

# Prédiction de l'énergie moléculaire

Apprentissage sous contraintes physiques

Les Marseillaises  
Ines BESBES, Sara ROOL

5 ModIA

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Description des données</b>	<b>2</b>
2.1	Pré-traitement des données . . . . .	2
2.1.1	Extraction des caractéristiques . . . . .	2
2.1.2	Chargement et structuration des données . . . . .	2
2.1.3	Intégration des énergies . . . . .	2
2.2	Visualisation des données . . . . .	2
2.2.1	En 2D . . . . .	3
2.2.2	En 3D . . . . .	3
2.3	Invariance par translation . . . . .	3
2.4	Visualisations exploratoires du jeu de données moléculaires . . . . .	4
<b>3</b>	<b>Description de la métrique</b>	<b>5</b>
<b>4</b>	<b>Matrice de Coulomb</b>	<b>6</b>
4.1	Description d'une matrice de Coulomb . . . . .	6
4.1.1	Formule . . . . .	6
4.1.2	Analyse des invariances . . . . .	6
4.1.3	Représentation visuelle . . . . .	7
4.2	Apprentissage . . . . .	8
4.2.1	Régression linéaire . . . . .	8
4.2.2	Random forest . . . . .	9
4.2.3	Réseau de neurones . . . . .	9
4.3	Analyse des résultats . . . . .	12
<b>5</b>	<b>Scattering</b>	<b>13</b>
5.1	Définition du scattering . . . . .	13
5.2	Densités électroniques approchées . . . . .	13
5.3	Transformée par ondelettes harmoniques solides . . . . .	13
5.4	Mise en oeuvre numérique . . . . .	14
5.4.1	Calcul des coefficients de scattering . . . . .	14
5.4.2	Sélection du meilleur modèle de régression . . . . .	14
5.5	Combinaison des descripteurs: scattering, matrices de Coulomb et attributs géométriques	14
5.5.1	Descripteurs issus de la transformée de scattering harmonique solide . . . . .	14
5.5.2	Ajout de caractéristiques géométriques simples . . . . .	15
5.5.3	Intégration de la matrice de Coulomb . . . . .	15
5.5.4	Combinaisons testées . . . . .	15
5.6	Résultats obtenus . . . . .	16
5.7	Analyse des résultats . . . . .	16
<b>6</b>	<b>Conclusion et piste d'amélioration</b>	<b>18</b>

# 1 Introduction

Ce projet vise à développer un modèle capable de prédire avec précision l'énergie d'atomisation  $E(\mathbf{r})$  de petites molécules organiques. Le défi principal réside dans le respect des contraintes de symétrie, notamment les invariances par translation, rotation et permutation des atomes dans la molécule.

On utilisera un sous-ensemble du jeu de données QM7-X, contenant plusieurs structures moléculaires avec des informations sur les positions atomiques et les énergies d'atomisation associées.

Ce rapport décrit les étapes de prétraitement des données, la modélisation et l'analyse des résultats obtenus par les différents modèles.

## 2 Description des données

Le jeu de données utilisé pour ce projet est un sous-ensemble de QM7-X, qui comprend 8236 structures de petites molécules organiques (6591 d'entraînement et 1645 de test). Chaque structure est décrite par les positions atomiques en trois dimensions et des informations supplémentaires sur les types d'atomes présents. Les données sont organisées en deux dossiers principaux : **atoms** et **energies**.

Le dossier **atoms** contient les configurations moléculaires au format xyz, séparées en ensembles d'entraînement et de test. Le dossier **energies** contient un fichier CSV avec deux colonnes : **id** et **energy**, où **energy** représente l'énergie d'atomisation de chaque molécule dans l'ensemble d'entraînement.

L'objectif principal est de produire un fichier CSV similaire pour l'ensemble de test, contenant les énergies d'atomisation prédites par un modèle développé.

### 2.1 Pré-traitement des données

Afin de préparer les données à l'analyse et à la modélisation, on réalise plusieurs étapes de prétraitement. Les données brutes des fichiers xyz sont lues et transformées en un format structuré pour faciliter la manipulation des données.

#### 2.1.1 Extraction des caractéristiques

Afin d'extraire les positions atomiques, les types d'atomes et les charges de chaque fichier xyz, on a développé la fonction `extract_features` à l'aide de la bibliothèque `ase`. Chaque molécule est ainsi représentée par une matrice de positions atomiques et un vecteur de charges atomiques. Pour uniformiser les données, chaque molécule a été complétée avec des zéros pour atteindre un nombre maximal de 23 atomes. Cela permet d'être sûr qu'on ait une taille constante pour chaque entrée.

#### 2.1.2 Chargement et structuration des données

De plus, la fonction `load_all_xyz` a été développée pour charger et trier les fichiers xyz par identifiant, correspondant à l'ordre des énergies dans le fichier CSV. Les positions et les charges ont été stockées dans des tableaux NumPy de dimensions  $(N, \text{max\_atoms}, 3)$  pour les positions et  $(N, \text{max\_atoms})$  pour les charges, où  $N$  est le nombre total de molécules.

#### 2.1.3 Intégration des énergies

Les énergies d'atomisation ont été lues à partir du fichier CSV et associées aux molécules correspondantes. Les identifiants des molécules ont été convertis en chaînes de caractères et triés pour garantir l'alignement correct avec les données de positions et de charges.

## 2.2 Visualisation des données

On va à présent visualiser les molécules pour mieux comprendre leur structure et leur distribution des atomes associée.

### 2.2.1 En 2D

Dans un premier temps, on a réalisé des visualisations en 2D. Trois molécules ont été sélectionnées aléatoirement à partir de l'ensemble de données d'entraînement et visualisées en utilisant la bibliothèque `matplotlib`.

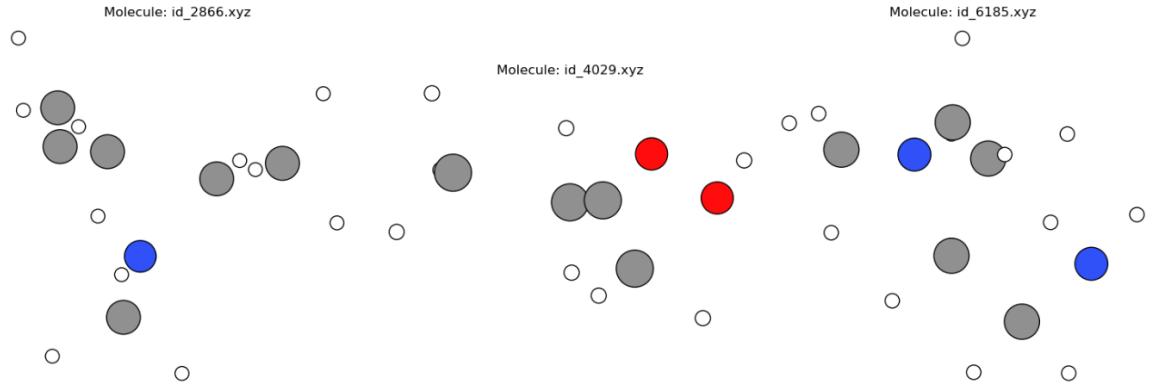


Figure 1: Visualisation 2D de trois molécules sélectionnées aléatoirement. Les atomes sont représentés par des cercles de différentes couleurs et tailles.

