JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

Rapport Technique

Découverte de matériaux donneurs à l'aide de l'apprentissage automatique : une approche par réseau de neurones profonds

---

Membres :
AMADOU SOUMANOU Omonlabakè Raliatou
CHERIF Wissem
CODO Loth-Marie Jg
MEVAH MBELLE Marlene Tatiana
RAOUMBE Axel

Encadré par:
Kekeli N'KONOU

Academic year : 2024 – 2025

Contents

1 Introduction

1.1 Contexte

La découverte de nouveaux matériaux joue un rôle central dans l'accélération de l'innovation technologique, en particulier dans le domaine de l'efficacité énergétique. Alors que la demande mondiale de solutions énergétiques durables augmente, les cellules solaires organiques (OSC) apparaissent comme une alternative prometteuse aux cellules photovoltaïques traditionnelles. Ces dispositifs se distinguent par leurs nombreux avantages, notamment leur faible coût de fabrication, leur flexibilité mécanique, leur facilité de production à grande échelle et leur faible toxicité. Cependant, l'amélioration de l'efficacité de la conversion de puissance (PCE) reste un défi majeur.

La qualité des matériaux constituant la couche active des OSC, composée de matériaux donneurs (polymères et petites molécules) et d'accepteurs, est un facteur décisif de leur performance. La découverte et l'optimisation de matériaux donneurs performants nécessitent des cycles expérimentaux longs, coûteux et gourmands en ressources, ce qui entrave l'adoption généralisée de cette technologie.

Dans ce contexte, l'intégration de techniques d'apprentissage automatique, en particulier l'apprentissage profond, offre une solution innovante. Celles-ci peuvent modéliser des données complexes et temporelles, ce qui les rend particulièrement adaptées à la prédiction des propriétés des matériaux et à la génération de nouvelles structures moléculaires optimisées pour maximiser le PCE.

1.2 Objectifs du projet

L'objectif principale de ce projet est d'accélérer la découverte de nouveaux matériaux donneurs pour les OSC en utilisant des techniques d'apprentissage automatique. En particulier, nous visons à développer un modèle basé sur l'apprentissage profond capable de prédire efficacement le PCE des matériaux donneurs. En parallèle, nous utiliserons des algorithmes de génération moléculaire pour créer de nouvelles structures moléculaires optimisées pour les OSC. En vu du développement de ce projet, nous écrivons un rapport technique dans le but de permettre à quiconque de prendre en main le modèle d'apprentissage développé et de l'utiliser pour prédire les propriétés des matériaux donneurs, en particulier le PCE.

2 Architecture du modèle

Pour répondre à la problématique de la découverte de nouveaux matériaux donneurs pour les OSC, nous nous appuyons sur les travaux de **Jinyu Sun et al.**[1] qui ont développé un modèle d'apprentissage profond pour la génération de molécules et du QuantumDeepField de **Masashi et al.**[2] capable de prédire les propriétés électroniques des molécules.

2.1 Modèle de prédiction de propriétés (PCE)

Dans ce modèle, nous parlerons du **Quantum Deep Field** (QDF) pour prédire les propriétés des matériaux donneurs telles que le PCE. L'architecture du modèle repose sur deux réseaux de neuronnes profonds majeures : le réseau de neurones de prédiction de propriétés et le réseau de neurones basés sur des contraintes physiques. L'architecture du modèle est illustrée ci-dessous.

