

Ecole Normale Supérieure Paris-Saclay  
Département Génie Mécanique

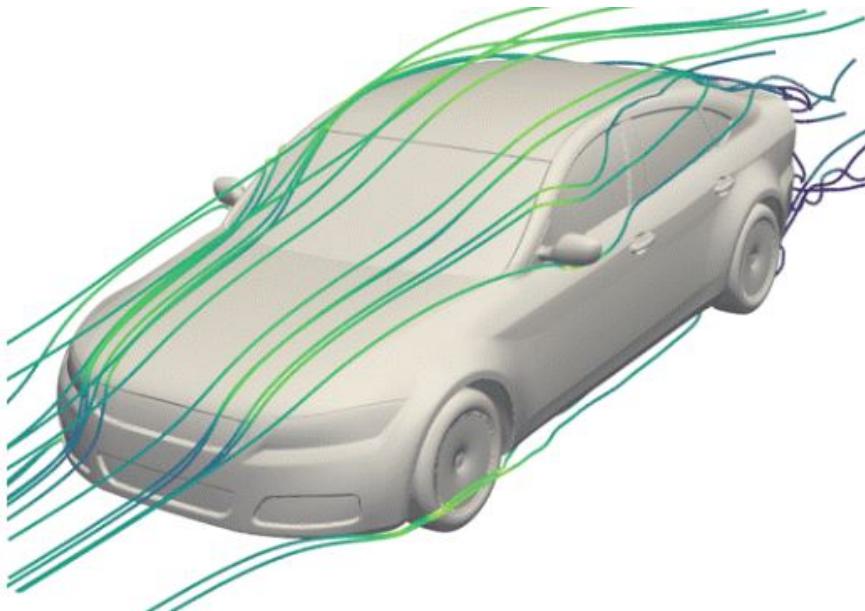
---

**MEMOIRE DE STAGE DE MASTER 1<sup>e</sup> ANNEE**

Effectué à l'Institut de Recherche en Ingénierie de l'Aragon (I3A)  
Sous la direction de Elias Cueto

**Génération de topologies de voitures optimisées selon des critères d'aérodynamisme par apprentissage profond**

**Bastien JACQUES**



Je remercie M. Cueto de m'avoir accueilli dans son équipe. Merci à lui et à toute l'équipe de l'Institut de Recherche en Ingénierie de l'Aragon de m'avoir mis dans les meilleures dispositions de travail.

Je remercie Antoine Petit pour le soutien qu'il a pu m'apporter durant ces 12 semaines de stage. Je le remercie également de m'avoir mis en contact avec M. Cueto et de m'avoir aidé dans ma recherche de stage.

Je tiens également à remercier particulièrement Carlos Bermejo Barbanoj, Mikel Martinez ainsi que Lucas Tesan pour l'aide qu'ils m'ont apporté durant ces 3 mois de stage.

Merci enfin à M. Guilhem pour sa disponibilité et ses conseils tout au long de ce stage.

Agradezco al Sr. Cueto por haberme acogido en su equipo. También doy las gracias a todo el equipo del Instituto Universitario de Investigación en Ingeniería de Aragón (I3A) por haberme brindado las mejores condiciones para trabajar.

Quiero agradecer especialmente a Carlos Bermejo Barbanoj, Mikel Martínez y Lucas Tesan por la ayuda que me han brindado durante estos tres meses de prácticas.

« *Pour chaque problème complexe, il existe une solution simple, compréhensible... et fausse.* »

— H. L. Mencken (attribution populaire, source non confirmée)

# Table des matières

<b>Table des matières</b>	<b>i</b>
<b>Table des figures</b>	<b>iii</b>
<b>Liste des tableaux</b>	<b>vi</b>
<b>Introduction</b>	<b>1</b>
1    Introduction . . . . .	1
<b>1  Mécanique des fluides et Aérodynamique</b>	<b>6</b>
1    Mécanique des fluides . . . . .	7
1.1    Régimes d'écoulements . . . . .	7
1.2    Equations de Navier-Stokes et nombre de Reynolds . . . . .	7
2    Aérodynamisme . . . . .	10
2.1    Coefficient de traînée . . . . .	10
2.2    Force de Trainée . . . . .	11
<b>2  Principes et applications de l'apprentissage automatique</b>	<b>13</b>
1    Apprentissage Automatique . . . . .	14
2    Apprentissage profond . . . . .	14
<b>3  De la base de données au modèle : structuration, traitement et représentation</b>	<b>16</b>
1    Base de données DrivAerNet++ . . . . .	17
1.1    Calcul de la surface projetée . . . . .	17
1.2    Calcul de la force de traînée . . . . .	18
1.3    Coefficient de traînée . . . . .	19
2    Représentation 3D pour l'apprentissage . . . . .	21
2.1    Représentaion en distance signée . . . . .	23
2.2    Décomposition de la base de données . . . . .	25
2.3    Coût computationnel . . . . .	26
<b>4  De la conception à l'optimisation du modèle d'apprentissage</b>	<b>27</b>
1    Structure du modèle . . . . .	28
1.1    Encodeur . . . . .	28

1.2	Décodeurs . . . . .	30
2	Entraînement . . . . .	31
2.1	Fonction coût . . . . .	32
3	Espace des hypothèses et espace des hyperparamètres . . . . .	33
3.1	Espace des hypothèses . . . . .	33
3.2	Espace des hyperparamètres . . . . .	33
3.3	Choix d'une hypothèse et des hyperparamètres . . . . .	34
3.4	Comparaison des résultats . . . . .	39
4	Coût computationnel . . . . .	43
<b>5</b>	<b>Résultats</b>	<b>44</b>
1	Présentation des résultats . . . . .	45
1.1	Prédiction du coefficient de traînée . . . . .	45
1.2	Reconstruction en distance signée . . . . .	47
1.3	Mesures sur en dehors de la distribution . . . . .	49
<b>6</b>	<b>Mesure de l'incertitude</b>	<b>52</b>
1	Avant propos . . . . .	53
2	Mesure de l'incertitude . . . . .	53
2.1	Dropout Monte-Carlo . . . . .	54
<b>7</b>	<b>Optimisation dans l'espace latent</b>	<b>58</b>
1	Démarche d'optimisation . . . . .	59
2	Recherche dans l'espace latent par bruit blanc . . . . .	59
3	Optimisation dans l'espace latent par descente de gradient . . . . .	61
<b>Conclusion</b>		<b>65</b>
<b>Mentions légales</b>		<b>66</b>
<b>Bibliographie</b>		<b>67</b>
0.1	Démonstration Equation de Navier-Stokes . . . . .	72
0.2	Couche limite . . . . .	74
0.3	Décollement de couche limite, surpression et dépression . . . . .	74
1	Apprentissage Automatique . . . . .	76
1.1	Présentation de l'algorithme du scheduler . . . . .	76
1.2	Loss Eikonal . . . . .	76

# Table des figures

1	Diversité des géométries des modèles de voitures dans le dataset DrivAer-Net++ . . . . .	3
2	Détails d'un échantillon du dataset DrivAerNet++ . . . . .	3
1.1	Trajectoires de particules fluides d'un écoulement laminaire . . . . .	7
1.2	Trajectoires de particules fluides d'un écoulement turbulent . . . . .	7
1.3	Écoulement de fumée . . . . .	9
1.4	Surface projetée . . . . .	10
1.5	Contrainte de cisaillement . . . . .	11
1.6	Distribution de pression . . . . .	11
1.7	Contraite de cisaillement projetée selon l'axe de l'écoulement . . . . .	11
1.8	Distribution de pression arrière . . . . .	11
1.9	$C_d(x)$ . . . . .	12
2.1	L'intelligence artificielle (IA) et ses sous domaines . . . . .	15
3.1	Distribution du coefficient de traînée dans la base de données. . . . .	19
3.2	Fastback . . . . .	19
3.3	Notchback . . . . .	19
3.4	Estateback . . . . .	19
3.5	Distribution des valeurs de $C_d$ pour les trois types de carrosseries . . . . .	20
3.6	Corrélation de Pearson entre $C_d$ et l'échelle des échantillons . . . . .	24
3.7	250k points d'échantillonage . . . . .	25
3.8	Distribution des valeurs de SDF dans un échantillon de 250k points . . . . .	25
3.9	$SDF(\epsilon)$ . . . . .	25
4.1	Structure de l'encodeur . . . . .	29
4.2	Fonction de transfert d'un neurone d'une couche fully-connected . . . . .	29
4.3	Modèle de l'encodeur-décodeur à deux têtes . . . . .	31
4.4	Structure de l'encodeur . . . . .	38
4.5	Structure du décodeur SDF sans skip-connection . . . . .	38
4.6	Structure du décodeur SDF avec skip-connection . . . . .	39
4.7	Représentation des performances de différentes structures de GenNet . . . . .	40
4.8	Mesure des performances sur la prédiction de $C_d$ (MSE) selon la configuration du réseau et la fonction de coût utilisée. . . . .	40

4.9	Mesure des performances sur la distance de Chamfer selon la configuration du réseau et la fonction de coût utilisée. . . . .	40
4.10	Hallucinations lors de la reconstruction SDF . . . . .	41
4.11	Évolution de la loss Eikonal sur le set de train et de validation au cours des 250 epochs. . . . .	42
4.12	Évolution de $\ \nabla \text{SDF}\ $ sur le set de validation au cours des 250 epochs. . . . .	42
4.13	Evolution de la complexité (nombre de paramètres du modèle) en fonction de la structure et de la configuration choisie. . . . .	42
4.14	Évolution de la loss $\mathcal{L}(\text{SDF}) = \mathcal{L}_{\text{SDF}}^{\delta}$ sur le set de train et de validation au cours des 250 epochs. . . . .	43
4.15	Évolution de la loss $\mathcal{L}(C_d)$ sur le set de train et de validation au cours des 250 epochs. Valeurs mesurées sur les coefficients normalisés ce qui explique la différence avec la figure 4.7 . . . . .	43
5.1	Erreur relative moyenne sur le set de test . . . . .	46
5.2	Corrélation entre $\hat{C}_d$ et $C_d$ pour les modèles GenNet . . . . .	46
5.3	Distance de Chamfer mesurée sur le set de test pour le modèle S1. . . . .	47
5.4	Distance de Chamfer mesurée sur le set de test pour le modèle S1'. . . . .	47
5.5	Espace latent $\mathcal{Z}_{\text{train}}$ . . . . .	48
5.6	Morphing de formes . . . . .	49
5.7	Echantillons de ModelNet40 . . . . .	50
5.8	Distribution de la SDF associée à un maillage donné en entrée du modèle (Orange) et la SDF prédite (Bleu) pour les mêmes points d'échantillonage. . . . .	50
5.9	Visualisation des composantes principales des vecteurs latents associés au train, validation et OOD. . . . .	51
6.1	Mesure de l'incertitude par MC dropout. . . . .	55
6.2	Couverture empirique avec MC dropout calibré vs couverture théorique pour une loi Normale. . . . .	57
7.1	Génération de nouveaux vecteurs latents par bruit blanc . . . . .	59
7.2	3 véhicules générées par bruit blanc adapté tels que $\hat{C}_d < C_d^{\text{DrivAerNet++}}$ avec filtrage sur le maillage généré (car défauts au niveau des roues). . . . .	60
7.3	Génération avec bruit blanc constant pour $\sigma = 0.01$ . . . . .	60
7.4	Génération avec bruit blanc adapté avec $a^2 = 0.5$ . (+ filtrage car défauts au niveau des roues). . . . .	60
7.5	Corrélation de Spearman entre la distance de Mahalanobis et l'incertitude épistémique. . . . .	62
7.6	Optimisation de la moyenne $\mu = \langle \hat{C}_d \rangle$ prédite par MC dropout . . . . .	63
7.7	Évolution de l'incertitude épistémique mesurée par MC dropout . . . . .	63
7.8	Évolution du vecteur latent dans l'espace (PCA1, PCA2) pour $\lambda_d = 0$ et $\lambda_{\text{reg}} = 1$ . . . . .	63
7.9	Évolution du vecteur latent dans l'espace (PCA1, PCA2) pour $\lambda_d = 0.005$ et $\lambda_{\text{reg}} = 1$ . . . . .	63

---

7.10 Géométrie générée au bout de 1000 itérations avec la fonction coût régularisée pour $\lambda_{\text{reg}} = 1$ et $\lambda_d = 0.01$ . . . . .	64
11 Volume de contrôle . . . . .	72

# Liste des tableaux

3.1	Comparaison des coefficients de traînée pour différents types de véhicules.	20
3.2	Calcul et traitement de la base de données . . . . .	26
4.1	Hyperparamètres utilisés . . . . .	34
4.2	Hyperparamètres utilisés pour le choix des hypothèses . . . . .	36
4.3	Configurations évaluées. . . . .	37
4.4	Coût computationnel de l’entraînement . . . . .	43
5.1	Comparaison des performances de modèles de l’état de l’art pour la régression sur formes 3D. Ces performances sont toutes évaluées sur le set de test de la base de données DrivAerNet++ [14] . . . . .	45
5.2	Temps d’entraînement, d’inférence et complexité des modèles. . . . .	45
5.3	Comparaison de modèles génératifs 3D. CD : Distance de Chamfer (mean). Seulement GenNet prédit le coefficient de traînée $C_d$ en plus de faire une reconstruction en distance signée. Les benchmarks font références aux datasets utilisés pour la mesure des performances des différents modèles. . . . .	47
6.1	Couverture associée à $\pm 1\sigma$ , $\pm 2\sigma$ , $\pm 3\sigma$ pour le dropout non calibré, calibré et attendu pour une loi normale. Les tests de couverture ont été réalisés sur le set de validation. . . . .	57

# Introduction

## Abstract

L'amélioration aérodynamique de nos véhicules est un enjeu majeur afin de limiter nos émissions en gaz à effets de serre à l'origine du réchauffement climatique. Ce travail propose une méthode de génération de topologies de voitures optimisées selon leur coefficient de traînée en utilisant des modèles d'apprentissage profond. Les modèles sont entraînés sur le dataset DrivAerNet++ qui contient 8000 géométries variées et détaillées de voitures civiles ainsi que les résultats des simulations CFD (Computational Fluid Dynamics) de haute fidélité associées. La représentation des formes se fait par l'utilisation d'une distance signée (SDF), continue et différentiable, permettant l'apprentissage de réseaux de neurones profonds. GenNet permet de prédire le coefficient de traînée sur le jeu de données de test avec une erreur relative moyenne de 1,8% ainsi que reconstruire des formes complexes avec précision, obtenant une distance de Chamfer moyenne de  $0,826 \times 10^3$  entre les maillages reconstruits et d'origine du jeu de test. La génération inverse a permis la construction de nouvelles géométries de véhicules précises et plausibles avec une amélioration du coefficient de traînée de 0,22% par rapport à la meilleure géométrie du jeu d'entraînement de la base de données.

## 1 Introduction

En début d'année 2020, l'Australie a été frappée par le pire incendie de son histoire, brûlant plus de 10 millions d'hectares, détruisant des habitations de milliers de familles, faisant ainsi disparaître des communautés toutes entières. 34 personnes ont perdu la vie [27]. Les incendies et les fumées toxiques dégagées par ces feux ont également tué plus d'un milliard d'animaux et ont profondément modifié leur environnement [50]. Ces incendies ont eu lieu suite à l'année la plus chaude jamais enregistrée jusque-là.

Depuis l'an 2000, ce sont chaque année plus de 270 gigatonnes de glace qui fondent à travers le monde. On estime que les glaciers ont tous perdu entre 2% et 39% de leur masse de glace localement entre 2000 et 2023, produisant une perte totale de plus de 5% de la masse de glace du globe. La fonte des glaces est à l'origine de la montée des eaux :

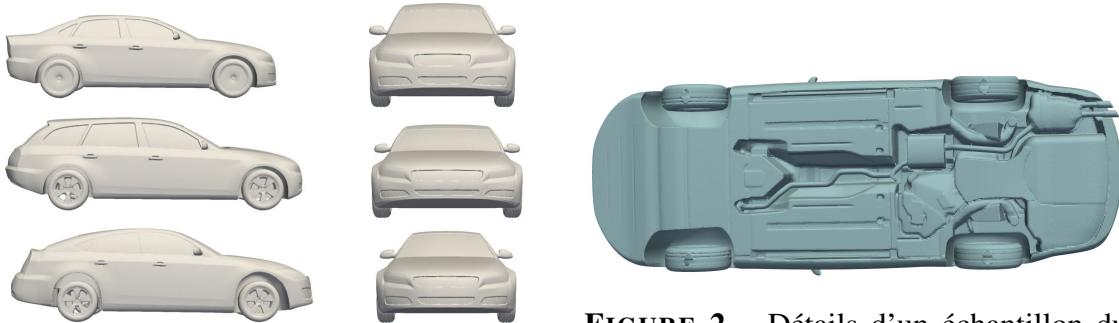
entre 1901 et 2018, le niveau moyen des océans a augmenté d'environ 15 cm à 25 cm avec une accélération notable depuis 2013 (4,62 mm d'augmentation par an entre 2013 et 2022)[25]. De plus, les glaciers montagneux représentent la plus grande source d'eau douce au monde, leur fonte menace la sécurité alimentaire et en eau de deux milliards de personnes[49].

Ces événements ont mis la question du changement climatique au centre de nos sociétés. La plupart des États comme la France ont pris des mesures pour limiter l'impact des activités humaines sur l'environnement. L'accord de Paris[1], signé en 2015, a par exemple pour objectif de limiter le réchauffement global de la planète nettement sous les 2°C et poursuivre l'objectif de 1.5°C de hausse des températures (par rapport à l'ère préindustrielle dont la période 1850-1900 est généralement prise pour référence par les experts du GIEC). A l'échelle européenne, des lois ont également été mises en place comme la loi Européenne sur le climat[2], obligeant la neutralité carbone des états membres de l'Union Européenne (UE) d'ici 2050 et visant une diminution de 55% des émissions en 2030 (référence par rapport à 1990). En 2023, les directives énergies renouvelables (RED III)[3] ont pour objectif de valoriser les énergies renouvelables à hauteur de 42,5 % du mix énergétique européen (UE). A l'échelle de l'individu, des initiatives sont mises en place afin de limiter les émissions de chacun : transport en commun, construction et rénovation de logements afin de les rendre énergétiquement performants ... Des innovations sont également réalisées dans un grand nombre de secteurs de l'industrie et notamment celle du transport. Le secteur du transport représente 15 % des émissions de gaz à effet de serre (en équivalent  $CO_2$ ) mondiales, sur ce total, 75 % proviennent du transport routier, soit un sous-total de 11 % environ. Les véhicules particuliers comptent pour 45 % des émissions du transport routier, contribuant ainsi à environ 5 % des émissions globales de  $CO_2$  émises par l'homme[26]. Dans certains pays comme les Etats-Unis, les voitures particulières représentent environ 16 % des émissions de gaz à effet de serre du pays[5]. Face à ce constat et grâce aux avancées technologiques en matière de stockage de l'énergie, la dernière décennie a vu apparaître de nouveaux véhicules : la voiture électrique ainsi que la voiture à hydrogène. Ces solutions sont censées, à terme, remplacer nos véhicules diesel ou essence actuels. Pourtant, des doutes subsistent toujours sur leur efficacité [6], notamment dans les pays où la production d'énergie est fortement carbonée, la production des batteries est également très émissive . Le passage à l'électrique peut ainsi être très compliqué à mettre en place à grande échelle si bien que certains pays ont revu leur politique publique concernant l'électrification de leur parc automobile. Le Japon par exemple, a revu son engagement de ventes 100 % électriques de véhicules légers d'ici 2035 (stratégie Green Growth) face à la faible part de marché actuelle des véhicules électriques[4, 32, 42, 44].

Pour autant, bien que les questions de l'énergie utilisée soient au cœur du débat actuel, d'autres moyens permettant d'améliorer les performances en matière d'émissions de gaz à effet de serre de nos véhicules existent. L'un de ces moyens consiste en l'amélioration des performances aérodynamiques des véhicules. L'aérodynamisme joue un rôle très important dans les émissions qu'un véhicule génère, plus une voiture est aérodynamique, moins

elle est freinée par l'air qui l'entoure. Créer ou conserver du mouvement nécessite donc un apport d'énergie plus faible de la source, ce qui garantit une consommation réduite. Cette approche possède de plus l'avantage d'être applicable à de nombreux véhicules différents (voitures, camions, bus ...) et à pouvoir s'appliquer quelle que soit la source d'énergie utilisée (électrique, essence, diesel ...). L'aérodynamisme d'un véhicule est impacté par la forme de celui-ci ainsi que la rugosité des matériaux composants l'extérieur de la voiture.

Dans cette étude, nous proposons une méthode permettant de générer des formes 3D optimisées topologiquement selon des contraintes d'aérodynamisme (coefficient de traînée) en utilisant des modèles d'apprentissage automatique profonds sur la base de données DriAerNet++[14] comprenant 8000 simulations CFD (Computational Fluid Dynamics) sur des géométries de voitures variées et détaillées.



**FIGURE 1** – Diversité des géométries des modèles de voitures dans le dataset DrivAerNet++ [14]

**FIGURE 2** – Détails d'un échantillon du dataset DrivAerNet++

Ce mémoire s'accompagne d'un dépôt de code disponible sur Github

<sup>1</sup>.

1. [github.com/Bastien-Jacques/GenNet](https://github.com/Bastien-Jacques/GenNet)

---

## Notations

### Mécanique des fluides et aérodynamique

Symbole	Description
$T$	Température ( $K$ )
$\rho$	Masse volumique ( $Kg/m^3$ )
$P$	Pression ( $Pa$ )
$V$	Vitesse ( $m.s^{-1}$ )
$P_{\text{dyn}} = \frac{1}{2}\rho v^2$	Pression dynamique ( $Pa$ )
$\mu$	Viscosité dynamique ( $Pa.s$ )
$m$	Masse ( $Kg$ )
$t$	Temps ( $s$ )
$L$	Longueur ( $m$ )
$v^*$	Vitesse adimensionnée
$x^*$	vecteur position adimensionné
$t^*$	Temps adimensionné
$p^*$	Pression adimensionnée
$\Omega$	Volume de contrôle ( $m^3$ )
$\partial\Omega$	Frontière du volume de contrôle ( $m^2$ )
$d\Omega$	Volume infinitésimal ( $m^3$ )
$dS$	Surface infinitésimale ( $m^2$ )
$f_v$	Force volumique ( $N/m^3$ )
$\mathbf{T}$	Force surfacique ( $N/m^2$ )
$\sigma$	Tenseur des contraintes ( $Pa$ )
$\tau$	Tenseur des efforts de cisaillement ( $Pa$ )
$\mathbf{n}$	Vecteur normal à la surface
$\nabla$	Divergence ou gradient en fonction du contexte
$\nabla$	Divergence
$\Delta$	Laplacien
$\mathbf{1}$	Matrice identité
$\delta$	Symbôle de Kronecker
$C_d$	Coefficient de traînée
$F_d$	Force de traînée ( $N$ )
$A$	Surface projetée frontale ( $m^2$ )
$U$	Vitesse à l'infini ( $m.s^{-1}$ )
$\otimes$	Produit dyadique

---

## Base de données et apprentissage

Symbol	Description
$\Omega$	Volume d'un corps ( $m^3$ )
$\Omega_d$	Forme discrétisée en maillage
$F_{\text{pressure}}$	Force de pression ( $N$ )
$F_{\text{cis}}$	Force de cisaillement ( $N$ )
$N_{\text{cells}}$	Nombre de cellules d'un maillage
$S_i$	Surface d'une cellule i ( $m^2$ )
$r$	Corrélation de Pearson
$\mathcal{N}(\mu, \sigma^2)$	Loi Normale de moyenne $\mu$ et d'écart-type $\sigma$
$z$	Vecteur latent
$n_{\text{lat}}$	dimension des vecteurs latents
$\mathcal{Z}$	Espace latent
$\phi$	Fonction d'activation
$n_{\text{epochs}}$	Nombre d'éPOCHS
$\mathbf{E}$	Encodeur
$\mathbf{D}_{\text{SDF}}$	Décodeur géométrique
$\mathbf{D}_\phi :$	Décodeur physique
$C_d$	Coefficient de traînée vrai
$\hat{C}_d$	Coefficient de traînée prédit
$SDF$	Distance signée vraie
$\theta$	Poids du réseau
$n_{\text{layers}}$	Nombre de couches du réseau
$d_{\text{hidden}}$	Nombre de neurones par couche cachée
$\mathcal{F}$	Espace des hypothèses
$f$	fonction de l'espace des hypothèses
$\mathcal{D}$	Distribution inconnue des données
$\mathcal{S}$	Echantillon tiré de $\mathcal{D}$
$\mathcal{H}$	Espace des hyperparamètres
$h$	Jeu d'hyperparamètres
$\mathcal{L}$	Fonction de coût
$\ell$	Fonction de perte élémentaire
$\mathcal{L}_{\text{Eikonal}}$	Fonction coût Eikonal
$\mathcal{L}_2$	Fonction coût des moindres aux carrés
$\mathcal{L}_1^\delta$	Fonction de coût Clamped
$\delta$	Paramètre de la fonction coût Clamped
$\mathbb{E}$	Espérance
$\mathcal{R}$	Risque : espérance de la perte sur la distribution $\mathcal{D}$
$\hat{\mathcal{R}}$	Risque empirique : moyenne des pertes sur l'échantillon $S$

# Chapitre 1

## Mécanique des fluides et Aérodynamique

*Dans ce premier chapitre, nous verrons les bases de la mécanique des fluides, les équations de Navier-Stokes et le nombre de Reynolds. On introduira ces notions afin d'expliquer les phénomènes qui entrent en jeu lorsqu'il s'agit de quantifier les performances aérodynamiques d'un véhicule.*

### Sommaire

---

<b>1</b>	<b>Mécanique des fluides</b>	<b>7</b>
1.1	Régimes d'écoulements	7
1.2	Equations de Navier-Stokes et nombre de Reynolds	7
<b>2</b>	<b>Aérodynamisme</b>	<b>10</b>
2.1	Coefficient de traînée	10
2.2	Force de Trainée	11

---

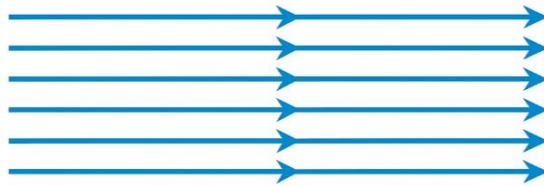
# 1 Mécanique des fluides

## 1.1 Régimes d'écoulements

On distingue trois régimes d'écoulement différents :

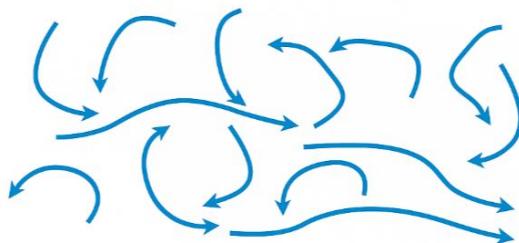
- **Écoulement laminaire** : Les particules d'air glissent parfaitement les unes sur les autres sans échange de particules entre elles. Elles suivent un mouvement rectiligne et parallèle. L'écoulement laminaire est un régime d'écoulement caractérisé par une diffusion de quantité de mouvement ( $p = mv$ ) élevée et une convection faible, c'est-à-dire peu de mouvement interne.