### 2.2.2 En 3D

On a également développé des visualisations en 3D afin de mieux comprendre les positions relatives des atomes dans l'espace tridimensionnel.

3D Molecule #4029

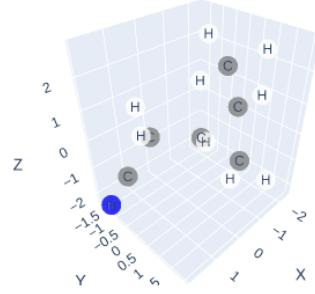


Figure 2: Visualisation 3D d'une molécule montrant les positions des atomes dans l'espace.

Cette visualisation est importante pour pouvoir observer les structures moléculaires et vérifier la conformité des modèles aux contraintes de symétrie.

## 2.3 Invariance par translation

L'invariance par translation est une propriété fondamentale des molécules qui doit être respectée par les modèles de prédiction d'énergie. Cela signifie que l'énergie d'une molécule ne doit pas changer si la molécule est translatée dans l'espace. Pour illustrer cette propriété, on a visualisé une molécule dans sa configuration originale ainsi que dans des configurations translatées verticalement et horizontalement.

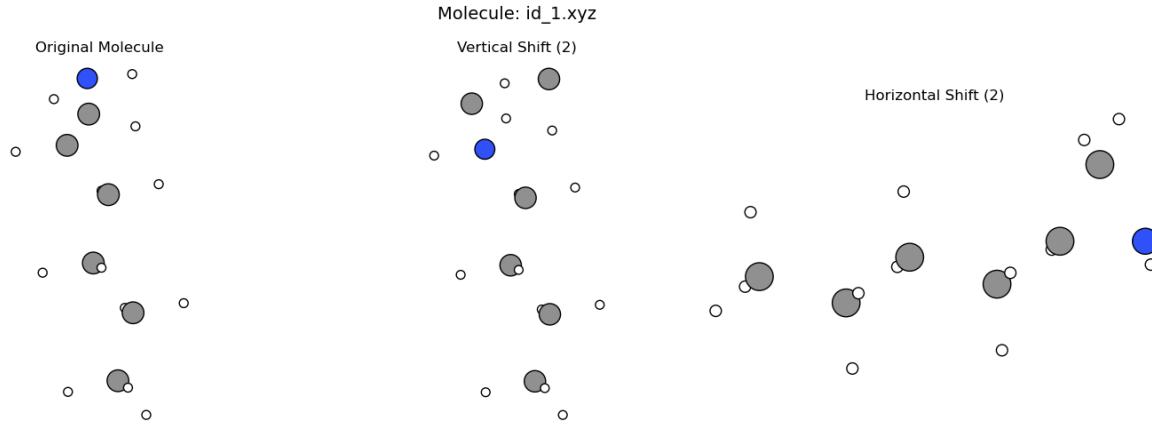


Figure 3: Illustration de l'invariance par translation d'une molécule.

## 2.4 Visualisations exploratoires du jeu de données moléculaires

Pour mieux comprendre les caractéristiques du jeu de données, on a réalisé plusieurs visualisations exploratoires. Les graphiques ci-dessous permettent d'analyser différentes propriétés des molécules, telles que le nombre d'atomes, la distribution des numéros atomiques, le numéro atomique moyen par molécule, et la distance moyenne des atomes par rapport au centroïde de la molécule.

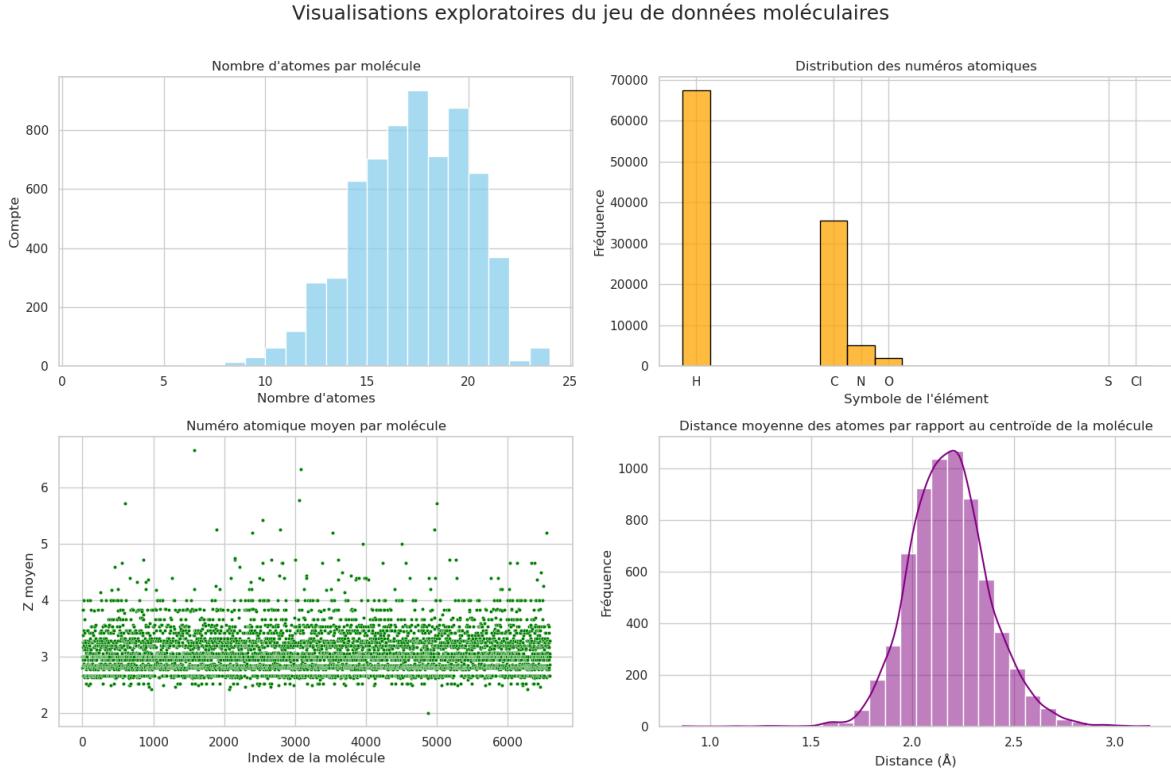


Figure 4: Visualisations exploratoires du jeu de données moléculaires.