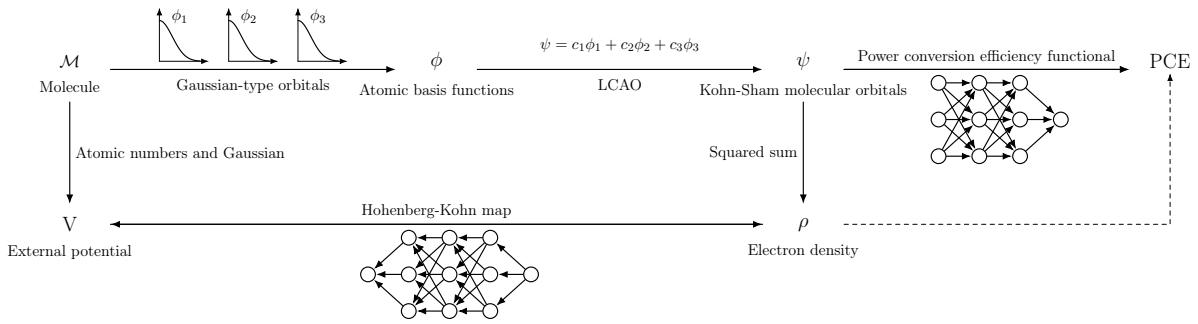

Figure 1: Quantum Deep Field (QDF) pour la prédiction de PCE

2.1.1 Représentation de la molécule

La molécule d'intérêt, notée $\mathcal{M}$, est d'abord représentée sous forme de coordonnées atomiques et de numéros atomiques. Cette représentation sert de point de départ pour la génération des fonctions de base atomiques.

2.1.2 Construction des orbitales atomiques de type gaussien

À partir de la structure moléculaire, des orbitales atomiques de type gaussien (ϕ_1, ϕ_2, ϕ_3, ...) sont générées. Ces fonctions de base servent à décrire la distribution électronique autour de chaque atome.

2.1.3 Combinaison linéaire des orbitales atomiques (LCAO)

Les orbitales atomiques sont combinées linéairement pour former les orbitales moléculaires de Kohn-Sham (ψ), selon la méthode LCAO ($\psi = c_1\phi_1 + c_2\phi_2 + c_3\phi_3$). Cette étape permet de modéliser la structure électronique de la molécule.

2.1.4 Densité électronique et HK_map

La densité électronique (ρ) est obtenue en sommant les carrés des orbitales moléculaires. Cependant, minimiser uniquement la fonction de perte $\mathcal{L}_{PCE}$ ne garantit pas que le modèle appris soit physiquement significatif.

En effet, en raison de la forte non-linéarité du réseau de neurones profond (DNN), il est possible d'obtenir un PCE correcte même si les orbitales de Kohn-Sham ψ ne sont pas valides.

Cela signifie que le modèle ne garantit pas que les orbitales $\psi(\mathbf{r})$ produisent la bonne densité électronique $\rho(\mathbf{r}) = \sum_{n=1}^{N} |\psi_n(\mathbf{r})|^2$.

Pour remédier à ce problème, une contrainte supplémentaire est imposée sur $\psi(\mathbf{r})$ basée sur le théorème de Hohenberg-Kohn, qui assure que le potentiel externe $V(\mathbf{r})$ est une fonction unique de la densité électronique.

2.1.5 PCE

Le PCE est calculé à partir des orbitales moléculaires. Il est exprimé par un DNN, ce qui permet d'évaluer l'efficacité de conversion d'énergie de la molécule.

En somme, on a 2 prédictions princiaples effetuées par le modèle :

- **Prédiction du potentiel** $V(\mathbf{r})$: La première, qui est une prédiction sur le potentiel,sert de contraintes physiques pour ajuster le modèle de prédiction de PCE.
- **Prédiction du PCE** : La deuxième prédiction est le PCE de la molécule.

2.2 Modèle de genération de molécules

Ce modèle repose sur une approche par modélisation générative utilisant un Auto-Encodeur Variationnel (VAE), combinée à une préparation des données chimiques. L'objectif est de créer un système capable de représenter fidèlement les structures moléculaires dans un espace latent continu, puis d'utiliser ces représentations pour générer de nouvelles structures prometteuses en termes d'efficacité photovoltaïque

Le choix du VAE s'explique par sa capacité à encoder des données complexes dans un espace latent structuré, tout en permettant la génération d'exemples nouveaux par simple échantillonnage. Dans notre cas, cela permet de transformer des chaînes SMILES en vecteurs continus, puis de régénérer des chaînes moléculaires viables à partir de ces vecteurs. Le VAE est particulièrement utile pour éviter les erreurs syntaxiques communes lors de la génération de nouvelles molécules.

Le modèle VAE utilisé dans ce projet se compose de trois parties principales :

- **Encodeur** : transforme les SMILES vectorisés en une distribution latente gaussienne.
- **Espace latent** : échantillonnage dans cet espace pour produire des représentations moléculaires nouvelles.
- **Décodeur** : reconstruit des SMILES valides à partir des vecteurs latents.