D'un point de vue microscopique, deux particules fluides voisines à un instant donné restent voisines aux instants suivants.



**FIGURE 1.1 – Trajectoires de particules fluides d'un écoulement laminaire**

- **Écoulement turbulent** : L'écoulement est très désordonné, les particules se mélangent et ne suivent ni une trajectoire rectiligne ni parallèle. Certaines particules peuvent se déplacer dans le sens opposé de la masse, formant des tourbillons. Les vitesses d'écoulement et la pression subissent des changements chaotiques. La turbulence est causée par l'excès d'énergie cinétique dans certaines parties de l'écoulement du fluide, l'énergie cinétique en excès venant contrecarrer l'effet d'amortissement apporté par la viscosité du fluide.



**FIGURE 1.2 – Trajectoires de particules fluides d'un écoulement turbulent**

## 1.2 Equations de Navier-Stokes et nombre de Reynolds

En mécanique des fluides, les équations de **Navier-Stokes** sont des équations aux dérivées partielles non linéaires permettant de décrire le comportement des fluides **Newtoniens** (les

gaz et les liquides). Bien que l'existence de solutions analytiques n'ait pas été prouvée, ces équations sont utilisées dans de nombreux domaines pour modéliser le comportement des fluides. Les équations de **Navier-Stokes** sont utilisées numériquement afin d'apporter des solutions à des problèmes de mécanique des fluides complexes. Les solutions sont approchées en discrétilisant le problème à traiter (méthode des **éléments finis, volumes finis, différences finies...**).

Les équations de **Navier-Stokes** découlent de deux principes fondamentaux de la mécanique des milieux continus :

- **Principe de conservation de la masse** : Soit un volume de contrôle  $\Omega$  entouré par une surface fermée  $\partial\Omega$ , la masse d'un fluide contenue dans ce volume de contrôle est conservée au cours du temps. **Toute variation de masse dans ce volume de contrôle résulte uniquement d'un flux de masse entrant ou sortant.**

On a :

$$m = \int_{\Omega} \rho \, d\Omega$$

La conservation de la masse au cours du temps donne donc :

$$\frac{\partial m}{\partial t} = 0 \longrightarrow \frac{\partial}{\partial t} \int_{\Omega} \rho \, d\Omega = 0 \longrightarrow \int_{\Omega} \frac{\partial \rho}{\partial t} + \nabla(\rho \cdot \mathbf{v}) \, d\Omega = 0$$

Soit :

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \cdot \mathbf{v}) = 0$$

Qui peut se développer sous la forme :

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{v}$$

En régime permanent ( $\frac{\partial \rho}{\partial t} = 0$ ) et pour un fluide incompressible ( $\nabla \rho = 0$ ), on obtient :

$$\nabla \cdot \mathbf{v} = 0 \tag{*}$$

- **Principe de conservation de la quantité de mouvement** : Soit  $\Omega$  un volume de contrôle et  $\partial\Omega$  la surface fermée qui l'entoure. La variation de la quantité de mouvement d'un fluide contenu dans  $\Omega$  est égale à la somme des forces exercées sur ce fluide, c'est-à-dire la résultante des forces de volume (la gravité par exemple) et des forces de surface (pressions et contraintes visqueuses). On peut donc écrire :

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{v} \, d\Omega = \int_{\Omega} \mathbf{f}_v \, d\Omega + \int_{\partial\Omega} \mathbf{T} \, dS$$

où  $\mathbf{f}_v$  est une force volumique ( $N/m^3$ ),  $\mathbf{T}$  est une force surfacique s'exerçant sur  $\partial\Omega$  et  $\mathbf{n}$  la normale sortante à  $\Omega$  pour toute surface infinitésimale  $dS$  de  $\partial\Omega$ .

Le **théorème de Reynolds** donne :

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{v} \, d\Omega = \int_{\Omega} \frac{\partial(p\mathbf{v})}{\partial t} \, d\Omega + \int_{\partial\Omega} \rho \mathbf{v} (\mathbf{v} \cdot \mathbf{n}) \, dS$$

Sachant de plus :

$$\mathbf{T} = \boldsymbol{\sigma} \cdot \mathbf{n} \text{ et } \boldsymbol{\sigma} = -p\mathbf{1} + \boldsymbol{\tau}$$

On obtient (en négligeant la force volumique) :

$$\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \mu \Delta \mathbf{v} \quad (1.1)$$

Qui s'écrit sous sa forme adimensionnée :

$$\frac{\partial \mathbf{v}^*}{\partial t^*} + \mathbf{v}^* \cdot \nabla^* \mathbf{v}^* = -\nabla^* p^* + \frac{1}{Re} \Delta^* \mathbf{v}^*$$

On fait alors apparaître le **nombre de Reynolds** dans les équations de **Navier-Stokes**[8, 35] :

$$Re = \frac{\rho U L}{\mu}$$

- Si  $Re \ll 1$ , l'écoulement est **lamininaire**.
- Si  $Re \gg 1$ , l'écoulement est **inertiel**, potentiellement **turbulent**.



**FIGURE 1.3 –** Ecoulement de fumée, présence d'un régime laminaire à l'origine de l'émission de la fumée, l'écoulement est structuré, l'ensemble des particules de gaz se déplacent uniformément vers le haut. On observe ensuite une zone limite, l'écoulement devient de plus en plus chaotique, les particules de fumée ne se déplacent plus uniformément formant des tourbillons. A l'origine de l'écoulement, la longueur caractéristique de l'écoulement est très faible, la vitesse est également faible (vitesse nulle à l'origine), on a donc  $Re = \frac{\rho U L}{\mu} \ll 1$  ce qui garantie en effet un écoulement laminaire. La fumée, sous l'action de la poussée d'Archimède est accélérée verticalement vers le haut, sa vitesse augmente ainsi que sa longueur caractéristique de sorte que le nombre de Reynolds croît. Une fois la valeur limite atteinte, l'écoulement devient turbulent. (Image Onera)

## 2 Aérodynamisme

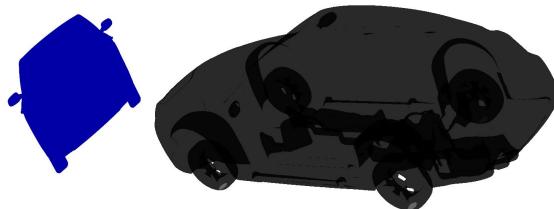
### 2.1 Coefficient de traînée

Les zones de surpression, de dépression, ainsi que les efforts de cisaillement que l'air exerce sur les parois d'un véhicule, ont un impact direct sur ses performances aérodynamiques. Pour mesurer à quel point ces effets freinent un véhicule, et ainsi quantifier ses performances, on utilise le **coefficent de traînée**  $C_d$ . Le coefficient de traînée est une **grandeur adimensionnelle** largement utilisée pour quantifier la performance aérodynamique d'un objet exposé à un écoulement. Le coefficient de traînée est indépendant du fluide utilisé, de sa vitesse d'écoulement à l'infini ainsi que de la taille du corps considéré, puisque sans dimension. Ces propriétés le rendent donc très utile pour comparer les performances aérodynamiques de plusieurs véhicules en ne prenant en compte que leur géométrie et en supprimant les effets d'échelle. Le coefficient de traînée est l'un des deux coefficients construits à partir du théorème de **Pi-Buckingham** appliqué au problème de la force de traînée (avec le **nombre de Reynolds**) et s'exprime comme :

$$C_d = \frac{F_d}{\frac{1}{2}\rho U^2 A}$$

$F_d$  correspond à la **force de traînée** que l'écoulement d'air exerce sur le véhicule,  $\rho$  est la **masse volumique** du fluide,  $U$  la **vitesse du fluide à l'infini** et  $A$  la **surface de référence**.

Le calcul du coefficient de traînée repose sur le choix d'une **surface de référence**  $A$ , qui doit être cohérent avec la physique de l'écoulement pour garantir des comparaisons valides. En aérodynamique, notamment dans le domaine automobile, la convention largement adoptée est d'utiliser la **surface projetée frontale** [7, 24], c'est-à-dire la surface **orthogonale** à la direction de l'écoulement. Utiliser une autre surface, comme la surface mouillée ou une surface arbitraire, invaliderait l'interprétation du coefficient de traînée en tant qu'indicateur de performance aérodynamique normalisée, et empêcherait toute comparaison fiable entre formes ou entre publications.



**FIGURE 1.4 – Surface projetée d'un échantillon du dataset DrivAerNet++**

## 2.2 Force de Trainée

La force exercée par un fluide sur une surface  $\partial\Omega$  peut se calculer comme suit :

$$\mathbf{T} = \int_{\partial\Omega} \boldsymbol{\sigma} \cdot \mathbf{n} \, dS = \int_{\partial\Omega} (-p\mathbf{1} + \boldsymbol{\tau}) \cdot \mathbf{n} \, dS$$

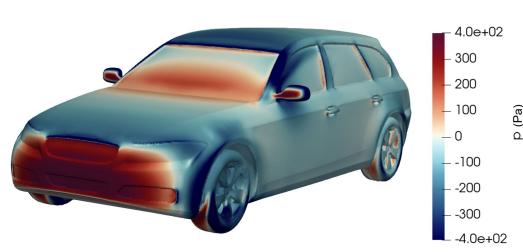
La force de traînée est la composante dirigée selon la direction de l'écoulement (souvent  $e_x$ ) de cette force. Soit :

$$F_d = - \underbrace{\int_{\partial\Omega} p \mathbf{n} \cdot \mathbf{e}_x \, dS}_{\text{force de pression}} + \underbrace{\int_{\partial\Omega} \boldsymbol{\tau} \mathbf{n} \cdot \mathbf{e}_x \, dS}_{\text{force de cisaillement}}$$

Connaissant la distribution de pression ainsi que les forces de cisaillement exercées par un écoulement de fluide sur la surface d'un véhicule, on peut ainsi déterminer le coefficient de traînée d'un véhicule.



**FIGURE 1.5** – Norme de la contrainte de cisaillement calculé par CFD sur un échantillon de DrivAerNet++ [14]



**FIGURE 1.6** – Distribution de pression (relative à  $p_{atm}$ ) calculée par CFD sur un échantillon de DrivAerNet++



**FIGURE 1.7** – Contraite de cisaillement projetée selon l'axe de l'écoulement ( $e_x$ ) :  $\boldsymbol{\tau} \cdot \mathbf{n} \cdot e_x$  pour un échantillon du dataset DrivAerNet++



**FIGURE 1.8** – Distribution de pression (relative à  $p_{atm}$ ) à l'arrière d'un échantillon de DrivAerNet++

On remarque que les zones de surpression les plus importantes sont localisées au niveau du pare-chocs et du pare-brise de la voiture. Cela s'explique puisque ces zones sont exposées avec une forte incidence à des flux d'air importants et à grande vitesse. Les zones latérales, le capot ainsi que le toit sont exposés à de légères dépressions causées par décollement de couche limite (voir Annexe A). L'arrière de la voiture est également soumis à une dépression due à la cassure géométrique forte de la voiture, créant un décollement inertiel (voir annexe A).

Les côtés du véhicule ainsi que le toit et le plancher n'ont pas d'impact majeur sur la force de pression que l'air exerce sur le véhicule puisque les normales à ces surfaces sont orthogonales à la direction d'écoulement de l'air, on a donc  $\mathbf{n} \cdot \mathbf{e}_x \approx 0$ . En revanche, ces grandes surfaces contribuent à la force de cisaillement.

Le pare-choc ainsi que le pare-brise contribuent fortement à la force de pression puisque les pressions sont élevées et les normales à ces surfaces sont presque colinéaires à la direction d'écoulement, on a en effet :

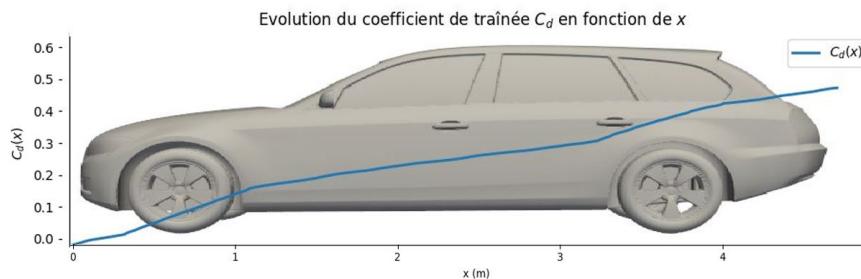
$$p > 0 \text{ et } \mathbf{n} \cdot \mathbf{e}_x \approx -1 \longrightarrow -p \mathbf{n} \cdot \mathbf{e}_x \gg 0$$

Ainsi, ces zones de surpression créent une force qui pousse la voiture vers l'arrière demandant un apport d'énergie supplémentaire pour contrer cette force de pression.

La zone arrière de la voiture est-elle soumise à une dépression. On a donc  $p < 0$  ce qui entraîne également une force de pression positive.

$$p < 0 \text{ et } \mathbf{n} \cdot \mathbf{e}_x \approx +1 \longrightarrow -p \mathbf{n} \cdot \mathbf{e}_x \gg 0$$

La zone de dépression à l'arrière du véhicule 'aspire' le véhicule vers l'arrière, ce qui l'empêche d'avancer sans un apport d'énergie supplémentaire.



**FIGURE 1.9 –** Evolution du coefficient de traînée  $C_d(x)$  où  $x$  représente l'abscisse liée à la longueur de la voiture pour un échantillon du dataset DrivAerNet++. On remarque que le coefficient de trainée augmente fortement à l'avant et l'arrière du véhicule à cause des zones de surpression/dépression alors qu'il augmente plus faiblement pour  $x \in [1, 3.5] \text{ m}$ , correspondant à la partie centrale de la voiture moins exposée à ces zones de fortes ou faibles pressions.

## Chapitre 2

# Principes et applications de l'apprentissage automatique

*Ce chapitre fait office d'introduction à l'apprentissage automatique et à l'apprentissage profond. Seront définis ici les principes fondateurs de ces théories ainsi qu'une liste de leurs applications.*

### Sommaire

---

1	Apprentissage Automatique . . . . .	14
2	Apprentissage profond . . . . .	14

---

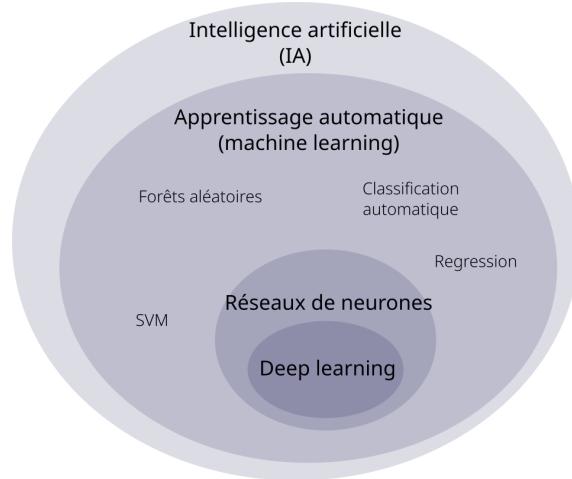
## 1 Apprentissage Automatique

L'**apprentissage automatique** est une branche de l'**intelligence artificielle** consistant à faire apprendre des comportements à des machines. L'**apprentissage automatique** se distingue par l'utilisation de **données** : un programme apprend à interpréter des résultats liés à ces données. Cette phase d'**apprentissage** est réalisée lors de l'**entraînement** du modèle. Une fois cette phase d'entraînement terminée, le modèle est capable, s'il a été correctement entraîné, de **prédir des résultats** à partir de données jamais vues lors de son entraînement. Cette capacité est évidemment très puissante, puisqu'elle permet d'une part, de traiter des quantités de données importantes rapidement et d'autre part de détecter des informations fines (appelées '**signaux faibles**') dans les données qui échappent à l'humain, permettant de générer des prédictions compliquées, voire impossibles à réaliser pour un humain (prédiction de cancer sur des images médicales, prédiction des contraintes dans une structure complexe sous chargement...). Bien que les modèles d'intelligence artificielle soient en mesure de traiter des problèmes de plus en plus complexes et aient dépassé l'humain dans bon nombre de domaines (1997 : victoire aux échecs du modèle IA 'Deep Blue' face à Garry Kasparov alors considéré comme le meilleur joueur du monde), certaines tâches extrêmement simples pour des humains sont encore difficiles pour des machines (reconnaissance faciale, reconnaissance d'émotions ... ), c'est le **paradoxe de Moravec**. On peut expliquer ce paradoxe par plusieurs raisons :

- L'Homme a évolué durant des millions d'années en apprenant à s'**adapter** à son **environnement**, rendant simples des aptitudes nécessaires à sa survie (distinguer un humain d'un animal, distinguer des émotions...).
- Les tâches simples sont souvent des **problèmes mal posés**, la reconnaissance visuelle par exemple dépend de l'angle de vue, de la luminosité ... rendant l'apprentissage pour un modèle compliqué.
- A contrario, les tâches compliquées pour l'homme (échecs, jeu de go ...) sont des **problèmes bien posés**, structurés par des **règles** limitant l'ambiguité ainsi qu'un **cadre mathématique clair**.

## 2 Apprentissage profond

L'**apprentissage profond** (**Deep Learning**) est une branche de l'**intelligence artificielle** (**IA**) consistant en l'utilisation de **réseaux de neurones** artificiels structurés sous forme de **couches** (plus le nombre de couches est important, plus le réseau est dit 'profond'). Ces structures permettent de résoudre des problèmes complexes, en s'appuyant sur des **données d'entraînement**. Plus les données sont en nombre important, meilleure sera la capacité du modèle d'apprendre à partir de celles-ci.



**FIGURE 2.1** – L'intelligence artificielle (IA) et ses sous domaines. Le **deep learning** est une branche du **machine learning**, (apprentissage automatique). L'apprentissage automatique est fondé sur la théorie de l'apprentissage statistique. Image tiré du Livre 'Deep Learning' de Ian Goodfellow [16]

Le **machine learning** comprend une variété importante de **modèles** permettant de résoudre des problèmes selon les grands types suivants :

- **Régression** : Prédiction d'une valeur continue (ex : température, coefficient de traînée ...)
- **Classification** : Prédire une classe discrète (ex : spam/non-spam, chat/chien ...)
- **Ségmentation** : Classer chaque pixel/point (ex : imagerie médicale).
- **Séries temporelles** : Prédictions dépendantes du temps (ex : cours boursier ...)

Dans notre cas, nous utilisons un modèle de **Deep Learning** afin de prédire le coefficient de traînée d'un véhicule, ainsi que de générer des formes de véhicules optimisées selon le coefficient de traînée. Ces deux tâches sont des tâches de régression (voir Chapitre 4).

# Chapitre 3

## De la base de données au modèle : structuration, traitement et représentation

*Ce chapitre introduit la structure de la base de données DrivAerNet++ utilisée pour ce travail, les objectifs de son utilisation, tous les éléments de prétraitement des données réalisés en amont de l'entraînement de notre modèle d'apprentissage profond ainsi que la méthode de représentation des données utilisée pour garantir un bon entraînement de notre modèle.*

### Sommaire

---

<b>1</b>	<b>Base de données DrivAerNet++</b>	<b>17</b>
1.1	Calcul de la surface projetée	17
1.2	Calcul de la force de traînée	18
1.3	Coefficient de traînée	19
<b>2</b>	<b>Représentation 3D pour l'apprentissage</b>	<b>21</b>
2.1	Représentaion en distance signée	23
2.2	Décomposition de la base de données	25
2.3	Coût computationnel	26

---

## 1 Base de données DrivAerNet++

La base de données DrivAerNet++[14] contient 8000 exemplaires de voitures aux géométries variées et détaillées ainsi que les simulations CDF associées réalisées avec le logiciel `OpenFoam`. La grande quantité de données permet une approche data-driven et justifie l'utilisation de modèles d'apprentissage profond.

Chaque échantillon de la base de données peut être décomposé selon trois informations essentielles :

- **La Structure du maillage** : chaque échantillon possède un maillage décrivant la géométrie de la voiture. Le maillage est défini comme un ensemble de points ('vertices') avec leurs coordonnées dans le repère du véhicule, ainsi que les liens entre chaque point formant des 'faces' ou 'cellules'.
- **Le champ de pression** : chaque échantillon possède le champ de pression associé à la simulation CFD. Chaque cellule du maillage se voit attribuer une valeur de pression (valeur scalaire).
- **Le champ des vecteurs de cisaillement** : chaque échantillon possède le champ de l'effort de cisaillement. Il est attribué à chaque cellule du maillage le vecteur correspondant à l'effort de cisaillement s'exerçant localement ( $\tau = \underline{\tau} \cdot \mathbf{n}$  où  $\mathbf{n}$  est la normale sortante à la cellule).

La première étape consiste à calculer le coefficient de traînée de chaque échantillon de la base de données. Pour cela, on rappelle l'expression du coefficient de traînée :

$$C_d = \frac{F_d}{\frac{1}{2}\rho U^2 A}$$

avec :

$$F_d = - \int_{\partial\Omega} p \mathbf{n} \cdot \mathbf{e}_x dS + \int_{\partial\Omega} \boldsymbol{\tau} \cdot \mathbf{n} \cdot \mathbf{e}_x dS$$

et A la surface projetée.

### 1.1 Calcul de la surface projetée

La surface projetée est définie comme :  $A = p_{yz}(\Omega)$  la projection orthogonale de  $\Omega$  sur le plan  $yz$ .

On peut calculer cette surface comme :

$$A = \int_{\partial\Omega} \max(0, \mathbf{n} \cdot \mathbf{e}_x) dS$$

En effet,  $\mathbf{e}_x$  correspondant à l'axe de la longueur de la voiture (c'est-à-dire la normale au plan  $(yz)$  sur lequel on projette la surface), le produit scalaire  $\mathbf{n} \cdot \mathbf{e}_x$  correspond à la projection du vecteur normal à la surface sur le plan  $(yz)$ . On ajoute également la fonction  $x : \rightarrow \max(x)$  afin de ne compter qu'une seule fois les surfaces.

Dans notre cas, toute forme  $\Omega$  est discrétisée en un maillage  $\Omega_d$ , on peut donc calculer  $A$  comme :

$$A = \sum_i^{N_{cells}} \max(0, \mathbf{n}_i \cdot \mathbf{e}_x) S_i = \begin{bmatrix} \max(0, \mathbf{n}_0 \cdot \mathbf{e}_x) \\ \vdots \\ \max(0, \mathbf{n}_{N_{cells}} \cdot \mathbf{e}_x) \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ \vdots \\ S_{N_{cells}} \end{bmatrix}$$

où  $\mathbf{n}_i$  est la normale à la cellule i,  $S_i$ , la surface de la cellule i,  $N_{cells}$  le nombre de cellules contenues dans le maillage  $\Omega_d$ . La forme vectorisée permet d'optimiser la vitesse des calculs.

## 1.2 Calcul de la force de traînée

De la même manière que pour la surface projetée, on peut calculer la force de traînée (qui se décompose en une force de pression et une force de cisaillement) pour un maillage en utilisant l'expression de  $F_d$  discrétisée :

$$F_{pressure} = -\sum_i^{N_{cells}} p_i \mathbf{n}_i \cdot \mathbf{e}_x S_i = \left( \begin{bmatrix} p_0 \\ \vdots \\ p_{N_{cells}} \end{bmatrix} \circ \begin{bmatrix} S_0 \\ \vdots \\ S_{N_{cells}} \end{bmatrix} \right) \cdot \left( \begin{bmatrix} \mathbf{n}_0 \cdot \mathbf{e}_x \\ \vdots \\ \mathbf{n}_{N_{cells}} \cdot \mathbf{e}_x \end{bmatrix} \right)$$

Où  $\circ$  est le produit de Hadamard.

Enfin, la force de cisaillement :

$$F_{cis} = \sum_i^{N_{cells}} \underline{\underline{\tau}} \mathbf{n} \cdot \mathbf{e}_x = \sum_i^{N_{cells}} \boldsymbol{\tau} \cdot \mathbf{e}_x = \begin{bmatrix} \boldsymbol{\tau}_0 \\ \vdots \\ \boldsymbol{\tau}_{N_{cells}} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{e}_x \\ \vdots \\ \mathbf{e}_x \end{bmatrix}$$

Avec  $F_d$  et  $A$ , on peut calculer  $C_d$  pour chaque échantillon de la base de données. Pour réaliser ces calculs, on utilise les librairies `Numpy` [19] et `Pyvista` [46] qui permettent de faire du calcul vectoriel ainsi que de la lecture et du traitement de fichiers vtk.

## 1.3 Coeffcient de traînée

### 1.3.1 Distribution du coefficient de traînée

Les coefficients de traînée des échantillons de la base de données ont tous été calculés avec OpenFoam dans les mêmes conditions de simulation :  $U_\infty = 30 \text{ m/s}$ ,  $\rho = \rho_{\text{air}} = 1.225 \text{ Kg/m}^3$  et  $\mu = \mu_{\text{air}} = 1,81 \cdot 10^{-5} \text{ Pa.s}$ . Une fois la phase de calcul des coefficients de traînée terminée, il est important de comprendre sa distribution au sein du dataset afin de réaliser les opérations de prétraitement adéquates :

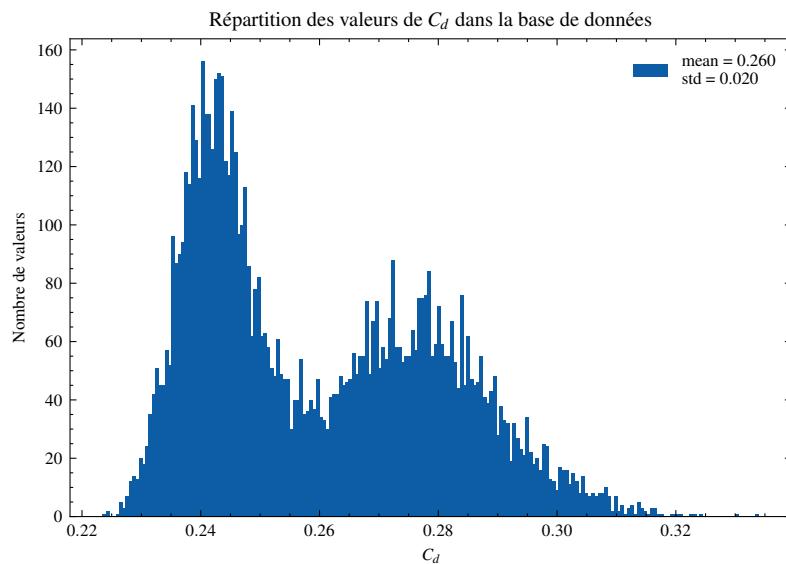


FIGURE 3.1 – Distribution du coefficient de traînée dans la base de données.