Le graphique en haut à gauche de la figure 4 montre la distribution du nombre d'atomes par molécule. On observe que la majorité des molécules contiennent entre 10 et 23 atomes, avec un pic notable autour de 17 à 20 atomes.

D'autre part, on observe dans le graphique en haut à droite montre que l'hydrogène (H) est de loin l'atome le plus fréquent, suivi par le carbone (C). De plus, cela se confirme dans le graphique en bas à

gauche qui montre que la plupart des molécules ont un numéro atomique moyen autour de 3 à 5. Ceci est donc cohérent avec la prédominance des atomes de carbone et d'hydrogène.

Enfin, on observe dans le graphique en bas à droite que la majorité des distances moyennes se situent entre 1.0 et 2.5 Ångströms ( $\text{\AA}$ ), ce qui donne une indication de la taille et de la compacité des molécules dans le jeu de données.

### 3 Description de la métrique

Pour évaluer la performance des modèles de prédiction de l'énergie d'atomisation des molécules, on va utiliser l'erreur quadratique moyenne (Root Mean Square Error, RMSE). Cette métrique permet de mesurer l'écart entre les valeurs prédites par le modèle et les valeurs réelles des énergies d'atomisation.

La RMSE est définie comme suit : soit  $\tilde{E}(\mathbf{r}_{id})$  la prédiction du modèle pour une configuration moléculaire  $\mathbf{r}_{id}$ , et  $E(\mathbf{r}_{id})$  l'énergie réelle correspondante. Pour un ensemble de test contenant  $D$  configurations moléculaires  $\{\mathbf{r}_{id}\}_{id \leq D}$ , l'erreur est calculée par :

$$\text{RMSE} = \sqrt{\frac{1}{D} \sum_{id=1}^D (E(\mathbf{r}_{id}) - \tilde{E}(\mathbf{r}_{id}))^2}$$

## 4 Matrice de Coulomb

### 4.1 Description d'une matrice de Coulomb

#### 4.1.1 Formule

Une matrice de Coulomb est une matrice symétrique carrée de taille  $N \times N$ ,  $N$  étant le nombre d'atomes, définie par la formule ci-dessous:

$$M_{ij}^{Coulomb} = \begin{cases} 0.5Z_i^{2.4} & \text{si } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{si } i \neq j \end{cases}$$

avec  $R_{ij} = \|R_i - R_j\|$  la distance entre les noyaux atomiques  $i$  et  $j$ . Et  $Z_i$  le nombres de charge nucléaire de l'atome  $i$ . Chaque élément de la matrice encode une interaction électrostatique entre deux atomes.

Les éléments situés sur la diagonale correspondent à l'interaction d'un atome avec lui-même. C'est une approximation de l'énergie d'un atome isolé.

Les éléments hors diagonaux représentent l'interaction électrostatique entre les noyaux atomiques  $i$  et  $j$ , selon la loi de Coulomb. C'est une mesure de l'attraction/répulsion entre atomes, en tenant compte de leur charge et distance.

#### 4.1.2 Analyse des invariances

Une matrice de Coulomb est invariante par translation et par rotation :

- Translation: la distance  $R_{ij}$  ne change pas si on translate toute la molécule dans l'espace,
- Rotation: les distances entre paires d'atomes sont également invariantes par rotation.

Ces deux invariances garantissent que les valeurs de la matrice ne dépendent pas de la position absolue de la molécule ce qui est essentiel pour décrire une molécule.

Cependant, une matrice de Coulomb n'est pas invariante aux permutations d'atomes car la structure de la matrice dépend de l'ordre des atomes dans la liste d'entrée. Cela signifie que deux représentations identiques d'une même molécule peuvent aboutir à des matrices distinctes si l'ordre de liste des atomes diffère. C'est un problème car l'énergie est invariante à la permutation des atomes.

Pour résoudre ce problème, on propose deux solutions:

- On fait un tri canonique avec l'argument `sorted_12`. Ce tri permet de fixer un ordre canonique pour les atomes basé sur l'intensité globale de leurs interactions, ce qui rend la représentation indépendante de l'ordre initial des atomes.
- On utilise des permutations aléatoires de l'ordre des atomes dans la matrice de Coulomb. Cela permet de générer plusieurs représentations valides d'une même molécule, ce qui permet d'augmenter artificiellement la taille du jeu de données. On fait 5 permutations différentes par molécule.

Ainsi, la matrice devient invariante à la permutation des atomes d'entrée.

#### 4.1.3 Représentation visuelle

Voici la représentation (normalisée et non normalisée) de 4 molécules :

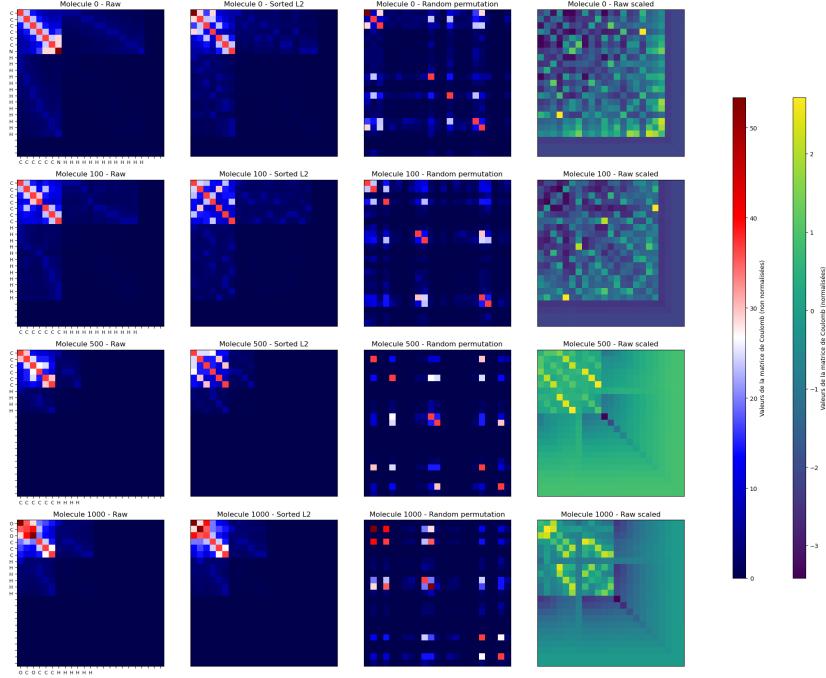


Figure 5: Représentation de matrices de Coulomb pour quatre molécules et selon différentes transformations.