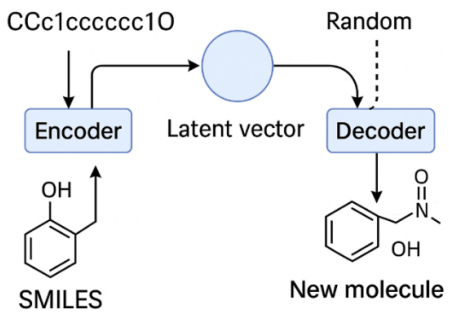

Figure 2: Architecure VAE

Cette architecture permet non seulement de reproduire fidèlement les molécules du dataset, mais aussi de générer des candidats inédits pour les OPV.

Chaque molécule passe par plusieurs étapes :

- **Nettoyage des données** : vérification de l'intégrité des SMILES, suppression des anomalies et standardisation.
- **Vectorisation** : conversion caractère par caractère ou via des embeddings, pour produire des vecteurs traitables par les modèles.
- **Normalisation** : application de padding et de mise à l'échelle, pour assurer une homogénéité des entrées.

Ces transformations permettent au réseau d'extraire les caractéristiques pertinentes des molécules tout en préservant leur validité chimique.

2.3 Architecure globale du modèle

L'architecture globale du modèle combine les deux parties précédentes : le modèle de prédiction de propriétés et le modèle de génération de molécules.

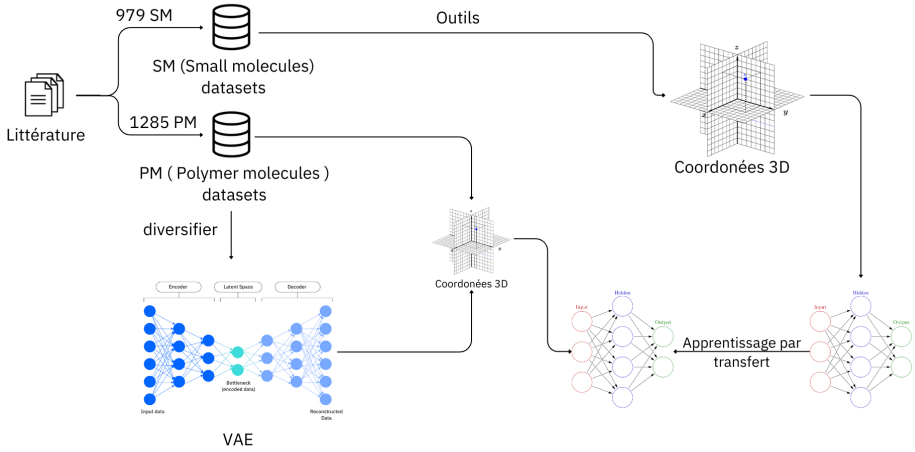

Figure 3: Architecure globale du modèle

On peut voir sur le modèle qu'il y a une série d'étape pour notre achitecture :

- **Tranformations des molécules en 3D** : La première chose à faire est de transformer les molécules en coordonnées pour l'entrainement du modèle.
- **Entrainement du modèle QDF-SM** : Entrainer dans un premier le modèle sur les molécules SM tranformées.
- **Transfert Learning du modèle QDF-PM** : Faire de l'apprentissage par transfert sur les molécules PM tranformées.
- **Génération de molécules** : Génération de nouvelles molécules qui sont à tester sur les modèles entrainés (pour la prédiction).

En résumé, l'architecture proposée combine un modèle génératif basé sur un Auto-Encodeur Variationnel (VAE) pour la création de nouvelles structures moléculaires et un modèle de prédiction de propriétés (Quantum Deep Field) pour évaluer leur efficacité photovoltaïque. Cette approche intégrée permet non seulement d'explorer efficacement l'espace chimique à la recherche de nouveaux matériaux donneurs, mais aussi de prédire les propriétés de matériaux tels que le PCE.

3 Datasets

3.1 Overview sur les Datasets

Les datasets utilisés dans ce projet sont issus du dépôt GitHub de Jinyu Sun et al.[1] sur le DeepDonor. Dans ce dépôt on trouve deux types de datasets : SM (Small Molecules) et PM (Polymer Molecules). Pour pouvoir évalué le modèle plus tard il nous fallait les mêmes données que celles utilisées par Jinyu Sun et al.[1]

Le datset SM est constitué de **979 SM** tandis que le datasets PM est constitué de **1285 PM** et comportent des molécules qui proviennent de la littérature. Ils sont sont au format **csv** et dans chacun d'eux on retrouve les molécules encodés en **SMILES** et leurs **PCE** respectifs.

