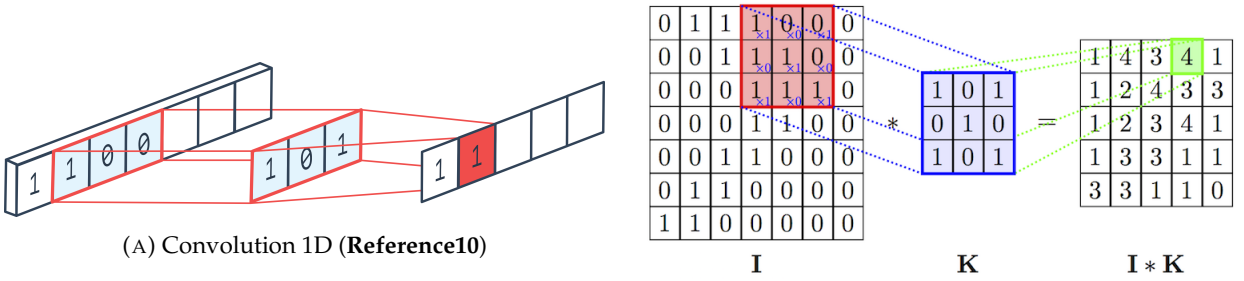
Dans les architecture de CNN typiques, la couche de convolution est généralement suivi d'une étape dite de détection. Dans cette étape, les résultats de la convolution (linéaires) sont passés à une fonction non linéaire au niveau d'une couche d'activation. Nous détaillerons la notion d'activation dans les sections suivantes. Après cette étape de détection, le Pooling est généralement appliqué pour modifier les résultats encore plus profondément.

4.3.2 Le max-pooling

L'opération de Pooling permet de réduite la taille des données (« downsampling »). Une fonction de Pooling transforme les entrées voisines par une fonction d'agrégation statistique. Plusieurs fonctions d'agrégations peuvent êtres utilisées. Par exemple, le Max-pooling renvoi le maximum parmi les entrées sur un domaine appelé « pool-size » (rectiligne en 1D et rectangulaire en 2D).

En général, l'opération de Pooling permet de rendre la représentation approximativement invariante aux petites variations dans l'input. Parlant de l'identification d'objets dans une image par

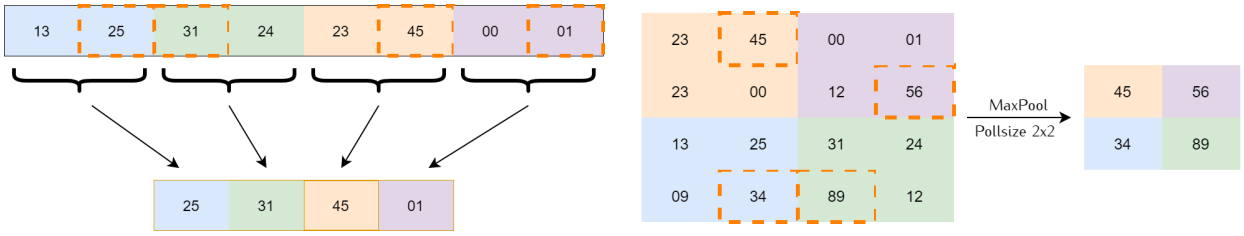
10. Lorsqu'on utilise Theano, les convolutions sont effectivement des convolution comme definie en **??** et **??**.



(A) Convolution 1D (**Reference10**)

(B) Convolution 2D (**Reference11**)

FIGURE 4.8 – Illustration d'une convolution (cros-corrélation) 1D/2D en mode "valid" (aucun « padding » de 0 ne sera ajouté et la sortie S aura une taille inferieure a l'entree I). La taille du noyaux que nous utiliserons sera de 3 (en 1D) et (6,2) (en 2D). Un autre paramètre important pour réduire la taille de la sortie est le "stride" ; il caractérise de l'écart entre deux applications consécutives du noyaux de convolution K. Nous le prenons égale à 1 (en 1D) et (1,1) (en 2D) de façon à couvrir tous les indices valides de l'entrée I.



(A) Max-pooling 1D

(B) Max-pooling 2D

FIGURE 4.9 – Opération de Max-pooling en 1D/2D avec un « pool size » de 2 (en 1D) et de 2×2 (en 2D).

exemple, "*L'invariance par translations locales (petites translations) peut être utile si on est plus intéressé par la présence de l'objet que par sa localisation exacte.*"(**Reference5**).

Dans le problème inverse que nous résolvons, on aimerait non seulement détecter la présence du saut de densité, mais aussi ses coordonnées exactes. Cela nous amène donc à considérer dans un premier temps une architecture avec Max-pooling (figure **??**), et dans un deuxième temps, sans Max-pooling (figure **??**).

4.3.3 Flatten

L'opération d'aplatissage (ou Flatten) permet de transformer les données en quittant de la forme tensorielle (2D avec plusieurs canaux) à une forme vectorielle. Il s'agit en réalité d'une étape de préparation pour les couches denses (complètement connectées).

4.3.4 Les couches denses

Dans ces couches, tous les neurones d'une couche sont connectés à tous les neurones de la couche précédente. Une couche dense (ou « fully connected ») prend les résultats d'une convolution/Pooling et en ressort des poids. Les couches de convolution ayant appris des aspects particuliers des données, la couche dense est un moyen facile d'apprendre des combinaisons non linéaires de ces dernières.

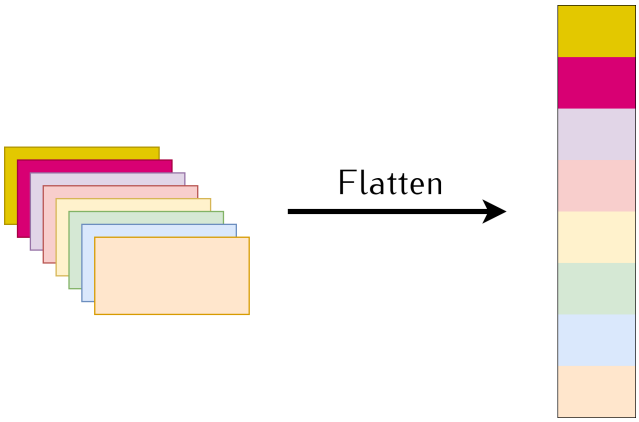


FIGURE 4.10 – Illustration d'une opération de Flatten). Qu'on soit en 1D ou en 2D, le Flatten assure que la sortie est un vecteur sur lequel une couche Dense peut opérer.