La première colonne correspond à la matrice de Coulomb sans modification, c'est-à-dire qu'on conserve l'ordre d'entrée du fichier. Toutefois, on a vu précédemment que cette option n'est pas invariante à la permutation des atomes. La deuxième colonne est la version avec le tri canonique et la troisième colonne est la version avec un tri aléatoire (et augmentation du jeu de données). Ces représentations sont invariantes à la permutation des atomes. Et la dernière colonne est la version normalisée sur tout le jeu de données de la première colonne.

Pour la suite, on va conserver les matrices de Coulomb non normalisées pour préserver l'information physique. En effet, les valeurs absolues dans la matrice de Coulomb sont directement liées aux interactions électrostatiques entre les atomes, et elles contiennent l'échelle physique de l'énergie. De fait, normaliser ces valeurs risque de détruire cette échelle et par conséquent de rendre l'apprentissage plus compliqué.

De plus, dans notre cas, toutes les matrices ont la même taille avec à l'ajout de zéros (padding). Ces zéros n'ont pas de sens physique. Lorsque l'on normalise les données, cela peut donner trop d'importance aux valeurs non nulles, car la moyenne est réduite par les zéros. Cela peut aussi fausser la variance, car les zéros ajoutés peuvent diminuer ou exagérer l'importance des vraies interactions.

On verra par la suite que dans la pratique, des données normalisées peuvent nous donner de meilleurs résultats, même si la théorie nous suggère l'inverse.

## 4.2 Apprentissage

On considère dans cette partie différentes transformations pour les matrices de Coulomb:

- **X\_raw**: les matrices de Coulomb sans permutations ni tri,
- **X\_raw\_scaled**: la version normalisée de **X\_raw**,
- **X\_sorted**: les matrices de Coulomb avec un tri canonique,
- **X\_sorted\_scaled**: la version normalisée de **X\_sorted**,
- **X\_augmented**: les matrices de Coulomb avec 5 permutations aléatoires par molécule,
- **X\_augmented\_scaled**: la version normalisée de **X\_augmented**,

### 4.2.1 Régression linéaire

Dans cette partie, on applique deux types de régressions régularisées: Ridge et Lasso. De plus, même si on a expliqué précédemment qu'il est préférable de ne pas utiliser les matrices normalisées en raison de la perte potentielle d'information physique, on a tout de même voulu tester leurs effets. On obtient les résultats suivant:

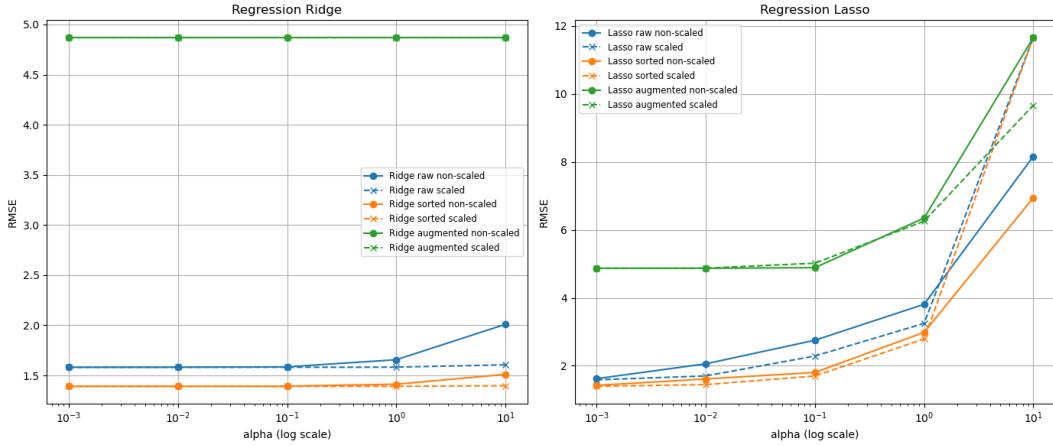


Figure 6: Résultats pour une régression Ridge et Lasso sur des matrices de Coulomb selon différentes transformations et différents  $\alpha$ .

De manière générale, la régression Ridge donne de meilleurs résultats que la régression Lasso. Les valeurs de RMSE se situent dans une plage plus restreinte, entre 1.4 et 2.0 selon la valeur de  $\alpha$ , ce qui traduit une bonne stabilité du modèle. Ceci n'est pas le cas pour **X\_augmented**.

De plus, on constate pour Ridge que les matrices normalisées produisent parfois une meilleure RMSE pour  $\alpha > 0.1$ . Cependant, pour de faibles valeurs de  $\alpha$ , la normalisation n'apporte pas de différence significative.

Toutefois, la régression Ridge échoue sur le jeu de données augmenté. La RMSE reste élevée, autour de 5, quelle que soit la valeur de  $\alpha$ . Ce résultat indique que la structure du jeu de données augmenté introduit une complexité ou un bruit que Ridge ne parvient pas à compenser.

Concernant la régression Lasso, les performances sont plus variables. La RMSE fluctue entre 1.5 et 12, selon les cas. Lasso donne de meilleurs résultats avec les matrices de Coulomb non transformées ou triées canoniquement, sans normalisation, pour de grands  $\alpha$ . Cependant, pour des petits  $\alpha$ , ce sont les matrices normalisées qui produisent de légers meilleurs résultats.

Ridge utilise la norme  $L_2$  pour réduire les coefficients sans les annuler. Cela permet de garder toutes les variables, ce qui est utile dans notre cas car les coefficients contiennent de l'information. Lasso utilise la norme  $L_1$  et peut annuler certains coefficients. Cela peut être un problème si aucune variable n'est vraiment inutile. C'est pour cela que Ridge donne de meilleurs résultats.

Il faut cependant souligner que ces résultats, bien que corrects sur le jeu d'entraînement, risquent de mal se généraliser sur un jeu de test, dû à la non invariance à la permutation des atomes, et à la perte d'information physique avec la normalisation des données.

Enfin, on constate que le meilleur modèle est obtenu avec une régression Ridge appliquée aux matrices de Coulomb triées canoniquement, non normalisées, et avec un  $\alpha = 0.01$ .

#### 4.2.2 Random forest

Les modèles de régression linéaire restent relativement simples. Dans cette section, on applique un régressseur de type Random Forest, un algorithme plus complexe capable potentiellement de capturer plus de dynamiques.