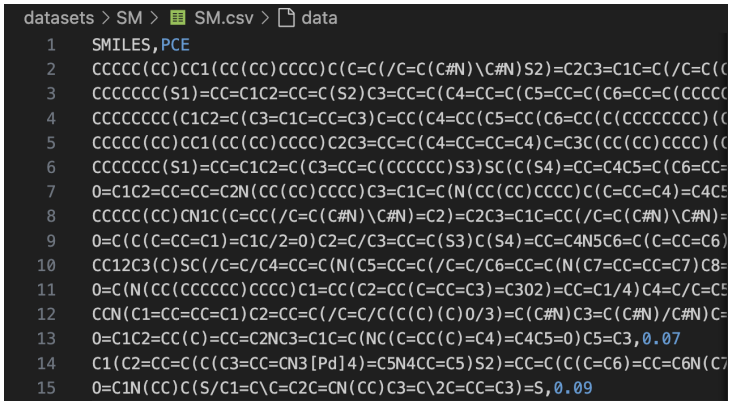

Figure 4: SMILES et PCE

Comme on peut le voir sur la figure ci-dessus, chaque ligne du fichier csv correspond à une molécule encodé en SMILES associées à leurs PCE respectif.

Un point important à noter est le fait qu'il y a une différence entre les PM et les SM. Cette différence réside au niveau de la masse moléculaires. Ainsi, ce point est intéressant à prendre en compte pour éviter toute confusion pour la suite.

3.2 Datasets de coordonnées 3D

Les datasets évoqués précédement servent de point d'entrée pour le modèle de prédiction (Quantum Deep Field). Pour permettre au modèle de prédiction de capter la structure moléculaires des molécules il faut des données numériques d'où l'idée de transformer les SMILES en coordonnées 3D. De plus, à partir de ces données numériques ils pourraient être possibles de prédire d'autres propriétés électroniques liés aux molécules.

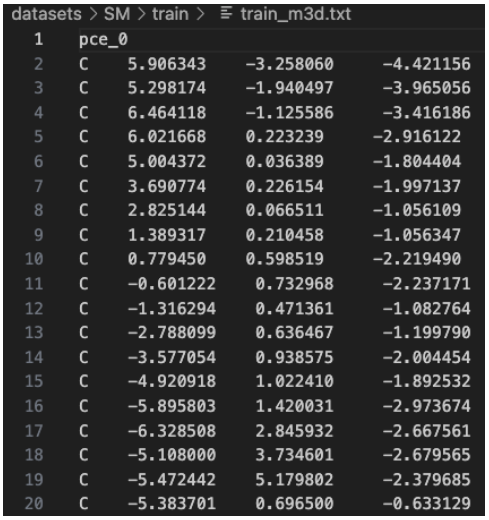

Figure 5: Coordonnées 3D d'une molécule

Pour chaque molécules, on génére les coordonnées 3D de chaque atomes. Ensuite on récupère ces coordonnées 3D on effectue un prétraitement pour les transformer en fichier `.npy` pour qu'ils puissent enfin être utilisé par le modèle de prédiction.

4 Prérequis

Pour la suite de ce document, l'ensemble du projet a été réalisé avec un environnement virtuel (conda).

4.1 Vérification de la présence d'un GPU et installation de CUDA

Pour profiter de l'accélération matérielle (ce qu'on vous recommande fortement) lors de l'entraînement de modèles, il est important de vérifier si votre ordinateur dispose d'un GPU compatible.

- **Sous Windows :** Ouvrez le gestionnaire de tâches (`Ctrl+Shift+Esc`) et vérifiez l'onglet "Performances" pour voir si un GPU nvidia est détecté.

```
lspci | grep -i nvidia
```

Installation de CUDA :

1. Rendez-vous sur le site officiel de NVIDIA : `https://developer.nvidia.com/cuda-downloads`. Pour l'Installer typeprenez la version local
2. Sélectionnez votre système d'exploitation et suivez les instructions d'installation.
3. Après installation, vérifiez la version de CUDA avec la commande :

```
nvcc --version
```

Remarque : Assurez-vous que la version de CUDA installée est compatible avec la version de PyTorch que vous souhaitez utiliser (voir la documentation officielle de PyTorch).