On remarque que la distribution est bimodale avec un premier mode autour de la valeur  $C_d \approx 0,25$  et un deuxième autour de  $C_d \approx 0,28$ . Cette bimodalité s'explique par la composition de la base, qui comprend trois types de carrosseries : *Fastback*, *Notchback* et *Estateback*.



FIGURE 3.2 – Carrosserie de type *Fastback*.

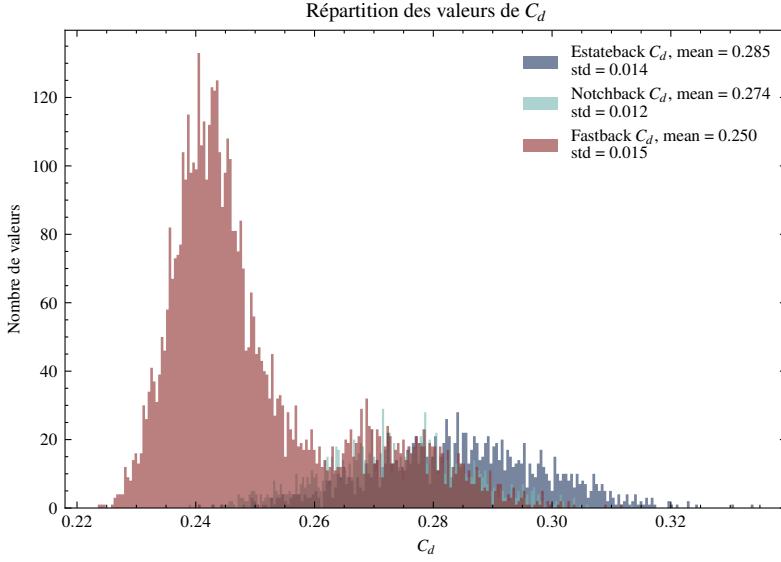


FIGURE 3.3 – Carrosserie de type *Notchback*.



FIGURE 3.4 – Carrosserie de type *Estateback*.

La base de données est composée de 5200 exemplaires de carrosserie de type *Fastback* ainsi que de 1400 de *Notchback* et 1400 de *Estateback*. On peut représenter les distributions du coefficient de traînée pour chacun de ces types de carrosserie.



**FIGURE 3.5** – Distribution des valeurs de  $C_d$  pour les trois types de carrosseries

On remarque que les voitures avec une carrosserie de type *Fastback* possèdent globalement un coefficient de traînée plus faible que les carrosseries de type *Notchback* et *Estateback*. Cette observation est bien reconnue dans la littérature concernant l'aérodynamique des voitures [21, 23, 28, 45].

	moyenne $\mu$	écart-type $\sigma$	minimum	maximum
Fastback	0.249	0.014	0.223	0.311
Notchback	0.273	0.011	0.238	0.313
Estateback	0.284	0.014	0.240	0.333

**TABLE 3.1** – Comparaison des coefficients de traînée pour différents types de véhicules.

### 1.3.2 Standardisation du coefficient de traînée

Différentes méthodes de mise à l'échelle peuvent être envisagées pour rendre la variable  $C_d$  adaptée à l'apprentissage automatique. Nous présentons ici trois approches courantes, en discutant de leur pertinence dans notre cas.

La **Standardisation (Z-score scaling)**[20, 22]. Elle transforme toute variable  $x$  d'un ensemble de données X en :

$$x_{stnd} = \frac{x - \mu}{\sigma}$$

où  $\mu$  est la **moyenne** des valeurs de X et  $\sigma$  l'**écart-type** de X. La standardisation permet d'obtenir un ensemble de données  $X_{strd}$  de moyenne nulle et d'écart-type égal à 1 (voir démonstration Annexe B). La standardisation suppose néanmoins une certaine **symétrie** de la distribution des données, ce qui n'est pas le cas pour notre jeu de  $C_d$ .

- Une autre approche courante pour transformer une variable continue est l'utilisation d'un **Quantile Transformer**. Cette méthode consiste à remplacer chaque valeur par son **quantile empirique** (sa position dans la distribution triée des données), puis à mapper ces quantiles vers une distribution cible, souvent normale ou uniforme. Cette transformation présente l'avantage de corriger les asymétries (skewness) dans la distribution initiale et de produire une variable transformée adaptée à l'apprentissage par réseaux de neurones. Cependant, son **inversion** étant **approchée** (via interpolation sur les quantiles appris), cela peut introduire une **perte de précision** lors de la restitution du coefficient de traînée réel après prédiction. De plus, cette méthode peut perturber la structure relative des écarts entre échantillons, ce qui compromet l'interprétation physique des valeurs transformées. Enfin, la distribution initiale ne présente pas de skew extrême justifiant une transformation non linéaire aussi agressive.

#### Normalisation MinMax

- La **normalisation MinMax**[18] : consiste à normaliser les données de sorte à réduire la distribution de valeurs d'un ensemble de données à [0,1], on a :

$$x_{normed} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

La normalisation **MinMax** est très simple à mettre en place et permet de traiter les données sans a priori sur la distribution initiale. Dans notre cas, la normalisation est également intéressante dans la mesure où la distribution initiale ne présente pas de valeurs aberrantes qui pourraient écraser l'information ( $\min(C_d) \approx 0.223$  et  $\max(C_d) \approx 0.333$ ).

Pour ces raisons, une normalisation MinMax a été choisie afin de traiter la distribution de valeurs du coefficient de traînée du dataset.

## 2 Représentation 3D pour l'apprentissage

L'objectif étant de générer des formes de voitures optimisées selon leur coefficient de traînée, on distingue deux sous-objectifs :

- (1) : Apprendre à générer des formes.
- (2) : Apprendre à prédire le coefficient de traînée d'un véhicule.

Ces deux tâches nécessitent d'apprendre la représentation d'une forme, pour (1), cela permet ensuite de générer de nouvelles formes à partir de celles apprises, pour (2), cela permet de comprendre la géométrie de la voiture et ainsi de lui associer le bon coefficient de traînée.

Il existe quatre moyens principaux de représenter une forme 3D :

- La représentation sous forme de **maillage** : il s'agit de la représentation initiale des

voitures de la base de données. Cette méthode est très utilisée pour représenter les formes 3D. Néanmoins, cette structure ne permet pas de réaliser correctement la génération 3D puisque le maillage fixe une topologie empêchant le modèle de générer des formes mixtes tirées de celles qu'il a apprises lors de son entraînement.

- La représentation sous forme de **nuage de points**. On représente un objet par une distribution de points permettant de donner une description **discrète** d'un objet. Cette technique est couramment utilisée en apprentissage automatique, PointNet[39] par exemple, utilise cette méthode pour faire de la classification et segmentation d'objets 3D. Cette méthode ne permet néanmoins pas de générer des formes continues ainsi que des surfaces fermées.
- La représentation en **voxels** : La représentation en Voxels est la plus naturelle puisqu'elle est une extension directe de la représentation 2D des images : les **pixels**. Les voxels représentent une forme 3D en découplant l'espace tridimensionnel, on peut alors définir des formes en utilisant une grille d'occupation dense (occupé/non-occupé). Néanmoins, le coût de calcul augmente fortement à mesure que la résolution augmente, si bien que les méthodes actuelles ne permettent pas de générer des formes fidèlement (la résolution  $128^3$  est difficilement dépassée).

### Distance signée

- La représentation en **distance signée**. La représentation en distance signée (**Signed Distance Function : SDF**) consiste à définir la **distance Euclidienne** entre chaque point  $\mathbf{x}$  de l'espace et la surface  $\partial\Omega$  d'une forme  $\Omega$ . La distance signée est donc, en théorie, une fonction continue  $f : \mathbf{R}^3 \rightarrow \mathbf{R}$ . En pratique, calculer la distance signée de manière continue n'est pas possible, on peut alors discréteriser la SDF en calculant la distance séparant  $\partial\Omega$  avec des points d'échantillonnage placés dans  $\mathbf{R}^3$ . Soit  $\Omega$ , une forme dans  $\mathbf{R}^3$  et  $\partial\Omega$  son enveloppe, on peut définir la distance signée comme :

$$\text{SDF}(\mathbf{x}) = \begin{cases} -\text{dist}(\mathbf{x}, \partial\Omega) & \text{si } \mathbf{x} \in \Omega \\ +\text{dist}(\mathbf{x}, \partial\Omega) & \text{sinon} \end{cases}$$

Où  $\text{dist}(\mathbf{x}, \partial\Omega)$  est la distance Euclidienne entre  $\mathbf{x}$  et  $\partial\Omega$ . On peut alors définir la forme comme l'isosurface  $\text{SDF}(\mathbf{x}) = 0$ .

La distance signée est également une représentation très adaptée à l'entraînement d'un réseau de neurones. En effet, celle-ci est facilement différentiable, ce qui rend la propagation du gradient (voir Chapitre 4) aisée, permettant un entraînement de qualité. Enfin, la distance signée est adaptée à la génération de formes fermées ; en effet, il est possible pour un réseau de neurones d'apprendre la fonction continue  $f : \mathbf{x} \rightarrow \text{SDF}(\mathbf{x})$  à partir de sa forme discrète.

**Nous utiliserons donc la représentation en distance signée pour le reste de ce travail.**

## 2.1 Représentaion en distance signée

### 2.1.1 Normalisation et centrage

Avant de décrire les formes de chaque échantillon de la base de données sous forme de SDF, il est important de normaliser et centrer les maillages. En effet, la normalisation, qui consiste à réduire  $\mathbf{R}^3$  à  $[-1, 1]^3$  permet de contenir toute l'information selon la même échelle pour tous les échantillons de la base de données. Le centrage permet de corrélérer la SDF à la forme de la voiture uniquement et non plus à sa position dans l'espace. Ces deux procédés sont essentiels pour le bon apprentissage de notre modèle, sans cela, le modèle apprend des artefacts de position et d'échelle au lieu de la géométrie intrinsèque.

Le centrage est réalisé en décalant la position de chaque sommet (vertex) du maillage par le vecteur  $\mathbf{OC}$  où C est le centre du maillage, déterminé comme la moyenne des positions des sommets composant le maillage. On applique donc :

$$\mathbf{x}_{center} = \mathbf{x} - \mathbf{OC}$$

Dans un second temps, on peut procéder à la normalisation : on divise les coordonnées de chaque point par l'échelle définie comme :

$$\text{scale} = \max(|\mathbf{X}|)$$

où  $\mathbf{X}$  contient tous les vecteurs position de chaque sommet après centrage. On a donc :

$$\mathbf{x}_{normed} = \frac{\mathbf{x}}{\text{scale}}$$

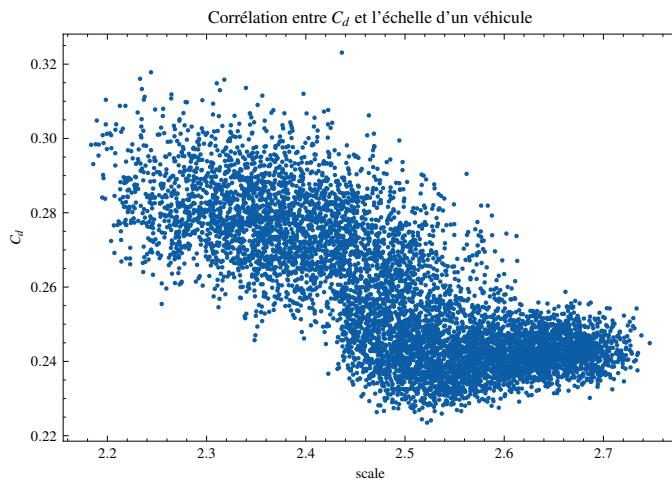
Cette **normalisation isotrope** des données, consistant à réduire  $\mathbf{R}^3$  à  $[-1, 1]^3$  pour chaque échantillon, permet donc un meilleur apprentissage qu'en utilisant les données brutes et enlève la notion d'**échelle** au modèle. C'est-à-dire que deux échantillons différents en échelle mais similaires en géométrie seront vus comme exactement similaires par le modèle. Le modèle est dit '**scale-invariant**', c'est-à-dire :

$$f(\mathbf{x}) = f(s.\mathbf{x}) \quad \forall s > 0$$

On peut multiplier l'entrée par une constante positive sans changer la prédiction.

Dans le cadre de ce travail, l'objectif étant de générer des formes aérodynamiques optimisées à partir de la **géométrie** seule. L'échelle absolue d'un véhicule n'étant ni un **critère de performance** (le coefficient de traînée est en effet non corrélé à l'échelle pour de grands Reynolds ( $R_e \approx 10^6$ )[13, 43, 54] ce qui est le cas des simulations CFD constituant la base de données DrivAerNet++) ni un **facteur de conception visé**, il est essentiel que le modèle n'apprenne que des relations géométriques. La normalisation isotrope utilisée permet de supprimer toute information liée à la taille absolue sans altérer la forme relative, de forcer l'apprentissage à se concentrer sur la géométrie relative des échantillons et de stabiliser l'entraînement.

Enfin, on remarque que l'échelle des véhicules présents dans la base de données possède une **corrélation de Pearson** significative avec le coefficient de traînée  $r = -0.73$ . Cette corrélation n'est pas d'origine **causale** puisque le coefficient de traînée ne dépend pas de l'échelle dans les conditions de simulation effectuées pour constituer la base de données [13, 43, 54]. Elle reflète la distribution statistique de la base où les véhicules de type 'Fastback' sont sur-représentés et possèdent un coefficient de traînée plus faible que les autres types de carrosserie, ainsi qu'une échelle absolue plus importante. La normalisation permet d'éviter que le modèle n'exploite l'association non physique entre échelle absolue et le coefficient de traînée.



**FIGURE 3.6 – Corrélation de Pearson entre  $C_d$  et l'échelle des échantillons**

La corrélation de Pearson  $r$  entre deux séries de données  $\mathbf{Y} = \{C_d^1, C_d^2, \dots, C_d^N\}$  et  $\mathbf{X} = \{\text{scale}_1, \text{scale}_2, \dots, \text{scale}_N\}$  est définie comme suit :

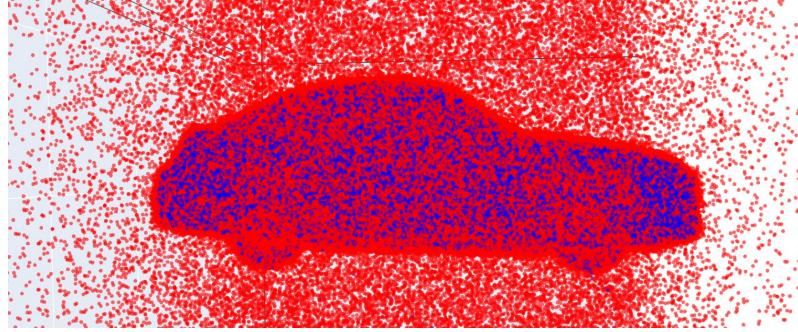
$$r = \frac{\text{cov}(\mathbf{X}, \mathbf{Y})}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

où  $\bar{x}$  (respectivement  $\bar{y}$ ) est la moyenne de  $\mathbf{X}$  (respectivement  $\mathbf{Y}$ ).  $r \in [-1, 1]$  mesure la corrélation linéaire entre la série de données  $\mathbf{Y}$  et la série  $\mathbf{X}$ . Si  $r \approx 0$ , il n'y a pas de corrélation, si  $r = 1$ , il y a corrélation parfaite et si  $r = -1$ , il y a corrélation linéaire inverse.

### 2.1.2 Distance signée

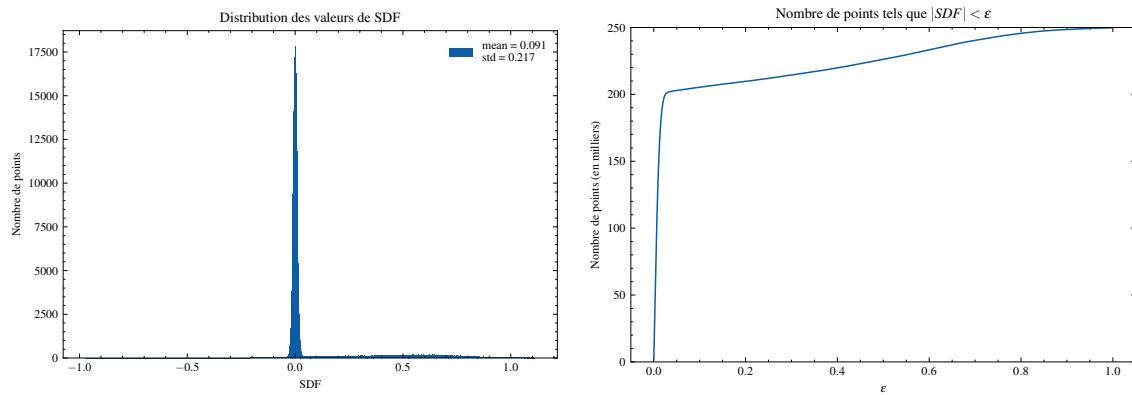
Afin de représenter au mieux une forme grâce à sa représentation en SDF, il est important d'échantillonner l'espace  $[-1, 1]^3$  avec un nombre important de points. Néanmoins, il est également bon de prendre en compte que **plus une forme est riche en points, plus l'entraînement du modèle est long à réaliser et plus le passage du maillage à la SDF est coûteux**. Afin de maximiser l'information tout en limitant le coût, une pratique courante consiste à échantillonner un **grand nombre de points proches de la surface**, permettant

de capter les **détails fins** de la géométrie, ainsi qu'une **faible quantité de points répartis aléatoirement dans le cube**  $[-1, 1]^3$ , permettant d'obtenir une '**vue générale**' de la forme.



**FIGURE 3.7 –** 250k points d'échantillonage. En bleu, les points proches de la surface ( $|SDF| < 0.01$ ), en rouge le reste.

Dans notre cas, chaque échantillon de la base de données sera constitué de **250k points** avec **80% proches de la surface** et les **20% restant placés uniformement dans le cube**. On définit  $X_{\partial\Omega}$  l'ensemble des points proches de la surface, on :  $X_{\partial\Omega} \sim \mathcal{N}(0, 0.01^2)$ , c'est-à-dire que les points de  $X_{\partial\Omega}$  sont placés à moyenne nulle de  $\partial\Omega$  et avec un écart type  $\sigma = 0.01$ . La SDF de chaque point par rapport au maillage est calculée à l'aide de la librairie `Open3d` [53] qui permet de réaliser du calcul optimisé en temps.



**FIGURE 3.8 –** Distribution des valeurs de SDF dans un échantillon de 250k points

**FIGURE 3.9 –** Nombre de points tels que  $|SDF| < \epsilon$  en fonction de  $\epsilon$

## 2.2 Décomposition de la base de données

Afin d'entraîner le réseau et quantifier ses performances, une pratique courante consiste à partager la base de données en 3 différents sets :

- Le set d'entraînement ('trainset'), ce sont l'ensemble des données qui vont être utilisées pour entraîner le modèle. En général, 80 % des données de la base sont utilisées comme données d'entraînement.

- Le set de validation : il s'agit de l'ensemble des données qui permettent de quantifier l'évolution de l'apprentissage du modèle. Les données de validation ne servent pas à l'apprentissage, mais sont utilisées en même temps que les données d'entraînement afin de mesurer la capacité de généralisation du modèle (voir si le modèle parvient à produire des prédictions satisfaisantes pour des données qui ne lui ont pas servi à s'entraîner). Le set de validation est très utile pour détecter le surapprentissage ('overfitting'), c'est-à-dire les cas où le modèle apprend par cœur les données d'entraînement, mais devient incapable de faire de la prédiction sur de nouvelles données. En général, 10 % des données de la base sont allouées au set de validation

- Le set de test. Les données de test sont isolées durant tout l'entraînement du modèle et ne servent que lors de l'inférence. Ces données permettent de quantifier les performances du modèle une fois son entraînement terminé. Les 10 % restants de la base de données sont utilisés pour constituer le set de test.

L'objectif étant de faire des prédictions de la meilleure qualité sur les données de test, il est évident que les sets d'entraînement et de validation doivent posséder une distribution statistique des données semblable à celle du set de test. Enfin, les différents sets ne doivent en aucun cas contenir des informations partagées : aucune donnée appartenant à un set ne doit être présente dans un autre (contamination), aucune information sur la distribution statistique d'un set ne doit être partagée à un autre set ('data-leakage'). Afin de respecter les deux premiers points, nous créons les différents sets en prenant aléatoirement des échantillons parmi l'ensemble des données de la base sans remise. L'aspect aléatoire permet de garantir des distributions statistiques cohérentes entre les trois ensembles. Enfin, afin d'éviter le partage d'information entre différents groupes, la normalisation du coefficient de traînée se fait en utilisant les valeurs maximales et minimales présentes dans le set d'entraînement uniquement. En effet, normaliser le set d'entraînement avec le minimum et le maximum global de la base de données présenterait un grand risque que le train-set possède des informations sur le set de test qu'il n'est donc pas censé avoir.

### 2.3 Coût computationnel

Dans cette partie, nous donnons un bref aperçu du coût de calcul et de traitement des données permettant l'entraînement de notre modèle.

	Type de ressource	Processeur	temps de calcul	mémoire
Calcul de $C_d$	CPU	Intel Xeon 12 GB RAM	4,2 CPU-hours	200 Ko
Calcul SDF	CPU	Intel Xeon 50 GB RAM	9,7 CPU-hours	28 Go

**TABLE 3.2** – Outils et ressources utilisées pour le calcul et traitement des données de la base de données. Les temps sont donnés en CPU-hours totales pour l'ensemble des 8000 échantillons.

# Chapitre 4

## De la conception à l'optimisation du modèle d'apprentissage

*Ce chapitre décrit en détail la structure du modèle d'apprentissage profond utilisé, définit les notions clés liées à l'apprentissage, l'ensemble des hyperparamètres qui interviennent dans l'utilisation du modèle, ainsi qu'une méthode permettant la recherche des meilleurs hyperparamètres.*

### Sommaire

---

<b>1</b>	<b>Structure du modèle</b>	<b>28</b>
1.1	Encodeur	28
1.2	Décodeurs	30
<b>2</b>	<b>Entraînement</b>	<b>31</b>
2.1	Fonction coût	32
<b>3</b>	<b>Espace des hypothèses et espace des hyperparamètres</b>	<b>33</b>
3.1	Espace des hypothèses	33
3.2	Espace des hyperparamètres	33
3.3	Choix d'une hypothèse et des hyperparamètres	34
3.4	Comparaison des résultats	39
<b>4</b>	<b>Coût computationnel</b>	<b>43</b>

---

# 1 Structure du modèle

Une fois la base de données créée et structurée, il est possible de concevoir le modèle d'apprentissage profond permettant de générer des formes de voitures optimisées selon le coefficient de traînée. L'objectif est ici de modéliser le lien entre géométrie 3D et coefficient de traînée à l'aide d'un réseau de neurones conçu spécifiquement pour exploiter les données de Signed Distance Function (SDF).

Le modèle retenu repose sur une architecture d'autoencodeur, c'est-à-dire un réseau composé de deux blocs : un encodeur (**E**) et un décodeur (**D**).

## 1.1 Encodeur

### Encodeur

Pour une forme  $\Omega$  donnée, on prépare un ensemble de paires  $\mathbf{X}_{i \in [1, N]}$  ( $N = 250\,000$  puisque 250 000 points d'échantillonnage par forme) tel que  $\mathbf{X}_i = \{\mathbf{x}_i, SDF(\mathbf{x}_i)\}$ . L'encodeur agit comme la fonction :

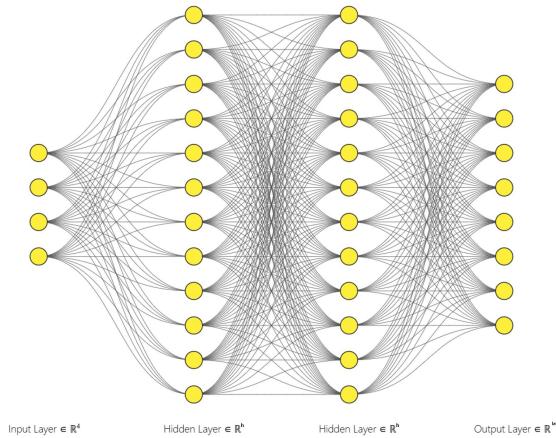
$$f_\theta(\mathbf{X}_i) = \mathbf{z}_i, \mathbf{z}_i \in \mathbf{R}^{n_{\text{lat}}}$$

Ainsi, la fonction  $f_\theta$  qui décrit le comportement de l'encodeur permet d'associer un vecteur latent  $\mathbf{z}_i$  de  $\mathbf{R}^{n_{\text{lat}}}$  à chaque couple  $\mathbf{X}_i = \{\mathbf{x}_i, SDF(\mathbf{x}_i)\}$ . On peut alors définir un vecteur latent encodant la forme entière en moyennant les  $\mathbf{z}_i$  :

$$\mathbf{z} = \frac{1}{N} \sum_{i=1}^N \mathbf{z}_i$$

D'autres choix permettent d'obtenir un vecteur global  $\mathbf{z}$  à partir de  $\mathbf{z}_i$ ,  $i \in [1, N]$ , le max pooling par exemple applique la transformation :  $z_j = \max_{i \in [1, N]} (z_{i,j})$ , chaque composante du vecteur  $\mathbf{z}$  final étant ainsi la composante maximale parmi les  $z_i$ . Le choix d'un vecteur latent moyen permet de contenir toute l'information d'une forme, ce que le max pooling ne pourrait pas faire.