Si f_i désigne la fonction représentant une couche dense, l'opération effectuée est la suivante : $f_i(X) = \phi(XW + b)$, ou X représente les entrées de la couche, b le biais, W la matrice des poids (une ligne par neurone d'entrée et une colonne par neurone de cette couche, excepté ceux du biais), et ϕ désigne la fonction d'activation que nous détaillerons plus tard. (**Reference8**)

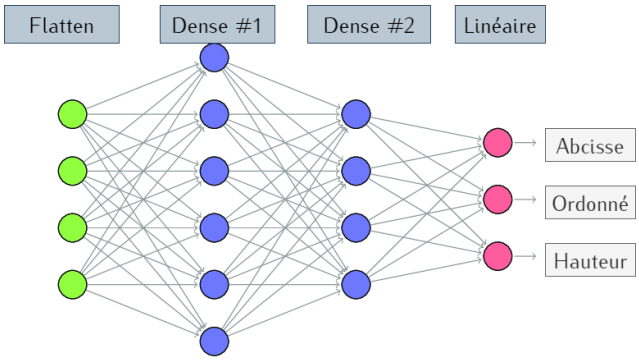


FIGURE 4.11 – Série de 3 couches denses « fully connected ». La couche en vert représente le résultat de l'opération de Flatten. L'utilisation des couches denses constitue la dernière étape des réseaux en figures **??** et **??**(Image adaptée de (**Reference9**))

4.4 Configurations de l'entrainement sous Keras

Keras propose une multitude d'options et d'hyper-paramètres pour tuner le modèle et son entrainement. Ceux qui ont été utilisés sont indiqués dans les sections suivantes.

4.4.1 Les hyper-paramètres

4.4.1.1 L'optimiseur

L'optimisation est une méthode d'accélération de l'entrainement. L'optimiseur Adam [11] combine les propriétés de deux autres algorithmes d'entrainement (AdaGrad et RMSProp) pour offrir une performance de taille.

11. Adaptative moment estimation

4.4.1.2 Activation RELU

La fonction d'activation introduit une non-linéaire entre les couche. L'avantage majeur de l'activation ReLU [12] par rapport aux autres fonctions d'activation c'est qu'elle n'active pas tous les neurones en même temps. D'un point de vue computationnel, elle est très efficace tout en produisant des résultats satisfaisants.

4.4.1.3 Le taux d'apprentissage

Il s'agit du paramètre le plus influant pour notre apprentissage. Il contrôle à quelle vitesse le modèle s'adapte au problème en déterminant de quelle quantité les poids des neurones seront mis à jour [13] après l'algorithme de « backpropagation ». S'il est très élevé, il peut rapidement conduire à solution non optimale ; s'il est très faible, le modèle peut reste figé (il faudra alors un nombre élevé d'époques pour potentiellement le débloquer).

Avec un taux d'apprentissage égale à `1e-4`, nous n'avons été capable que de détecter la hauteur du créneau en 1D ; et une réduction supplémentaire entraine une moins bonne convergence du modèle. En 2D, il a fallu descendre jusqu'à `1e-5` pour déterminer avec précision l'abscisse, l'ordonnée, et la hauteur du créneau.

4.4.1.4 Le « batch size »

Il s'agit de la taille de chaque paquet de données [14] passés au modèle durant une époque. Un « batch size » faible apporte du bruit au modèle vu qu'une portion aléatoire des données est utilisée pour mettre ajour les poids des neurones. Ceci assure une meilleure généralisation du modèle tout en limitant la quantité de données chargées dans la RAM à chaque époque.

4.4.1.5 Early-stopping

La technique d' Early-stopping sera notre moyen primaire de lutte contre le sur-apprentissage. Pour l'implémenter de façon efficace sous Keras, il nous faut une variable appelée `patience`. Il s'agit du nombre d'époques à attendre avant d'arrêter l'apprentissage de façon précoce. Nous arrêterons nos entrainements dès que le score R^2 (voir paragraphe **??**) sur le jeu de validation n'aura pas augmenté pendant 10 époques.

4.4.2 Les métriques

4.4.2.1 Loss MSE

Pendant la génération des données, on a pris soin de ne pas introduire de données aberrantes. La MSE(Mean Squared Error) qui est plus élevée sur les valeurs aberrantes que la MAE (Mean Absolute Error) est dont plus adaptée ici. Si les $\hat{y}_i$ désignent les prédictions et y_i les véritables cibles (valeurs observées), la MSE se définit par :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.4}$$

12. Rectified Linear Unit
13. Les poids des neurones ayant été initialisés de façon aléatoire
14. Nombre d'instances d'entrainement sélectionnés aléatoirement

4.4.2.2 Coefficient de détermination R^2

Le coefficient de détermination R^2 est très important en statistique. On peut l'obtenir par la formule :

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{4.5}$$

Avec

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad \text{et} \quad SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

Ou $\bar{y} = \sum_{i=1}^{n} y_i$ représente la moyenne des valeurs observées.

On peut remarquer que :
— Si le modèle prédit les valeurs attendues (observées), le score R^2 vaut 1.
— SI le modèle prédit toujours la valeur moyenne $\bar{y}$, le score R^2 vaut 0.
— Si les prédictions sont pires que la moyenne, le score R^2 est négatif.

En général, on voit que si les prédiction et les valeurs observées sont très corrélées (sans être égales), on aura un score R^2 ce qui n'est pas caractéristique des résultats. En effet, pour des taches de régression il se définit comme étant le carré du coefficient de corrélation entre les valeurs prédites et les valeurs observées.

Dans la suite de ce rapport, le score R2 sera présenté sous forme de pourcentage.

4.4.2.3 Un score personnalisé

On définit donc un nouveau score particulièrement adapté a nos données. On déclare qu'une prédiction est correcte si elle est suffisamment proche du label :
— au **dixième** près pour la position (suivant x ou y) car le domaine d'étude est $[0, 1]$ en 1D et $[0, 1] \times [0, 1]$ en 2D
— à l'**unité** près pour la hauteur car les hauteurs observées sont comprises entre 1 et 10 (en 1D) et 0.1 et 10 (en 2D)

Le score personnalisé est un score sévère (pourcentage des prédictions correctes) qui récompense les prédictions qui sont à la fois précisent en hauteurs et en position. La prédiction de la position est un problème majeur auquel nous avons fait face, elle cause généralement des valeurs faibles pour le score personnalisé.

4.5 Résultats

Nous résumons la section précédente en spécifiant les paramètres (et leurs définitions) utilises pour entrainer le modèle sous Keras.

Nous avons entrainé les architectures en 1D en ajustant les dimensions des couches de neurones convenablement. Observons à présent les résultats obtenus sur le jeu de données test.