4.2 Installation de Pytorch

Avant de commencer, munissez vous d'un environnement virtuel et assurez-vous d'avoir installé les dépendances nécessaires. L'ensemble des dépendances se retrouvent dans un fichier `requirements.txt` et sont facilement installables via `pip` ou `conda`. Activez votre environnement virtuel et vous pourrez commencez à utiliser les commandes `bash` du guide utilisateur sur votre terminal. Par ailleurs, une dépendance importante à prendre en compte est la library avec laquelle sont écrit les réseaux de neuronnes profonds : **Pytorch**.

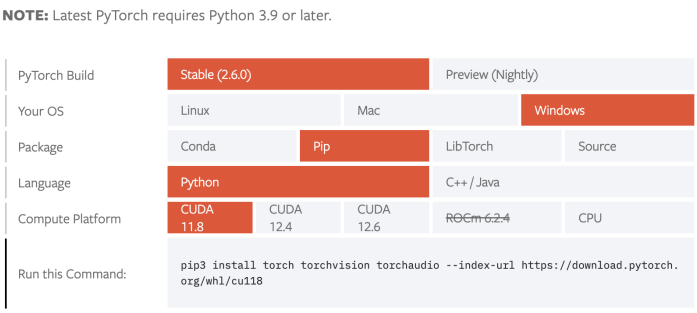

Figure 6: Page de téléchargement de Pytorch

Pour vérifier la disponibilité du GPU avec `PyTorch`, lancez dans un terminal Python :

```
import torch
print(torch.cuda.is_available())
```

Si la commande retourne `True`, votre GPU est prêt à être utilisé.

5 Guide utilisateur

Dans cette section, nous allons vous guider sur la prise main de l'ensemble du projet. Nous allons vous expliquer comment utiliser le modèle QDF dans les grandes lignes en une série d'étapes pour assurer la cohérence de la pipeline et prédire les propriétés des matériaux donneurs, en particulier le PCE. En ce qui concerne l'explication du code des notebooks au format `.ipynb` ainsi que les commentaires présents dans le code, ont été mis à disposition pour comprendre le fonctionnement de ce derniers. Aussi, nous nous accentuerons sur les traitements des deux datasets principaux que sont les PM (polymers molecules) et les SM (small molecules) et comment on parvient à les utiliser dans le modèle.
NB : On tient à rappeler que l'ensemble des commandes qui seront présentées par la suite ont été effectuées sur windows.

5.1 Arbre du projet

L'Arbre du projet est organisé de manière à faciliter la navigation et l'utilisation des différents composants du projet. Voici un aperçu de la structure de l'arborescence :

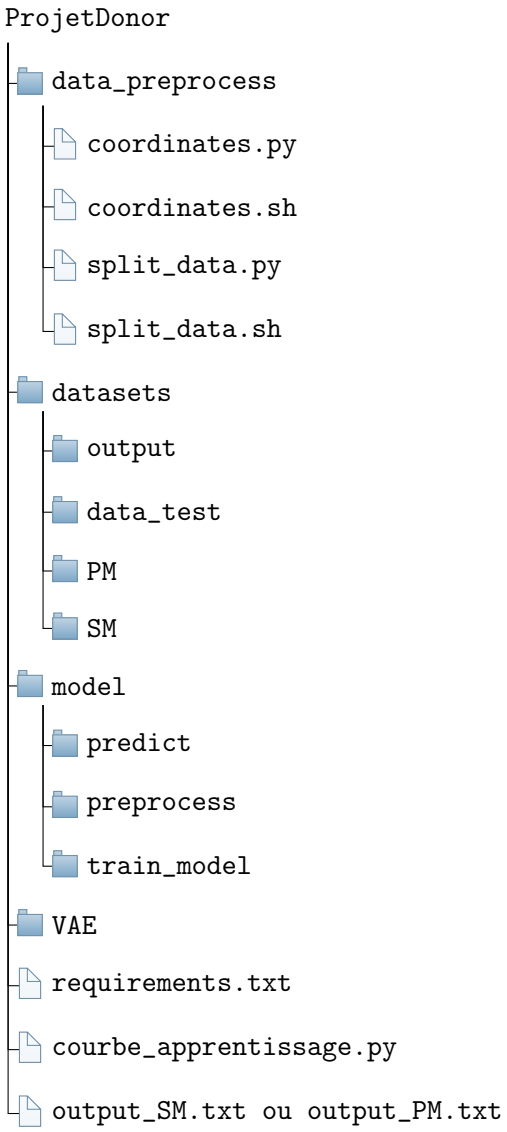

Figure 7: Arborescence du projet

Data_preprocess : Le dossier `data_preprocess` contient les scripts nécessaires pour prétraiter les données. En effet, la forme principale des molécules des datasets pour les PM (polymers molecules) et SM (small molecules) sont au format **SMILES** il va donc falloir les traiter pour les passer sous forme de coordonnées 3D en utilisant **RdKit**.