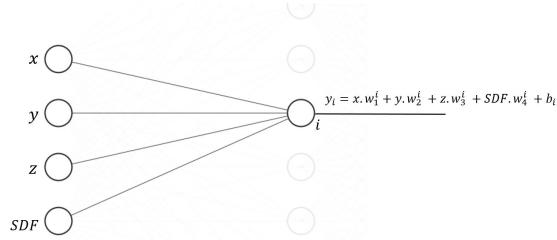
Ainsi, l'encodeur permet, à partir de la représentation sous forme de SDF d'une forme  $\Omega$ , d'encoder sa représentation sous forme d'un vecteur latent  $\mathbf{z}$  dans  $\mathbf{R}^{n_{\text{lat}}}$ . L'ensemble des vecteurs  $\mathbf{z}$  produits par l'encodeur (**E**) à partir d'un ensemble de données  $\mathcal{S}$  forme l'espace latent noté  $\mathcal{Z}$ . Plus le vecteur  $\mathbf{z}$  possède une dimension  $n_{\text{lat}}$  élevée, plus il contient d'informations sur la forme encodée.  $n_{\text{lat}}$  est donc un hyperparamètre à régler correctement pour permettre une information riche sans être trop lourde pour autant, ce qui ralentirait l'entraînement. Afin d'obtenir un vecteur latent à partir de la représentation implicite d'une forme en SDF, nous utilisons un réseau de neurones complètement connecté (Fully connected Neural Network) de poids  $\theta$  à entraîner sur le train-set.

**FIGURE 4.1 – Structure de l'encodeur**

Le réseau étant entièrement connecté, chaque neurone d'une couche  $i$  est connecté à chaque neurone de la couche  $i+1$ . La profondeur du réseau correspond au nombre de couches constituant le réseau (3 couches sur la figure 4.1). La profondeur  $n_{\text{layers}}$  est un hyperparamètre à régler. Le nombre de neurones des couches intermédiaires (hidden layers) (12 pour la figure 4.1) est également un hyperparamètre à régler  $d_{\text{hidden}}$ . Plus le réseau possède des couches comportant beaucoup de neurones, plus il est à même de générer une fonction  $f_\theta$  complexe. Enfin, la dernière couche permet de générer le vecteur latent, elle comporte donc  $n_{\text{lat}}$  neurones.

Chaque neurone du réseau possède ses propres poids ( $w$ ) ainsi qu'un biais ( $b$ ), qui réunissent définissent les paramètres du modèle. Pour obtenir une fonction  $f_\theta$  qui encode correctement l'information, il convient de trouver les meilleurs paramètres lors de l'entraînement (voir section Entraînement). Les poids et le biais d'un neurone interviennent dans la fonction de transfert de ce neurone. Soit un neurone  $i$  de la première couche auquel on donne en entrée un couple  $\{\mathbf{x}_j, SDF(\mathbf{x}_j)\}$  :

$$y_i = w_1^i x_j + w_2^i y_j + w_3^i z_j + w_4^i SDF_j + b^i = \mathbf{W}^i \mathbf{x}^i + b^i$$

**FIGURE 4.2 – Fonction de transfert d'un neurone d'une couche fully-connected**

Néanmoins, cette description du réseau ne permet d'avoir que des sorties linéaires (comme une succession de combinaisons linéaires) ce qui ne permettrait pas de traiter

les problèmes non linéaires. Ainsi, on ajoute une fonction d'activation  $\phi$  à la sortie de chaque neurone des couches intermédiaires afin d'ajouter de la non linéarité, de sorte que :

$$y_i = \phi(\mathbf{W}^T \mathbf{X} + b)$$

De nombreuses fonctions d'activation permettent de créer de la non linéarité. Dans notre cas, nous utilisons la fonction ReLU (Rectified Linear Unit) qui est une fonction d'activation standard pour les tâches de régression :

$$\text{ReLU}(x) = \max(0, x)$$

Cette fonction est nulle pour les valeurs négatives et linéaire pour les valeurs positives. Elle introduit une non-linéarité tout en conservant une grande simplicité computationnelle.

## 1.2 Décodeurs

Étant donné l'approche multi-objectifs du modèle (prédiction d'un coefficient de traînée et régression d'une distance signée), deux décodeurs sont utilisés.

### 1.2.1 Décodeur géométrique

#### Décodeur géométrique

Le premier décodeur, noté  $\mathbf{D}_{SDF}$  est le décodeur géométrique permettant la reconstruction d'une forme 3D à partir de son encodage sous forme de vecteur latent. Soit  $\mathbf{z}$  le vecteur latent généré par l'encodeur associé à une forme  $\Omega$ , le décodeur géométrique agit comme la fonction :

$$g_{\theta}^{SDF}(\mathbf{x}, \mathbf{z}) \approx SDF_{\Omega}(\mathbf{x}) \quad \forall \mathbf{x} \in [-1, 1]^3$$

Le décodeur géométrique est un réseau de neurones 'fully-connected' de la même manière que l'encodeur. Si bien entraîné, le décodeur géométrique associé à l'encodeur permet la reconstruction fidèle d'une forme en faisant de la prédiction sur la SDF d'un point de  $[-1, 1]^3$  pour la forme  $\Omega$  encodée dans le vecteur latent  $\mathbf{z}$ . Cela permet de créer de nouvelles formes en interpolant dans l'espace latent, c'est-à-dire que de nouvelles formes réalistes peuvent être générées à partir de nouveaux vecteurs latents encore jamais vus lors de l'entraînement du modèle. Le décodeur géométrique prend en entrée un point  $\mathbf{x} \in [-1, 1]^3$  ainsi que le vecteur latent associé à la forme encodée (vecteur de dimension  $n_{\text{lat}} + 3$ ) et produit la distance signée correspondant à ce point et ce vecteur latent.

De la même manière que pour l'encodeur, la fonction d'activation utilisée est ReLU. La fonction d'activation n'est en revanche pas appliquée aux sorties de la dernière couche, sans quoi les distances signées qui devraient être négatives (points à l'intérieur de  $\Omega$ ) seraient nulles, faussant la reconstruction.

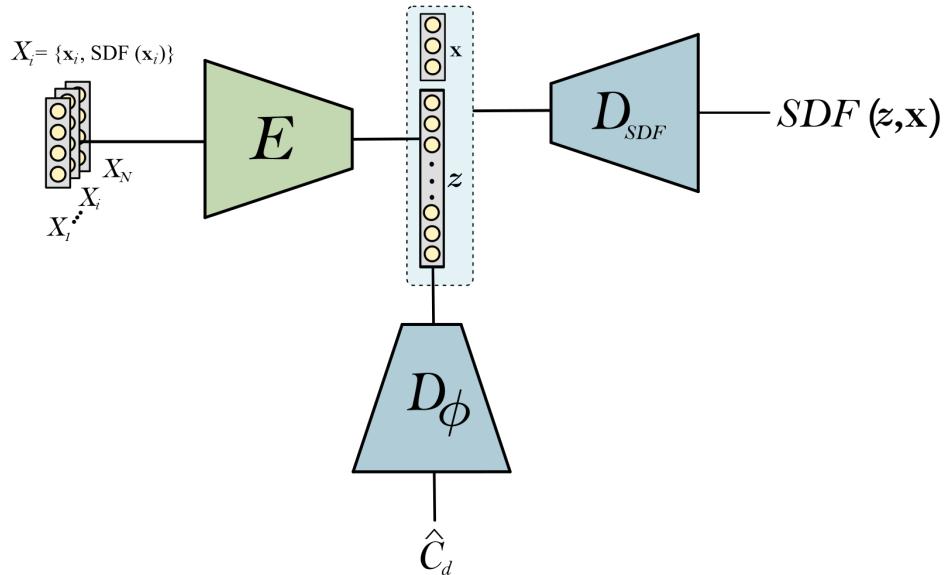
### 1.2.2 Décodeur Physique

#### Décodeur physique

En plus du décodeur géométrique, notre modèle doit être capable de prédire le coefficient de traînée d'un véhicule. Pour cela, nous utilisons également un réseau entièrement connecté avec activation ReLU. Soit  $\Omega$  est formé encodée par le vecteur latent  $z$ . Le décodeur physique ( $D_\phi$ ) agit donc comme la fonction :

$$g_\theta^\phi(z) = \hat{C}_d(\Omega) \approx C_d(\Omega)$$

L'association de l'encodeur et des deux décodeurs permet à la fois d'encoder la physique (coefficient de traînée) et la géométrie d'une forme dans un même vecteur latent. Lors de l'entraînement du modèle, l'espace latent est optimisé avec les poids de l'encodeur afin de représenter au mieux ces informations, permettant aux décodeurs de faire de bonnes prédictions sur la SDF et le coefficient de traînée.



**FIGURE 4.3 – Modèle de l'autoencodeur à deux décodeurs (géométrique et physique)**

## 2 Entraînement

Une fois le modèle défini ainsi que les données prêtées, il est possible de passer à la phase d'entraînement du modèle. L'entraînement consiste à déterminer les meilleurs paramètres  $\theta$  pour la fonction  $f$  du modèle selon un critère de performance choisi.

Dans le cadre de problèmes supervisés, on définit le **risque** comme l'**espérance** d'une

fonction de perte choisie  $l$  sur l'ensemble des données associées au problème à traiter  $\mathcal{D}$  (dans notre cas : tous les modèles de véhicules théoriquement existants).

$$\mathcal{R}(f) = \mathbb{E}_{\mathcal{D}}[l(f(\mathbf{x}), y)]$$

Bien sûr, la distribution  $\mathcal{D}$  est inconnue du modèle puisque l'ensemble des données d'entraînement  $\mathcal{S}$  ne peut que constituer un sous-ensemble de  $\mathcal{D}$ . On définit alors le **risque empirique** comme :

$$\hat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^n l(f(\mathbf{x}_i), y_i)$$

On définit ainsi l'objectif de l'entraînement comme :

$$\text{trouver } \theta^* = \arg \min_{\theta} \hat{\mathcal{R}}(f_{\theta})$$

Où  $f$  est une fonction de l'espace des hypothèses  $\mathcal{F}$  choisie et fixée en amont de l'entraînement.

Afin de déterminer le meilleur jeu de paramètres, on utilise une descente de gradient sur la fonction coût (identifiée ici au risque empirique)  $\mathcal{L}(f_{\theta}) = \hat{\mathcal{R}}(f_{\theta})$  permettant d'améliorer, par itérations, les poids du modèle à partir d'une configuration initiale par :

$$\theta := \theta - \eta \frac{\partial \mathcal{L}(f_{\theta})}{\partial \theta}$$

où  $\eta$  est le **taux d'apprentissage**, hyperparamètre à régler afin d'assurer un bon apprentissage. On effectue la descente de gradient **epochs** fois. Plus le nombre d'epochs est important, plus la descente de gradient est susceptible de fournir un bon jeu de poids.

Dans notre cas, nous utilisons l'optimizer `Adam`, variante de la descente de gradient, intégrant le momentum du gradient afin d'optimiser la convergence. Adam est très couramment utilisé pour les tâches de régression. Nous utilisons également le scheduler `ReduceLROnPlateau` (Voir Annexe B) de la librairie `Pytorch`, permettant de réduire d'un facteur  $\gamma$  le taux d'apprentissage si la Loss sur le set de validation n'a pas été améliorée au cours des  $\tau$  dernières epochs (patience). Dans notre cas, nous utilisons 250 epochs pour chaque entraînement, un taux d'apprentissage initialement fixé à  $10^{-3}$  avec un  $\gamma = 0.1$  et une patience de 10 epochs.

## 2.1 Fonction coût

Etant donné l'aspect multi-objectif du problème, il est nécessaire de définir une fonction coût permettant de traiter les deux objectifs à la fois. Pour cela, on définit une fonction coût dédiée à la reconstruction 3D, notée  $\mathcal{L}_{SDF}$ , une fonction coût de régularisation notée  $\mathcal{L}_{Eikonal}$  ainsi que la fonction coût permettant la prédiction du coefficient de traînée notée  $\mathcal{L}_{\phi}$ .

Deux fonctions coût  $\mathcal{L}_{SDF}$  ont été testées : la fonction coût  $\mathcal{L}_2$ , associée à la perte élémentaire  $l_2 = (\hat{SDF} - SDF)^2$  ainsi que la fonction coût  $\mathcal{L}_1^{\delta}$  associée à la perte élémentaire

$l_1^\delta = |\text{clamp}(\hat{SDF}, \delta) - \text{clamp}(SDF, \delta)|$ . La loss Clamped sature les grandes SDF permettant la concentration du réseau sur la reconstruction des points proches de l'isosurface 0.

La fonction de coût utilisée pour la prédiction du coefficient de traînée est la loss  $\mathcal{L}_2$  associée à la perte  $l_2 = (\hat{C}_d - C_d)^2$

La loss Eikonal est une fonction de régularisation. Elle ne se base pas sur la comparaison entre des prédictions et des valeurs vraies, mais impose une contrainte géométrique sur la distance signée :

$$l_{\text{Eikonal}} = (\|\nabla SDF\| - 1)^2$$

En effet, la SDF, de par sa définition, doit posséder une norme de son gradient égale à 1 (voir Annexe B). Injecter le résidu de cette propriété permet d'imposer cette contrainte sur les prédictions de SDF générées par le modèle, améliorant la structure globale des formes pouvant être générées.

La fonction de coût totale utilisée pour l'entraînement du modèle est définie comme :

$$\mathcal{L} = \mathcal{L}_{SDF} + \lambda \mathcal{L}_\phi + \lambda_{\text{reg}} \mathcal{L}_{\text{Eikonal}}$$

Les hyperparamètres  $\lambda$  et  $\lambda_{\text{reg}}$  permettent de pondérer l'entraînement sur les différentes tâches. La prédiction du coefficient de traînée étant par exemple une tâche plus simple que la reconstruction SDF, un  $\lambda$  faible permet de donner plus d'importance à la SDF et donc d'améliorer sa reconstruction.

### 3 Espace des hypothèses et espace des hyperparamètres

#### 3.1 Espace des hypothèses

Le choix d'une fonction  $f_\theta$  de l'espace des hypothèses  $\mathcal{F}$  est une démarche qu'il est primordial d'adopter afin de maximiser les performances d'un modèle. En effet, même un entraînement bien calibré et performant réalisé sur une fonction mal choisie donnera des résultats limités, voire médiocres.

Une fois le type de modèle fixé (par exemple, un réseau de neurones donné), l'espace des hypothèses  $\mathcal{F}$  correspond à l'ensemble des fonctions  $f$  que ce modèle est capable de représenter, selon sa structure (profondeur, nombre de neurones, activations, etc.).

#### 3.2 Espace des hyperparamètres

L'espace des hyperparamètres correspond à l'ensemble des hyperparamètres intervenant dans la résolution d'un problème d'apprentissage automatique. De même que pour les hypothèses, choisir un jeu d'hyperparamètres  $h \in \mathcal{H}$  est une tâche à la fois complexe et primordiale afin d'obtenir :

- une fonction  $f$  adéquate au problème traité, ni trop simple ni trop complexe, capable de traiter efficacement des données jamais vues lors de l'entraînement.

- Un entraînement de qualité, stable et convergent ainsi qu'une vitesse d'entraînement raisonnable.

On définit ici la liste des hyperparamètres intervenant dans le problème. Les valeurs grisesées correspondent aux hyperparamètres structurels ayant un impact direct sur l'espace des hypothèses  $\mathcal{F}$  :

Nom	Symbol	Valeur utilisée
Taux d'apprentissage	$\eta$	$1 \times 10^{-3}$
Taille de batch	$B$	-
Nombre d'epochs	$n_{\text{epochs}}$	250
Facteur de réduction (scheduler)	$\gamma$	0.1
Patience (scheduler)	$\tau$	10
Dimension du vecteur latent	$n_{\text{lat}}$	-
Nombre de couches cachées Encodeur	$n_{\text{layers}}^E$	2
Nombre de couches cachées Décodeur SDF	$n_{\text{layers}}^{SDF}$	6
Nombre de couches cachées Décodeur Physique	$n_{\text{layers}}^\phi$	1 - 3
Dimension des couches cachées	$d_{\text{hidden}}$	-
Fonction d'activation	$\phi$	ReLU
Skip connection	-	True/False
Optimiseur	-	Adam
Loss	$\mathcal{L}$	-
Poids de $C_d$	$\lambda$	-
Poids de Eikonal	$\lambda_{\text{reg}}$	-
Paramètre de la fonction Clamped	$\delta$	-
Dropout Rate	$p$	-

TABLE 4.1 – Hyperparamètres utilisés

### 3.3 Choix d'une hypothèse et des hyperparamètres

Différentes méthodes permettent de parcourir les espaces des hypothèses et des hyperparamètres. Une première méthode consiste à parcourir un ensemble de valeurs pour chaque hyperparamètre et à entraîner le modèle avec chaque combinaison, cette méthode est donc une recherche par grille (*grid search*). Elle permet d'obtenir la solution optimale parmi les différentes combinaisons d'hyperparamètres testées, mais est très coûteuse en ressources. Une deuxième méthode consiste à faire un nombre N de combinaisons en choisissant aléatoirement des valeurs d'hyperparamètres parmi les ensembles de valeurs de chacun. Cette méthode, connue sous le nom de recherche randomisée (*random search*), permet de limiter le coût computationnel, mais ne garantit pas d'atteindre l'optimum global. Ces méthodes sont généralement employées afin de trouver l'ensemble des hyperparamètres.

Dans notre cas, le choix a été fait de séparer l'objectif de recherche des meilleurs hyperparamètres en deux sous-objectifs :

- (1) Trouver les meilleurs hyperparamètres structurels (grisés dans la Table 4.1) en fixant les hyperparamètres non structurels, c'est-à-dire ceux qui n'influencent pas directement la capacité du modèle à approximer des fonctions (par exemple,  $\lambda, \lambda_{\text{reg}}, \delta$ ). Cette étape permet d'obtenir une hypothèse performante.
- (2) Trouver les meilleurs hyperparamètres non structurels associés au meilleur modèle trouvé à l'étape (1).

Cette décomposition en sous-objectifs clairs permet de déterminer des hyperparamètres performants efficacement, tout en gardant un contrôle sur la structure et donc la complexité de notre modèle. Elle permet aussi d'analyser plus finement les performances des modèles en distinguant clairement les contributions structurelles et non structurelles.

### 3.3.1 Recherche dans l'espace des hypothèses

Dans cette première partie, on cherche à construire la meilleure hypothèse étant donnée des hyperparamètres non structurels fixés. Afin de comparer les performances de nos modèles, il est nécessaire de définir des métriques permettant de quantifier les performances de chaque modèle (Par métrique, nous entendons ici toute fonction de mesure utilisée pour quantifier la performance ou la similarité, indépendamment du respect strict des axiomes mathématiques d'une métrique). Étant donné le caractère bi-objectif de notre problème, deux métriques ont été mises en place :

- La Distance de Chamfer (CD) permet de mesurer la similarité entre deux ensembles finis de points. Nous l'utilisons afin de comparer les reconstructions SDF avec la vérité terrain. Pour faire cela, nous extrayons un maillage à partir de l'isosurface  $SDF = 0$  pour la prédiction, puis nous échantillonons 30 000 points sur ce maillage et le maillage de vérité et mesurons la distance de Chamfer sur ces deux distributions. Soit deux ensembles de points  $P = \{p_1, p_2, \dots, p_n\}$  et  $Q = \{q_1, q_2, \dots, q_n\}$  tous deux dans  $\mathbb{R}^n$ , la distance de Chamfer est définie comme :

$$\text{CD}(P, Q) = \frac{1}{|P|} \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2 + \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \|q - p\|_2^2$$

Ainsi, l'objectif est de choisir le modèle qui minimise la distance de Chamfer.

Afin de quantifier les performances en matière de prédiction du coefficient de traînée de nos modèles et de les comparer avec l'état de l'art, quatre métriques sont couramment utilisées :

- L'erreur quadratique moyenne. Définie comme :

$$MSE = \frac{1}{N} \sum_{i=1}^N (\hat{C}_d - C_d)^2$$

- L'erreur absolue moyenne. Définie comme :

$$MAE = \frac{1}{N} \sum_{i=1}^N |\hat{C}_d - C_d|$$

- L'erreur maximale en valeur absolue. Définie comme :

$$MAX\ AE = \max(|\hat{C}_d - C_d|)$$

- Le score  $R^2$ , permet de mesurer la proportion de la variance de la variable cible : Défini comme :

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Ces métriques sont utilisées comme mesures de performance des prédictions du coefficient de traînée dans le cadre de la prédiction sur DrivAerNet++ [14]. Afin de différencier les performances des différents modèles, on ne mesurera que la MSE sur le jeu de validation pour les différents modèles. (Le reste des métriques sera calculé sur le jeu de test. Voir le Chapitre 5).

Ces métriques sont évaluées sur le jeu de validation (en adéquation avec celui fourni [14]) pour chaque modèle. Afin de garantir une reproductibilité et une interprétabilité de chacun des résultats associés à chaque modèle testé, tous les processus stochastiques ont été initialisés avec une graine identique. Cela concerne l'initialisation des poids ainsi que le processus de tirage des points pour l'application de la loss Eikonal. En toute rigueur, les batchs devraient également être identiques pour chaque entraînement, néanmoins, pour des raisons évidentes de temps de calcul et de mémoire des GPUs utilisés, les modèles les plus légers (moins de paramètres) ont été entraînés avec des batchs plus importants que les modèles les plus lourds. En pratique, pour un dataset d'entraînement large comme le nôtre et un nombre d'epochs important, cela ne crée aucune différence dans la convergence des différents modèles.

Tous les modèles ont été entraînés avec les hyperparamètres suivants fixés :

Nom	Symbole	Valeur utilisée
Poids de $C_d$	$\lambda$	0.05
Poids de Eikonal (si utilisée)	$\lambda_{reg}$	0.1
Paramètre de la fonction Clamped (si utilisée)	$\delta$	0.1
Dropout Rate	$p$	0.05

**TABLE 4.2 –** Hyperparamètres utilisés pour le choix des hypothèses

Ces valeurs d'hyperparamètres ont été choisies à l'aide de tests réalisés en amont :

- $\lambda = 0.05$  permet de donner plus de poids à la reconstruction SDF qu'à la prédiction du coefficient de traînée, qui est une tâche plus simple.

- $\lambda_{reg} = 0.1$  permet d'obtenir une bonne convergence de la norme du gradient de la SDF vers 1, sans compromettre la prédiction du  $C_d$  ni la reconstruction SDF basée sur les données.

-  $\delta = 0.1$  permet de prendre en compte la majorité des points d'échantillonnage de la SDF (voir Figure 3.9) tout en saturant correctement les SDF élevées inutiles à la reconstruction de l'isosurface  $SDF = 0$ .

Enfin, le dropout rate  $p = 0.05$ , correspond à la probabilité qu'un neurone donné d'une couche soit désactivé (mis à zéro) lors de chaque forward pass, de manière indépendante pour chaque neurone. Cela permet de réguler l'apprentissage en empêchant le surapprentissage. Le dropout rate a été volontairement choisi faible dans la mesure où un dropout trop important dégrade en moyenne les prédictions (introduction d'un biais). Ce dropout non nul sera utile lors de l'inférence du modèle. Le dropout est appliqué uniquement sur la dernière couche de la tête de sortie physique ( $\mathbf{D}_\phi$ ).

### 3.3.2 Démarche de recherche

On définit ici la démarche de recherche dans l'espace des hypothèses utilisée. 19 tests ont été effectués et comparés sur le jeu de validation.

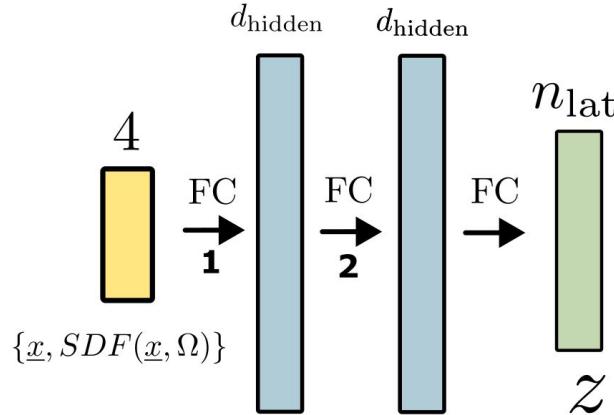
Variante	$n_{\text{layers}}^\phi$	Latent	Hidden	Skip	Eikonal	$\mathcal{L}_{SDF}$
<i>ES1</i>	1	128	256	✓	✓	$\mathcal{L}_1^\delta$
<i>ES2</i>	1	128	512	✓	✓	$\mathcal{L}_1^\delta$
<i>ES3</i>	1	256	512	✓	✓	$\mathcal{L}_1^\delta$
<i>S1</i>	1	128	256	✓	✗	$\mathcal{L}_1^\delta$
<i>S2</i>	1	128	512	✓	✗	$\mathcal{L}_1^\delta$
<i>S3</i>	1	256	512	✓	✗	$\mathcal{L}_1^\delta$
<i>E1</i>	1	128	256	✗	✓	$\mathcal{L}_1^\delta$
<i>E2</i>	1	128	512	✗	✓	$\mathcal{L}_1^\delta$
<i>E3</i>	1	256	512	✗	✓	$\mathcal{L}_1^\delta$
$\emptyset 1$	1	128	256	✗	✗	$\mathcal{L}_1^\delta$
$\emptyset 2$	1	128	512	✗	✗	$\mathcal{L}_1^\delta$
$\emptyset 3$	1	256	512	✗	✗	$\mathcal{L}_1^\delta$
<i>S1'</i>	3	128	256	✓	✗	$\mathcal{L}_1^\delta$
<i>S2'</i>	3	128	512	✓	✗	$\mathcal{L}_1^\delta$
<i>ES1'</i>	3	128	256	✓	✓	$\mathcal{L}_1^\delta$
<i>ES2'</i>	3	128	512	✓	✗	$\mathcal{L}_1^\delta$
$\emptyset 1^{(\mathcal{L}_2)}$	1	128	256	✗	✗	$\mathcal{L}_2$
$\emptyset 2^{(\mathcal{L}_2)}$	1	128	512	✗	✗	$\mathcal{L}_2$
$\emptyset 3^{(\mathcal{L}_2)}$	1	256	512	✗	✗	$\mathcal{L}_2$

**TABLE 4.3 –** Configurations évaluées. Lire : E = Avec Eikonal, S = Avec skip connection, ES = Eikonal + skip connection,  $\emptyset$  = sans Eikonal, sans skip connection.

