PROJET MOPSI

Instabilité de la dynamique hamiltonienne avec des potentiels appris par machine

15 mai 2024

Maryam El Yaagoubi, Félix Fourreau

École des Ponts
ParisTech

TABLE DES MATIÈRES

1 Introduction **3**

2 Choix de dataset **3**

3 Premier apprentissage **4**
3.1 Choix du potentiel . . . 4
3.2 Choix du schéma d'intégration . . . 5
3.2.1 L'algorithme de Verlet : Intuition . . . 5
3.2.2 L'algorithme de Verlet à Un Pas . . . 5
3.2.3 Verlet avec mémoisation . . . 6
3.3 Choix du réseau neuronal . . . 6
3.3.1 Motivation . . . 6
3.3.2 Résultats . . . 7

4 Architecture du réseau de neuronnes **9**
4.1 État de l'art . . . 9
4.1.1 Principes et Fonctionnement des GNNs . . . 9
4.1.2 Étude de Cas : GraphSAGE . . . 9
4.1.3 Importance de l'Équivariance . . . 10
4.2 Choix du Physnet . . . 10

5 Code **11**
5.0.1 Description Détaillée du Modèle . . . 11
5.0.2 Hyperparamètres et leur Rôle . . . 11

6 Analyse des résultats **12**
6.1 Molécule d'éthanol MD17 . . . 12
6.1.1 Sélection des bons modèles . . . 12
6.1.2 Impact de la taille du dataset d'entrainement . . . 13
6.1.3 Impact de la taille des batchs . . . 16
6.1.4 Impact du nombre de features . . . 17
6.1.5 Impact du cutoff . . . 17
6.1.6 Impact du poids de la force dans la loss . . . 18
6.1.7 Conclusion : Instabilité molécule d'éthanol . . . 18
6.2 Observation sur la molécule d'aspirine MD17 . . . 19

7 Conclusion **19**

8 Ouverture **20**

1 INTRODUCTION

La simulation de dynamique moléculaire, ancrée dans l'approximation de Born-Oppenheimer, constitue depuis longtemps le socle de la modélisation atomistique, ayant été récompensée par le prix Nobel de chimie en 2013 pour ses avancées significatives dans la compréhension des systèmes chimiques et biologiques complexes. Cependant, l'un des défis majeurs auxquels sont confrontées les simulations de dynamique moléculaire réside dans l'exactitude des modèles de prédiction de dynamique moléculaire classiques, limitant leur capacité à modéliser de manière prédictive les dynamiques et les fonctions des systèmes moléculaires.

Récemment, l'apprentissage automatique (ML) a émergé comme une approche prometteuse qui bouleverse les fondements de la simulation de la dynamique moléculaire. Basées sur des principes statistiques, ces modèles de prédiction de dynamique moléculaire visent à identifier une fonction de prédiction impartiale qui corréle de manière optimale un ensemble de structures moléculaires avec les énergies cibles données et, souvent, les forces utilisées comme données d'entraînement. Grâce à ses capacités de généralisation et à sa prédiction rapide sur des données non vues, ces modèles peuvent être utilisés pour accélérer la recherche de l'énergie minimale et de l'état de transition, l'analyse vibrationnelle, la simulation des spectres d'absorption et d'émission, l'exploration des réactions et des transitions structurelles, ainsi que la propagation de la dynamique des états fondamentaux et excités.

Un avantage et un inconvénient de l'apprentissage automatique est qu'il est possible de concevoir, à toutes fins utiles, un nombre infini de modèles de prédiction de dynamique moléculaire qui peuvent décrire la dynamique moléculaire. Ces modèles sont généralement construits à partir de deux composantes principales : l'algorithme de ML et son entrée X, le descripteur moléculaire. En choisissant un algorithme de ML spécifique, nous pouvons restreindre l'espace d'hypothèses des fonctions de mappage à une taille recherchable. Ces fonctions de mappage utilisées dans le processus d'apprentissage peuvent être commodément exprimées sous forme d'algorithmes de ML paramétriques ou non paramétriques. Dans le cadre de ce projet, nous utiliserons des algorithmes non paramétriques de type réseaux de neuronnes. Nous commencerons par utiliser un simple réseau feedforward pour prédire la dynamique moléculaire, puis nous utiliserons des architectures qui s'adaptent mieux à la structure moléculaire.

L'idée sera de visualiser les instabilités que présentent les trajectoires numériques afin de mieux les caractériser en utilisant des modèles de plus en plus complexes jusqu'à trouver le modèle minimal pour lequel la dynamique hamiltonienne explose pour un champ de force appris par réseau neuronal.

2 CHOIX DE DATASET

Dans le paysage diversifié des bases de données utilisées pour l'apprentissage des dynamiques hamiltoniennes en chimie moléculaire, MD17 se distingue comme un choix stratégique pour notre projet. Bien que des alternatives comme QM9 et QM7b offrent une richesse en propriétés moléculaires sur des échelles plus larges, MD17 est particulièrement adapté à l'étude des dynamiques moléculaires en raison de sa spécificité et de sa concentration sur les énergies moléculaires et les forces.

Il est vrai que le MD17 standard peut présenter un certain niveau de bruit numérique, dû en partie

aux méthodes de génération de données et aux paramètres de calcul. Cependant, cette caractéristique ne diminue pas nécessairement sa valeur pour nos besoins. En effet, le MD17 permet d'explorer environ vingt minimas énergétiques différents par molécule, offrant ainsi une vue complète et nuancée des paysages énergétiques moléculaires. Cette profondeur d'information est essentielle pour nos objectifs, car elle permet de capturer des comportements dynamiques subtils qui sont souvent négligés dans des datasets plus généraux.

Notre choix est également renforcé par le fait que MD17 est largement utilisé dans la littérature scientifique, ce qui nous fournit une base comparative solide pour évaluer la performance de nos modèles. La reconnaissance académique du MD17 signifie que de nombreux chercheurs ont testé et validé ses données dans divers contextes, attestant ainsi de sa pertinence malgré ses défis inhérents.