Datasets : Le dossier `datasets` contient les dossiers `PM` et `SM` eux même qui contiennent chacun les datasets (des PM et SM) encodés sous forme SMILES et les coordonnées 3D associées à ces datasets pour l'entrainement, la validation et le test. Le dossier `output` contient les résultats de la prédiction du PCE pour les PM et SM ainsi que les modèles d'entrainement associés.

Model : Le dossier `model` contient les scripts nécessaires pour l'entrainement du modèle QDF, la prédiction du PCE et le preprocess des données réalisés sur les datasets PM.csv et SM.csv.

Avant de commencer à expliquer les principales étapes du projet, il est important de prendre en compte le fait qu'au départ seuls les dossiers `PM` et `SM` contenant les fichiers `PM.csv` et `SM.csv` sont contenus dans le dossiers datasets. Au fur et à mesure qu'on effectue les étapes la strucutre change en rajoutant des dossiers.

5.2 Séparation du dataset

La première étape du projet consiste à réaliser une séparation du dataset (PM.csv ou SM.csv) que l'on veut utiliser en trois parties : une partie pour l'entrainement, une partie pour la validation et une partie pour le test. Pour cela on utilise le script `split_data.sh` qui se trouve dans le dossier `data_preprocess`.
Le contenu du script `split_data.sh` est le suivant :

Figure 8: Contenu du script split_data.sh

Dans cette image on peut voir que l'on utilise le script `split_data.py` qui se trouve dans le même dossier. Le premier argument (datapath) est le nom du fichier CSV à traiter (PM.csv ou SM.csv) et les autres arguments sont les noms des des dossiers de sortie.

Pour exécuter le script, il faut ouvrir un terminal **bash** sur un éditeur comme VS code, accéder au dossier data_preprocess avec la commande `cd data_preprocess` et lancer le scritpt depuis ce terminal avec la commande : `bash split_data.sh` sur windows.
NB : Si on veut fait un split des données sur le dataset PM il faudra juste remplacer SM par PM.

5.3 Transformation des SMILES en coordonnées 3D

La deuxième étape du projet consiste à transformer les SMILES en coordonnées 3D. Pour cela, on utilise le script `coordinates.sh` qui se trouve dans le dossier `data_preprocess`. Celà permettra d'avoir une base numérique pour l'entrainement des données sur le QDF.
Le script `coordinates.sh` est le suivant :

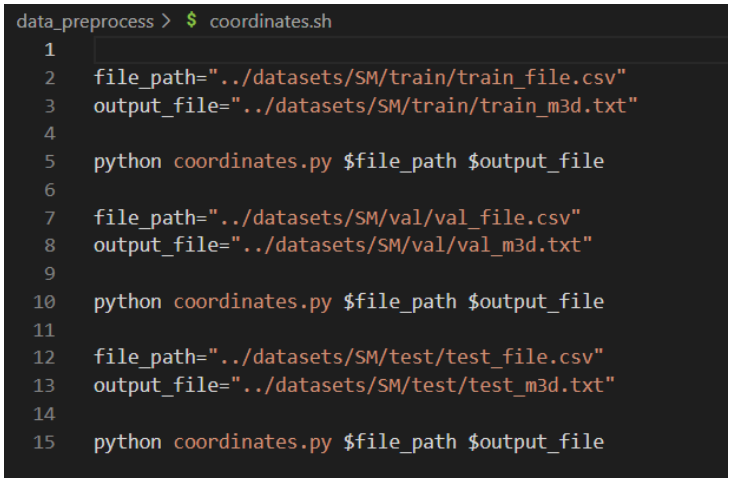

Figure 9: Contenu du script coordinates.sh

Sur l'image on peut voir que chaque variable `file_path` correspond au fichier créé lors de la séparation du dataset tandis que les variables `output_path` correspondent aux fichiers de sortie qui contiendront les coordonnées 3D des molécules.