### 3.3.3 Architectures utilisées

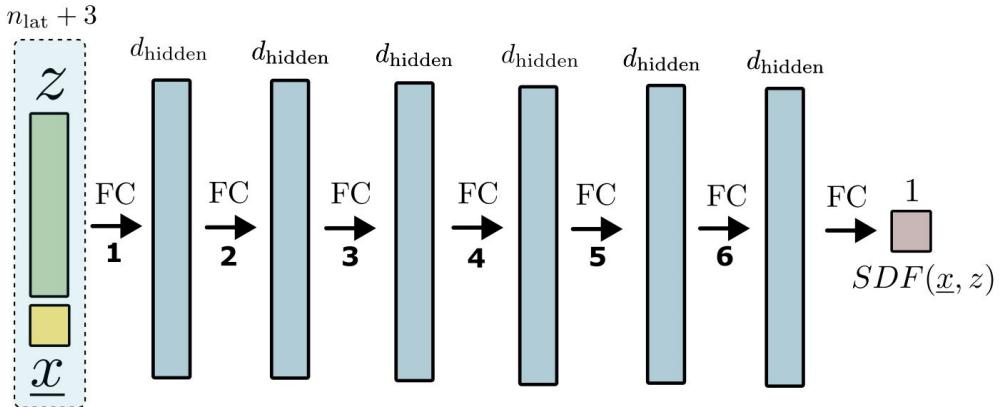
On définit ici les architectures de l'autoencodeur utilisées en détail.

La partie encodeur utilisée pour tous les tests est la même et est définie comme :



**FIGURE 4.4 – Structure de l'encodeur**

L'encodeur ne comporte que trois couches entièrement connectées (FC). La première transforme l'entrée  $\{\underline{x}, \text{SDF}(\underline{x}, \Omega)\}$  en un vecteur caché de dimension  $d_{\text{hidden}}$ . La dernière couche transforme le deuxième vecteur latent en un vecteur de dimension  $n_{\text{lat}}$ .

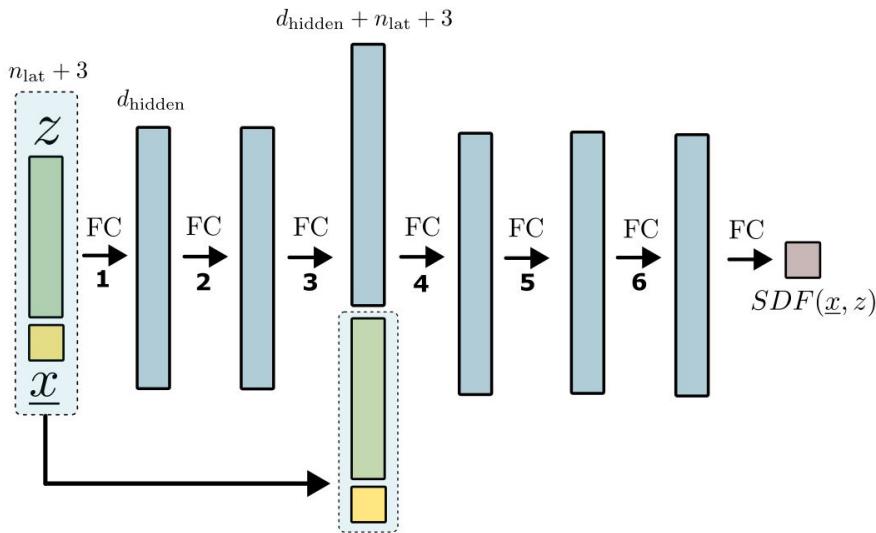


**FIGURE 4.5 – Structure du décodeur SDF sans skip-connection**

Le décodeur SDF ( $\mathbf{D}_{\text{SDF}}$ ) prend en entrée la concaténation du vecteur latent généré par l'encodeur ( $\mathbf{E}$ ) avec un vecteur  $\underline{x}$  correspondant à la coordonnée d'un point de l'espace  $[-1, 1]^3$  (lors de l'entraînement on utilise bien sûr les points  $\underline{x}$  qui ont servi à l'échantillonnage de la SDF c'est-à-dire le vecteur donné en entrée de l'encodeur, lors de l'inférence, on peut prendre tout point de  $[-1, 1]^3$ ). En pratique, on prendra des points répartis sur une

grille uniforme, permettant la reconstruction d'un maillage en utilisant Marching cubes de Trimesh). A l'aide de six couches entièrement connectées (FC), on peut déterminer la distance signée correspondant au point  $x$  et à la forme  $\Omega$  encodée par  $z$ .

On a également défini une version avec une skip connection. La skip connection consiste à réinjecter le vecteur latent ainsi que le vecteur  $x$  utilisés dans une couche intermédiaire du décodeur SDF (ici la troisième couche) en concaténant ces informations avec la sortie de la couche précédente. La skip connection permet de réinjecter de l'information lors de la propagation avant (*forward pass*) ainsi que de limiter le problème de la disparition du gradient (*vanishing gradient*) qui peut intervenir lors de la propagation inverse (*back propagation*) dans les réseaux profonds.



**FIGURE 4.6 – Structure du décodeur SDF avec skip-connection**

Enfin, le décodeur physique ( $D_\phi$ ) qui permet de prédire un coefficient de traînée  $\hat{C}_d$  à partir d'un vecteur latent  $z$ , est défini comme la succession de  $n_{\text{layers}}^\phi$  couches FC (La première prend en entrée le vecteur latent  $z$  tandis que la dernière renvoie la prédiction  $\hat{C}_d$ .

### 3.4 Comparaison des résultats

Etant donné notre modèle bi-objectif, une représentation pertinente des performances des 19 variantes de notre autoencodeur définies comme ci-dessus, consiste à représenter les métriques d'évaluation (calculées sur le jeu de validation) dans le plan, dans notre cas, on trace les résultats dans le plan ( $CD, MSE$ ). Les points oranges correspondent aux solutions non dominées. Soit  $\mathcal{S}$  l'espace des solutions, on dit qu'une solution  $x \in \mathcal{S}$  domine une solution  $y \in \mathcal{S}$  si et seulement si :

$$CD(x) \leq CD(y), MSE(x) \leq MSE(y)$$

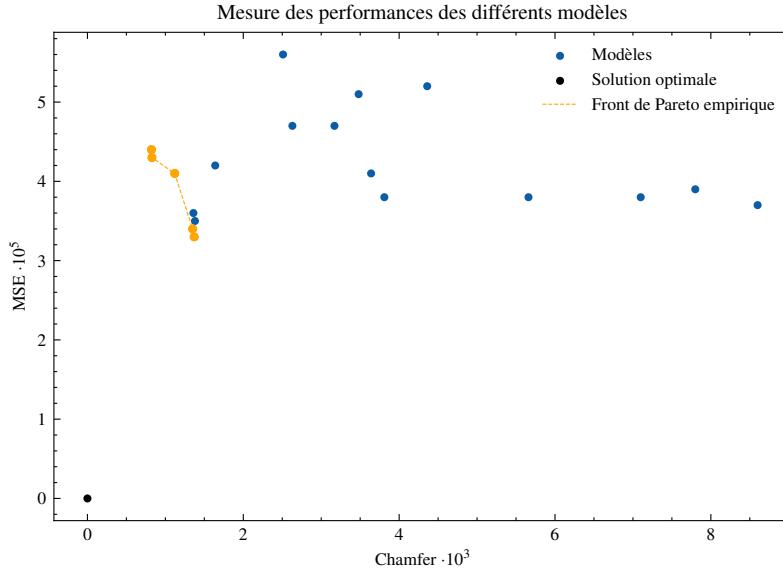
et

$$CD(x) < CD(y) \text{ ou } MSE(x) < MSE(y)$$

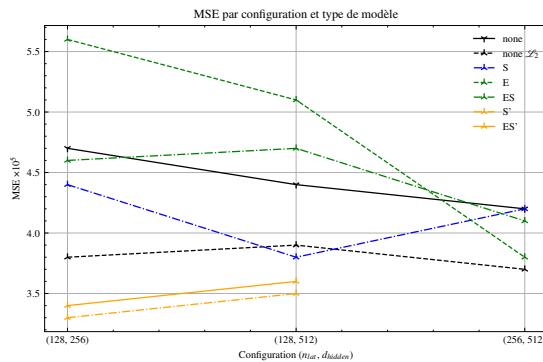
Cela signifie que  $x$  est au moins aussi bon que  $y$  sur tous les objectifs et strictement meilleur que  $y$  sur au moins un objectif. Ainsi, l'ensemble des solutions non dominées correspond à :

$$\mathcal{P} = \{x \in \mathcal{S} \mid \nexists y \in \mathcal{S} : y \text{ domine } x\}$$

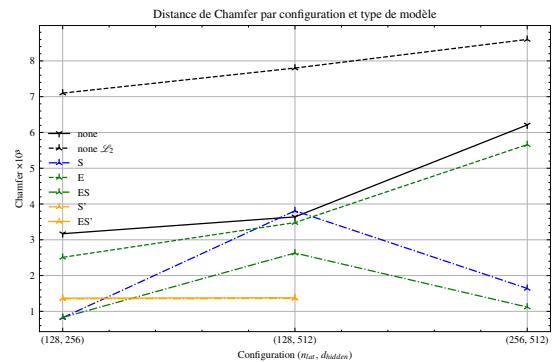
L'ensemble des solutions non dominées forme le front de Pareto.



**FIGURE 4.7** – Représentation des performances de différentes structures de GenNet mesurées sur le set de validation.



**FIGURE 4.8** – Mesure des performances sur la prédiction de  $C_d$  (MSE) selon la configuration du réseau et la fonction de coût utilisée.

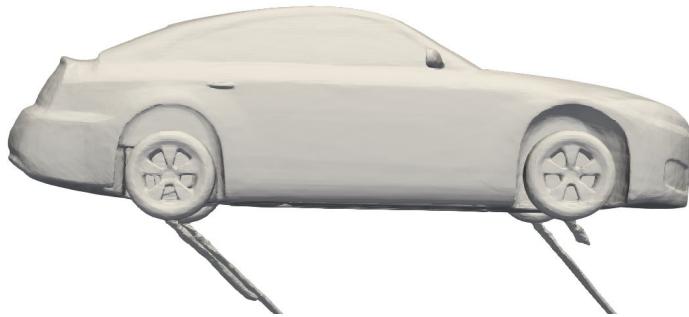


**FIGURE 4.9** – Mesure des performances sur la distance de Chamfer selon la configuration du réseau et la fonction de coût utilisée.

Le front de Pareto apparaît naturellement dans les tâches multi-objectifs, en effet, à partir d'un certain niveau de performances, il n'est plus possible d'améliorer les performances du modèle sur une métrique sans dégrader les autres. Dans notre cas, cinq solutions forment le front de Pareto (empirique).

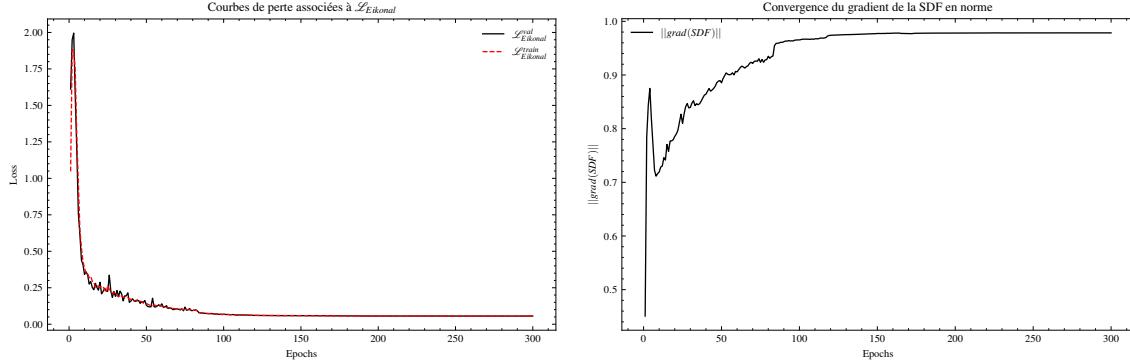
On remarque clairement que le modèle avec skip-connection dans le décodeur SDF ( $D_{SDF}$ ) surperforme les modèles qui n'en sont pas munis sur la distance de Chamfer. On remarque également une grande disparité des résultats sur la distance de Chamfer contre une faible dispersion concernant la MSE, ceci s'explique par la nature intrinsèquement plus simple de la prédiction du coefficient de traînée  $C_d$ , comparée à celle de la reconstruction SDF. On remarque également que les performances avec la structure à trois couches ( $n_{\text{layers}}^\phi = 3$ ) dans le décodeur physique ( $D_\phi$ ) sont meilleures concernant la prédiction de  $C_d$  que les structures à une couche ( $n_{\text{layers}}^\phi = 1$ ). Enfin, la fonction coût  $\mathcal{L}_{SDF} = L_2$  s'est montrée très sous-performante concernant la reconstruction par rapport à la loss  $\mathcal{L}_1^\delta$ .

On remarque également avec les figures ?? qu'une dimension de couche cachée importante  $d_{\text{hidden}}$  ne garantit pas de meilleurs résultats concernant la reconstruction SDF (distance de Chamfer) notamment dans le cas où le vecteur latent est de faible dimension. Cette observation s'observe en pratique, on remarque en effet que les modèles larges ont tendance à surapprendre des zones de l'espace, ayant pour résultat la création d'hallucinations lors de la reconstruction SDF. En pratique, nous avons observé ce phénomène uniquement au niveau des roues. Une analyse plus poussée des maillages du dataset est nécessaire afin de détecter des cas pathologiques pouvant produire ce problème. Ces hallucinations ont bien sûr pour effet de fortement dégrader la distance de Chamfer.



**FIGURE 4.10 –** Hallucinations lors de la reconstruction SDF

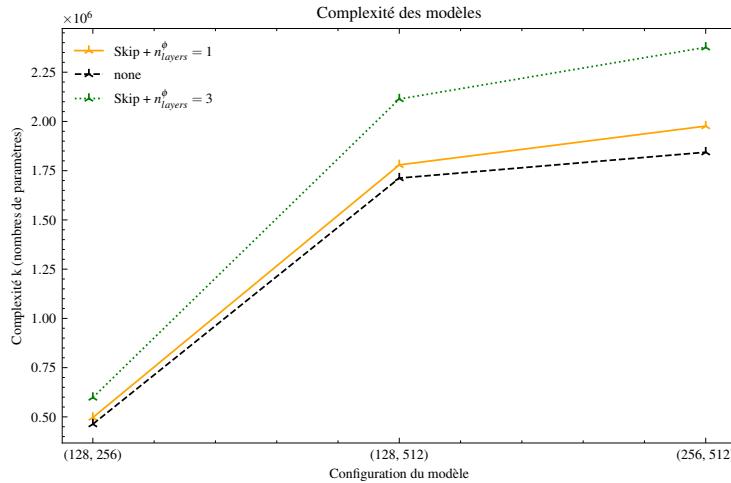
Ces observations nous poussent donc à utiliser des modèles plus simples. On remarque en effet que les modèles les plus performants ont pour configuration : ( $n_{\text{lat}} = 128$ ,  $d_{\text{hidden}} = 256$ ). Par ailleurs, on observe que l'ajout de la loss Eikonal n'a pas d'impact majeur sur la distance de Chamfer de ces modèles malgré la bonne convergence du gradient ( $\|\nabla SDF\| \rightarrow 1$ ), mais a tendance à dégrader la prédiction du coefficient de traînée.



**FIGURE 4.11** – Évolution de la loss Eikonal sur le set de train et de validation au cours des 250 epochs.

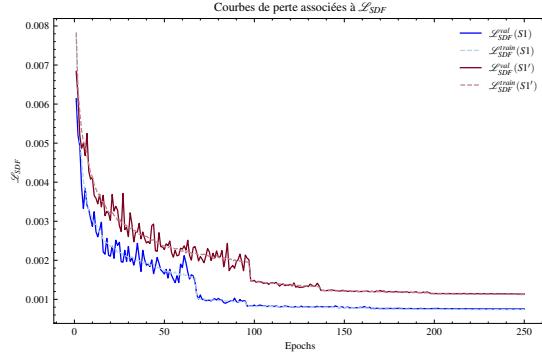
**FIGURE 4.12** – Évolution de  $\|\nabla \text{SDF}\|$  sur le set de validation au cours des 250 epochs.

Ainsi, **nous retenons deux modèles : S1 et S1'**. Le choix a été fait de sélectionner deux modèles différents, les deux seront utilisés différemment lors de l'inférence . Ces choix s'appuient d'une part sur l'observation des performances des différents modèles mais également sur leur complexité. En effet, plus un modèle est large et profond, plus il devient complexe :

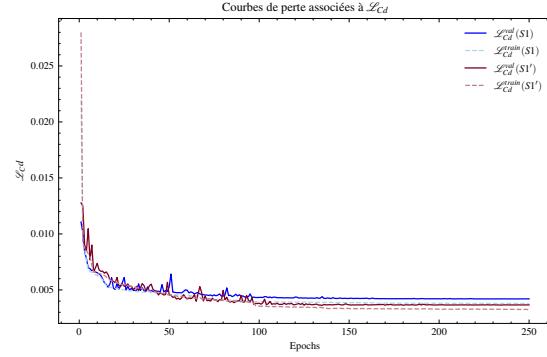


**FIGURE 4.13** – Evolution de la complexité (nombre de paramètres du modèle) en fonction de la structure et de la configuration choisie.

Opter pour une architecture simple présente un double avantage : d'une part, cela permet une exploration plus rapide des hyperparamètres non structurels, le temps d'entraînement étant généralement proportionnel au nombre de paramètres du modèle ; d'autre part, cela favorise une meilleure généralisation, en limitant le surapprentissage et en améliorant souvent les performances sur des données inédites. Un tel choix s'inscrit dans le respect du **rasoir d'Occam**, qui privilégie les solutions les plus simples parmi celles qui expliquent les observations de manière équivalente.



**FIGURE 4.14** – Évolution de la loss  $\mathcal{L}(SDF) = \mathcal{L}_{SDF}^\delta$  sur le set de train et de validation au cours des 250 epochs.



**FIGURE 4.15** – Évolution de la loss  $\mathcal{L}(C_d)$  sur le set de train et de validation au cours des 250 epochs. Valeurs mesurées sur les coefficients normalisés ce qui explique la différence avec la figure 4.7

Une fois la meilleure structure déterminée, c'est-à-dire le choix des hyperparamètres structurels fait, il est possible de passer à l'optimisation des hyperparamètres non structurels. Dans notre cas, les hyperparamètres que nous cherchons à optimiser sont  $\lambda$  ainsi que  $\delta$ . Pour des raisons de temps, cette recherche n'a pas été finalisée pour le moment, les résultats présentés dans le chapitre résultats sont donc associés aux deux modèles choisis S1 et S1' avec les hyperparamètres non structurels définis dans la Table 4.2.

## 4 Coût computationnel

Dans cette partie, nous donnons un aperçu du coût de calcul qui a été nécessaire à l'entraînement des 19 modèles présentés précédemment.

Type de ressource	Processeur	VRAM	temps de calcul total	Max	Min
GPU	A100 / 3090	40 Gb / 24 Gb	896 h	100 h	13h

**TABLE 4.4** – Outils et ressources utilisées pour l'entraînement des 19 modèles.

Tous les modèles ont été entraînés sur des GPUs Nvidia (A100 ou 3090). Les temps de calculs sont donnés en heures d'entraînement.

# Chapitre 5

## Résultats

*Ce chapitre rapporte les résultats concernant la prédiction du coefficient de traînée ainsi que la reconstruction SDF associée au modèle choisi précédemment. Une comparaison des performances à d'autres méthodes de reconstruction SDF et de prédiction du coefficient de traînée sera également faite.*

### Sommaire

---

<b>1</b>	<b>Présentation des résultats</b>	<b>45</b>
1.1	Prédiction du coefficient de traînée	45
1.2	Reconstruction en distance signée	47
1.3	Mesures sur en dehors de la distribution	49

---

# 1 Présentation des résultats

Après le choix des modèles, nous donnons ici un aperçu de leurs performances ainsi qu'une comparaison à d'autres moyens de prédiction du coefficient de traînée ainsi que la reconstruction en distance signée. Les performances sont calculées sur le jeu de test. L'ensemble des données de test n'a jamais été visualisé ou testé en amont de cette partie, par ailleurs, les deux modèles sélectionnés ont été testés simultanément, ainsi, aucun choix d'hyperparamètres ou de structure n'est basé sur quelque mesure qu'il soit réalisée sur les données de test.

## 1.1 Prédiction du coefficient de traînée

Dans cette partie, nous abordons les performances en matière de prédiction du coefficient de traînée pour les deux modèles sélectionnés. Les modèles sont évalués sur les métriques définies au chapitre 4.

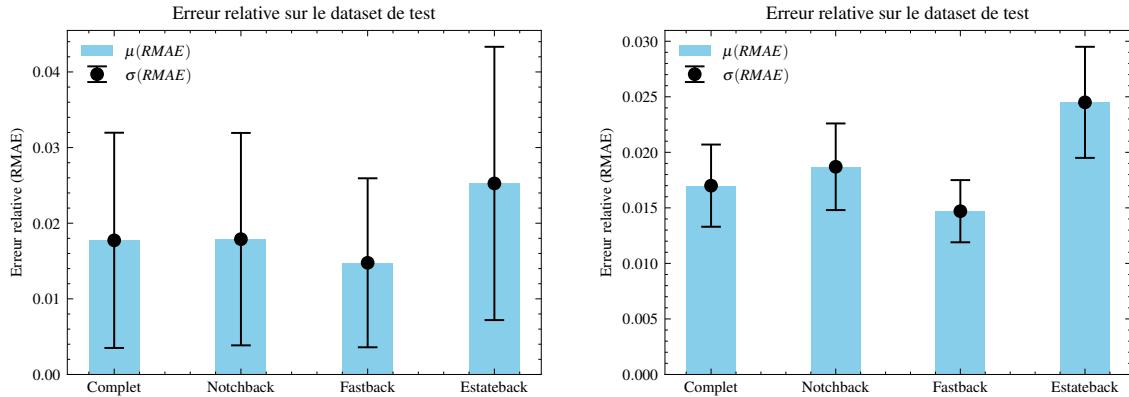
Modèle	MSE ↓ ( $\times 10^{-5}$ )	MAE ↓ ( $\times 10^{-3}$ )	Max AE ↓ ( $\times 10^{-2}$ )	$R^2 \uparrow$	Reconstruction SDF ?
PointNet[39]	12.000	8.850	10.180	0.826	✗
GCNN [30]	10.700	7.170	10.970	0.874	✗
PointNet++ [40]	7.813	6.755	3.463	0.896	✗
RegDGCNN [51]	8.010	6.910	8.800	0.901	✗
DeepGCN [33]	6.297	6.091	3.070	0.916	✗
MeshGraphNet [38]	6.000	6.080	2.965	0.917	✗
PointNeXt [41]	4.577	5.200	2.410	0.939	✗
FIGConvNet [12]	3.225	4.423	2.134	0.957	✗
TripNet [11]	<b>2.602</b>	<b>4.030</b>	<b>1.268</b>	<b>0.972</b>	✗
<b>GentNet S1</b>	<b>4.411</b>	<b>4.806</b>	<b>2.290</b>	<b>0.903</b>	✓
<b>GenNet S1'</b>	<b>3.406</b>	<b>4.490</b>	<b>2.077</b>	<b>0.913</b>	✓

**TABLE 5.1** – Comparaison des performances de modèles de l'état de l'art pour la régression sur formes 3D. Ces performances sont toutes évaluées sur le set de test de la base de données DrivAerNet++ [14]

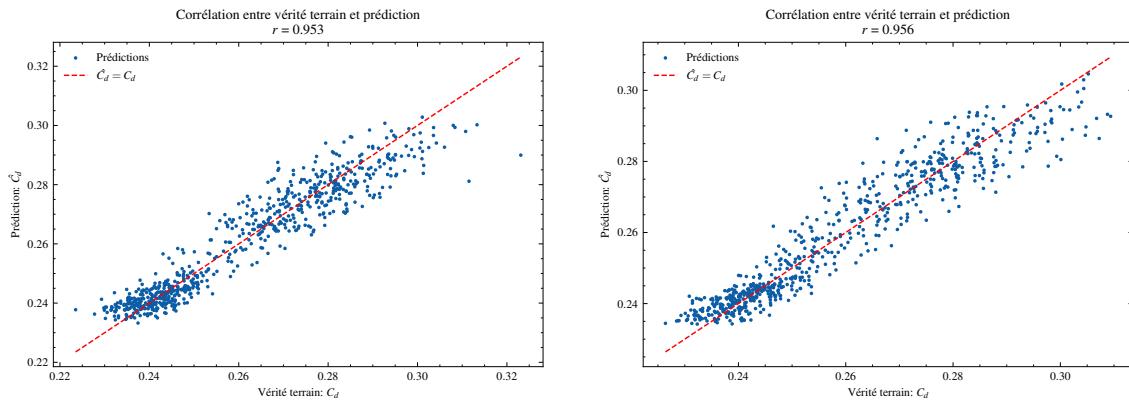
Modèle	Temps entraînement ↓	Temps d'inférence ↓	Paramètres
PointNet [39]	0.5 hrs	0.51 s	2,348,097
GCNN [30]	10.4 hrs	20.71 s	100,481
RegDGCNN [51]	3.2 hrs	0.52 s	3,164,257
<b>GenNet S1</b>	<b>23.7 hrs</b>	<b>17.02 s</b>	<b>496 642</b>
<b>GenNet S1'</b>	<b>61 hrs</b>	<b>24.14 s</b>	<b>598 276</b>

**TABLE 5.2** – Temps d'entraînement, d'inférence et complexité des modèles.

Nos deux modèles, **GenNet S1** et **GenNet S1'**, atteignent des performances comparables aux meilleures architectures actuelles pour la prédiction de  $C_d$ . Cette efficacité peut s'expliquer par l'utilisation de la Signed Distance Function (SDF), une représentation implicite continue, absente des autres méthodes comparées. Contrairement aux nuages de points utilisés par PointNet [39] et ses dérivés, ou aux maillages surfaciques employés par d'autres approches, la SDF permet une modélisation continue de la géométrie, facilitant son apprentissage via des couches fully-connected. En revanche, les temps d' entraînement et d'inférence de nos modèles sont plus élevés. Cela suggère que la densité d'échantillonnage de la SDF utilisée est supérieure à celle des nuages de points ou des maillages employés par les autres méthodes, ce qui pourrait partiellement expliquer la qualité des résultats obtenus. On donne ci-dessous les performances sur l'erreur relative ainsi qu'une représentation des performances sur la prédiction de  $C_d$  dans le plan  $(\hat{C}_d, C_d)$ :



**FIGURE 5.1** – Erreur relative moyenne (RMAE) sur le set de test pour les différents types de carrosseries. À gauche : modèle GenNet S1, à droite : modèle GenNet S1’.