Concernant les inquiétudes relatives à la stabilité, le choix de MD17 répond à un compromis entre précision et praticabilité. Les simulations requièrent des pas d'intégration très petits, et le nombre de pas nécessaire reste gérable avec MD17 sans compromettre la durée pratique des simulations. De plus, les instabilités observées pourraient, en effet, provenir des caractéristiques intrinsèques du dataset, mais notre approche de sparsification des données vise à surmonter ce problème en choisissant des échantillons qui maximisent la représentation des minimas énergétiques sans surcharge informationnelle (en pratique, la sélection aléatoire des échantillons s'avère adéquate pour nos objectifs : le dataset MD17 incluant environ une vingtaine de minimas énergétiques distincts pour chaque molécule, une taille d'échantillon de l'ordre de mille nous permet d'assurer une représentation exhaustive de ces minimas)

3
PREMIER APPRENTISSAGE

3.1 Choix du potentiel

Nous débutons nos expérimentations avec le potentiel de Müller-Brown, un choix judicieux pour les premières manipulations en raison de sa capacité à représenter des paysages énergétiques complexes à travers une formulation relativement simple. Le potentiel de Müller-Brown est particulièrement intéressant pour étudier les minima énergétiques et les barrières de transition, éléments clés dans la compréhension des trajectoires dynamiques moléculaires. Il est défini de la façon suivante :

$$V(x,y) = \sum_{i=1}^{4} K_i \exp\left(a_i(x-\beta_i)^2 + b_i(x-\beta_i)(y-\gamma_i) + c_i(y-\gamma_i)^2\right)$$

où les paramètres sont les suivants :

$$\begin{aligned}
K &= [-20, -10, -17, 1.5\},\\
a &= \{-1, -1, -6.5, 0.7\},\\
b &= \{0, 0, 11, 15\},\\
c &= \{-10, -10, -6.5, 0.7\},\\
\beta &= \{1, 0, -0.5, -1\},\\
\gamma &= \{0, 0.5, 1.5, 1\}.
\end{aligned}$$

Avec le potentiel bien défini, nous nous tournons vers l'aspect computationnel de notre étude, en particulier le choix du schéma d'intégration. Ce choix est essentiel pour assurer la précision et la stabilité de nos simulations numériques.

3.2 Choix du schéma d'intégration

Le schéma d'intégration est crucial pour la visualisation correcte de la dynamique à partir du potentiel prédit. Pour ce projet, nous avons sélectionné le schéma de Verlet pour son utilisation répandue en dynamique moléculaire et sa capacité à conserver l'énergie, propriété physique indispensable.

3.2.1 • L'algorithme de Verlet : Intuition

Un des algorithmes symplectiques les plus simples à mettre en œuvre et largement utilisé en dynamique moléculaire est l'algorithme de Verlet.

Il est facile de constater que ce schéma est réversible en temps et donc particulièrement adapté à l'étude des systèmes conservatifs dont les forces ne dépendent que de la position. En revanche, cet algorithme présente deux défauts.

Il s'agit d'un schéma à deux pas : l'itération ne peut démarrer que si l'on connaît q_0 et q_{-1}. Or, en général, les conditions initiales se résument par la donnée de q_0 et p_0. Il est d'usage alors d'initier l'itération à l'aide d'un développement de Taylor. Si l'on s'intéresse à l'évolution d'une grandeur faisant intervenir le moment (comme l'énergie cinétique), celle-ci se calcule à l'aide de la formule :

$$p_n = \frac{q_{n+1} - q_{n-1}}{2h} \tag{1}$$

On voit donc que le calcul du moment consiste à soustraire deux nombres voisins ce qui produit des erreurs d'arrondi importants -bien amplifiées dans le contexte de dynamiques hamiltoniennes. C'est pourquoi, on utilise en général une autre version algorithmique qui est mathématiquement équivalente à la version originale de Verlet mais qui présente l'intérêt d'être à un pas.

3.2.2 • L'algorithme de Verlet à Un Pas

L'algorithme de Verlet à un pas repose sur le schéma numérique suivant :

Algorithme de Verlet

1. Initialisation du pas de temps Δt et des conditions initiales : $t = 0$, $q = q_0$, $p = p_0$.
2. L'algorithme de Verlet s'écrit :

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{\Delta t}{2}\nabla V(q_n), \\ q_{n+1} &= q_n + \Delta t M^{-1} p_{n+1/2}, \\ p_{n+1} &= p_{n+1/2} - \frac{\Delta t}{2}\nabla V(q_{n+1}). \end{aligned}$$

3. On itère.

Choix d'hyperparamètres L'efficacité de l'algorithme de Verlet dans la préservation de l'énergie mécanique d'un système repose sur la sélection judicieuse de ses hyperparamètres, en particulier le pas de temps Δt. En effet, le pas de temps doit être une fraction de la période de vibration la plus rapide du système étudié pour garantir une simulation fiable. Dans le contexte des systèmes moléculaires, cela correspond typiquement à des pas de temps de l'ordre de la femtoseconde (10^{-15}s), une fraction de la période typique des vibrations moléculaires les plus rapides.

Le choix de ce pas de temps est crucial car il influence directement la précision de la conservation de l'énergie. La méthode symplectique, utilisée par l'algorithme de Verlet, permet de conserver une quantité d'énergie approximative de façon exacte. Cette conservation se manifeste par une oscillation de l'énergie autour de sa valeur initiale, dont l'amplitude est liée à l'ordre de la méthode numérique. Il est important de noter que la conservation de l'énergie est quasi maintenue sur des intervalles de temps qui croissent exponentiellement avec le pas de temps, ce qui souligne l'importance d'un choix adéquat de Δt.

Afin de garantir la stabilité à long terme des prédictions, une analyse rétrograde est souvent employée. Cette approche consiste à ajuster les hyperparamètres de manière à ce que l'erreur numérique ne croisse pas de façon inacceptable au fil des itérations. Ainsi, un choix approprié des hyperparamètres est essentiel pour assurer la fiabilité et la précision des simulations de dynamiques moléculaires effectuées par l'algorithme de Verlet.

3.2.3 • Verlet avec mémoisation

Dans l'algorithme de Verlet amélioré avec mémoïsation, le gradient $\nabla V(q)$ est calculé au début de l'étape 1 et stocké dans la variable `grad_cache`. Les étapes suivantes utilisent cette valeur précalculée, ce qui évite de recalculer le gradient aux étapes 1 et 3, améliorant ainsi l'efficacité de l'algorithme. Cela se traduit mathématiquement par :

$$\begin{aligned} p_{n+1/2} &= p_n - \frac{\Delta t}{2}\nabla V(q_n), \\ q_{n+1} &= q_n + \Delta t M^{-1} p_{n+1/2}, \\ p_{n+1} &= p_{n+1/2} - \frac{\Delta t}{2}\nabla V(q_{n+1}), \end{aligned}$$

où $\nabla V(q_n)$ est remplacé par `grad_cache` aux étapes 1 et 3.

Cette modification permet une réduction significative du coût de calcul lors de l'itération de l'algorithme de Verlet.

3.3 Choix du réseau neuronal

3.3.1 • Motivation

Le potentiel du Muller Brown, ne dépendant que des positions x et y dans un plan en deux dimensions, peut être appris efficacement par un réseau de neurones simples. Nous avons donc choisi une architecture de type Multilayer Perceptron (MLP). La structure d'un MLP consiste en un enchaînement de couches de neurones de tailles variables, séparées par des fonctions non linéaires. Les sorties de chaque neurone de ces couches sont envoyées aux couches suivantes jusqu'à obtenir en sortie une

prédiction. La détermination des poids de ces couches neuronales est primordiale pour obtenir des prédictions fiables.

L'ajustement de ces poids s'effectue lors de la phase d'entraînement du modèle en calculant l'impact de chacun de ces paramètres sur l'écart de la prédiction avec la valeur objectif, par le calcul du gradient. Les poids sont alors modifiés proportionnellement à leur impact sur la précision de la prédiction, dans le but d'atteindre après plusieurs itérations le minimum global de la fonction objectif. Remarquons que la vitesse et la précision de cette convergence sont données par le taux d'apprentissage et le nombre d'itérations de l'entraînement ; ces valeurs constituent, avec le nombre de poids du réseau, les hyperparamètres du MLP. Ceux-ci doivent être choisis en fonction du problème et de la précision souhaitée.