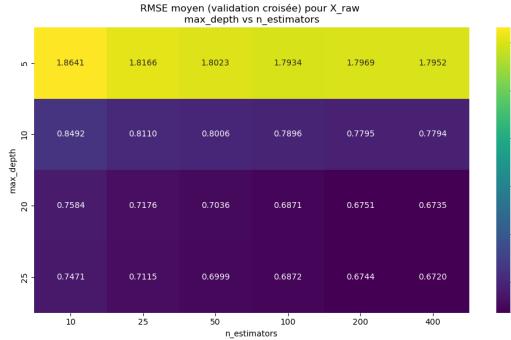


Figure 7: RMSE moyenne avec des forêts aléatoires entraînées sur `X_raw` selon différents hyperparamètres

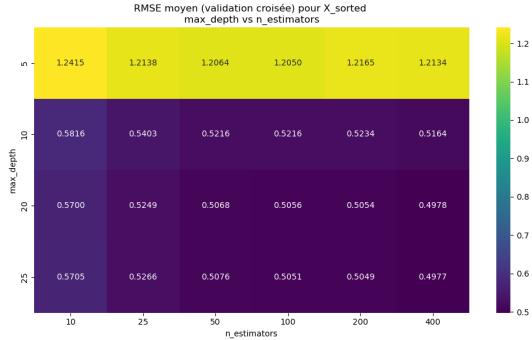


Figure 8: RMSE moyenne avec des forêts aléatoires entraînées sur `X_sorted` selon différents hyperparamètres

On décide de faire un apprentissage sur `X_raw`, malgré la non invariance par permutation.

#### Apprentissage pour `X_raw`:

On observe que plus `max_depth` est faible plus la RMSE est élevée, ce qui suggère un sous-apprentissage dû à une capacité de modélisation trop limitée. En augmentant donc `max_depth`, la RMSE diminue significativement, atteignant des valeurs proches de 0.67, ce qui traduit une meilleure adaptation du modèle aux données. De même, l'augmentation de `n_estimators` améliore globalement les performances, bien que les gains restents négligeables après 100 voir 200 estimateurs. Le meilleur résultat est obtenu avec `max_depth = 25` et `n_estimators = 400`, pour une RMSE minimale d'environ 0.672.

#### Apprentissage pour `X_sorted`:

De manière similaire, le modèle Random Forest montre une nette amélioration des performances lorsque `max_depth` dépasse 5. En effet, avec `max_depth = 5`, le RMSE reste élevé, traduisant un sous-apprentissage. En revanche, pour des profondeurs plus importantes (10 à 25), le modèle parvient à mieux capturer la structure des données, avec des RMSE nettement plus faibles.

On observe ici également que l'augmentation du nombre d'arbres, `n_estimators`, contribue généralement à une légère amélioration des résultats, bien que les gains deviennent négligeables au-delà de 100 estimateurs.

La combinaison optimale est `max_depth = 25` et `n_estimators = 400`, avec une  $RMSE \approx 0.4977$  en validation croisée.

On réentraîne les modèles pour avoir la RMSE sans validation croisée et on obtient  $RMSE_{raw} = 0.2251$  et  $RMSE_{sorted} = 0.1702$ . On constate de meilleurs résultats pour l'apprentissage fait avec `X_sorted`.

#### 4.2.3 Réseau de neurones

Dans cette partie, on souhaite réimplémenter le réseau de neurones multi-couches (MLP) proposé par Montavon et al. [1]

### **Transformation des matrices de Coulomb:**

L'article explique que chaque nombre de la matrice de Coulomb est important pour prédire la bonne sortie mais utiliser directement ces valeurs réelles comme entrée risque de conduire à un problème d'optimisation mal conditionné.

À la place, ils choisissent de décomposer chaque dimension de la matrice de Coulomb  $C$ , en convertissant la représentation en un tenseur 3D composé essentiellement de prédictats binaires.

$$x = \left[ \dots, \tanh\left(\frac{C - \theta}{\theta}\right), \tanh\left(\frac{C}{\theta}\right), \tanh\left(\frac{C + \theta}{\theta}\right), \dots \right]$$

Le vecteur  $x$  deviendrait l'entrée du modèle. Cette méthode de binarisation améliore la stabilité de l'apprentissage (meilleur conditionnement) et rend donc le modèle plus souple, donc potentiellement plus performant. Mais, d'après ce qu'ils expliquent, il faut suffisamment de données pour apprendre correctement à partir de cette représentation plus souple sinon les performances peuvent être moins bonnes. Dans l'article, ils proposent  $\theta = 1$ .

### **Architecture du réseau de neurones:**

Le réseau implémenté est un MLP comprenant :

- Une couche d'entrée de la taille de  $x$
- Deux couches cachées :
  - 400 neurones (activation sigmoïde)
  - 100 neurones (activation sigmoïde)
- Une couche de sortie donnant l'énergie

### **Paramètres initiaux et optimiseur:**

Les poids initiaux  $W_0$  et le learning rate  $\gamma$  sont instanciés respectivement à  $W_0 \sim \mathcal{N}(0, \frac{1}{\sqrt{m}})$  et  $\gamma = \frac{\gamma_0}{\sqrt{m}}$  avec  $m$  le nombre d'entrée et  $\gamma_0 = 0,01$ . De plus, ils normalisent les données d'entrée et les sorties de façon à avoir une moyenne nulle et un écart-type de 1. On essaiera également sans normaliser les données et/ou les sorties.

Concernant le choix de l'optimiseur, ils utilisent dans l'article une descente de gradient stochastique avec moyennage (ASGD). On essaiera également avec une descente de gradient stochastique basique (SGD).

### **Résultats obtenus pour l'entraînement:**

On entraîne le modèle pendant 1000 époques en testant différentes combinaisons d'optimiseurs et de techniques de transformation des données. Le jeu de données `X_sorted` contient 6591 échantillons, tandis que `X_augmented` en compte 32955, les deux jeux de données ont des entrées de tailles 1587 ( $23 \times 23 \times 3$ ). Pour la RMSE, on retransforme les sorties à leur échelle d'origine après la prédiction si elles ont été normalisées au préalable. On choisit une taille de batche de 25 comme dans l'article.

Transformation Coulomb	Entrées normalisées	Sorties normalisées	Optimiseur	RMSE d'entraînement
X_sorted	oui	oui	SGD	1.40417
X_sorted	oui	oui	ASGD	1.41009
X_sorted	oui	non	SGD	0.72356
X_sorted	oui	non	ASGD	0.72733
X_sorted	non	non	SGD	1.65289
X_sorted	non	non	ASGD	1.60336
X_augmented	oui	oui	SGD	1.49764
X_augmented	oui	oui	ASGD	1.52871
X_augmented	oui	non	SGD	0.24363
X_augmented	oui	non	ASGD	0.24126
X_augmented	non	non	SGD	1.44361
X_augmented	non	non	ASGD	1.16272

Table 1: Résultats RMSE de l’entraînement d’un réseau de neurones selon différents paramètres d’entrée

Les résultats des entraînements sont rassemblés dans la table ci-dessus.

Les plus faibles RMSE sont obtenues avec `X_augmented`, lorsque les entrées sont normalisées mais les sorties ne le sont pas. On obtient une RMSE de 0.24 pour les deux optimiseurs. Ces résultats sont particulièrement faibles comparés aux autres valeurs de RMSE aux alentours de 1.5 généralement.