4.5.1 Régression

4.5.1.1 En 1D

On obtient de très bonnes prédictions sur la hauteur de l'obstacle car celui-ci affecte directement l'amplitude des signaux sur le bord droit du domaine. Cela se traduit par un score R^2 très

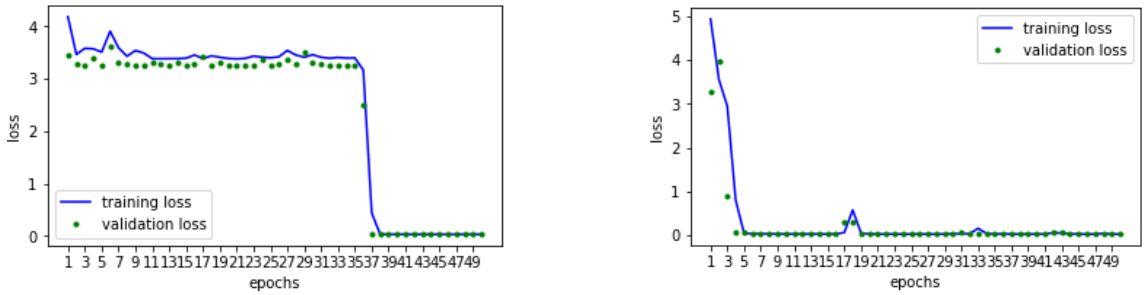
TABLE 4.1 – Liste des paramètres majeurs utilises pour l'entrainement. L'activation "relu" est utilisée sur les couches cachées et "linear" sur la couche de sortie

Paramètre	**Définition**	**Valeur 1D / 2D**
`learning_rate`	taux d'apprentissage	1e-4 / 1e-5
`batch_size`	taille d'un batch a chaque époque	32
`optimizer`	algorithme d'optimisation	Adam
`activation`	type de fonction d'activation	"relu" ou "linear"
`patience`	patience pour l'`early_stopping`	10
`epochs`	nombre d'époques	100
`kernel_size`	taille du noyau de convolution	3 / (6,2)

élevé. Ce score n'est pas assez indicatif vu que les prédictions sur la position du créneaux ne sont pas bonnes [15]. Le score personnalisé permet de capturer ce défaut et donne environ 25 %.

TABLE 4.2 – Résultats obtenus sur le jeu test en 1D. Le modèle avec Max-poling correspond à la figure **??**, et celui sans Max-pooling à la figure **??**, tous adaptés à la 1D. Ces valeurs représentent une moyenne obtenue sur plusieurs cycles d'entrainement/prédiction

Score	**Avec Max-pooling**	**Sans Max-pooling**
R^2	99.49 %	99.50 %
Personnalisé	26.50 %	28.21 %



(A) 1D svec Max-pooling

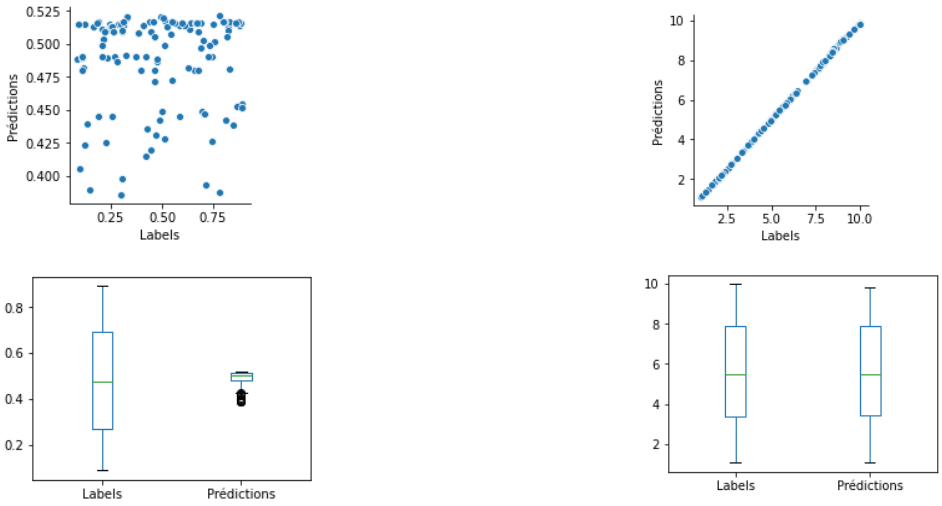
(B) 1D sans Max-pooling

FIGURE 4.12 – Comparaison de la vitesse de décroissance de la loss en 1D

Même si aucun n'est capable de correctement prédire la position du créneau, le modèle sans Max-pooling semble mieux se comporter sur ce jeu de données. Pour les illustrations qui vont suivre, nous utiliserons donc ce dernier modèle (sans Max-pooling). Nous commençons par une illustration de la corrélation entre les labels (valeurs observées) et les prédictions (figure **??**).

Pour observer les meilleures et les pires prédictions du modèle, il nous faut une mesure de la distance entre les prédiction et les labels. On définit donc la norme ci-bas (en prenant soins de

15. Le score R^2 élevé pourrait s'expliquer par le fait que le poids de la hauteur dans la corrélation est plus fort que celui de la position



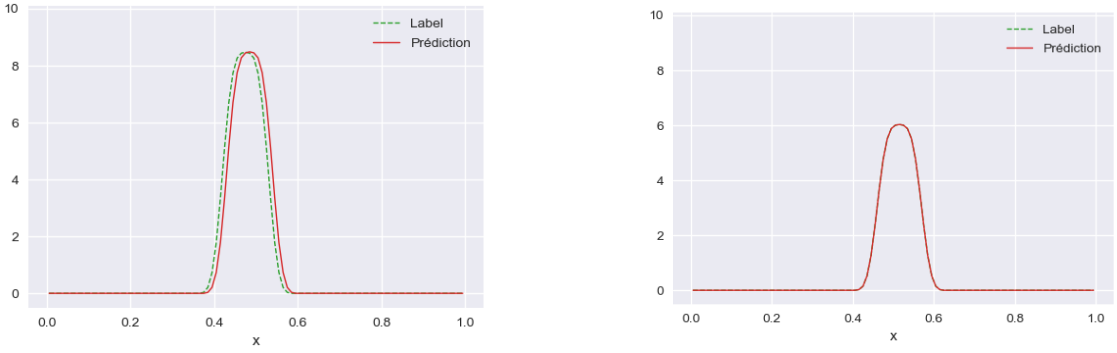
(A) Position (B) Hauteur

FIGURE 4.13 – Illustration de la corrélation entre les labels et les prédictions obtenues par le modèle sans Max-pooling en 1D. On peut observer l'exactitude des prédictions pour la hauteur mais un échec sur la position. En effet, les prédictions de la position du créneau sont concentrée autour de la moyenne 0.5.

normaliser la hauteur).

$$\text{Norme} = \sqrt{\text{Position}^2 + \left(\frac{\text{Hauteur}}{10}\right)^2}$$

Observons donc les meilleures prédictions du modèle (sans Max-pooling) (figure **??**).