3.3.2 • Résultats

Nous avons d'abord généré une base de données d'apprentissage comportant 50 000 énergies et gradients, pour lesquelles les valeurs de potentiels associées n'étaient pas supérieures au potentiel de Gibbs à température ambiante en ordre de grandeur proche de 0. Nous avons testé une multitude d'architectures, parmi lesquelles nous avons retenu la plus simple, celle obtenant les meilleurs résultats : 4 couches pour apprendre l'énergie ((2,32),(32,64),(64,32),(32,1)), et 4 couches pour apprendre les gradients selon x et y ((2,32),(32,64),(64,32),(32,2)). Toutes ces couches sont séparées par une non-linéarité de type ReLU. Nous avons fixé le taux d'apprentissage à 0.001, la taille des lots à 64, et le nombre d'epochs à 1000. Par ailleurs, nous avons utilisé l'erreur quadratique comme fonction de perte et l'optimiseur Adam pour accélérer la convergence de l'entraînement. Voici les résultats obtenus pour l'entraînement du modèle, qui converge rapidement :

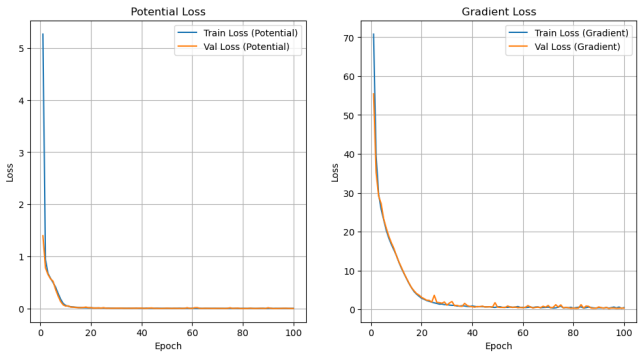


Figure 1 – Evolution des loss pour l'entrainement du MLP

L'exploration de la dynamique donne des résultats prometteurs. En effet, il n'y a pas de divergence du potentiel ni d'explosion de l'énergie : la dynamique semble cohérente et aucune instabilité n'apparaît après 100 000 pas pour 100 positions initiales différentes, le tout avec un faible nombre de paramètres et de données d'entraînement. Ces observations sont encourageantes pour l'évaluation de dynamiques plus complexes par des réseaux de neurones.

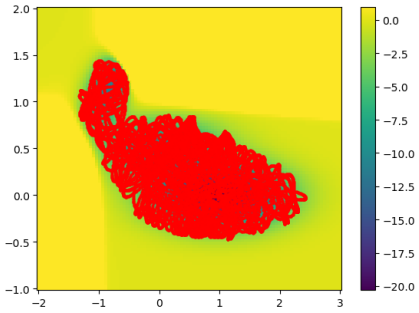
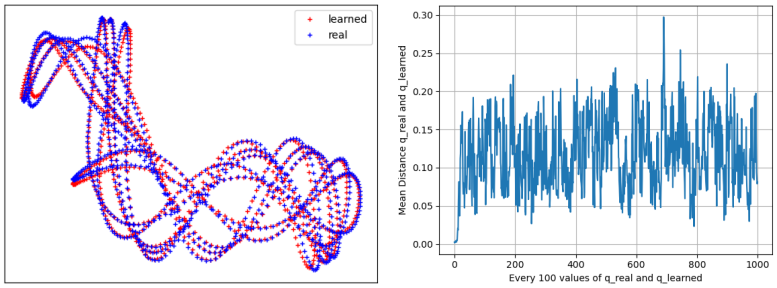


Figure 2 – Trajectoire particule dans potentiel de Muller Brown

Cependant, on observe que même si la trajectoire reste confinée dans un puits de potentiel, elle s'écarte progressivement de la trajectoire attendue. Cette divergence pourrait représenter un inconvénient pour une application concrète à l'étude de la dynamique moléculaire, notamment pour des modèles et des situations plus complexes.



(a) Comparaison dynamique apprise et réelle

(b) Evolution de la distance

On vérifie d'autre part la stabilité de l'energie du system tout le long de la dynamique. Cette stabilité est essentielle pour la dynamique Hamiltonienne est renforce l'idée que nous utiliserons cet algorithme dans toute la suite du projet pour evaluer la dynamique moléculaire

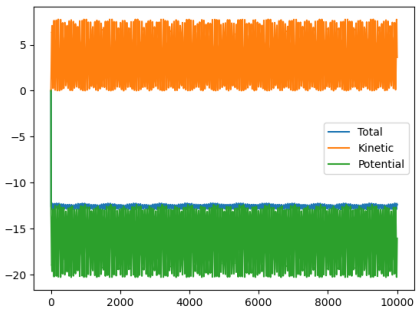


Figure 3 – Evolution des loss pour l'entrainement du MLP

4
ARCHITECTURE DU RÉSEAU DE NEURONNES

4.1 État de l'art

Les réseaux neuronaux en graphe (GNNs) sont conçus pour traiter des données représentées sous forme de graphes, exploitant la structure riche en informations des relations entre les nœuds pour réaliser diverses tâches analytiques.

4.1.1 • Principes et Fonctionnement des GNNs

Les GNNs utilisent le paradigme du *message passing*, où chaque nœud échange des informations avec ses voisins. Ce mécanisme permet aux nœuds de combiner leurs propres caractéristiques avec celles de leur voisinage pour générer des représentations globales du graphe.

Fonction d'Agrégation La fonction d'agrégation est au cœur de l'architecture des GNNs. Elle définit comment les informations provenant des différents voisins d'un nœud sont combinées pour mettre à jour l'état de ce nœud. Mathématiquement, cela peut être exprimé par exemple comme suit :

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ij}} \cdot h_j^{(l)} \cdot W^{(l)}\right)$$

où :
— $h_i^{(l)}$ représente le vecteur de caractéristiques du nœud i à la couche l,
— $\mathcal{N}(i)$ désigne le voisinage du nœud i,
— c_{ij} est un facteur de normalisation basé sur le degré des nœuds,
— $W^{(l)}$ est la matrice de poids partagée par tous les nœuds à la couche l,
— $\sigma(\cdot)$ est une fonction d'activation.

Cette formulation permet aux nœuds de mettre à jour leurs états en intégrant efficacement les informations issues de leur environnement local.

Méthodes d'Agrégation Communes Les fonctions d'agrégation varient selon les spécificités du modèle et les exigences de la tâche. Les méthodes courantes incluent :

Pooling : Le pooling peut être réalisé par max pooling ou mean pooling, où l'on agrège respectivement le maximum ou la moyenne des caractéristiques des voisins après les avoir éventuellement transformées via une fonction appliquée de manière indépendante à chaque voisin.