**FIGURE 5.2** – Relation entre vérité terrain ( $C_d$ ) et prédiction ( $\hat{C}_d$ ) sur le set de test pour les modèles GenNet S1 (à gauche) et GenNet S1’ (à droite).

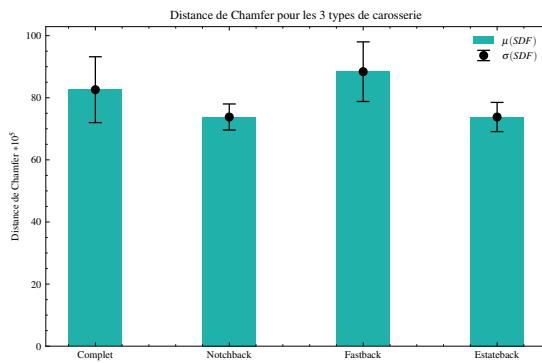
On observe une légère surperformance du modèle S1' par rapport au modèle S1. Par ailleurs, on remarque que le type de carrosserie Fastback, sur-représenté dans la base de données d'entraînement, est globalement favorable à la prédiction du coefficient de traînée. Cette observation montre la dépendance aux données des modèles d'apprentissage profond.

## 1.2 Reconstruction en distance signée

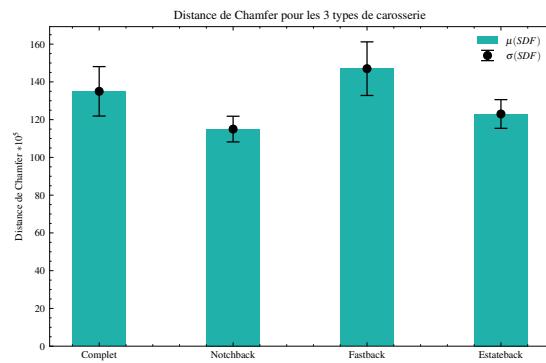
Dans cette section, nous comparons les performances de nos deux modèles sur la génération 3D. Les performances des différents modèles sont données par rapport à la distance de Chamfer (CD). Nos deux modèles ont de moins bonnes performances que l'état de l'art. On peut expliquer cela par le caractère bi-objectif de notre problème. Par ailleurs, DeepSDF [37] qui est le seul modèle utilisant la distance signée utilise deux fois plus de points d'échantillonnage que nous, expliquant en partie la surperformance de ce modèle. Enfin, DeepSDF s'appuie sur un auto-décodeur, différent de la structure en auto-encodeur que nous utilisons. Cette structure permet de meilleures performances mais est plus longue à l'inférence.

Méthode	$CD \times 10^3$ (moyenne)	Pred. $C_d$ ?	$n_{SDF}$	Benchmark
OGN [48]	0.167	✗	-	ShapeNet [10]
AtlasNet-Sph [17].	0.210	✗	-	ShapeNet [10]
AtlasNet-25 [17]	0.157	✗	-	ShapeNet [10]
DeepSDF [37]	0.084	✗	500k	ShapeNet [10]
<b>GenNet S1</b>	<b>0.826</b>	✓	250k	DrivAerNet++ [14]
<b>GenNet S1'</b>	<b>1.35</b>	✓	250k	DrivAerNet++ [14]

**TABLE 5.3 –** Comparaison de modèles génératifs 3D. CD : Distance de Chamfer (mean). Seulement GenNet prédit le coefficient de traînée  $C_d$  en plus de faire une reconstruction en distance signée. Les benchmarks font référence aux datasets utilisés pour la mesure des performances des différents modèles.

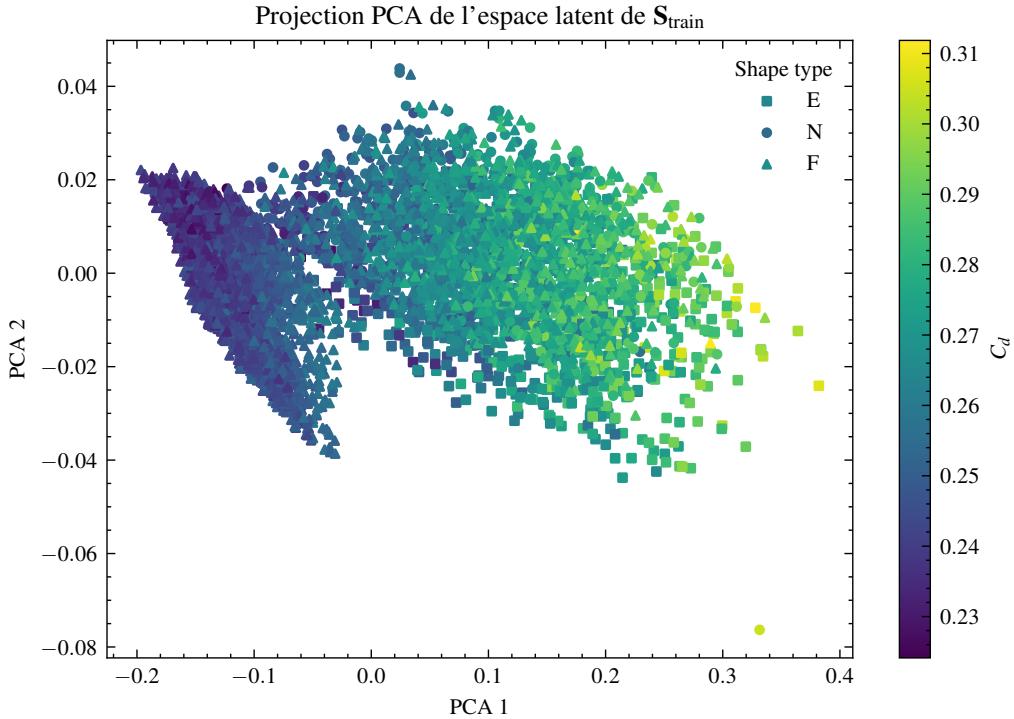


**FIGURE 5.3 –** Distance de Chamfer mesurée sur le set de test pour le modèle S1.



**FIGURE 5.4 –** Distance de Chamfer mesurée sur le set de test pour le modèle S1'.

Malgré de moins bonnes performances que l'état de l'art, notre modèle permet la génération de nouvelles formes détaillées. En effet, on montre ici que l'interpolation entre formes dans l'espace latent (*morphing*) permet de générer de nouvelles formes plausibles. On donne ici une représentation PCA (*principal component analysis*) (voir Annexe B) de l'espace latent d'entraînement (ensemble des vecteurs latents associés aux données du set d'entraînement). PCA1 et PCA2 sont les composantes décrivant le plus de variance :

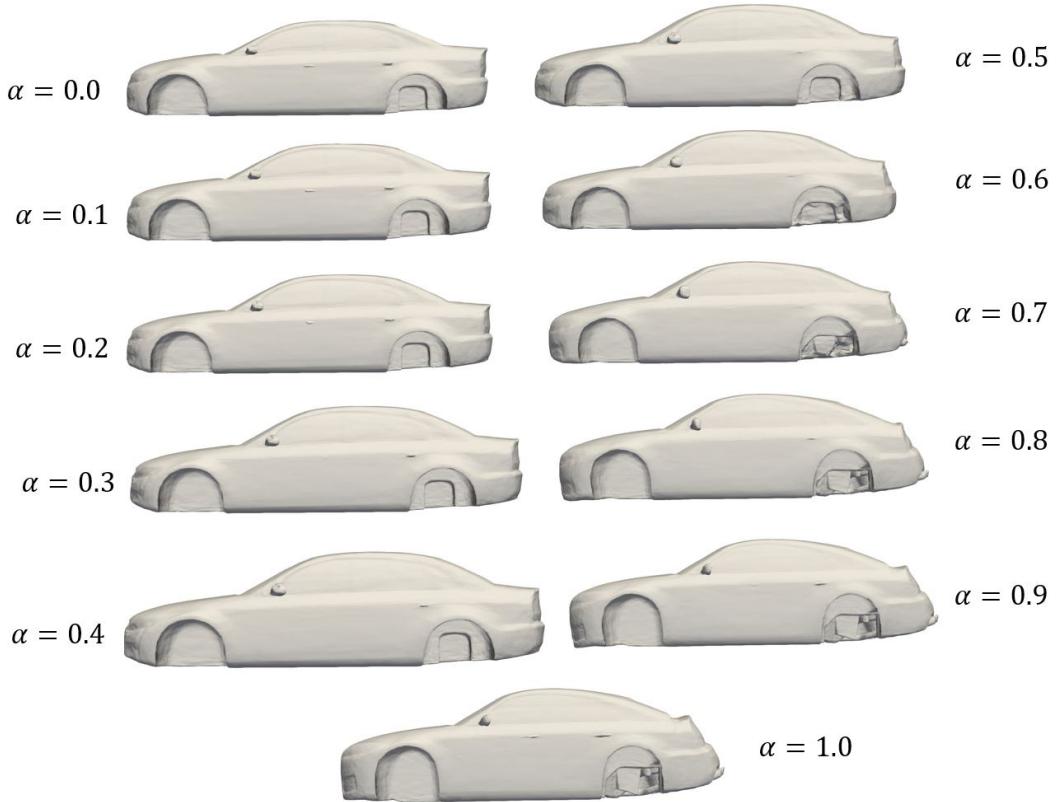


**FIGURE 5.5 – Espace latent  $\mathcal{Z}_{\text{train}}$**

On remarque que l'espace latent est structuré en deux pôles. Un premier se trouve à  $PCA1 > 0$ , comprenant majoritairement les véhicules de type Notchback et Estateback, de coefficient de traînée modéré à élevé. Le second pôle se trouve à  $PCA1 < 0$  et comporte majoritairement des véhicules de type Fastback, à faible coefficient de traînée. Cette observation montre que l'espace latent est ordonné (du moins les deux composantes principales) avec une logique de discrimination sur le coefficient de traînée.

Afin de visualiser la capacité d'interpolation dans l'espace latent, une technique couramment empruntée consiste à interpoler entre des vecteurs latents connus  $z_0$  et  $z_1$  et à visualiser les résultats. Pour cela, nous avons choisi deux modèles initialement dans le set de test avec des carrosseries différentes : un échantillon de type notchback ( $\alpha = 0$ ) de vecteur latent  $z_0$  ainsi qu'un échantillon de type fastback ( $\alpha = 1$ ) de latent  $z_1$ . Nous réalisons une interpolation linéaire entre ces deux formes comme :

$$z(\alpha) = \alpha z_0 + (1 - \alpha) z_1 \quad \text{avec } \alpha \in \{0, 0.1, \dots, 0.9, 1.0\}$$



**FIGURE 5.6** – Morphing entre deux carrosseries différentes :  $\alpha = 0 \longrightarrow$  Notchback,  $\alpha = 1.0 \longrightarrow$  Fastback.

On remarque que la génération de formes est précise et 'douce', c'est-à-dire que les formes générées permettent un passage d'une carrosserie notchback vers fastback progressif. L'ensemble des géométries réalisées avec  $\alpha \neq 0.0, \alpha \neq 1.0$  correspond à de nouvelles formes, qui n'étaient dans aucun des jeux.

### 1.3 Mesures sur en dehors de la distribution

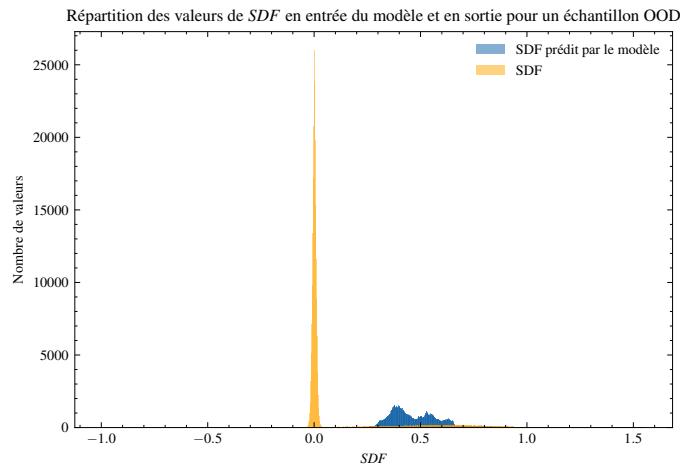
Les architectures des réseaux de neurones actuellement employées en deep learning ont une forte capacité d'**interpolation**, c'est-à-dire qu'un modèle bien entraîné peut générer/prédire des informations complexes si elles sont assez proches de ce qui a été vu lors de l'entraînement. Ces modèles ont en revanche des lacunes évidentes en matière d'**extrapolation**. L'extrapolation correspond à la faculté de faire de la prédiction nettement en dehors des données apprises. On distingue généralement deux distributions permettant de quantifier les performances en matière de généralisation d'un modèle : le **set ID** pour *in distribution* permet de mesurer les performances en matière d'interpolation. Le set ID est généralement issu du même dataset que le train et validation (c'est le cas du set de test utilisé précédemment). Le **set OOD** pour *out of distribution* permet quant à lui de quantifier les performances en extrapolation. Les données OOD sont issues de datasets significativement différents de celui ayant servi d'entraînement. Afin de quantifier les

performances de notre modèle en OOD, nous utilisons la base de données ModelNet40 [52], comprenant des maillages de 40 classes d’objets parmi lesquelles se trouvent des véhicules.



**FIGURE 5.7 – Echantillons de ModelNet40**

N’ayant pas d’étiquette  $C_d$  associée à ces véhicules, les tests ont été réalisés uniquement afin de visualiser les performances en termes de reconstruction SDF. Pour cela, nous avons échantillonné la SDF à partir des maillages de ces véhicules dans les mêmes conditions que les véhicules du dataset DrivAerNet++ (250k points + normalisation isotrope).

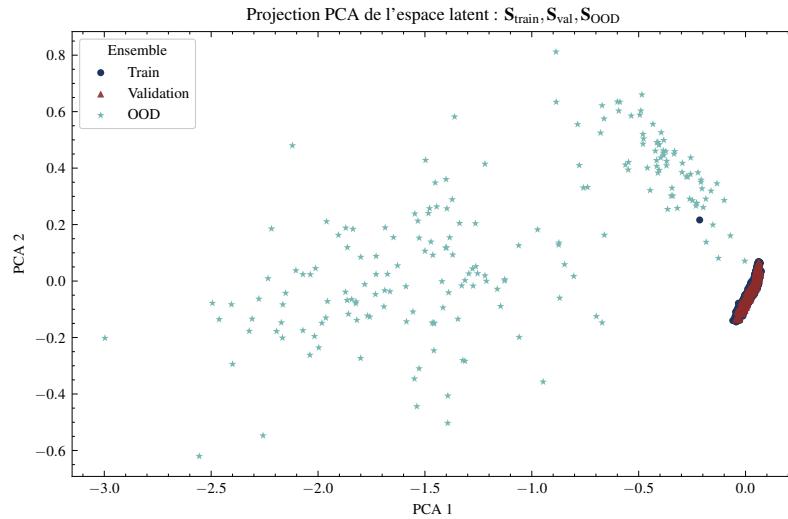


**FIGURE 5.8 – Distribution de la SDF associée à un maillage donné en entrée du modèle (Orange) et la SDF prédite (Bleu) pour les mêmes points d’échantillonage.**

On remarque que la SDF n’est pas stable, en effet :  $SDF \neq f_\theta \circ g_\theta(SDF)$  où  $f_\theta$  correspond à la fonction de transfert du décodeur SDF ( $\mathbf{D}_{SDF}$ ) et  $g_\theta$  la fonction de transfert associée à l’encodeur ( $\mathbf{E}$ ). On remarque par ailleurs que la SDF vérifie  $SDF_i \neq 0 \forall i \in [1, 250\,000]$  ce qui ne permet pas d’extraire d’isosurface et donc de reconstruire un maillage.

On peut expliquer ce problème en regardant la distribution des échantillons de ModelNet40 dans l’espace latent. Il apparaît clairement que les vecteurs latents associés aux données OOD sont très éloignés du set de train. On remarque en revanche que le set de validation est très proche du set de train. Une mesure de la distance de Mahalanobis

$d = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}$  (où  $\boldsymbol{\mu}$  est le vecteur de train moyen et  $\Sigma$  la matrice de covariance associée au set de train) confirme cette observation puisque  $\langle d_{val} \rangle \approx 13$  et  $\langle d_{OOD} \rangle \approx 148020$ .



**FIGURE 5.9 –** Visualisation des composantes principales des vecteurs latents associés au train, validation et OOD.

Cette observation soulève la nécessité de mesurer et quantifier avec précision l'incertitude émise sur les prédictions de notre modèle.

# Chapitre 6

## Mesure de l'incertitude

*Mesurer l'incertitude sur les prédictions du modèle est primordial. Pour cela, nous présentons dans ce chapitre une méthode permettant une estimation de l'incertitude sur la prédiction du coefficient de traînée de notre modèle basée sur l'utilisation d'un dropout de Monte-Carlo (MC dropout).*

### Sommaire

---

<b>1</b>	<b>Avant propos</b>	53
<b>2</b>	<b>Mesure de l'incertitude</b>	53
2.1	Dropout Monte-Carlo	54

---

## 1 Avant propos

Mesurer l'incertitude sur un résultat est une démarche nécessaire afin de donner un niveau de confiance associé à un résultat. La mesure de l'incertitude sur les prédictions faites par un modèle de Deep Learning est un sujet de plus en plus mis en avant dans la communauté, en effet, les modèles d'aujourd'hui remplissent des tâches complexes et critiques, où la mesure de l'incertitude sur un résultat est primordiale (médecine, industrie ...). Dans notre cas, déterminer le niveau de confiance que l'on peut avoir sur notre prédiction permet de gagner beaucoup de temps et d'optimiser le design inverse (voir chapitre 7), en effet, connaître l'incertitude sur une prédiction permet de tester uniquement les géométries de voitures qui sont à la fois performantes (coefficient de traînée faible) et robustes (incertitude sur la mesure de  $C_d$  faible).

## 2 Mesure de l'incertitude

On distingue deux types d'incertitudes :

- **L'incertitude aléatoire** (ou incertitude intrinsèque) correspond à la variabilité des données, due à un bruit de mesure, une variabilité physique ou un processus stochastique. Elle est irréductible, ce qui signifie qu'un modèle parfait entraîné sur une infinité de données donnerait toujours des résultats soumis à une incertitude non nulle. Dans notre cas, le bruit vient du calcul numérique du coefficient de traînée par CFD et de l'échantillonnage de la SDF à partir du mesh.
- **L'incertitude épistémique** (incertitude du modèle) traduit le manque de connaissance du modèle sur sa relation entrée-sortie. Elle est liée aux paramètres appris et reflète le fait que le modèle n'a pas été suffisamment entraîné dans certaines régions de l'espace d'entrée.

Dans notre cas, il est raisonnable de penser que l'incertitude aléatoire est faible étant donné la nature des données d'entrée. En effet, les calculs CFD, surtout pour des maillages fins comme ceux utilisés pour représenter les véhicules constitutants la base de données DrivAerNet++, sont fiables et à faible incertitude. Pour ces raisons, nous nous intéressons à la mesure de **l'incertitude épistémique**.

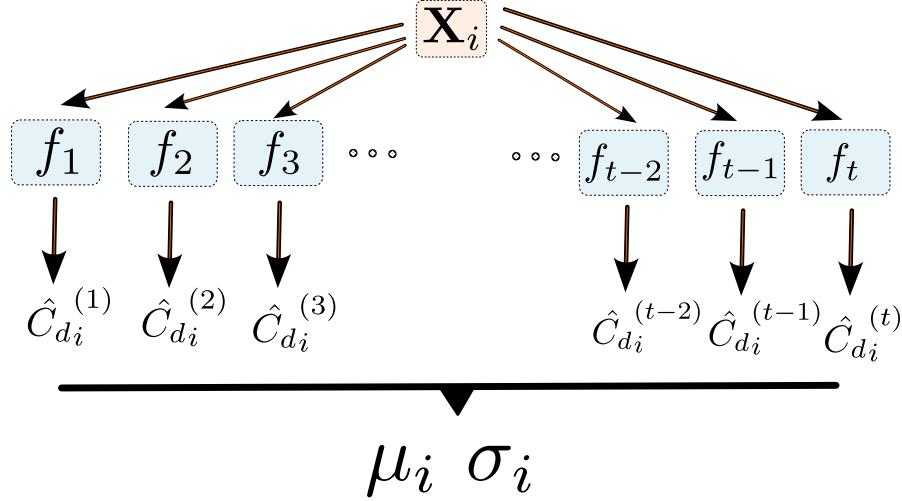
De nombreuses méthodes permettent la mesure de l'incertitude épistémique liée à l'utilisation d'un modèle d'apprentissage profond. Les **réseaux Bayésiens** [9] (BNN), contrairement aux réseaux de neurones classiques, qui apprennent un ensemble fixe de poids  $\theta$  en minimisant une fonction de coût, modélisent l'**incertitude sur les poids** eux-mêmes. Chaque paramètre du réseau est vu comme une variable aléatoire, décrite par une distribution de probabilité. L'objectif devient alors d'estimer la **distribution a posteriori**  $p(\theta | \mathcal{D})$  des poids sachant les données d'apprentissage  $\mathcal{D}$ . La prédiction pour une nouvelle entrée  $x$  est obtenue en marginalisant sur les poids :

$$p(y | x, \mathcal{D}) = \int p(y | x, \theta) p(\theta | \mathcal{D}) d\theta.$$

c'est-à-dire qu'on intègre toutes les prédictions possibles  $p(y \mid x, \theta)$ , pondérées par la probabilité que ces poids soient vrais d'après les données  $\mathcal{D}$ . Cette méthode, bien qu'elle soit très fiable et robuste, est compliquée à mettre en oeuvre. Une alternative aux approches bayésiennes consiste à utiliser des *Deep Ensembles* [31], une méthode empirique simple mais efficace pour quantifier l'incertitude. Elle consiste à entraîner  $N$  fois le même modèle, chacun avec une initialisation aléatoire différente. Chaque entraînement converge vers un minimum local potentiellement différent, ce qui permet d'explorer indirectement une portion de l'espace des poids. L'incertitude sur la prédiction est alors estimée à partir de la variance des sorties des  $N$  modèles, typiquement en utilisant l'écart-type. Les Deep Ensembles bien qu'ils soient très simples à mettre en place demandent beaucoup de ressources en calcul. Une autre approche plus récente pour estimer l'incertitude est *SWAG* [34] (Stochastic Weight Averaging-Gaussian). Cette méthode repose sur l'observation que, durant l'entraînement stochastique, les poids d'un réseau oscillent dans un voisinage proche d'un minimum de la fonction de perte. SWAG consiste à capturer ces fluctuations en moyennant les poids visités en fin d'entraînement (stochastic weight averaging), puis en ajustant une distribution gaussienne (approchée par un low-rank plus diagonale) autour de cette moyenne. Cette gaussienne approximative  $q(\mathbf{w})$  sur les poids permet alors d'échantillonner efficacement différents ensembles de poids pour produire des prédictions stochastiques, sans avoir à réentraîner plusieurs modèles comme dans les deep ensembles. SWAG constitue donc une solution peu coûteuse et scalable pour approximer une postérieure bayésienne, tout en étant compatible avec les entraînements standards en deep learning. Cette méthode bien que simple et peu coûteuse n'a pas été utilisée au profit d'une mesure d'incertitude par MC dropout [15].

## 2.1 Dropout Monte-Carlo

Afin de mesurer l'incertitude épistémique associée à notre modèle, nous utilisons la méthode du **dropout Monte-Carlo** (MC Dropout). Le MC dropout consiste à inférer le modèle en maintenant le dropout activé avec un ratio  $p$  similaire à celui utilisé lors de l'entraînement du modèle ( $p = 0,05$  dans notre cas). On peut alors réaliser  $T$  passes avec une même donnée d'entrée  $\mathbf{X}_i$  et obtenir  $T$  résultats différents. Le dropout n'est activé que sur la dernière couche du décodeur physique ( $\mathbf{D}_\phi$ ), pour plusieurs raisons. D'une part, appliquer du dropout sur un trop grand nombre de couches peut dégrader la prédiction moyenne, en introduisant un biais trop important. Pour limiter cet effet, nous avons donc retenu une valeur modérée de  $p = 0,05$  appliquée uniquement à la dernière couche. D'autre part, nous avons choisi de ne pas activer le dropout dans le décodeur géométrique ( $\mathbf{D}_{SDF}$ ), afin de ne pas complexifier une tâche déjà difficile qu'est la reconstruction implicite de la forme par reconstruction SDF. Par ailleurs, les informations nécessaires à la reconstruction de la forme via la SDF et à la prédiction du coefficient de traînée  $\hat{C}_d$  étant mutualisées dans le même vecteur latent  $z$ , il nous a semblé suffisant d'estimer l'incertitude uniquement sur la prédiction de  $\hat{C}_d$ . Enfin, il est bien plus simple et rapide de quantifier une incertitude sur un scalaire (comme le coefficient de traînée) que sur une fonction implicite SDF :  $\mathbb{R}^3 \longrightarrow \mathbb{R}$  définie sur tout l'espace tridimensionnel.



**FIGURE 6.1** – Mesure de l'incertitude par MC dropout.

A chaque forward pass, un masque de neurones aléatoire différent est défini. Ainsi, chaque passe correspond à une hypothèse plausible  $f_t$  issue de la distribution a posteriori implique des paramètres du réseau (ie ensemble des configurations possibles de poids compatibles). En effectuant  $T$  passes sur la même entrée, on obtient  $T$  prédictions différentes  $\{\hat{C}_d^{(1)}, \hat{C}_d^{(2)}, \dots, \hat{C}_d^{(t)}, \dots, \hat{C}_d^{(T)}\}$ , dont la moyenne  $\mu = \frac{1}{N} \sum_{t=1}^T \hat{C}_d^{(t)}$  fournit l'estimation finale et l'écart-type  $\sigma = \sqrt{\frac{1}{N} \sum_{t=1}^T (\hat{C}_d^{(t)} - \mu)^2}$  une approximation de l'incertitude épistémique. Les moyennes et écart-types sont bien sûr calculés sur les valeurs dénormalisées.