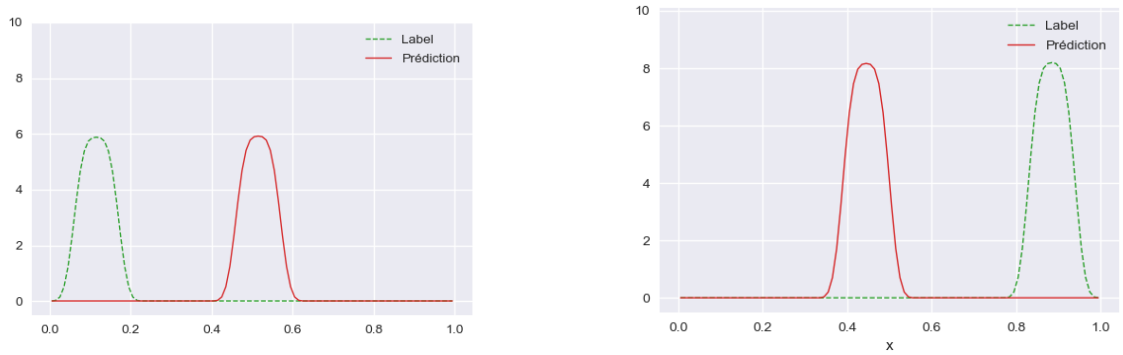


(A) Label = [0.488 8.474]
Prédiction = [0.49 8.486]

(B) Label = [0.522 6.032]
Prédiction = [0.527 6.033]

FIGURE 4.14 – Les meilleures prédictions 1D. Il s'agit ici d'une reconstruction manuelle de la densité à partir des vecteurs en titre de figure en (A) et (B). La première coordonnée indique la position du saut de densité, et la deuxième sa hauteur. On confirme que les bonnes prédictions des positions sont proches du milieu du domaine.

Les pires prédictions du modèle permettent de mieux illustrer les problèmes rencontrés avec l'apprentissage 1D (**??**).



(A) Label = [0.125 5.886]
Prédiction = [0.524 5.925]

(B) Label = [0.892 8.195]
Prédiction = [0.454 8.167]

FIGURE 4.15 – Les pires prédictions du modèle (sans Max-pooling) 1D. La différence se joue au niveau de la position du créneau comme indiquée à la figure **??**. Les labels les plus éloignés du milieu conduisent aux prédictions les plus mauvaises.

Pour améliorer ce score personnalisé, il faut passer en 2D. En effet, le problème inverse n'est pas forcément bien posé dans le sens ou plusieurs entrées peuvent donner la même sortie. En 1D, on ne peut mesurer la sortie que sur un seul bord du domaine, ce qui limite beaucoup notre apprentissage.

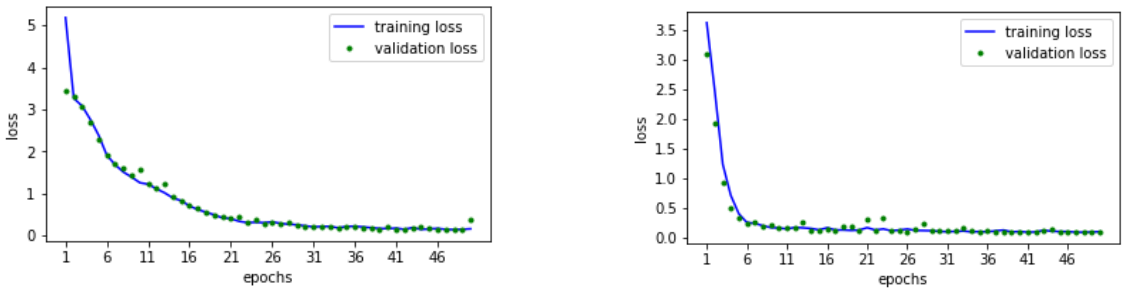
4.5.1.2 En 2D

Comme attendu, le réseau est capable de détecter non seulement la hauteur de l'obstacle, mais aussi son abscisse et son ordonnée. Notre score personnalisé nous permet de confirmer cela dans le tableau **??**.

TABLE 4.3 – Résultats obtenus sur le jeu test en 2D

Score	Avec Max-pooling	Sans Max-pooling
R^2	94.80 %	98.81 %
Personnalisé	55.75 %	93.50 %

On constate que le modèles sans l'opération de Max-pooling est globalement meilleure que son homologue avec Max-pooling du à l'application de l'Early-stopping. La figure **??** permet d'observer cela à travers la vitesse de convergence du modèle.

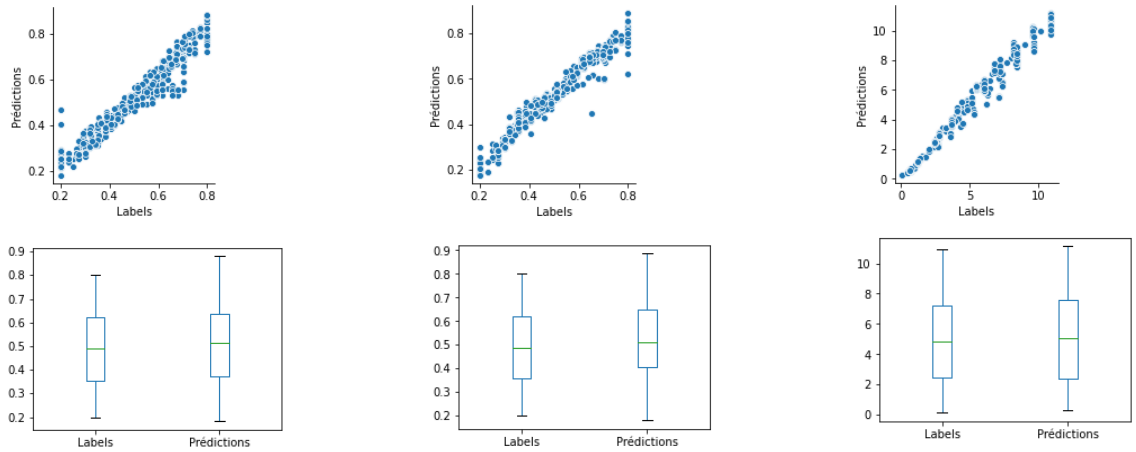


(A) 2D avec Max-pooling

(B) 2D sans Max-pooling

FIGURE 4.16 – Comparaison de la vitesse de décroissance de la loss en 2D

Comme nous l'avons fait en 1D, Le modèle sans Max-pooling sera utilisé par la suite pour illustrer la corrélation entre les prédictions (de l'abscisse x, de l'ordonne y, et de la hauteur) et leurs labels respectifs (figure **??**).