LSTM Une autre méthode utilise des réseaux LSTM pour traiter les informations des voisins de manière séquentielle, ce qui peut capturer des dépendances complexes entre les caractéristiques des nœuds dans des ordres spécifiques.

4.1.2 • Étude de Cas : GraphSAGE

GraphSAGE est un exemple notable de GNN qui utilise des fonctions d'agrégation pour générer des embeddings de nœuds qui peuvent être appliqués inductivement à des graphes non vus lors de

l'entraînement. GraphSAGE est particulièrement adapté aux graphes riches en caractéristiques, comme ceux rencontrés dans les données biologiques ou de citation.

Algorithme de Propagation Avant de GraphSAGE L'algorithme de GraphSAGE procède par agrégation itérative des caractéristiques des voisins pour chaque nœud, en utilisant une série de fonctions d'agrégation et de matrices de poids :

Graph $G(V, E)$; caractéristiques d'entrée $\{x_v, \forall v \in V\}$; profondeur K ; matrices de poids $W_k, \forall k \in \{1, ..., K\}$; non-linéarité σ ; fonctions d'agrégation $AGGREGATE_k, \forall k \in \{1, ..., K\}$; fonction de voisinage $N : v \rightarrow 2^V$ Représentations vectorielles z_v pour tous les $v \in V$ $v \in V$ $h_v^0 \leftarrow x_v$ $k = 1...K$ $v \in V$ $h_N^k(v) \leftarrow AGGREGATE_k(\{h_u^{k-1}, \forall u \in N(v)\})$ $h_v^k \leftarrow \sigma(W_k \cdot \text{CONCAT}(h_v^{k-1}, h_N^k(v)))$ $v \in V$ $h_v^k \leftarrow \frac{h_v^k}{\|h_v^k\|_2}$ $v \in V$ $z_v \leftarrow h_v^K$

4.1.3 • Importance de l'Équivariance

Dans le contexte des GNNs appliqués aux données physiques, l'équivariance aux transformations comme les rotations et les translations assure que les prédictions restent valides sous différentes orientations ou positions des objets étudiés.

4.2 Choix du Physnet

Le réseau Physnet est particulièrement intéressant en raison de sa capacité à construire des réseaux neuronaux qui sont équivariants par rapport au groupe Euclidien E(3), incluant les translations, rotations, et réflexions de l'espace tridimensionnel.

L'architecture garantit que les valeurs numériques des quantités physiques ou géométriques (comme les positions ou orientations) changent de manière prévisible sous des transformations de l'espace, sans perturber les règles sous-jacentes que le modèle doit apprendre. Caractéristique cruciale pour les dynamiques moléculaires. D'autre part, grâce à l'équivariance intégrée, il permet une modélisation précise même avec moins de données, car il n'est pas nécessaire de réapprendre les comportements sous différentes orientations ou positions des objets étudiés. Il est également à noter que l'architecture de physnet est très simple et requière moins de paramètres pour atteindre une performance similaire ou supérieure à celle des réseaux neuronaux ordinaires, car ils tirent parti des symétries des données pour réduire la complexité du modèle (en lançant le modèle, on a obtenu une performance presque aussi bonne que celle du Gemnet pour une fraction du temps de calcul). En l'occurence, nous avons pu executer des calculs avec un million de pas de temps en moins d'une heure avec le supercalculateur du CEA Paris Saclay Jean Zay.

5
CODE

Le script implémente un modèle de Physnet.

5.0.1 • Description Détaillée du Modèle

Le modèle utilise E3x pour construire des interactions atomiques au sein d'un graphe moléculaire, où chaque atome est un nœud et chaque lien potentiel entre atomes est une arête. Ce graphe est traité à travers un réseau de message-passing qui intègre les positions relatives et les types atomiques pour prédire des propriétés telles que les énergies et les forces atomiques.

5.0.2 • Hyperparamètres et leur Rôle

Features et Max Degree Le modèle utilise des vecteurs de caractéristiques de dimension `features = 32` et un `max_degree = 2`. Ces paramètres définissent la complexité des interactions entre les caractéristiques dans le graphe, influençant directement la capacité du modèle à capturer des détails fins dans les données de structure moléculaire mais aussi le nombre de paramètres du modèle.

Num Iterations et Num Basis Functions Avec `num_iterations = 3` et `num_basis_functions = 32`, le modèle effectue plusieurs passes de message-passing pour raffiner les prédictions des propriétés atomiques. Chaque itération permet d'agréger plus d'informations locales et distantes, augmentant potentiellement la précision mais aussi le coût computationnel et donc le temps de calcul.

Cutoff Le `cutoff = 3.0` définit la portée spatiale maximum des interactions dans le graphe. Il est crucial pour limiter les effets des atomes éloignés et pour concentrer l'apprentissage sur les interactions significatives, réduisant le bruit dans les prédictions.

Batch Size et Learning Rate Le modèle est entraîné avec un `batch_size = 30` et un `learning_rate = 0.01`. Ces paramètres sont essentiels pour contrôler la vitesse de convergence du modèle et la stabilité de l'apprentissage. Un batch size plus petit peut conduire à des mises à jour plus fréquentes mais potentiellement plus bruyantes, tandis qu'un taux d'apprentissage modéré aide à éviter de surajuster sur des détails mineurs des données d'entraînement.

Training Dynamics Le modèle utilise une fonction de perte qui combine les erreurs sur les énergies prédites et les forces, pondérée par `forces_weight = 1.0`. Cette pondération assure que les prédictions de force sont considérées aussi importantes que les prédictions d'énergie, ce qui est crucial pour les applications où les forces moléculaires jouent un rôle dynamique important.

6
ANALYSE DES RÉSULTATS

L'idée est de voir comment va se comporter le modèle pour différentes valeurs d'hyperparamètres, notamment dans quels cas a-t-il tendance à exploser et lorsqu'il n'explose pas, quelle est la variance de l'énergie totale (à nombres de pas fixés).

6.1 Molécule d'éthanol MD17

La molécule d'éthanol est la molécule la plus petite du dataset md17. C'est donc avec celle-ci que nous essaierons de trouver les hyperparamètres qui impactent le plus les instabilités. En effet, elle nécessite un entraînement plus léger et donc des temps de calcul plus faibles. Cependant, ceux-ci restent tout de même très longs. Ainsi, la dynamique ne sera toujours évaluée que sur 100 000 pas, ce qui équivaut à un temps de dynamique de 0,1 ns. Les hyperparamètres évalués pour étudier le comportement de l'éthanol seront les suivants :

```
L_num_trains=[100*k for k in range(1,10,2)]
L_num_basis_functions=[32]
L_cutoff=[3,4]
L_batch_size=[10,30,50]
L_num_valid=[k//5 for k in L_num_trains]
L_learning_rate=[0.01]
L_forces_weight=[0.01,0.1]
L_features=[32,64]
max_degree=2
```