### 2.1.1 Calibration de l'incertitude

Bien que le MC dropout permette de donner une approximation de l'incertitude épistémique, il est souvent mal calibré, c'est-à-dire que les valeurs d'incertitudes, bien qu'elles soient corrélées au niveau de certitude qu'a le modèle concernant une prédiction (une prédiction hors distribution aura une incertitude plus élevée qu'une prédiction dans la distribution), sont soit trop élevées, soit trop faibles (ce qui est notre cas). Pour pallier ce problème, il est d'usage de procéder à une calibration visant à réhausser ou diminuer les valeurs d'incertitudes. La méthode la plus simple consiste à trouver un facteur  $\alpha$  cohérent tel que  $\sigma_{\text{cal}} = \alpha \sigma$ . Dans notre cas, nous cherchons  $\alpha$  par une minimisation de la **log-vraisemblance négative** (*negative log likelihood NLL*). Soit  $p(C_d | f_\theta(\mathbf{x}))$  la vrai-

semblance, on définit la NLL comme :

$$NLL = -\log p(C_d | f_\theta(\mathbf{x}))$$

Minimiser la NLL est équivalent à maximiser la vraisemblance (qui correspond à la probabilité d'obtenir une valeur vraie  $C_d$  sachant une hypothèse  $f_\theta$  et des données  $\mathbf{x}$ ). On fait l'hypothèse de bruit Gaussien pour l'erreur ce qui signifie :  $\epsilon \sim \mathcal{N}(0, \sigma^2)$  avec  $\sigma^2 = \sigma_{al}^2 + \sigma_{epi}^2$ .

On a  $C_d = \mu - (\mu - C_d) = \mu - \epsilon$ , étant donné  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ , on a :

$$p(C_d | f_\theta(\mathbf{x})) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2)$$

Etant supposé  $\sigma_{al}$  faible devant  $\sigma_{epi}$ , on a :  $\sigma \approx \sigma_{epi}$ , d'où finalement :

$$p(C_d | f_\theta(\mathbf{x})) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma_{epi}^2) = \frac{1}{\sqrt{2\pi\sigma_{epi}^2}} \exp\left(-\frac{1}{2}\left(\frac{C_d - \mu(\mathbf{x})}{\sigma_{epi}}\right)^2\right)$$

On a ici l'expression de la vraisemblance associée à une seule prédiction  $\mu(\mathbf{x})$  (calculée comme la prédiction moyenne sur les  $T$  passes de MC dropout), on peut définir la vraisemblance sur le dataset de validation composé de  $N$  données :

$$p(\mathbf{C}_d | \mathbf{f}_\theta, \mathbf{S}_{val}) = \prod_{i=1}^N p(C_d^{(i)} | \mathbf{f}_\theta, \mathbf{S}_{val})$$

D'où :

$$NLL = \frac{1}{2} \sum_{i=1}^N [\log(2\pi\sigma_{epi}^{(i)}) + \left(\frac{C_d^{(i)} - \mu(\mathbf{x})^{(i)}}{\sigma_{epi}^{(i)}}\right)^2]$$

Comme stipulé auparavant,  $\sigma_{epi}$  donné par le MC dropout n'est pas calibré, on remplace donc  $\sigma_{epi}$  par  $\sigma_{cal} = \alpha\sigma_{epi}$  dans la NLL et on cherche  $\alpha^* = \arg \min_\alpha NLL(\alpha)$ .

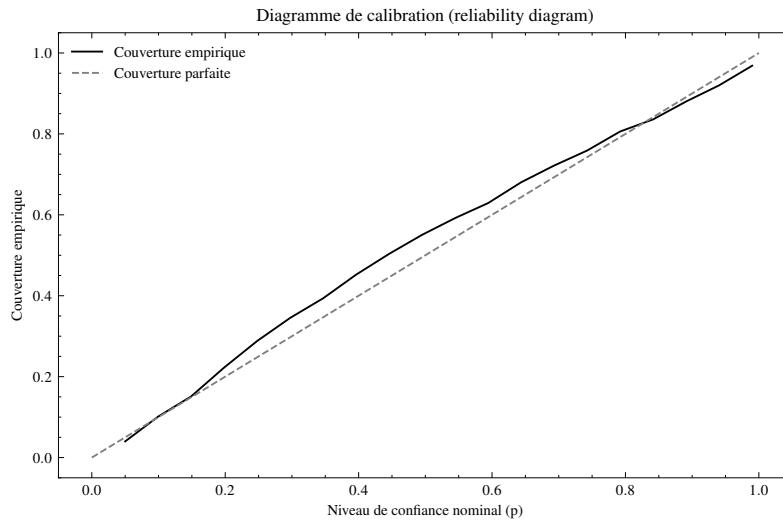
On a donc :

$$NLL(\alpha) = \frac{1}{2} \sum_{i=1}^N [\log(2\pi\alpha\sigma_{epi}^{(i)}) + \left(\frac{C_d^{(i)} - \mu(\mathbf{x})^{(i)}}{\alpha\sigma_{epi}^{(i)}}\right)^2]$$

En résolvant  $\frac{\partial NLL(\alpha)}{\partial \alpha} = 0$ , on trouve :

$$\alpha^* = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\frac{C_d^{(i)} - \mu_i}{\sigma_{epi}^{(i)}}\right)^2}$$

On peut alors calibrer  $\sigma_{epi}$ . Dans notre cas, on trouve  $\alpha \approx 9.846$ , permettant de passer la NLL de  $NLL(\alpha = 1) \approx 42.049$  (pré-calibration) à  $NLL(\alpha = \alpha^*) \approx -3.637$  (post-calibration).



**FIGURE 6.2** – Couverture empirique avec MC dropout calibré vs couverture théorique pour une loi Normale.

	$\pm\sigma$	$\pm 2\sigma$	$\pm 3\sigma$
Dropout non calibré ( $\alpha = 1$ )	10, 27%	19, 83%	30, 53%
Dropout calibré ( $\alpha = \alpha^*$ )	76, 18%	94, 01%	98, 29%
Attendu	68, 3%	95, 4%	99, 7%

**TABLE 6.1** – Couverture associée à  $\pm 1\sigma$ ,  $\pm 2\sigma$ ,  $\pm 3\sigma$  pour le dropout non calibré, calibré et attendu pour une loi normale. Les tests de couverture ont été réalisés sur le set de validation.

La calibration du dropout avec  $\alpha = \alpha^*$  permet de donner à  $\sigma_{cal}$  une valeur interprétable pour le calcul d'intervalles de confiance associés à une loi normale. On remarque néanmoins que l'intervalle de confiance à  $\pm\sigma$  associé au dropout calibré est légèrement trop conservateur.

# Chapitre 7

## Optimisation dans l'espace latent

*Etant donné la capacité de notre modèle de générer de nouvelles formes par interpolation dans l'espace latent ainsi que la prédiction du coefficient de traînée associé, il est désormais possible de faire de la génération inverse afin de générer des formes de voitures plausibles à faible coefficient de traînée.*

### Sommaire

---

<b>1</b>	<b>Démarche d'optimisation</b>	<b>59</b>
<b>2</b>	<b>Recherche dans l'espace latent par bruit blanc</b>	<b>59</b>
<b>3</b>	<b>Optimisation dans l'espace latent par descente de gradient</b>	<b>61</b>

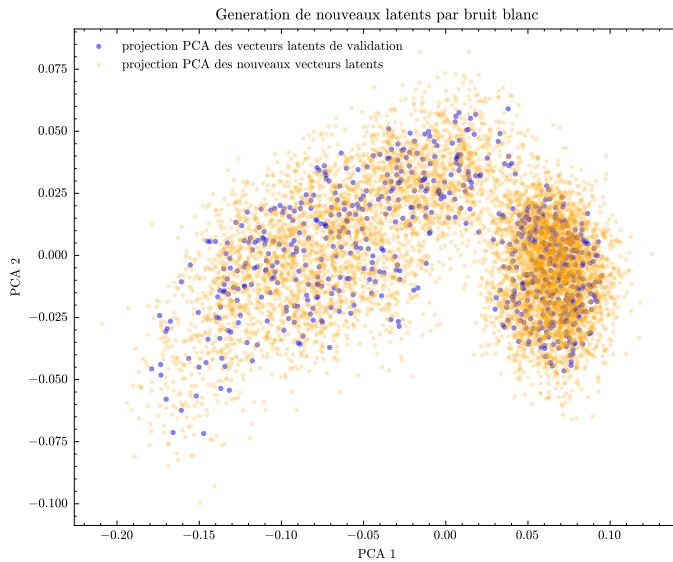
---

## 1 Démarche d'optimisation

La génération inverse consiste, à partir d'un vecteur latent  $z$  à générer la forme associée et ainsi à créer de nouvelles formes à partir de celles apprises. Cette démarche peut être couplée à une optimisation du coefficient de traînée. En effet, le modèle étant capable de déterminer le coefficient de traînée associé à un vecteur latent  $z$ , il est possible de calculer le coefficient de traînée associé à chaque nouvelle forme que l'on souhaite générer à partir d'un vecteur latent jamais vu lors de l'entraînement.

## 2 Recherche dans l'espace latent par bruit blanc

Plusieurs démarches d'optimisation du coefficient de traînée peuvent être mises en place, il est par exemple possible de générer des vecteurs latents proches de ceux vus lors de l'entraînement en appliquant un bruit Gaussien sur ces valeurs de sorte que :  $z_{new} = z + \mathcal{N}(0, \sigma^2)$  où  $\sigma$  est choisi, une valeur élevée permet d'explorer le latent amplement, tandis qu'une faible valeur permet de rester proche des valeurs apprises lors de l'entraînement. Cette approche a l'avantage d'être très simple à mettre en place mais ne permet pas de chercher dans des directions permettant l'optimisation du coefficient de traînée, en effet, cette méthode est 'exploratoire', un choix des meilleures formes générées selon  $C_d$  doit être réalisé en amont.



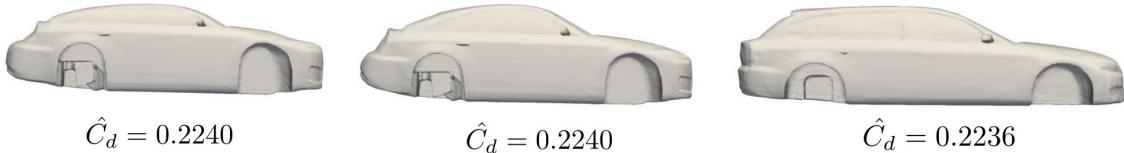
**FIGURE 7.1 –** Génération de nouveaux vecteurs latents par bruit blanc ( $\sigma = 0.01$  ici)

Dans cette section, nous utilisons le modèle GenNet S1. La recherche par bruit blanc a été effectué en générant 1 million de vecteurs latents tels que :

$$z_{gen} = z_{train} + \epsilon$$

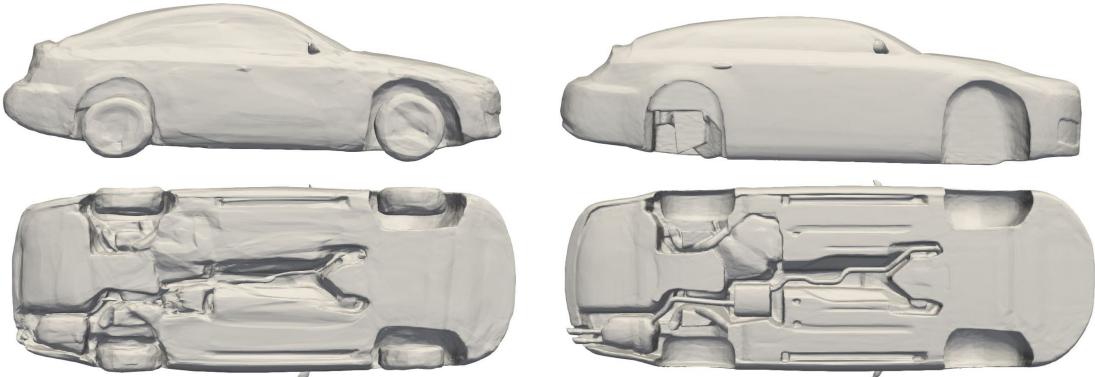
où  $\epsilon_i \sim \mathcal{N}(0, a^2 * \mathbb{V}(z_i))$ ,  $i$  représente la i-ième composante du vecteur et  $\mathbb{V}(z_i)$  la variance décrite par la i-ème composante de l'espace latent de train (mesurée par PCA). Cette

expression du bruit a pour avantage de créer un bruit d'intensité adapté à chaque composante. Nous avons ensuite visualisé les maillages reconstruits à partir des distances signées des échantillons générés avec des coefficient de traînée tels que  $C_d^{\text{gen}} < C_d^{\text{DrivAerNet++}}$ . Nous avons généré 700 000 vecteurs latents par addition du bruit blanc aux vecteurs de train adapté avec  $a^2 = 0.5$ . Après un filtrage des meilleurs résultats, nous obtenons 3 géométries telles que  $\hat{C}_d < C_d^{\text{DrivAerNet++}}$ .



**FIGURE 7.2** – 3 véhicules générées par bruit blanc adapté tels que  $\hat{C}_d < C_d^{\text{DrivAerNet++}}$  avec filtrage sur le maillage généré (car défauts au niveau des roues).

Nous avons également réalisé des tests avec bruit blanc constant, d'écart type  $\sigma = 0.01$ . On observe une différence marquante entre les résultats associés aux deux méthodes :



**FIGURE 7.3** – Génération avec bruit blanc constant pour  $\sigma = 0.01$

**FIGURE 7.4** – Génération avec bruit blanc adapté avec  $a^2 = 0.5$ . (+ filtrage car défauts au niveau des roues).

Cette méthode de recherche dans l'espace latent a permis l'obtention de bons résultats, il reste dans un second temps à valider les coefficients de traînée prédis par calcul CFD (OpenFoam) dans les mêmes conditions que celles utilisées pour calculer les coefficients de la base DrivAerNet++.

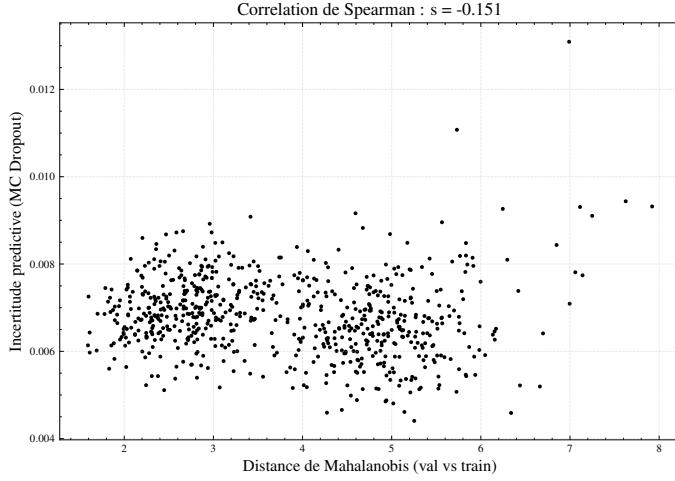
### 3 Optimisation dans l'espace latent par descente de gradient

Une deuxième méthode consiste à réaliser une descente de gradient dans l'espace latent  $\mathcal{Z}$ . En effet, de la même manière qu'on réalise une descente de gradient lors de l'entraînement sur l'espace des poids  $\theta$  afin de minimiser une fonction coût, on peut réaliser une descente de gradient afin de minimiser une fonction judicieusement choisie lors de l'inférence du modèle. Le problème de la descente de gradient se formalise alors comme :

$$\text{trouver } z^* = \arg \min_z \mathcal{L}(z)$$

Cette méthode est optimisée puisqu'elle permet, avec une bonne fonction coût  $\mathcal{L}$ , d'aller dans les directions permettant la minimisation du coefficient de traînée. Cette méthode est en revanche plus difficile à mettre en place, d'abord parce que réaliser une descente de gradient sur  $z$  par rapport à  $C_d$  nécessite que la relation  $\hat{C}_d(z)$  ne soit pas linéaire, sans quoi une seule direction ne peut être exploitée lors de la descente, cela nécessite donc l'utilisation de plusieurs couches cachées pour le décodeur physique ( $\mathbf{D}_\phi$ ) ce qui complexifie le modèle. Enfin, le choix de la fonction coût ayant un impact direct sur l'optimisation, il est nécessaire de procéder à différents tests permettant de choisir la meilleure fonction à minimiser.

Étant donné l'objectif de minimisation du coefficient de traînée  $C_d$ , une loss naturelle serait simplement  $\mathcal{L} = C_d$ , néanmoins, cette loss pousse le réseau à produire des résultats faux en allant vers des zones de l'espace latent où les prédictions sont très incertaines. Une démarche fréquente consiste à ajouter une fonction coût de régularisation  $\|z\|_2^2$ . Néanmoins, cette régularisation suppose que l'espace latent d'entraînement est centré en 0, ce qui est généralement vrai pour les encodeurs variationnels (VAE) mais faux pour les auto-encodeurs comme celui utilisé ici, par ailleurs le choix de la norme  $L_2$  est fait sur l'a priori que l'espace latent  $\mathcal{Z}$  est isotrope ce qui est souvent faux (faux dans notre cas). Enfin, cette méthode de régularisation, qui à pour objectif implicite de rester dans des zones avec incertitude sur la prédiction faible en limitant la distance entre un  $z$  choisi et l'espace latent d'entraînement, suppose une corrélation entre la distance  $\|z\|_2$  et l'incertitude sur la mesure. Cette hypothèse s'avère être fausse dans notre cas, en effet, nous pouvons montrer que la distance de Mahalanobis  $d = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}$  (qui permet de prendre le centre  $\boldsymbol{\mu}$  de l'espace latent d'entraînement comme mesure de distance ainsi que l'anisotropie de  $\mathcal{Z}$  (grâce à la matrice de covariance  $\Sigma$ )) mesurée entre chaque point du set de validation et du set d'entraînement ainsi que l'incertitude sur la prédiction de  $\hat{C}_d$  pour chaque vecteur latent du set de validation est totalement décorrélée :



**FIGURE 7.5** – Corrélation de Spearman (voir Annexe B) entre la distance de Mahalanobis mesurée entre les vecteurs latents du set de validation  $\mathcal{S}_{\text{val}}$  et le centre de l'espace latent associé à  $\mathcal{S}_{\text{train}}$  et l'incertitude épistémique mesurée par MC dropout pour chaque échantillon de  $\mathcal{S}_{\text{val}}$ .

La mesure de l'incertitude épistémique via MC dropout nous permet d'améliorer la régularisation en postulant la loss  $\mathcal{L}$  à minimiser :

$$\mathcal{L} = \mathbb{E}_T[\hat{C}_d^{(t)}] + \lambda_{\text{reg}} \cdot \sqrt{\mathbb{V}_T[\hat{C}_d^{(t)}]} + \lambda_d \cdot (e^d - e^{<d>}) = \mu + \lambda_{\text{reg}} \cdot \sigma_{\text{epi}} + \lambda_d \cdot (e^d - e^{<d>})$$

$\mu$  représente la prédiction moyenne faite sur les T passes de MC dropout,  $\sigma_{\text{epi}}$  l'incertitude mesurée,  $d$  la distance de Mahalanobis au plus proche vecteur du set de train et  $< d >$  la distance minimale moyenne entre deux vecteurs latents du set d'entraînement. L'ajout de l'exponentielle de la distance par rapport au plus proche vecteur d'entraînement permet de pénaliser fortement une éventuelle sortie du manifold. En effet, générer de nouvelles formes loin de l'espace latent appris par le réseau (extrapolation) donne de mauvais résultats en matière de reconstruction SDF et des prédictions du coefficient de traînée très peu fiables. Par ailleurs, générer de nouvelles formes en interpolant dans l'espace latent d'entraînement revient à générer de nouvelles formes en interpolant dans l'espace paramétrique des géométries de véhicules du set d'entraînement ce qui est désiré.

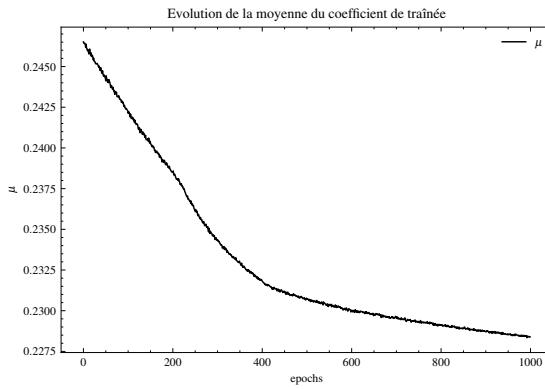
$\lambda_{\text{reg}}$  et  $\lambda_d$  sont des hyperparamètres propres à l'inférence du modèle, qui doivent être paramétrés afin de donner un poids variable à l'incertitude dans la loss et à la distance au plus proche voisin.  $\sigma_{\text{epi}}$  est l'incertitude épistémique calibrée mesurée par MC dropout.

On minimise cette fonction coût avec l'optimizer Adam (variante de la descente de gradient classique) :

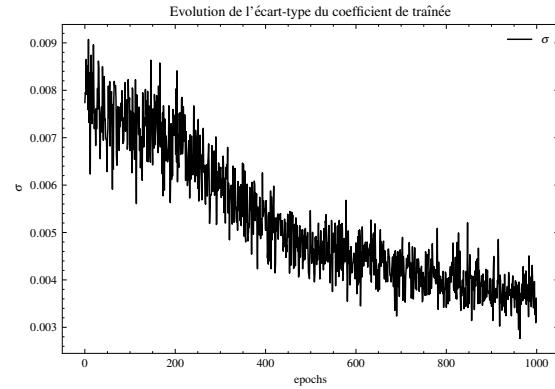
$$z_i := z_i - \eta \frac{\partial \mathcal{L}}{\partial z}$$

Plusieurs méthodes d'initialisation du latent peuvent être envisagées (vecteur aléatoire, vecteur moyen sur le set d'entraînement...). Les résultats de l'optimisation ci-dessous

montrent l'évolution du coefficient de traînée prédit ainsi que l'incertitude associée pour une initialisation avec un vecteur de l'espace latent d'entraînement correspondant à une géométrie de type Fastback. Les tests ont été réalisés en effectuant 1000 itérations (notées epochs ci-dessous) avec  $\lambda_d = 0.005$  et  $\lambda_{\text{reg}} = 1$ .

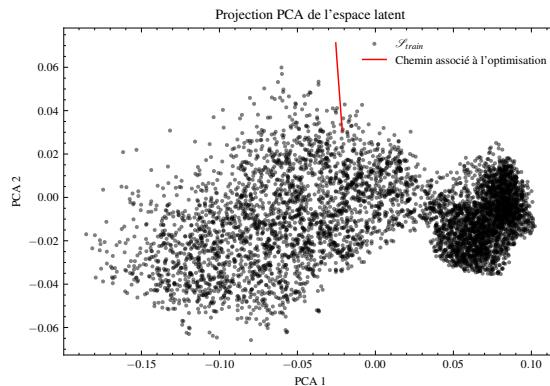


**FIGURE 7.6 – Optimisation de la moyenne  $\mu = \langle \hat{C}_d \rangle$  prédite par MC dropout**

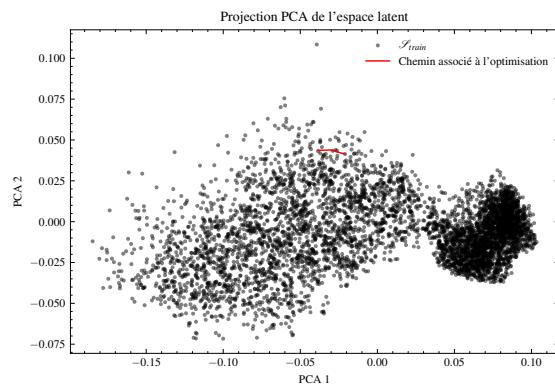


**FIGURE 7.7 – Évolution de l'incertitude épistémique mesurée par MC dropout**

La fonction coût avec régularisation sur la distance de Mahalanobis et l'incertitude permet la convergence du coefficient de traînée vers  $\hat{C}_d \approx 0.224$  ainsi que la diminution de l'incertitude sur la prédiction.



**FIGURE 7.8 – Évolution du vecteur latent dans l'espace (PCA1, PCA2) pour  $\lambda_d = 0$  et  $\lambda_{\text{reg}} = 1$ .**



**FIGURE 7.9 – Évolution du vecteur latent dans l'espace (PCA1, PCA2) pour  $\lambda_d = 0.005$  et  $\lambda_{\text{reg}} = 1$ .**

On remarque que l'optimisation sort du manifold de train, ce qui n'est pas souhaité. Un ajustement avec  $\lambda_d = 0.005$  permet de rester dans le manifold et de contrôler l'exploration dans l'espace latent.



**FIGURE 7.10** – Géométrie générée au bout de 1000 itérations avec la fonction coût régularisée pour  $\lambda_{\text{reg}} = 1$  et  $\lambda_d = 0.01$

Cette méthode n'a pour le moment pas permis d'obtenir des formes détaillées et plausibles. Une paramétrisation fine des hyperparamètres de la fonction coût pourrait permettre de générer de nouvelles formes optimisées selon un critère de minimisation du coefficient de traînée.