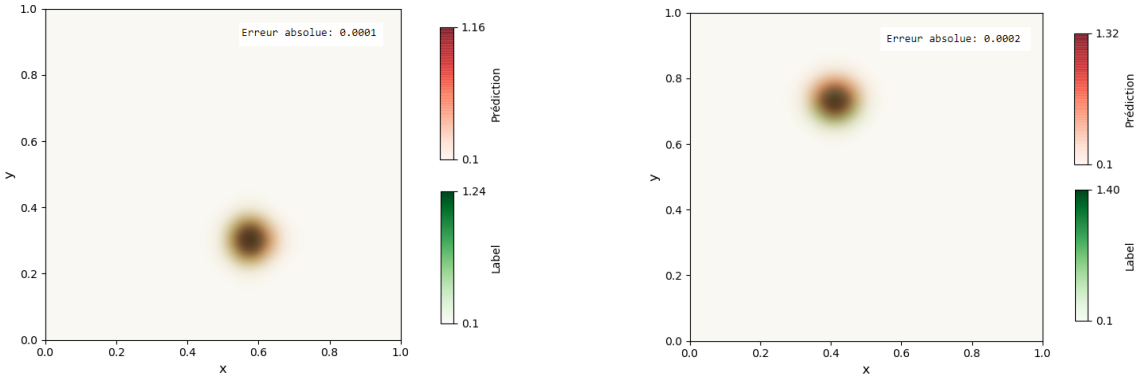


(A) Abscisse (B) Ordonnée (C) Hauteur

FIGURE 4.17 – Illustration de la corrélation entre les labels et les prédictions obtenues par le modèle sans Max-pooling en 2D. On peut observer que toutes les trois informations sont relativement bien corrélées, d'où le score R2 élevé (tableau **??**).

Pour observer les meilleures et les pires prédictions du modèle, on définit donc la norme ci-bas.

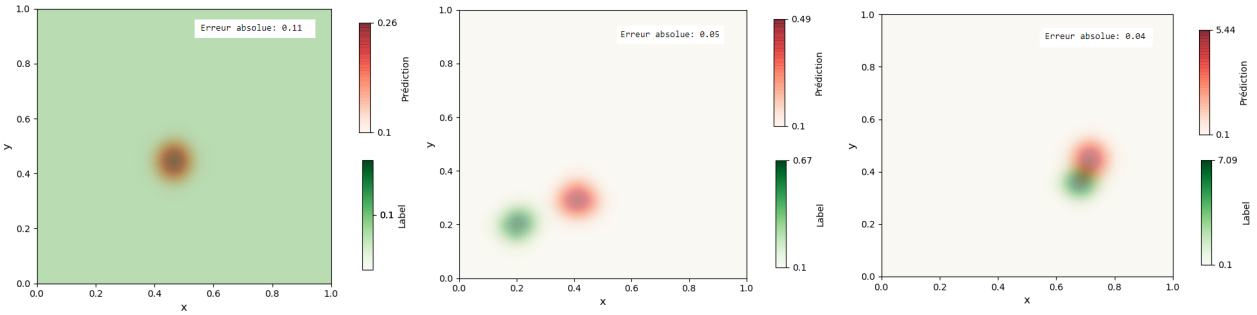
$$\text{Norme} = \sqrt{\text{Abcisse}^2 + \text{Ordonee}^2 + \left(\frac{\text{Hauteur}}{10}\right)^2}$$



(A) Label = [0.574 0.303 1.24]
Prédiction = [0.58 0.302 1.157]

(B) Label = [0.405 0.724 1.392]
Prédiction = [0.409 0.736 1.318]

FIGURE 4.18 – Les meilleures prédictions 2D. Ces images ont été reconstruites à partir des vecteurs "Labels" et "Prédiction" issus de l'apprentissage. La première coordonnée indique l'abscisse x du saut de densité, la deuxième son ordonnée y, et la troisième sa hauteur. L'interpolation bicubique a été utilisée pour obtenir des images plus nettes.



(A) Label = [0.2 0.65 0.1]
Prédiction = [0.466 0.448 0.267]

(B) Label = [0.2 0.2 0.736]
Prédiction = [0.407 0.298 0.494]

(C) Label = [0.674 0.358 7.096]
Prédication = [0.712 0.449 5.462]

FIGURE 4.19 – Les pires prédictions du modèle 2D. Sans surprise la plus mauvaise des prédictions s'obtient lorsque le créneau est absent (A). On remarque en général que les pires prédictions sont faites lorsque le créneau se situe très proche de L'extrémité (B).

En ce qui concerne la généralisation du modèle à d'autres formes d'obstacles, je n'ai pas eu l'occasion de comparer les modèles avec et sans Max-pooling. Le modèle sans Max-pooling risque alors d'être moins performant conformément à la théorie (voir paragraphe **??**). Sous Keras, le modèle à prouver être capable d'apprendre en continu, du moment que les entrées sont toutes normalisées et ont la même forme.

4.5.2 Classification

Durant le stage, il a fallu effectuer une classification mutilabel sur les données en 2D. Elle permet de placer l'obstacle dans une catégorie définie à partir de la source. La classification permet de détecter juste l'ordonnée de l'obstacle. La structure des entrées est quasiment la même que pour la régression en 2D, sauf qu'il manque les signaux sur la gauche (ou se trouve la source). En ce qui concerne les sorties, l'image ci-dessous décrit mieux leur structure :