Figure 4 – Hyperparametres évalués pour la l'apprentissage de l'ethanol

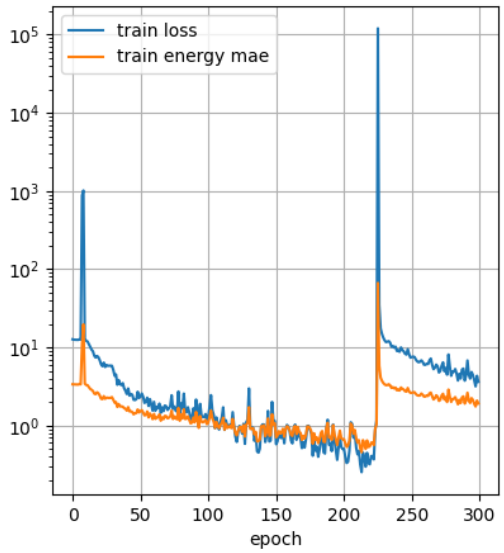
Nous étudierons donc l'influence des hyperparamètres classiques des réseaux de neurones (features, batch size et training size), mais aussi celle des hyperparamètres propres à la dynamique moléculaire (force weight et cutoff). Nous n'avons pas pu étudier l'influence de tous les hyperparamètres en raison d'une limite de temps, mais nous avons été orienté vers les paramètres soupçonnés d'influer le plus sur les instabilités par M. Marinica

6.1.1 • Sélection des bons modèles

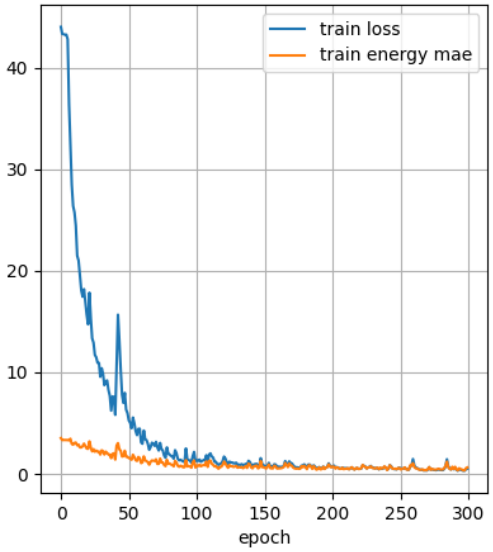
L'idée de cette partie est de conserver uniquement les hyperparamètres qui permettent un entraînement correct du modèle, c'est à dire un entrainement qui laisse à penser par les indicateurs de loss sur l'entrainement et la validation que celui-ci peut estimer avec précision la force d'une molécule test.

Modèles sous entrainés

La diminution de la loss lors de l'entraînement est une condition nécessaire pour un modèle efficace. Il n'est donc pas utile de tester des modèles qui ne convergent pas puisque l'instabilité inhérente est simplement causée par un mauvais entrainement. Ce type d'entrainement se reconnait par l'évolution de sa loss :



(a) Mauvais Entrainement



(b) Bon Entrainement

On supprime de cette manière tous les modèles tels que la loss et la loss mae d'entrainement (mean absolute error) de l'énergie sont pour les 10 dernières epochs toujours supérieures à 3. Cette valeur permet de supprimer 15 combinaisons d'hyperparametres dont la majorité comportait 64 features et une taille d'entrainement de 100.

Cela s'explique notamment par le fait qu'un modele avec 64 features et 100 molécules d'entrainement peut avoir des difficultés à converger.

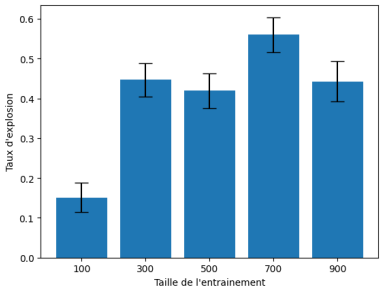
Modèles sur entrainés

A l'opposé un modele avec 64 features et 1000 molécules d'entrainement peut overfitter sur les données d'entrainement. On peut de cette même manière supprimer les modèles qui overfittent les données en procédant de façon similaire mais cette fois sur la loss de validation. On ne conserve alors que les modèles dont la loss de validation est supérieure à 3. On ne retire cependant avec cette méthode que quelques modèles avec toujours un dataset de 1000 molécules.

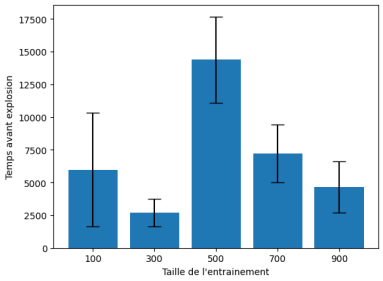
6.1.2 • Impact de la taille du dataset d'entrainement

Nous passerons rapidement cette partie tant la taille du dataset d'entrainement est une composante classique des modèles de DeepLearning. Ici et dans toute la suite la taille du dataset de validation sera 20% de la taille de celui d'entrainement et le dataset test sur lequel seront évalués les modéles sera composé de molécules non comprises dans la validation et dans le train.

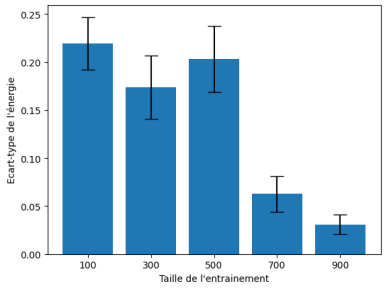
Nous avons souhaité selectionner la taille de dataset qui maximisait les explosions afin de pouvoir observer des instabilités avec un nombre de pas de temps raisonnable. Ainsi dans un premier temps nous avons entrainé des modèles pour chaque hyperparametres à taille d'entrainement fixé. Ensuite chaque modèle a été testé sur 100 molécules tirées aléatoirement sur le dataset entier pour 100 000 pas on obtient les résultats suivants :



(c) Taux d'explosion



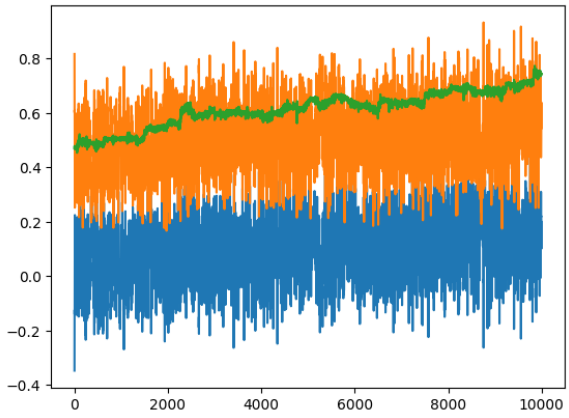
(d) t moyen avant explosion



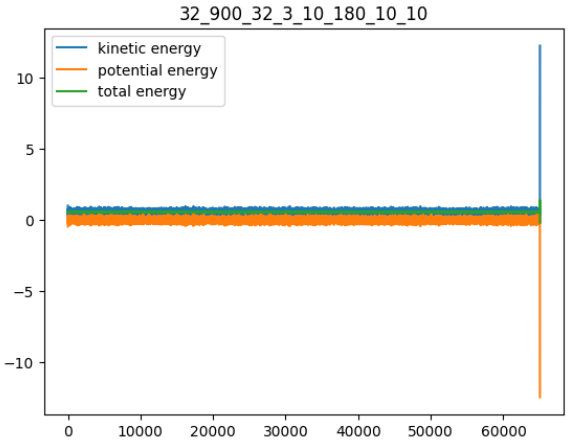
(e) Variance energie totale (sans explosion)

Afin d'obtenir ces résultats, nous avons considéré qu'une molécule avait explosé si il existait un pas de temps tel que le gradient de l'énergie dépassait 1. Ce critère est plutot arbitraire mais semble toutefois représenter assez fidèlement les explosion de gradient. On peut de cette manière trouver le pas de temps à partir duquel la molécule explose. D'autre part on a aussi voulu représenter la variance de l'énergie pour les molécules qui n'explosaient pas, c'est à dire pour les molécules dont le gradient de leur énergie sur les 100 000 pas de temps ne dépassait jamais 1.