Dans le cas des deux transformations, `X_sorted` et `X_augmented`, on observe que la normalisation des sorties a tendance à dégrader les performances. Par exemple, avec des entrées normalisées, le passage de sorties non normalisées à normalisées fait passer le RMSE de 0.72 à 1.40 pour `X_sorted` et de 0.24 à 1.49 pour `X_augmented`, soit une dégradation notable. Ceci est sûrement dû à une perte d’information physique contenue dans les sorties.

La transformation `X_augmented` fournit de meilleurs résultats que `X_sorted`. Cela suggère que l’augmentation des données apporte une meilleure capacité de généralisation aux données de test.

Enfin, l’optimiseur ASGD surpassé SGD dans plusieurs cas, en particulier lorsqu’il est combiné avec des données augmentées et des sorties non normalisées. Cela suggère que l’utilisation d’une moyenne des poids peut apporter plus de stabilité ou de précision à l’entraînement dans ce contexte. Cependant, il est important de noter que l’entraînement avec l’optimiseur ASGD prend beaucoup plus de temps, environ 1,5 à 2 fois plus long, que avec SGD.

Finalement, au vu du tableau ci dessus et du temps de calcul, on choisit les modèles avec entrées normalisées et sorties non normalisées avec un optimiseur SGD entraînés sur `X_sorted` et sur `X_augmented`.

### 4.3 Analyse des résultats

On considère dans cette partie différentes transformation pour le jeu de données de test:

- `X_test_raw`: les matrices de Coulomb des données de test sans permutations ni tri,
- `X_test_sorted`: les matrices de Coulomb des données de test avec un tri canonique,
- `X_test_augmented`: les matrices de Coulomb des données de test avec une permutation aléatoire.

On sélectionne les meilleurs modèles pour chaque technique d'apprentissage utilisée. Ensuite, on effectue des prédictions avec le jeu de test et on évalue la RMSE en soumettant les résultats sur Kaggle. On obtient le tableau suivant:

Modèle	Description du modèle	Transformation des données de test	RMSE de train	RMSE de test
Régression	Ridge avec $\alpha = 0.01$ et entraîné sur <code>X_raw</code> non normalisé	<code>X_test_raw</code> et non normalisé	1.58	1.657
Régression	Ridge avec $\alpha = 0.01$ et entraîné sur <code>X_sorted</code> non normalisé	<code>X_test_sorted</code> et non normalisé	1.39	1.486
Random forest	Entraîné sur <code>X_raw</code> avec <code>n_estimators = 400</code> et <code>max_depth = 25</code>	<code>X_test_raw</code> et non normalisé	0.225	0.626
Random forest	Entraîné sur <code>X_sorted</code> avec <code>n_estimators = 400</code> et <code>max_depth = 25</code>	<code>X_test_sorted</code> et non normalisé	0.170	0.497
Réseau de neurones	Entraîné sur <code>X_sorted</code> binarisé puis normalisé et l'optimiseur SGD	<code>X_test_sorted</code> binarisé puis normalisé selon les données d'entraînement	0.724	11.010
Réseau de neurones	Entraîné sur <code>X_augmented</code> binarisé puis normalisé et l'optimiseur SGD	<code>X_test_augmented</code> binarisé puis normalisé selon les données d'entraînement	0.243	2.929
Réseau de neurones	Entraîné sur <code>X_augmented</code> binarisé puis normalisé et l'optimiseur SGD	<code>X_test_raw</code> binarisé puis normalisé selon les données d'entraînement	0.243	2.466

Table 2: Table comparative des RMSE d'entraînement et de test pour différent modèle.

On observe que les modèles de régression Ridge avec un paramètre de régularisation  $\alpha = 0.01$  montrent des performances moyennes. Concernant le modèle entraîné `X_raw`, l'erreur sur les données de test est de 1.657. Cette erreur diminue à 1.486 lorsque les données sont triées, `X_sorted`, ce qui suggère une légère amélioration grâce au tri des variables.

D'autre part, on remarque que les modèles Random Forest sont bien plus précis que les régressions linéaires. Ils obtiennent des erreurs de test de 0.626 et 0.497 respectivement sur les données brutes et triées. Le tri améliore encore les performances, et cet effet est plus marqué pour les Random Forest, probablement parce que la structure des arbres profite d'un ordre plus logique dans les variables. Les faibles erreurs d'entraînement montrent que ces modèles s'ajustent bien aux données, mais la différence avec les erreurs de test indique un léger surapprentissage.

Enfin, les réseaux de neurones donnent des résultats très variables selon la transformation des données. Un réseau entraîné sur des données triées, binarisées et normalisées montre un fort surapprentissage, avec une faible erreur d'entraînement mais une très grande erreur de test. Cependant, l'ajout de données augmentées améliore significativement la généralisation, réduisant les erreurs de test à 2.929 et 2.466. On a testé le modèle entraîné sur des données invariantes avec des données ordonnées différemment pour vérifier sa robustesse, et cela semble fonctionner. Cela montre que l'augmentation des données et un bon prétraitement aident à mieux utiliser les réseaux de neurones, même si leurs performances restent inférieures à celles des forêts aléatoires sur ces données.

## 5 Scattering

### 5.1 Définition du scattering

La transformée de scattering est une méthode d'extraction de caractéristiques multirésolution basée sur des ondelettes. Elle permet de générer des descripteurs invariants aux translations et rotations, et stables aux petites déformations géométriques. Dans ce projet, on utilise la **transformée de scattering harmonique solide 3D** appliquée à des cartes de densité électronique.

Les coefficients de scattering sont calculés jusqu'à l'ordre 2, en utilisant différents paramètres ( $J$  et  $L$ ), et servent de base à la prédiction d'énergies moléculaires.

### 5.2 Densités électroniques approchées

Chaque molécule est représentée par plusieurs densités électroniques tridimensionnelles, construites à partir de gaussiennes centrées aux positions atomiques. On considère trois canaux:

- **Canal total** ( $\rho^{\text{tot}}$ ): encode l'identité atomique et reflète la structure complète.
- **Canal de valence** ( $\rho^{\text{val}}$ ): sensible à la réactivité chimique et aux liaisons.
- **Canal de cœur** ( $\rho^{\text{core}}$ ): encode l'organisation interne des noyaux atomiques, utile pour l'énergie atomique isolée.

Chaque densité s'écrit comme une somme de gaussiennes pondérées par la charge appropriée:

$$\rho_x(u) = \sum_k \gamma_k g(u - r_k)$$

où  $\gamma_k$  est la charge (totale, de valence ou de cœur) de l'atome  $k$ ,  $r_k$  sa position, et  $g$  une gaussienne tridimensionnelle de largeur  $\sigma$ .