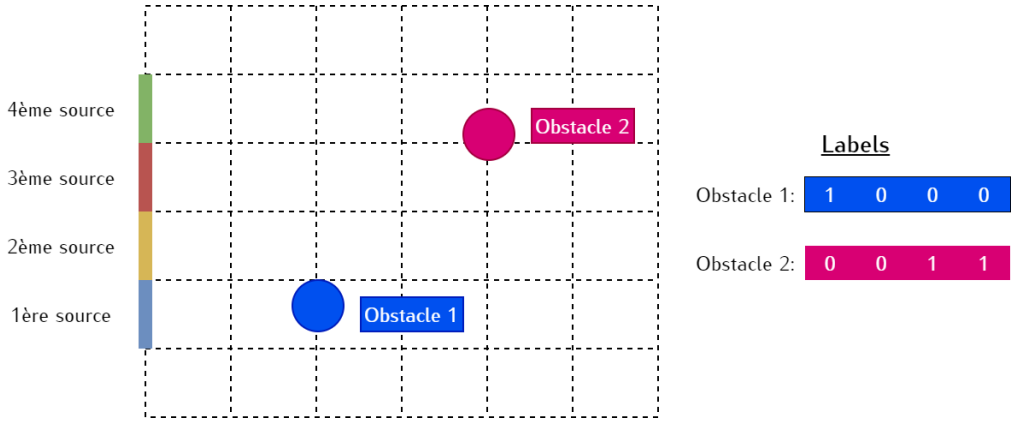


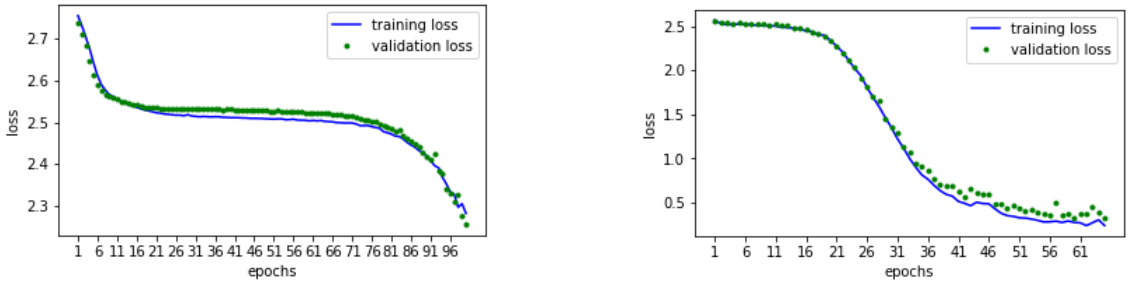
FIGURE 4.20 – Description des labels pour la classification multilabel en 2D. Un label est marque 1 si l'obstacle se trouve dans le champ de la source correspondante

Les données utilisées pour la classification ont une taille différente des autre. On a moins d'itérations en temps (40 au lieu de 168) mais un maillage beaucoup plus fin (90×90 au lieu de 28×28). Lu modèle de CNN utilisé ressemble à celui pour la régression. Une importante différence est qu'on utilise une activation "sigmoid" à la place de l'activation "linear" au niveau de la couche de sortie. On obtient donc en sortie des probabilités qu'il faut classer par catégories. Le modèle est

entrainé avec des hyper-paramètres identiques à ceux utilisés durant la régression. Les résultats sont présentés dans la table **??**

TABLE 4.4 – Résultats obtenus pour la classification en 2D. Le score sévère favorise les prédictions qui sont exactes au véritable label sur toutes les 4 colonnes. Le seuillage permet d'augmenter la précision des résultats en se fixant un nouveau seuil à partir duquel classer les sorties du réseau de neurones. Le score de « Binary accuracy » procuré par Keras est calculé avant le seuillage.

Score	**Avec Max-pooling**	**Sans Max-pooling**
Binary accuracy	75.00 %	98.86 %
Score sévère après seuillage	27.27 %	95.45 %



(A) 2D avec Max-pooling (B) 2D sans Max-pooling

FIGURE 4.21 – Comparaison de la vitesse de décroissance de la loss (Multilabel Cros-sentropy) pour la classification en 2D

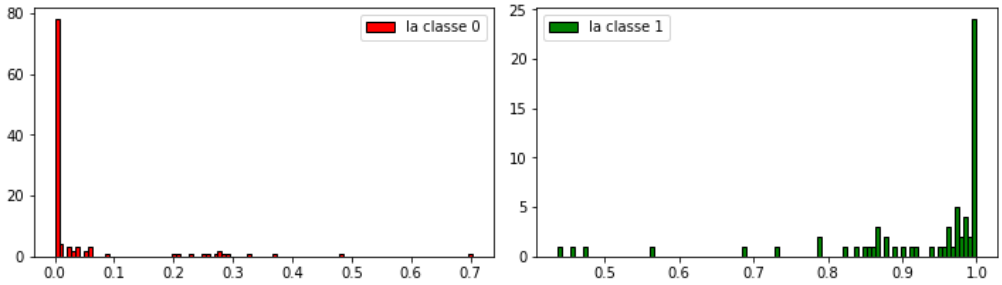


FIGURE 4.22 – Confiance du modèle sans Max-pooling en ses prédication. En rouge la classe 0 qui indique l'absence du créneau devant la source ; et en vert la classe 1. Cette figure indique que le modèle se trompe très rarement, ce qui confirme le résultat obtenu à la table **??**.

Dans ce cas aussi, le modèle sans Max-pooling semble meilleure (figures **??** et **??**). Cependant nous ne disposons pas d'assez de données pour vérifier celui qui se généralise mieux. Ceci étant une classification, les meilleures prédictions du modèle sont exactement les labels attendus et les pires prédictions ne diffèrent que de peu (de 1) de leurs cibles. Il est intéressant de constater qu'il n'y a aucune prédiction aberante, par exemple : un obstacle se trouve en face des sources 1 et 3 sans recontrer la source 2.

Chapitre 5

Bilan du stage

5.1 Ressources utilisées

Les ressources utilisées durant le stage varient en nature et en fonction.

5.1.1 Écriture de code

— **Visual Studio Code** : Pour l'édition du code en langages C++ et LaTeX grâce à ses nombreuses fonctionnalités. Les extensions CMake, LateX Workshop et LTeX ont considerableme facilite les travaux.
— **Google Colab** : Pour facilitation de l'apprentissage grâce à ses GPU. Les librairies Python les plus utilisées ici sont Numpy, Pandans, et Keras.
— **Jupyter** : Pour les tâches ne nécessitant pas trop de ressource (visualisation, sauvegarde en format PARQUET). Les librairies Python majeures utilisées ici sont Numpy, Pandas et Matplotlib.
— **Kile** : Pour l'écriture du premier brouillon du rapport en Latex.
— **Draw.io** : Pour les illustrations.

5.1.2 Communication

Les communications se sont effectuées principalement par messagerie électronique. J'ai aussi eu l'occasion de communiquer avec les professeurs en présentiel à trois reprises.

5.2 Journal de bord

5.2.1 Semaines 1 et 2

— 15 juin : Réunion de début de Stage par Google Meet.
— 16 juin : Demande aux professeurs de vérifier un exemple de simulation 1D, avant de me lancer la génération des données.
— 17 juin : Remarque du problème de formation et de propagation d'un créneau indésirable sur l'énergie, le flux et la température.
— 18 juin : Rédaction d'un nouveau schéma par M. Franck (pour l'étape 1) qui devrait conserver l'équilibre.
— 22 juin : Détection de la source du problème d'apparition du créneau indésirable, et définition d'un nouveau terme S'.
— 23 juin : Confirmation de l'exactitude des simulations 1D et début de la génération des données avec 500 mailles.
— 25 juin : Demande d'aide à M. Vigon pour la configuration de la fonction d'activation de la couche de sortie sous Keras.

5.2.2 Semaines 3 et 4

— 3 juillet : Rencontre avec M. Navoret pour discuter des avancements. Prise de connaissance d'une des raisons potentielles du problème de mauvaise prédiction de la position du créneau sur la densité en 1D. Proposition de plusieurs solutions par M. Navoret, entre autre de partir d'un signal stationnaire sinusoïdal et d'introduire l'onde à un temps t*>0.
— 6 juillet : Nouvelles simulations effectuées en vue d'observer la différence entre les effets de deux densités différentes. Continuation vers des nouvelles simulations avec 300 mailles.
— 8 juillet : Décroissance du taux d'apprentissage à la suggestion de M. Franck mais non-amélioration des résultats d'apprentissage.
— 9 juillet : Passage aux réseaux convolutif grâce à M. Vigon.
— 11 juillet : Plot du début des oscillations, des maximum, des minimum à la demande de M. Vigon, afin de mieux observer les effets de deux créneaux de densités différentes.