Les résultats obtenus sont suprenants puisqu'ils ne montrent pas de tendance générale à la diminution des instabilités lorsque la taille de l'entrainement augmente. Cette tendance ne peut pas etre du à l'overfitting jusqu'alors non detecté puisque l'instabilité n'augmente pas plus après 300 molécule. On peut tout de même identifier deux types d'instabilités, celles pour lesquelles l'énergie diverge progressivement de la valeur initiale (ecart type de l'énergie élevé) et celles qui correspondent à une explosion soudaine de l'énergie (taux d'explosion élevé). Graphiquement elles se distingue facilement :



(f) Divergence progressive de l'énergie

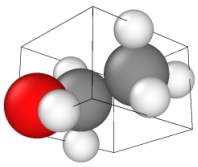


(g) Explosion de l'énergie totale

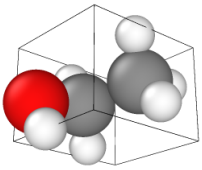
Ainsi, on observe que la taille de la base de données d'entraînement influe moins sur le taux d'explosion des molécules que sur la variance de l'énergie de la dynamique. Pour les bases d'entraînement de 100 molécules, les deux phénomènes semblent se compenser : un faible taux d'explosion est accompagné d'une forte divergence de l'énergie lorsqu'elle n'explose pas. Pour l'autre valeur extrême d'entraînement, on distingue un comportement opposé : avec 900 molécules d'entraînement, la dynamique explose plus souvent, mais quand elle ne le fait pas, elle est beaucoup plus stable. Cela

correspond aux deux types d'instabilités identifiées plus tot.

Remarquons que dans les résultats nous avons observé un nombre significatif d'explosions quasi instantannées de l'énergie, ce comportement surprennant pourrait venir de l'initialisation de la molécule avant le passage dans le modèle. En effet celle ci est placée dans une configuration qui assure une majoration de la force pour chaque atome afin de partir d'une configuration stable. Cette configuration est atteinte par optimisation selon la méthode Broyden-Fletcher-Goldfarb-Shanno (BFGS) qui peut avoir tendance à ne pas converger assez vite, c'est pourquoi nous avons limité le nombre de pas de la méthode à 100, si au bout de ces 100 pas elle n'est pas parvenue à descendre en dessous de la limite de force (fmax=0.05) alors on test le modele sur un atome qui n'est pas nécessairement dans une configuration stable et qui possède une énergie elevée, et qui est donc dans un état dont le modele ne peut assurer une énergie constante.

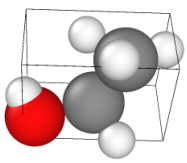


(h) Avant optimisation

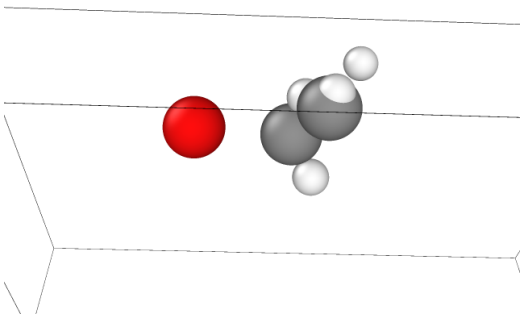


(i) Apres optimisation

On peut par exemple observer le comportement d'une molécule non optimisée sur ovito [1] on remarque que celle ci a tendance à beaucoup bouger avec une ampleur croissante jusqu'au moment ou l'atome d'hydrogene "entre" dans l'atome d'oxygen entrainant le detachement de l'atome du reste de la molécule. Cette éjection rapide entraine une augmentation conséquente de l'énergie cinétique, l'énergie totale ne plus être conservée. Dans la suite du projet, lorsque les graphiques montrent une explosion de l'énergie cinétique, cela signifie qu'un atome ou plus se sont décrochés de la molécule.



(j) Avant explosion



(k) Apres explosion

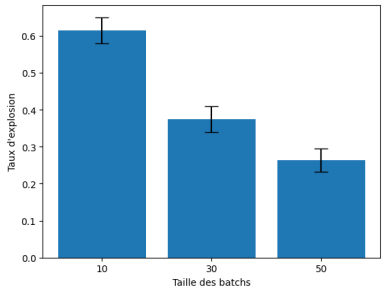
Ces cas de figure ne nous intéresse pas, c'est pourquoi ils ont été écarté de tous les résultats pendant

ce projet.

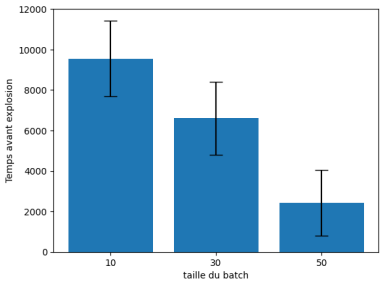
6.1.3 • Impact de la taille des batchs

La taille des batch est un paramètre crucial dans l'entraînement des modèles d'apprentissage automatique, notamment lorsqu'ils sont exécutés sur des GPU. Une taille de batch plus grande peut accélérer la convergence du modèle en permettant une exploitation plus efficace du parallélisme. Cependant les gros batch peuvent avoir plus de difficulté à converger vers des minimas puisque les grosses mise à jour de poids qu'ils induisent peuvent outrepasser ces minimas. D'autre part des batch trop importants peuvent entraîner une saturation de la mémoire du gpu. Dans notre étude, nous avons limité la taille des batchs à 50 et nous avons évité ce problème.