### 5.3 Transformée par ondelettes harmoniques solides

Les ondelettes harmoniques solides sont définies par:

$$\psi_\ell^m(u) = \frac{1}{(2\pi)^{3/2}} e^{-\|u\|^2/2} \|u\|^\ell Y_\ell^m\left(\frac{u}{\|u\|}\right)$$

où  $Y_\ell^m$  est une fonction sphérique harmonique. Elles sont dilatées à l'échelle  $2^j$  par:

$$\psi_{j,\ell}^m(u) = 2^{-3j} \psi_\ell^m(2^{-j}u)$$

La transformée d'ordre 1 est obtenue via une convolution et une agrégation par norme euclidienne sur  $m$ :

$$U[j, \ell](u) = \left( \sum_{m=-\ell}^{\ell} |\rho * \psi_{j,\ell}^m(u)|^2 \right)^{1/2}$$

Puis, les **coefficients invariants** de premier ordre sont obtenus par intégration:

$$S[j, \ell, q] = \int_{\mathbb{R}^3} |U[j, \ell](u)|^q du$$

Pour capturer des interactions à plusieurs échelles, on calcule également les coefficients de second ordre:

$$U[j, j', \ell](u) = \left( \sum_{m=-\ell}^{\ell} |U[j, \ell] * \psi_{j', \ell}^m(u)|^2 \right)^{1/2}, \quad j' > j$$

$$S[j, j', \ell, q] = \int_{\mathbb{R}^3} |U[j, j', \ell](u)|^q du$$

## 5.4 Mise en oeuvre numérique

### 5.4.1 Calcul des coefficients de scattering

Le calcul des coefficients de scattering repose sur plusieurs étapes, décrites ci-dessous.

1. **Chargement des données:** les fichiers `.xyz` sont parcourus avec la fonction `load_all_xyz`, qui appelle en interne `extract_features` pour extraire les positions atomiques et les numéros atomiques (charges nucléaires).
2. **Construction des densités électroniques:** pour chaque molécule, les densités `full_batch`, `val_batch` et `core_batch` sont construites à l'aide de la fonction `generate_weighted_sum_of_gaussians` qui superpose des gaussiennes centrées sur chaque atome, pondérées par les charges.
3. **Discrétisation spatiale:** l'espace est discréteisé sur une grille  $M \times N \times O$ .
4. **Application de la transformée de scattering:** pour chaque canal (nuclear, valence, core), les coefficients de scattering d'ordre 0 sont calculés par `TorchBackend3D.compute_integrals(...)` et ceux d'ordre 1 et 2 par l'opérateur `HarmonicScattering3D(...)` défini avec `J`, `L`, `sigma` et les puissances intégrales.
5. **Traitements par lots:** la boucle principale itère par batchs de taille `batch_size=8` pour limiter la consommation mémoire sur GPU. Elle assemble les tenseurs `order_0` et `orders_1_and_2` au fil des itérations. Ils sont ensuite transférés en mémoire CPU, convertis en `numpy` puis aplatis et enregistrés au format `.npy`.
6. **Reproduction sur l'ensemble de test:** les mêmes étapes sont appliquées à l'ensemble de test.

Les coefficients de scattering pour les ensembles d'entraînement et de test sont concaténés avec:

```
train_scattering_coef = np.concatenate([order_0_train, orders_1_and_2_train], axis=1)
test_scattering_coef = np.concatenate([order_0_test, orders_1_and_2_test], axis=1)
```

### 5.4.2 Sélection du meilleur modèle de régression

Pour sélectionner le modèle le plus performant, on a codé la fonction `find_best_model` qui teste plusieurs modèles (`LinearRegression`, `Ridge`, `SVR`, `RandomForest`, `KNN`, `DecisionTree`, `GradientBoosting`, `XGBoost`), avec une validation croisée (par défaut  $K = 5$ ), en optimisant la RMSE. Le modèle retenu est celui ayant la plus petite RMSE. Il est ensuite ajusté avec `best_model.fit(X_train, y_train)`. Enfin, on utilise le modèle pour faire effectuer les prédictions de l'énergie moléculaire sur les données de test.

## 5.5 Combinaison des descripteurs: scattering, matrices de Coulomb et attributs géométriques

Afin d'améliorer la prédiction de l'énergie moléculaire, plusieurs représentations ont été étudiées, seules ou combinées.

### 5.5.1 Descripteurs issus de la transformée de scattering harmonique solide

Deux configurations de scattering ont été évaluées, en accord avec l'article original [2]:

- **Configuration 1 (J=2, L=3):** utilisée dans les expériences principales du projet. Paramètres: grille  $96 \times 64 \times 48$  et  $q \in \{0.5, 1, 2, 3\}$ .

- **Configuration 2 (J=4, L=3):** reproduit les réglages de l'article. Paramètres: grille identique et  $q \in \{0.5, 1, 2, 3, 4\}$ .

### 5.5.2 Ajout de caractéristiques géométriques simples

Pour enrichir la description moléculaire, des descripteurs géométriques sont extraits à partir des coordonnées atomiques et des charges nucléaires, via la fonction `compute_all_features`:

- Nombre total d'atomes et charge totale,
- Moyenne et écart-type des numéros atomiques,
- Moyenne et écart-type des distances interatomiques (`compute_pairwise_distance_stats`),
- Moyenne des distances au centre et variance spatiale (`compute_geometric_features`),
- Fréquences d'apparition des types atomiques : H, C, N, O, F, S (`compute_atomic_type_frequencies`),

Ces descripteurs ont été conçus pour être invariants vis-à-vis de plusieurs transformations géométriques, ce qui est crucial pour garantir que les représentations moléculaires ne dépendent pas du choix de coordonnées ni de l'ordre d'entrée des atomes.

Trois propriétés d'invariance ont été testées et vérifiées numériquement :

- **Invariance par translation:** un déplacement global de la molécule dans l'espace ne modifie pas les descripteurs. Ceci est vérifiée avec `test_translation_invariance`, avec une différence maximale de  $1.33 \times 10^{-15}$ .
- **Invariance par permutation:** l'ordre des atomes dans le fichier `.xyz` n'influence pas le résultat. Ceci est vérifié avec `test_permutation_invariance`, avec une différence maximale de  $1.78 \times 10^{-15}$ .
- **Invariance par rotation:** une rotation arbitraire 3D de la molécule ne modifie pas les distances ou les distributions spatiales. Ceci est vérifié avec `test_rotation_invariance`, avec une différence maximale de  $2.22 \times 10^{-15}$ .

Ces descripteurs sont ensuite concaténés aux coefficients de scattering et/ou aux matrices de Coulomb pour enrichir les vecteurs d'entrée.