5.2.3 Semaines 5 et 6

— 13 juillet : Rencontre avec M. Navoret et M. Franck à la fac. Devant la persistance du problème de non-détection de la position du créneau, l'implémentation du problème en 2D semble être la solution appropriée.
— 14 juillet Reformulation 2D du schéma de « splitting » et adaptation du code 1D en 2D.
— 19 juillet : Fin du codage 2D et présentation des résultats.
— 25 juillet : Ajustement de la gamme de couleurs pour les visualisations et passage à la génération des données sur 90x90 mailles.

5.2.4 Semaines 7, 8, et 9

— 5 août : Rencontre avec M. Franck à la fac. Proposition de solutions pour la non-détection de la position du créneau en 2D par résolution d'un système proche de l'équation de la chaleur, après affichage par lignes de niveaux. La possibilité d'adopter un obstacle s'étendant sur toute la verticale est envisagée. Prise de connaissance des délais pour la rédaction du rapport.
— 6 août : Rédaction et envoi du plan du rapport de stage.
— 7 août : Proposition de réduction drastique de la résolution spatiale, et proposition de nouvelles idées par M. Vigon, entre autre la considération d'un obstacle considérable plus opaque.
— 8 août : Nouvel apprentissage avec des simplifications majeures qui fonctionne. Amélioration des résultats et continuation du rapport.
— 15 août : Soumission du premier brouillon du rapport.

5.3 Difficultés rencontrées et solutions apportées

Un résume du déroulement du stage est présente à la figure **??**. Les points les plus marquants du stage y sont représentés. On peut aussi y voire les difficultés majeures auxquelles j'ai été confronte.

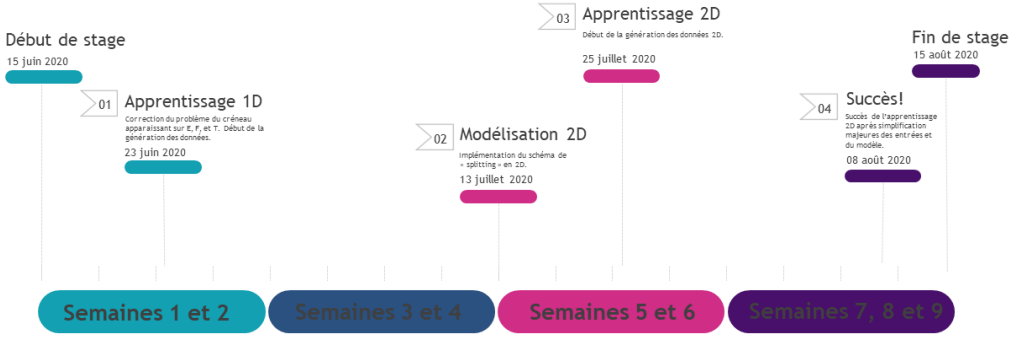


FIGURE 5.1 – Résumés des tournants du stage. Les détails du déroulement peuvent être obtenues dans la section **??**.

5.3.1 Apparition d'un créneau indésirable

Au commencement du stage, le code de calcul 1D n'était pas au point. En effet, dès qu'on plaçait un créneau sur la densité, un créneau correspondant se formait puis se propageait sur les signaux E, F, et T. Le problème a été résolu par rajout d'un terme S' au niveau de la deuxième équation du schéma de « splitting ».

5.3.2 Detection de la position du crenau

La détection de la position du saut de densité a été un problème récurant durant le stage. À la fin stage, aucune solution (si elle existe) n'a été trouvée pour le problème inverse en 1D. Cependant en 2D le problème a été résolu par diminution du taux d'apprentissage à $1e-5$ et augmentation du nombre d'époques. Il est bien connu que l'entrainement des réseaux de neurones peu diverger si le taux d'apprentissage est trop élevé. Quant au nombre d'époques, je n'en faisais pas suffisamment pour voir le modèle converger. Une solution bien plus rapide aurait été d'automatiser la recherche des hyper-paramètres, chose que je n'ai apprise qu'à la fin du stage.

Pour revenir à l'apprentissage en 1D, il est important de mentionner de façon spéculative quelques pistes d'étude qui, bien combinée, peuvent améliorer les résultats 1D :

— **deux apprentissages différents** : en créant un modèle sépare pour l'apprentissage de la position et un autre pour la hauteur
— **élimination des mauvais labels** : on pourrait supprimer de l'apprentissage tous les exemples qui ont un label trop proche du bord du domaine (par exemple ne garder que ceux compris entre 0.2 et 0.8 comme en 2D).

Cela dit, la solution trouvée en 2D est loin d'être optimale ; elle souffre de deux problèmes majeurs :

— **pas de generalisation** : le modèle finalement retenu ne comporte pas de couche de max-pooling qui aurait probablement aidé a la généralisation. Aucun test n'a été fait pour vérifier le niveau de généralisation de ce modèle.
— **model trop lourd** : le nombre de paramètres est très élevé dû à la taille des entrées `(168, 28, 48)`, ce qui necessite un espace memoire tres important au moment de la sauvegarde des poids (2Go).

Toutes ces quatre pistes d'amélioration n'ont pas été implémentées pour cause de temps. En effet, la gestion de temps a été un vrai problème tout au long du stage.

5.3.3 Gestion du temps

La gestion du temps durant le stage n'a pas été facile. Au moment d'implémenter le schéma en 2D (ce qui n'était pas initialement prévu), j'ai longuement hésiter sur l'option la plus rapide. J'ai pu compter sur les conseils de M. Navoret pour surmonter cet obstacle.

Aussi, je me suis rendu compte des délais bien en retard, j'ai du me débrouiller pour terminer l'apprentissage et améliorer les résultats tout en rédigeant le rapport. Cela dit, je n'ai pas réussi à faire une partie essentielle qui consiste à étudier l'apport du max-pooling dans la généralisation du modèle de CNN.

5.4 Les apports du stage

Ce stage a été enrichissant pour moi sur plusieurs fronts :

5.4.1 Expérience en développement

J'ai gagné de l'expérience en développement C++ et Python, tout en me développant un portfolio. J'ai beaucoup appris sur l'API de Pandas, Matplotlib, et de Keras. J'ai à présent une large base de données de code réutilisable pour d'autres taches.

5.4.2 Équations aux dérivées partielles

J'ai pu observer directement quelques astuces utilisées par MM. Navoret et Franck pour vérifier la validité d'une simulation. Pour l'équation du transfert radiatif, j'ai compris la nécessité de partir d'un état d'équilibre radiatif.