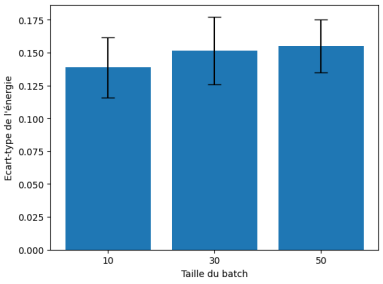
Nous avons donc étudié trois tailles de batchs différentes : 10, 30 et 50, qui reflètent des tailles généralement utilisées pour des tâches de deep learning classiques. Nous obtenons les résultats suivants :



(l) Taux d'explosion



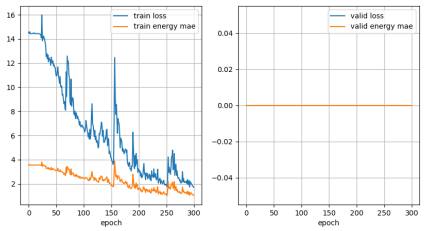
(m) t moyen avant explosion



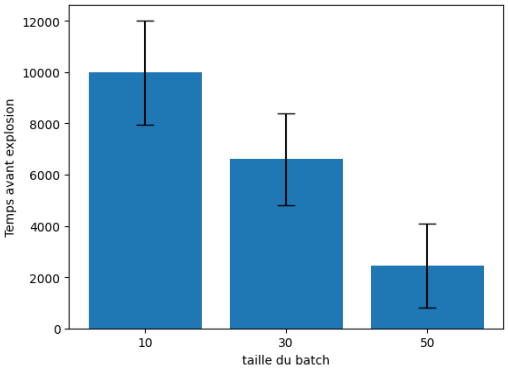
(n) Variance energie totale (sans explosion)

Nous observons donc que le taux d'explosion decroit avec la taille des batchs, cela peut venir du fait que si les plus petits batchs permettent de converger plus précisement vers un minimum de loss, ils sont plus sensibles aux aléas du gradient stochastique puisque le gradient du batch correspond à la moyenne des gradients de ses composantes. Ainsi lorsqu'il atteint un minimum de loss, le modèle est très efficace (cf t moyen avant explosion plus élevé) mais lorsqu'il ne permet pas une convergence suffisante de la loss, le modéle créé est ineficace (cf taux d'explosion élevé). Et au contraire des gros batchs vont plus souvent parvenir à converger (cf faible taux d'explosion) mais lorsque'ils ne convergent pas les conséquences sur le modèle sont mauvaises (cf temps avant explosion faible).

Remarquons que le faible temps moyen avant explosion pour une taille de batch de 50 n'est pas du à une incompatibilité avec les modèles entrainés sur 100 molécules (ce que nous pensions initialement). En effet on observe aucun différence significative en les supprimant des données :



(o) num train=100 ; batch size=50

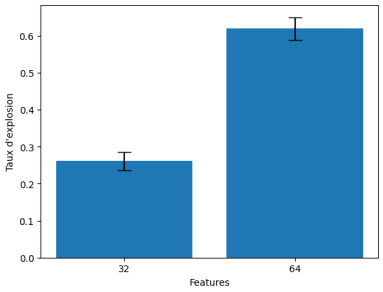


(p) t moyen avant explosion

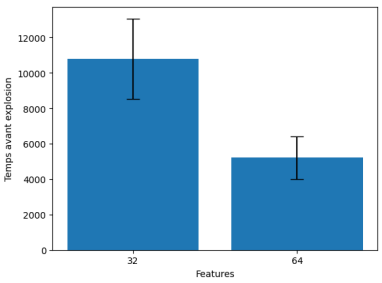
En fait, la taille des batchs contribue principalement au deuxième type d'instabilité, c'est-à-dire les explosions. En effet la variance de l'énergie reste relativement stable lorsque les explosions ne se produisent pas. Par conséquent, la taille des batchs n'affecte pas significativement le premier type d'instabilité (divergence progressive de l'énergie).

6.1.4 • Impact du nombre de features

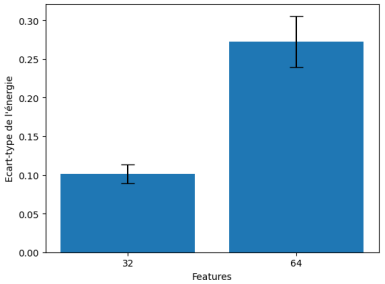
L'impact du nombre de features a déja été rapidement mentionné dans la partie 6.1.1, et plusieurs modéles entrainés avec 64 features ne présentaient pas un entrainement satisfaisant et ont été retiré. Ainsi en évaluant les modèles restants à nombre de features fixé, on obtient les résultats suivants :



(q) Taux d'explosion



(r) t moyen avant explosion



(s) Variance energie totale (sans explosion)

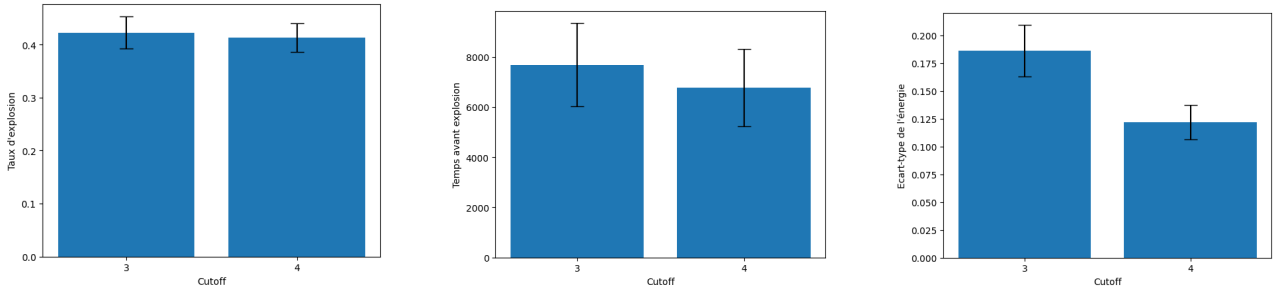
On observe directement que pour l'éthanol un nombre de features de 64 est beaucoup trop important pour prédir la force. Comme annoncé précédemment, la loss ne parvient pas à converger.

Ainsi un trop grand nombre de features contribue à tous les types d'instabilités.

6.1.5 • Impact du cutoff

Le "cutoff" représente le rayon de coupure à partir duquel les interactions inter atomique sont négligées. Un cutoff plus court permet de capturer des interactions à courte portée avec une précision plus élevée, tandis qu'un cutoff plus long inclura des interactions à plus longue portée, ce qui peut être important pour certaines propriétés moléculaires. Cependant, un cutoff plus long augmente gé-

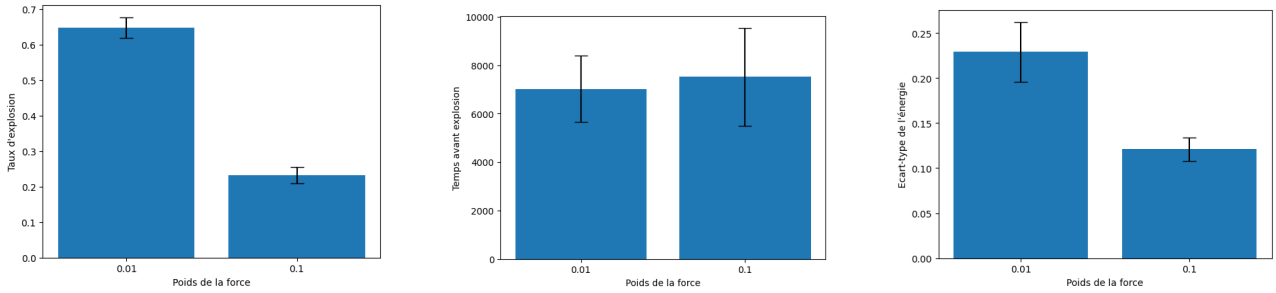
néralement le coût de calcul de l'entrainement. On obtient les résultats suivants en testant pour les deux valeurs de cutoff fixées :



On remarque que le cutoff n'impacte que très légérement les prédictions cela peut être du au fait que la molécule d'éthanol est très simple et que la géne stérique des atomes entre eux est trés faible et ceux ci reste globalement éloignés les un des autres.