# Conclusion

Ce mémoire s'inscrit dans le cadre d'un projet de génération de formes 3D optimisées, couplant apprentissage de la géométrie (via SDF) et prédiction de propriétés physiques (ici le coefficient de traînée aérodynamique). L'objectif principal était de développer un modèle capable à la fois d'encoder efficacement une forme en un vecteur latent, mais aussi d'optimiser ce vecteur en fonction de critères physiques, tout en tenant compte de l'incertitude prédictive. Pour cela, nous avons utilisé une structure dite d'autoencodeur, entraîné à prédire la distance signée (SDF) d'un échantillon du dataset DrivAerNet++, tout en apprenant simultanément à estimer son coefficient de traînée. Un mécanisme d'optimisation du vecteur latent  $z$ , sous contrainte, a ensuite été mis en œuvre pour générer des formes minimisant  $C_d$ , en exploitant le MC Dropout afin de prendre en compte l'incertitude épistémique. Une régularisation par distance de Mahalanobis a également été introduite, permettant de maintenir l'optimisation dans le voisinage du manifold des formes connues, évitant ainsi des extrapolations non physiques. Les résultats obtenus montrent que, malgré un grand potentiel d'amélioration, il est possible de générer des formes plausibles et détaillées tout en prédisant un coefficient de traînée avec précision. Ce travail présente cependant certaines limites. D'une part, la quantité de données reste modeste (8000 échantillons) au regard de la diversité morphologique possible. D'autre part, l'apprentissage du modèle repose sur des données simulées (CFD), elles-mêmes soumises à des approximations numériques. Enfin, l'optimisation du latent reste sensible aux hyperparamètres, en particulier aux poids de régularisation qui pourraient être améliorées dans le futur. Le modèle est également améliorable, notamment via l'exploration approfondie de l'espace des hyperparamètres. Il serait également possible d'intégrer des mécanismes tels que les Fouriers Features [47] afin de mieux représenter les parties à hautes fréquences de la SDF, utiliser une fonction de coût hétéroscédastique [29] ou à poids auto-adaptatifs... Il serait également intéressant d'entraîner, en parallèle de GenNet, un modèle permettant la prédiction des caractéristiques (longueur, largeur, angles ...) d'un véhicule à partir d'une représentation en SDF. Cela permettrait d'avoir un modèle génératif ainsi que les dimensions associées (dimensions dans  $[-1, 1]^3$ )

## **Mentions légales**

Sauf cas contraire, toutes les figures, illustrations et images présentes dans ce mémoire ont été réalisées par l'auteur. Toute reproduction ou réutilisation, totale ou partielle, sans autorisation explicite de l'auteur, est interdite.

# Bibliographie

- [1] (2015). Paris Agreement. United Nations Framework Convention on Climate Change (UNFCCC). Adopted 12 December 2015, entered into force 4 November 2016, 196 Parties.
- [2] (2021). Regulation (EU) 2021/1119 – european climate law. Official Journal of the European Union, L243. Adopted 30 June 2021.
- [3] (2023). Directive (EU) 2023/2413 – renewable energy directive iii. Official Journal of the European Union, L328. Adopted 18 October 2023 ; increases target to 42.5
- [4] (2023). Japan aims for 100 InfluenceMap / Climate Action Tracker. Target includes hybrids, PHEVs, EVs ; part of Green Growth strategy.
- [5] AGENCY, U. E. P. (2023). Transportation sector emissions. EPA. Transportation accounts for 29% of total U.S. GHG ; passenger vehicles contribute 58% of transport emissions.
- [6] AL-GHAILI, A. M., KASIM, H., ARIS, H. et AL-HADA, N. M. (2022). Can electric vehicles be an alternative for traditional fossil-fuel cars with the help of renewable energy sources towards energy sustainability achievement ? *Springer*.
- [7] BAJADA, K. (2024). The aerodynamics of a generic suv : from optimization to general trends.
- [8] BIRD, R. B., STEWART, W. E. et LIGHTFOOT, E. N. (2002). *Transport Phenomena*. Wiley, 2nd édition.
- [9] BLUNDELL, C., CORNEBISE, J., KAVUKCUOGLU, K. et WIERSTRA, D. (2015). Weight uncertainty in neural networks. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.
- [10] CHANG, A. X., FUNKHOUSER, T., GUIBAS, L., HANRAHAN, P., HUANG, Q., LI, Z., SAVARESE, S., SAVVA, M., SONG, S., SU, H., XIAO, J., YI, L. et YU, F. (2015). Shapenet : An information-rich 3d model repository. *arXiv preprint arXiv :1512.03012*.
- [11] CHEN, Q., ELREFAIE, M., DAI, A. et AHMED, F. (2025). TripNet : Learning large-scale high-fidelity 3d car aerodynamics with triplane networks. In *ICML (Conference track)*. Preprint : <https://arxiv.org/abs/2503.17400>.

- [12] CHOY, C. e. a. (2025). Figconvnet : Fine-grained inference graph convolutional networks. Unpublished work.
- [13] EL HASADI, Y. et PADDING, J. (2023). A generalized model for predicting the drag coefficient of arbitrary bluff shaped bodies at high reynolds numbers. *Chemical Engineering Science*. arXiv preprint.
- [14] ELREFAIE, M., DAI, A. et AHMED, F. (2025). Drivaernet : A parametric car dataset for data-driven aerodynamic design and prediction.
- [15] GAL, Y. et GHAHRAMANI, Z. (2016). Dropout as a bayesian approximation : Representing model uncertainty in deep learning. *International Conference on Machine Learning (ICML)*.
- [16] GOODFELLOW, I., BENGIO, Y. et COURVILLE, A. (2016). *Deep Learning*. MIT Press.
- [17] GROUEIX, T., FISHER, M., KIM, V. G., RUSSELL, B. et AUBRY, M. (2018). Atlasnet : A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 216–224.
- [18] HAN, J., KAMBER, M. et PEI, J. (2011). *Data Mining : Concepts and Techniques*. Elsevier, 3rd édition.
- [19] HARRIS, C. R., MILLMAN, K. J., van der WALT, S. J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., WIESER, E., TAYLOR, J., BERG, S., SMITH, N. J. et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.
- [20] HASTIE, T., TIBSHIRANI, R. et FRIEDMAN, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer, 2nd édition.
- [21] HATCH, A. M. et SIMS-WILLIAMS, D. B. (2006). Flowfield study of fastback automobile shapes. *SAE Technical Paper*, (2006-01-0809).
- [22] HUANG, L., QIN, J., ZHOU, Y., ZHU, F., LIU, L. et SHAO, L. (2020). Normalization techniques in training dnns :methodology, analysis and application. *Arxiv*.
- [23] HUCHO, W.-H. (1998). *Aerodynamics of Road Vehicles*. SAE International, 4th édition.
- [24] HUNG TRAN, T., HIJKURO, M., ANYOJI, M., UCHIDA, T., NAKASHIMA, T. et SHIMIZU, K. (2023). Surface flow and aerodynamic drag of ahmed body with deflectors. *Experimental Thermal and Fluid Science*.
- [25] IPCC (2021). Climate change 2021 : The physical science basis. <https://www.ipcc.ch/report/ar6/wg1/>. Sixth Assessment Report, Working Group I.

- [26] IPCC (2022). Chapter 10 : Transport. Working Group III, AR6. Transport sector accounts for roughly 15% of total GHG emissions and 23% of CO<sub>2</sub> from energy.
- [27] JOHNSTON, F. H. et al. (2021). Air pollution events from forest fires and the associated health burden in australia : a retrospective study. *The Lancet Planetary Health*, 5(7):e361–e370.
- [28] KATZ, J. (2016). *Automotive Aerodynamics*. John Wiley & Sons.
- [29] KENDALL, A. et GAL, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision ? *In Advances in neural information processing systems*, pages 5574–5584.
- [30] KIPF, T. N. et WELLING, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv :1609.02907*.
- [31] LAKSHMINARAYANAN, B., PRITZEL, A. et BLUNDELL, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
- [32] LEUSSINK, D. (2025). Honda scales back ev plans, focuses on hybrids. *Reuters*. Downgrades EV target to 20
- [33] LI, G., MULLER, M., THABET, A. et GHANEM, B. (2019). Deepgcns : Can gcns go as deep as cnns ? *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9267–9276.
- [34] MADDOX, W. J., GARIFOV, T., IZMAILOV, P., VETROV, D. et WILSON, A. G. (2019). A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.
- [35] MUNSON, Okiishi, H. (2020). *Fox and McDonald's Introduction to Fluid Mechanics*. Wiley, 10th édition.
- [36] PANTON, R. L. (2013). *Incompressible Flow*. Wiley, 4th édition.
- [37] PARK, J. J., FLORENCE, P., STRAUB, J., NEWCOMBE, R. et LOVEGROVE, S. (2019). Deepsdf : Learning continuous signed distance functions for shape representation.
- [38] PFAFF, T., FORTUNATO, M., SANCHEZ-GONZALEZ, A. et BATTAGLIA, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv :2010.03409*.
- [39] QI, C. R., SU, H., MO, K. et GUIBAS, L. J. (2017a). Pointnet : Deep learning on point sets for 3d classification and segmentation. *Arxiv*.

- [40] QI, C. R., YI, L., SU, H. et GUIBAS, L. J. (2017b). Pointnet++ : Deep hierarchical feature learning on point sets in a metric space. *In Advances in neural information processing systems*, pages 5099–5108.
- [41] QIAN, Y., GOJCIC, Z., WIESER, A. et BIRDAL, T. (2022). Pointnext : Revisiting pointnet++ with improved training and scaling strategies. *In Advances in Neural Information Processing Systems*, volume 35, pages 22955–22967.
- [42] REUTERS (2022). Toyota chief lobbied government to favour hybrids over evs. *Reuters*. Toyota pressed for greater emphasis on hybrid vehicles.
- [43] SCHEWE, G. (2000). Reynolds-number effects in flow around more-or-less bluff bodies. *Wind Engineering / Fluid Mechanics experiments. Tests à haut Reynolds sur corps bluff.*
- [44] SCIENCE DAILY (2024). Japan's ev policy may only reduce 1050
- [45] SOVRAN, G. et BOHN, M. (1981). Formulae for the aerodynamic drag of automobiles. *SAE Technical Paper*, (810184).
- [46] SULLIVAN, C. R. et KASZYNSKI, A. (2019). Pyvista : 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk). *The Journal of Open Source Software*, 4(37):1450.
- [47] TANCIK, M., SRINIVASAN, P. P., MILDENHALL, B., FRIDOVICH-KEIL, S., RA-GHAVAN, N., SINGHAL, U., RAMAMOORTHI, R., BARRON, J. T. et NG, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *In Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 7537–7547.
- [48] TATARCHENKO, M., DOSOVITSKIY, A. et BROX, T. (2018). Octree generating networks : Efficient convolutional architectures for high-resolution 3d outputs. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3857–3866.
- [49] UNESCO/ONU (2025). Glacier melt threatens water supplies for two billion people. UN World Water Development Report / Carbon Brief. Receding snow and ice in mountain regions severely affect water and food security.
- [50] VAN EEDEN, L. M., EHMKE, G. et et AL. (2020). Animal mortality during the 2019–2020 australian bushfires : a review. *WWF Australia Report*. <https://www.wwf.org.au/news/news/2020/3-billion-animals-impacted-by-australian-bushfires>.
- [51] WANG, Y., SUN, Y., LIU, Z., SARMA, S. E., BRONSTEIN, M. M. et SOLOMON, J. M. (2019). Dynamic graph cnn for learning on point clouds. *In ACM Transactions on Graphics (TOG)*, volume 38, pages 1–12. ACM.

- [52] WU, Z., SONG, S., KHOSLA, A., YU, F., ZHANG, L., TANG, X. et XIAO, J. (2015). 3d shapenets : A deep representation for volumetric shapes. *In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1912–1920.
- [53] ZHOU, Q.-Y., PARK, J. et KOLTUN, V. (2018). Open3D : A modern library for 3D data processing. *arXiv :1801.09847*.
- [54] Álvaro VERGARA, WEI, D. et FUENTES, R. (2023). Drag coefficient for irregularly shaped grains : rotational dependence at high reynolds numbers. *arXiv preprint arXiv :2308.05272*.

## Annexe A

### Mécanique des fluides et aérodynamique

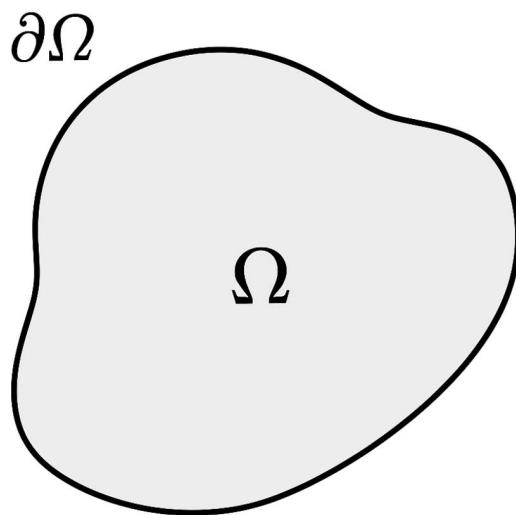


FIGURE 11 – Volume de contrôle

#### 0.1 Démonstration Equation de Navier-Stockes

Le théorème de Reynolds donne :

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{v} \, d\Omega = \int_{\Omega} \frac{\partial(p\mathbf{v})}{\partial t} \, d\Omega + \int_{\partial\Omega} \rho \mathbf{v} (\mathbf{v} \cdot \mathbf{n}) \, dS$$

Et on a de plus :

$$\mathbf{T} = \boldsymbol{\sigma} \cdot \mathbf{n}$$

Ainsi, le théorème de la divergence donne :

$$\int_{\partial\Omega} \mathbf{T} \, dS = \int_{\Omega} \nabla \cdot \boldsymbol{\sigma} \, d\Omega$$

et

$$\int_{\partial\Omega} \rho \mathbf{v}(\mathbf{v} \cdot \mathbf{n}) dS = \int_{\Omega} \nabla(\rho \mathbf{v} \otimes \mathbf{v}) d\Omega$$

Qui devient dans le cadre d'un fluide à densité constante :

$$\int_{\Omega} \rho \mathbf{v} \nabla \mathbf{v} d\Omega$$

On a donc sous forme locale :

$$\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \nabla \mathbf{v} \right) = \mathbf{f}_v + \nabla \boldsymbol{\sigma}$$

Or, la loi de comportement donne :

$$\boldsymbol{\sigma} = -p \mathbf{1} + \boldsymbol{\tau}$$

avec  $\boldsymbol{\tau} = \mu(\nabla \mathbf{v} + \nabla^T \mathbf{v})$  le tenseur des efforts visqueux.

On a donc :

$$\nabla \boldsymbol{\sigma} = \nabla(-p \mathbf{1} + \boldsymbol{\tau})$$

On a de plus :

$$[\nabla(-p \mathbf{1})]_i = \partial_j(-p \delta_{ij}) = -\partial_j(p \delta_{ij})$$

Or :  $\delta_{ij} = 0$  si  $i \neq j$  donc :

$$\nabla \cdot (-p \mathbf{1}) = -\nabla p$$

On a enfin :

$$\nabla \boldsymbol{\tau} = \mu \nabla(\nabla \mathbf{v} + \nabla^T \mathbf{v})$$

Ce qui donne en indiciel :

$$(\nabla \boldsymbol{\tau})_i = \partial_j \tau_{ij} = \mu \partial_j (\partial_j v_i + \partial_i v_j) = \mu (\partial_j \partial_j v_i + \partial_j \partial_i v_j)$$

La vitesse étant de classe  $C^2$  (c'est-à-dire deux fois dérivable et de dérivées continues pour chaque variable) pour tout problème bien posé, on peut appliquer le théorème de **Schwartz** qui donne :

$$\partial_j \partial_i v_j = \partial_i \partial_j v_j$$

Cette écriture fait intervenir l'expression indicelle de  $\nabla \mathbf{v}$  égale à 0 pour masse volumique constante (**conservation de la masse** (\*)).

On obtient donc [36] :

$$(\nabla \boldsymbol{\tau})_i = \mu \partial_j \partial_j v_i = \mu (\Delta v)_i$$

Finalement, on obtient (en négligeant la force volumique) :

$$\boxed{\rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \mu \Delta \mathbf{v}} \quad (\text{N.S})$$

## 0.2 Couche limite

Lorsqu'un fluide se déplace autour d'un corps  $\Omega$  solide, l'écoulement est perturbé par l'interaction qu'il a avec l'obstacle. On remarque que le fluide en contact avec  $\partial\Omega$  a une vitesse nulle à cause des frottements visqueux avec la paroi. En s'éloignant de la surface, l'interaction du fluide avec  $\partial\Omega$  devient inexisteante de sorte que le fluide tend vers une vitesse constante  $U_e$ . On définit la **couche limite** comme la zone de fluide où  $U \leq U_e$ . La couche limite se décompose souvent en trois zones distinctes :

- La **couche limite laminaire** : l'écoulement présent dans cette zone est laminaire, la convection du fluide est faible, l'écoulement est ordonné.
- La **phase de transition** : l'écoulement fluide contenu dans la couche limite passe d'un état laminaire à turbulent.
- La **couche limite turbulente** : L'écoulement contenu dans la couche limite devient turbulent : la convection du fluide contenu dans la couche devient élevée.

Cette transition entre régime laminaire et turbulent s'explique par les **aspérités** présentes sur la paroi. Plus celle-ci est **rugueuse**, plus l'écoulement atteint un régime turbulent rapidement. On note  $\delta(x)$  l'épaisseur de la couche limite à une position  $x$ , c'est-à-dire la distance nécessaire au fluide pour atteindre sa vitesse  $U_e$  à une position  $x$ . La couche limite turbulente possède une épaisseur supérieure à celle de la couche limite laminaire. En effet, les turbulences, bien qu'elles permettent une augmentation rapide et soudaine de la vitesse du fluide en s'éloignant de la paroi vont également le freiner lorsqu'on s'en éloigne progressivement, c'est un phénomène dit : '**d'échange d'élan bilatéral**'. À l'inverse, le gradient de vitesse dans la couche limite laminaire est faible mais permet une augmentation continue de celle-ci, créant une couche limite plus fine.

On peut déterminer la nature d'une couche limite quantitativement grâce au **nombre de Reynolds** :

$$R_e(x) = \frac{\rho U_e \delta(x)}{\mu}$$

si  $R_e < R_{lim}$  : la couche limite est laminaire

si  $R_e \approx R_{lim}$  : la couche limite n'est pas clairement définie, il s'agit de la zone de transition.

si  $R_e > R_{lim}$  : la couche limite est turbulente.

## 0.3 Décollement de couche limite, surpression et dépression

A l'échelle **microscopique**, la pression exercée par un fluide sur une paroi correspond à la **quantité de mouvement** transférée par le fluide sur la paroi grâce aux impacts des

particules de ce fluide sur cette même surface. Ainsi, les surfaces exposées à des flux d'air importants avec des **vitesses élevées** sont souvent des zones de **surpression** d'un véhicule.

On a vu précédemment que la couche limite est la zone proche de la surface où la vitesse de l'écoulement est ralentie de par l'interaction qu'a le fluide avec la surface. Bien qu'on puisse croire que la couche limite suit nécessairement les contours de la surface, on observe en pratique que les couches limites ont parfois tendance à s'en éloigner, c'est ce qu'on appelle un **décollement de couche limite**. On observe deux types de décollements :

- **Le décollement inertiel** : Survient lorsque le corps possède une **discontinuité géométrique** forte. Si le fluide possède une **inertie** trop importante localement, il ne lui est pas possible de suivre la courbure du corps. On peut alors observer la création d'une **poche d'eau morte** entre la surface et la couche limite décollée. Ces zones sont caractérisées par une **faible vitesse** du fluide, une **recirculation** du fluide, pouvant créer des **tourbillons**, ainsi qu'une forte **dépression**.

- **Le décollement par gradient de pression** : On a vu que la vitesse d'écoulement d'un fluide proche d'une surface est faible (d'autant plus si la couche limite est laminaire) à cause de l'interaction 'visqueuse' qu'a le fluide avec le corps (**force de cisaillement**). Si la pression augmente dans le sens de l'écoulement, c'est-à-dire que :  $\frac{\partial p}{\partial x} > 0$  (**reverse pressure**), alors l'énergie cinétique du fluide devient trop faible pour contrer ce gradient de pression de sorte que l'écoulement change de direction : on a une zone de recirculation et donc une chute de pression.

# Annexe B

## 1 Apprentissage Automatique

### 1.1 Présentation de l'algorithme du scheduler

---

#### Algorithme 1 : Scheduler ReduceLROnPlateau

---

**Entrée :** Learning rate initial  $\eta_0$ , facteur de réduction  $\gamma < 1$ , patience  $p$ ,

compteur  $c = 0$ , meilleure perte  $\mathcal{L}_{\min} = +\infty$

**Sortie :** Nouveau learning rate  $\eta$

```
1 foreach epoch do
2   Calculer la perte de validation  $\mathcal{L}_{\text{val}}$ ;
3   if  $\mathcal{L}_{\text{val}} < \mathcal{L}_{\min}$  then
4      $\mathcal{L}_{\min} \leftarrow \mathcal{L}_{\text{val}}$ ;
5      $c \leftarrow 0$ ;
6   else
7      $c \leftarrow c + 1$ ;
8     if  $c \geq p$  then
9        $\eta \leftarrow \gamma \cdot \eta$ ;
10       $c \leftarrow 0$ ;
```

---

### 1.2 Loss Eikonal

La *loss eikonale* est un terme de régularisation directement inspiré de l'**équation eikonale**, une équation aux dérivées partielles largement utilisée en optique géométrique et en traitement du signal pour modéliser la propagation de fronts d'onde à vitesse constante. Cette équation s'écrit sous la forme :

$$\|\nabla\phi(\mathbf{x})\| = 1$$

où  $\phi(\mathbf{x})$  est une fonction scalaire (par exemple une fonction de distance ou un temps d'arrivée), et  $\nabla\phi$  désigne son gradient. Dans le contexte des distances signées (SDF), cette équation traduit le fait que la SDF idéale doit vérifier localement une norme unitaire du gradient, c'est-à-dire :  $\|\nabla\text{SDF}(\mathbf{x})\| = 1$ .

Ce critère est utilisé comme contrainte dans l'apprentissage de réseaux de type implicite, afin de garantir une consistance géométrique de la fonction apprise. On introduit alors un terme de régularisation dans la fonction de coût totale :

$$\mathcal{L}_{\text{eikonal}} = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left( \left\| \nabla_{\mathbf{x}} \hat{\text{SDF}}(\mathbf{x}) \right\|_2 - 1 \right)^2$$

où  $\hat{\text{SDF}}(\mathbf{x})$  est la prédiction du réseau pour un point  $\mathbf{x}$  de l'espace  $\mathcal{X}$ . Ce terme pénalise toute déviation de la norme du gradient par rapport à l'unité, et agit ainsi comme une régularisation favorisant la plausibilité géométrique de la reconstruction.

### 1.2.1 Demonstration

On rappelle que la SDF dérive de la distance Euclidienne avec pour expression :

$$\text{SDF}(\mathbf{x}) = \begin{cases} -\text{dist}(\mathbf{x}, \partial\Omega) & \text{si } \mathbf{x} \in \Omega \\ +\text{dist}(\mathbf{x}, \partial\Omega) & \text{sinon} \end{cases}$$

avec  $\text{dist}$  faisant référence à la distance Euclidienne et  $\Omega$  un corps quelconque. On a ainsi :

$$\text{SDF}(\mathbf{x}) = \begin{cases} -\sqrt{(x - x_H)^2 + (y - y_H)^2 + (z - z_H)^2} & \text{si } \mathbf{x} \in \Omega \\ +\sqrt{(x - x_H)^2 + (y - y_H)^2 + (z - z_H)^2} & \text{sinon} \end{cases}$$

Avec  $\mathbf{x}_H$  faisant référence au plus proche point de  $\mathbf{x}$  dans  $\partial\Omega$ . On a alors :

$$\nabla \text{SDF}(\mathbf{x}) = \begin{bmatrix} \frac{(x - x_H)}{\text{SDF}(\mathbf{x}, \Omega)} \\ \frac{(y - y_H)}{\text{SDF}(\mathbf{x}, \Omega)} \\ \frac{(z - z_H)}{\text{SDF}(\mathbf{x}, \Omega)} \end{bmatrix}$$

Soit :

$$\|\nabla \text{SDF}(\mathbf{x}, \Omega)\|_2 = \sqrt{\nabla \text{SDF}(\mathbf{x}, \Omega) \cdot \nabla \text{SDF}(\mathbf{x}, \Omega)} = \sqrt{\frac{(x - x_H)^2 + (y - y_H)^2 + (z - z_H)^2}{\text{SDF}^2}} = 1$$

## Analyse en Composantes Principales (PCA)

L'**Analyse en Composantes Principales** (ou **PCA** pour *Principal Component Analysis*) est une méthode de réduction de dimension largement utilisée pour analyser et visualiser des données multivariées. Elle consiste à projeter les données initiales dans un nouvel espace orthogonal de dimension plus faible, tout en conservant un maximum de variance.

Soit une matrice de données  $\mathbf{X} \in \mathbb{R}^{n \times d}$ , contenant  $n$  échantillons de dimension  $d$ . La PCA repose sur les étapes suivantes :

- **Centrage** des données : soustraction de la moyenne sur chaque variable.

-**Calcul de la matrice de covariance :**

$$\mathbf{C} = \frac{1}{n} \mathbf{X}^\top \mathbf{X}$$

-**Diagonalisation** de la matrice de covariance : on calcule les vecteurs propres (axes principaux) et les valeurs propres associées (variances expliquées).

-**Projection** des données sur les  $k$  premiers vecteurs propres, correspondant aux plus grandes valeurs propres.

Le résultat est un nouvel ensemble de variables non corrélées (les composantes principales), classées par ordre décroissant de variance expliquée.

**Remarques :**

- La somme des valeurs propres représente la variance totale des données.
- En sélectionnant les premières composantes (celles dont les variances sont les plus élevées), on conserve l'information la plus significative.
- La PCA est sensible à l'échelle des variables ; il est souvent nécessaire de normaliser les données au préalable.

## Corrélation de Spearman

La **corrélation de Spearman** (notée  $s$  en Chapitre 7) est une mesure non paramétrique du degré de monotonie entre deux variables aléatoires. Contrairement à la corrélation de Pearson, elle ne mesure pas la linéarité, mais la force et la direction d'une relation monotone entre deux séries de données.

Elle s'appuie sur les **rangs** des valeurs plutôt que sur leurs valeurs numériques absolues. Elle est définie comme le coefficient de corrélation de Pearson entre les rangs des données.

Soient deux séries de données  $(x_i)_{i=1}^n$  et  $(y_i)_{i=1}^n$ , on note  $rg(x_i)$  et  $rg(y_i)$  leurs rangs respectifs dans leurs ensembles. Alors :

$$s = \frac{\text{cov}(rg(x), rg(y))}{\sigma_{rg(x)} \sigma_{rg(y)}}$$

Dans le cas sans ex-aequo, on peut utiliser la formule simplifiée :

$$s = 1 - \frac{6 \sum_{i=1}^n d_i^2}{n(n^2 - 1)} \quad \text{où} \quad d_i = rg(x_i) - rg(y_i)$$

**Propriétés :**

- $s \in [-1, 1]$ .
- $s = 1$  : relation strictement croissante.
- $s = -1$  : relation strictement décroissante.
- $s \approx 0$  : absence de relation monotone.
- Elle est robuste aux valeurs aberrantes et aux non-linéarités.