5.4.3 Réseaux de neurones

Ce stage m'a permis de percevoir la puissance des réseaux de neurones. J'ai appris à quel point le taux d'apprentissage est important, comme le rappelle cette citation du livre de référence Deep Learning : *"The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate."* (**Reference5**).

Tous ces enseignements me poussent à me poser quelques questions concernant le « batch size ». Lors de l'apprentissage, il a fallu entrainer le modèle en utilisant la méthode d' **augmentation du « batch size »** pour obtenir les premiers "bons" résultats. Cette méthode référencée ici montre que beaucoup de questions restent à résoudre dans le domaine du Deep Learning.

5.4.4 Expérience de recherche

En tant que première expérience dans un environnement de recherche tel que l'UFR, j'ai pu me familiariser avec le milieu. J'ai notamment appris que les résultats ne doivent pas toujours être ceux auxquels on s'attend, du moment que l'on a une explication pour l'échec.

Chapitre 6

Conclusion

Pour conclure, j'ai effectué mon stage de master 1 en tant que stagiaire en calcul scientifique/data scientist à l'UFR de mathématiques et d'informatique de l'Université de Strasbourg. Lors de ce stage de deux mois, j'ai pu mettre en pratique mes connaissances (en calcul scientifique, en méthodes numériques, et en développement C++ et Python) acquises durant ma formation de CSMI. Je me suis confronté au problème inverse de reconstruction de la densité d'un domaine par un CNN après avoir simulé la propagation du signal dans ce dernier en 2D.

Ce stage fut très enrichissant pour moi car il m'a permis d'approfondir mon savoir sur les notions de backpropagation et de descente de gradient utilises dans les réseaux de neurones. J'ai aussi gagné beaucoup d'expérience de développement logiciel et je me suis familiarisé avec l'environnement de Keras. Ce stage m'a aussi permis de comprendre le déroulement d'une activité de recherche, et à quel point une bonne organisation et un certain degré d'autonomie sont importants. Un point sur lequel j'aurais voulu mieux m'améliorer est la théorie des EDP, en particulier l'intuition derrière la définition des flux numériques.

Cette expérience de stage fut centrée autour de la problématique de l'apport des réseaux de neurones dans la résolution des problèmes inverse, spécialement dans la détection des tumeurs [1]. Le modèle de CNN construit est la preuve qu'un réseau de neurones est capables de reconstruire des sauts de densité de nature cylindrique. L'utilisation du max-pooling est sans doute un atout important pour la généralisation du modèle (détection de formes d'obstacles diverses, différentes expressions des opacités, etc.).

Fort de cette expérience et de ses nombreux enjeux, j'aimerais beaucoup par la suite, via un prochain stage, affiner les résultats à l'aide d'un apprentissage en continu en passant à la détection de plusieurs sauts de densités par exemple. On pourrait aussi envisager la reconstruction des opacités d'absorption et de dispersion. Tout ceci pourrais conduite ultimement à la création d'un tomographe et un déploiement en milieu médical.

1. Les tumeurs sont assimilables à des créneaux, des sauts de densite, ou obstacles

Annexe A

Comment reproduire les resultats ?

A.1 Exécution du code 1D/2D

Les codes de calculs 1D (développé durant le projet et utilise pour l'apprentissage durant ce stage) et 2D se trouvent dans deux dépôts Github différents :

— 1D : `https://github.com/desmond-rn/projet-inverse`
— 2D : `https://github.com/desmond-rn/projet-inverse-2d`

Pour compiler le code dans les deux cas, il faut idéalement passer par un conteneur Docker. Cependant la compilation peut aussi marcher si on passe par CMake directement. Ensuite il faut lancer l'exécutable `transfer` avec un fichier de configuration. Les détails ainsi que la définition des paramètres 1D/2D du fichier de configuration sont définis dans les fichiers `README.md` des dépôts 1D et 2D respectif.

A.2 Lecture du format binaire

Pour passer d'une simulation (C++) à sa visualisation ou à l'apprentissage (Python), les données peuvent être sauvegardées au format binaire à l'aide du paramètre `export_type binary`. Lorsque c'est le cas, il faut se servir de la fonction `read_sds_version01` (disponible dans le notebook principal de tout l'apprentissage Regression3.ipynb) pour les lire.

Les données **train**, **val** et **test** que nous avons utilisé ont été sauvegardées sous le format binaire de Numpy. Ceci permet de faciliter la reproduction de l'apprentissage.

A.3 Exécution des notebooks

Deux catégories de notebooks ont été créés dans les dépôts github. Les deux premiers sont exécutables directement après clonage du dépôts. Il s'agit de :

— Visualisation : Il permet de visualiser les résultats d'une simulation exportée en CSV dans le fichier `data/df_simu.csv`
— Sauvegarde : Il permet de transformer des données du format CSV au format binaire PARQUET rapidement lisible par Pandas.

Les autres notebooks sont exécutables sur Google Colab. Ils ne sont exécutables que si on dispose des données train, val et test. Il faut alternativement télécharger le modèle déjà entrainé. Dû à leur taille considérable, ces données (ainsi que les notebooks pour l'apprentissage) ne sont pas disponible sur les dépôts et doivent être téléchargés séparément sur Google Drive,

Annexe B

Comment faire des prédictions avec ce modèle ?

Pour faire des prédications, il suffit de disposer d'un jeu de données ayant la forme bien particulière décrite à l'une des figures **??** ou **??**. Il faut ensuite charger le modèle à l'aide de Keras et ensuite compiler.

B.1 Normalisation des données

En plus d'avoir la forme appropriée, les données doivent être normalisées i.e toute les énergies doivent êtres divisées par leur maximum (en valeur absolue). Il en est de même pour le flux et la température.

B.2 Chargement du modèle

Le modèle a été sauvegardé sous la convention HDF5 de Keras. Après l'avoir chargé, il faut le compiler avec l'optimiseur Adam (et un taux d'apprentissage de `1e-4` ou `1e-5` suivant qu'on soit en 1D ou 2D). Pour que la compilation fonctionne, il faut impérativement inclure la fonction de calcul du sore R^2 indiquée ci-dessous.

```
from keras import backend as K

 def r2_score(y_true, y_pred):
    SS_res =  K.sum(K.square(y_true - y_pred), axis=-1)
    SS_tot = K.sum(K.square(y_true - K.mean(y_true)), axis=-1)
    return 1.0 - SS_res/(SS_tot + K.epsilon())
```

B.3 Entrainer le modele en continu

Le modèle peut être entrainé en continu. Après l'avoir chargé, on peut l'apprendre à détecter d'autres formes d'obstacles sous différentes conditions. Il suffit déjà de disposer de telles données. Pour l'apprentissage en continu, on pourra utiliser la fonction `train_on_batch` de Keras.