6.1.6 • Impact du poids de la force dans la loss

La loss dans le modèle comprend une composante energétique et une composante pour la force. En ajoutant cette composante en gradient à la fonction de perte, on enrichit le paysage de la loss. Ce paysage de loss plus détaillé, permet au modèle de converger vers des minimas plus profonds et plus réalistes, ce qui se traduit par des prédictions plus précises et plus fiables de l'énergie potentielle et des forces. De plus, cette approche permet également de mieux généraliser le modèle à des données nouvelles ou peu fréquentes, car il apprend les relations subtiles entre les structures moléculaires et leurs énergies associées, ainsi que les forces qui les stabilisent.



On observe ici l'importance de la prise en compte de la force dans la loss, celle ci permet de réduire drastiquement le taux d'explosion des molécules par 3 et la variance de l'énergie par 2. De cette manière on identifie le poids de la force dans la loss comme un facteur déterminant de l'instabilité de la dynamique moléculaire apprise.

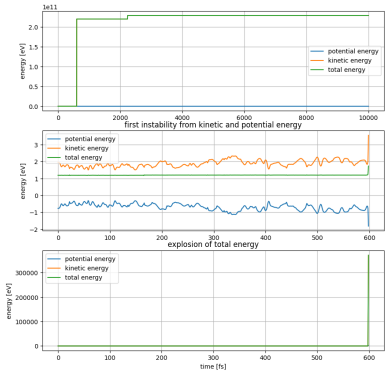
6.1.7 • Conclusion : Instabilité molécule d'éthanol

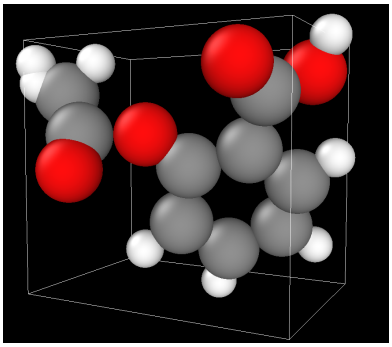
On a distingué deux types d'instabilités pour la molécule d'éthanol :
— Instabilité 1 : Divergence progressive de l'énergie totale de sa valeur initiale
— Instabilité 2 : Explosion soudaine de l'énergie
Les hyperparamètres contribuant à ces instabilité sont :
— Instabilité 1 :
  — Nombre faible de données d'entrainement
  — Nombre de features élevé
  — Poids faible de la force dans la loss

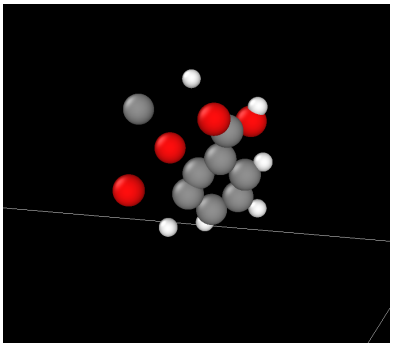
— Instabilité 2 :
  — Grand nombre de données d'entrainement
  — Taille de batch faible
  — Nombre de features élevé
  — Poids faible de la force dans la loss

6.2 Observation sur la molécule d'aspirine MD17

Nous pouvons tester ces observations sur la molécule d'aspirine en l'entraînant avec 1000 et 10 000 molécules d'entraînement, en utilisant les paramètres mentionnés précédemment qui maximisent l'instabilité de type 2. En testant chaque modèle sur 100 molécules, nous observons une explosion de l'énergie de plus de 95% pour des pas de temps inférieurs à 100 000 pour les deux modèles.







7 CONCLUSION

Il a été observé deux types d'instabilités dans la dynamique apprise par le réseau neuronal à graphes PhysNet (GNN). La première est une divergence progressive de l'énergie, en partie attribuable à la méthode de Verlet, tandis que la seconde correspond à une augmentation soudaine de l'énergie de la molécule, pouvant résulter d'un détachement complet des atomes ou d'un changement de conformation. Peu importe leur origine, ces instabilités ne sont pas physiquement acceptable et doivent être minimisées pour garantir la viabilité de l'utilisation des réseaux de neurones dans la dynamique moléculaire. Il convient de noter qu'avec la combinaison optimale des paramètres étudiées, nous observons tout de même une explosion dans 4% des cas avant 100 000 pas de dynamique. Les causes de ces instabilités résiduelles peuvent être multiples, telles que les paramètres non testés ou même l'architecture du GNN elle-même. Ainsi, une nouvelle approche est nécessaire si l'on souhaite maintenir l'utilisation des GNN pour l'apprentissage de la dynamique

8 OUVERTURE

L'intégration de la hessienne dans la prédiction par les réseaux de neurones représente une avancée significative dans la modélisation de données complexes. En pratique, cela implique l'extension du modèle pour qu'il retourne non seulement la prédiction de base ou la dérivée première (force et énergie), mais aussi la dérivée seconde ou hessienne. Mathématiquement, la hessienne, qui est le tableau des dérivées secondes d'une fonction, offre une vue détaillée de la courbure locale de la surface de réponse modélisée par le réseau. En ajoutant cette information lors de l'entraînement, on améliore la capacité du modèle à appréhender la géométrie sous-jacente de l'espace des données.

L'utilisation de la hessienne est particulièrement avantageuse pour prévenir les problèmes liés aux puits de potentiel et à d'autres discontinuités non linéaires, qui peuvent entraîner des instabilités ou des erreurs de généralisation, notamment dans les zones peu denses de l'espace des données où le modèle peut surajuster. Techniquement, la prédiction de la hessienne peut être réalisée en employant la fonctionnalité d'autograd pour ajouter automatiquement cette matrice aux données d'entraînement, ceci permettrait une prédiction plus fiable face aux variations locales des entrées, réduisant ainsi les instabilités

RÉFÉRENCES

[1] Stukowski, Alexander. Visualization and analysis of atomistic simulation data with OVITO-the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering*, 18(1), 2010.

[2] Gabriel Stoltz. An Introduction to Computational Statistical Physics. Pages 34-46.

[3] Ramil, Mouad and Boudier, Caroline and Goryaeva, Alexandra M. and Marinica, Mihai-Cosmin and Maillet, Jean-Bernard. On Sampling Minimum Energy Path. *Journal of Chemical Theory and Computation*, 2022, 18(10) :5864-5875.

[4] Maxime Labonne. Hands-On Graph Neural Networks Using Python : Practical techniques and architectures for building powerful graph and deep learning apps with PyTorch. Book.