### 5.5.3 Intégration de la matrice de Coulomb

Afin d'améliorer nos résultats, on a également considéré de concaténer la matrice de Coulomb aux coefficients de scattering. Comme dit dans la partie 1, les matrices de Coulomb sont connues pour modéliser les interactions électrostatiques entre atomes. On les génère comme suit :

- `create_train_coulomb_matrix` pour les données d'entraînement.
- `create_test_coulomb_matrix` pour les données de test.

Deux variantes sont testées :

- **Brute (raw):** sans permutation, ordre atomique initial.
- **Triée (sorted):** ordonnée selon la norme  $L^2$  des lignes pour assurer l'invariance aux permutations atomiques.

La version triée est principalement utilisée pour ses propriétés d'invariance.

### 5.5.4 Combinaisons testées

On a comparé les performances de plusieurs représentations :

1. **Scattering avec J=2 , L=3:** coefficients de scattering uniquement.
2. **Scattering avec J=4, L=3:** coefficients de scattering uniquement.

3. **Scattering avec  $J=2$ ,  $L=3 + features$  géométriques:** ajout des descripteurs géométriques simples.
4. **Scattering avec  $J=4$ ,  $L=3 + features$  géométriques:** ajout des descripteurs géométriques simples.
5. **Scattering avec  $J=2$ ,  $L=3 + Coulomb$ :** concaténation avec la matrice de Coulomb triée.
6. **Scattering avec  $J=2$ ,  $L=3 + Coulomb + features$ :** fusion des trois types de descripteurs.

Pour chaque combinaison, l'ensemble d'apprentissage  $X_{\text{train}}$  est formé en concaténant les vecteurs, et un modèle est entraîné pour prédire l'énergie moléculaire (contenue dans  $y_{\text{train}}$ ).

## 5.6 Résultats obtenus

Le tableau ci-dessus synthétise les performances obtenues pour différentes combinaisons de descripteurs, en mesurant la RMSE sur les ensembles d'apprentissage et de test.

Combinaison	Modèle	RMSE Train	RMSE Test
Scattering avec $J=2$ , $L=3$	Ridge ( $\alpha = 0.001$ )	0.98	0.713
Scattering avec $J=2$ , $L=3 + features$ géométriques	Ridge ( $\alpha = 0.01$ )	0.24	0.242
Scattering avec $J=4$ , $L=3$	Ridge ( $\alpha = 0.01$ )	1.14	0.799
Scattering avec $J=4$ , $L=3 + features$ géométriques	Ridge ( $\alpha = 0.01$ )	0.23	0.24
Scattering avec $J=2$ , $L=3 + Coulomb$ (sorted)	Ridge ( $\alpha = 0.1$ )	0.36	0.43
Scattering avec $J=2$ , $L=3 + Coulomb$ (sorted) + features	LinearRegression	0.20	0.33
Scattering avec $J=2$ , $L=3 + Coulomb$ (raw)	Ridge ( $\alpha = 0.0001$ )	0.29	0.32
Scattering avec $J=2$ , $L=3 + Coulomb$ (raw) + features	Ridge ( $\alpha = 0.1$ )	0.22	0.215

Table 3: Table comparative des RMSE d'entraînement et de test pour différentes combinaisons de modèles.

## 5.7 Analyse des résultats

On observe, dans un premier temps, que l'ajout de caractéristiques géométriques améliore significativement la performance, indépendamment de la valeur de  $J$ . En effet, la combinaison des coefficients de scattering avec des descripteurs géométriques simples réduit fortement la RMSE, particulièrement sur l'ensemble de test, passant par exemple de 0.713 à 0.242. Cela suggère que ces descripteurs apportent une information complémentaire pertinente.

De plus, quand on augmente la complexité du scattering, en utilisant une valeur plus élevée de  $J$ , on ne garantit pas une meilleure généralisation. Par exemple, même si l'utilisation de  $J = 4$  fournit plus de coefficients que  $J = 2$ , les performances en test restent similaires, voire parfois inférieures, en l'absence de caractéristiques géométriques. Par exemple, la RMSE de test passe de 0.799 pour  $J = 4$  à 0.713 pour  $J = 2$ .

D'autre part, l'ajout des matrices de Coulomb triées de manière canonique peut être bénéfique, bien que leur impact dépende de la combinaison utilisée. Par exemple, la combinaison `scattering + Coulomb + features` donne de bons résultats avec une RMSE de test de 0.33, mais pas nécessairement meilleurs que ceux obtenus avec `scattering + features` seul, qui atteint une RMSE de test de 0.242.

Par ailleurs, on observe que la combinaison `scattering + Coulomb (raw) + features` donne un score de RMSE en test encore plus faible, environ 0.215, mais on a décidé de ne pas l'utiliser car il ne respecte pas les contraintes physiques. En effet, contrairement à la version triée, la matrice de

Coulomb brute n'est pas invariante par permutation des atomes. Par conséquent, même si la RMSE est meilleure, le modèle utilisant Coulomb (raw) est considéré comme non fiable. Son utilisation sur des données permutées pourrait entraîner des prédictions incohérentes.

## 6 Conclusion et piste d'amélioration

Dans ce projet, on a exploré différentes représentations de molécules afin de prédire leur énergie d'atomisation. On a d'abord utilisé les matrices de Coulomb, en tenant compte de leurs propriétés d'invariance et des implications de la normalisation. Ensuite, on a comparé des modèles de régression linéaire, des forêts aléatoires et des réseaux de neurones, avec des performances marquées en faveur des forêts aléatoires et des MLP entraînés sur des données augmentées.

L'approche par transformée de scattering a ensuite permis de construire des descripteurs invariants et interprétables. L'ajout de caractéristiques géométriques simples a significativement amélioré les performances, atteignant une RMSE de test de 0.240. La combinaison avec la matrice de Coulomb brute a offert des résultats encore meilleurs, mais au prix d'une perte d'invariance, ce qui peut entraîner une mauvaise généralisation.

Pour poursuivre ce travail, il peut être intéressant d'explorer d'autres représentations, telle que l'ACP, afin de mieux comprendre la structure des descripteurs et potentiellement améliorer la généralisation des modèles.

## Références

- [1] Grégoire Montavon, Katja Hansen, Siamac Fazli, Matthias Rupp, Franziska Biegler, Andreas Ziehe, Alexandre Tkatchenko, O. Anatole von Lilienfeld, Klaus-Robert Müller. **Learning Invariant Representations of Molecules for Atomization Energy Prediction.** In Advances in Neural Information Processing Systems (NeurIPS), vol. 25, 2012.
- [2] Michael Eickenberg, Georgios Exarchakis, Matthew Hirn, Stéphane Mallat, Louis Thiry. **Solid Harmonic Wavelet Scattering for Predictions of Molecule Properties.** *Journal of Chemical Physics*, vol. 148, pp. 241732, 2018.