5.4 Préparation des données pour le modèle

La troisième étape du projet consiste à préparer les données pour le modèle QDF. Pour cela, on utilise le script `preprocess.sh` qui se trouve dans le dossier `model/preprocess`.

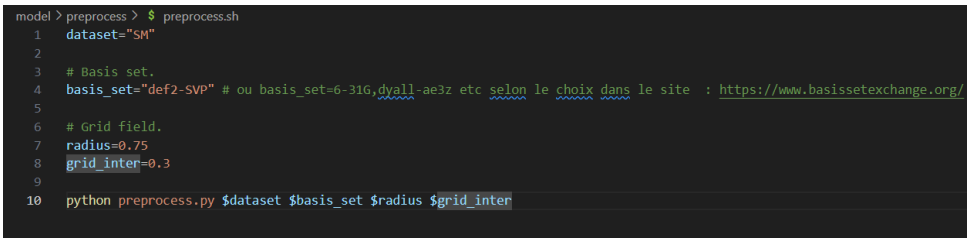

Figure 10: Contenu du script preprocess.sh

La première variable `dataset` sert à définir le nom du dataset surlequel on souhaite faire du preprocessing. La deuxième variable `basis_set` sert à définir l'ensemble de fonctions de bases utilisé. Les variables `radius` et `grid_size` servent à définir le rayon et la taille de la grille utilisée pour le champ électrique associé à chaque atomes.

5.5 Entrainement du modèle

La quatrième étape du projet consiste à entrainer le modèle QDF. Pour cela, on utilise le script `QDF_SM.sh` ou `QDF_PM.sh` qui se trouve dans le dossier `model/train_model`. Un ensemble de plusieurs paramètres sont à définir dans le script `QDF_SM.sh` ou `QDF_PM.sh` :

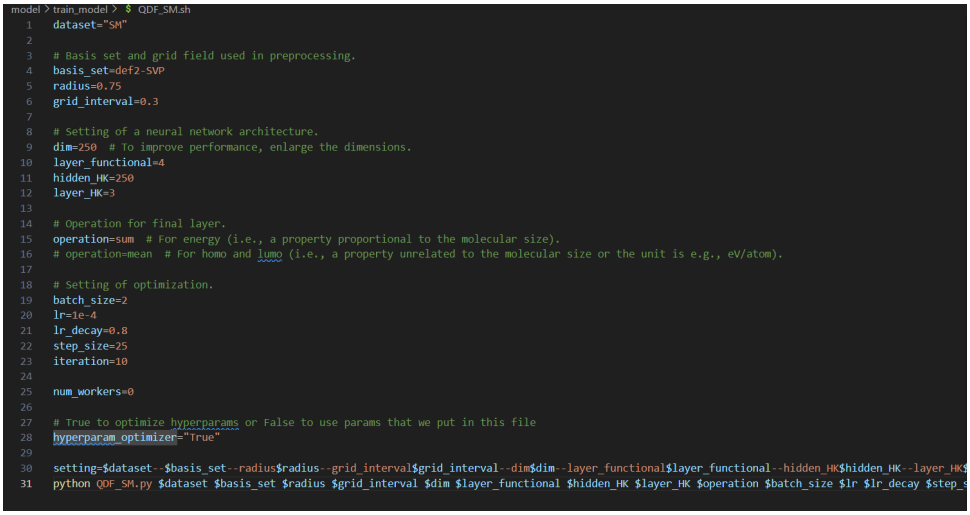

Figure 11: Variable contenu dans QDF_SM.sh

Une variable importante qui influence la façon dont le modèle va apprendre est `hyperparam_optimizer`. Les hyperparamètres sont enregistrés dans un fichier qui servira de transfert learning pour l'entrainement du modèle QDF pour les PM.

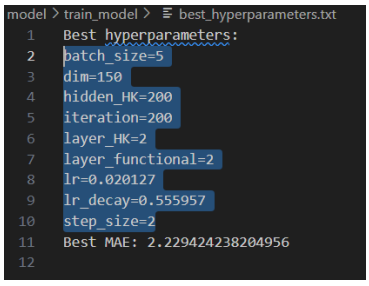

Figure 12: Hyperparamètres optimisés

On utilise les hyperparamètres optimisés pour le modèle QDF sur les SM pour entrainer le modèle QDF sur les PM.

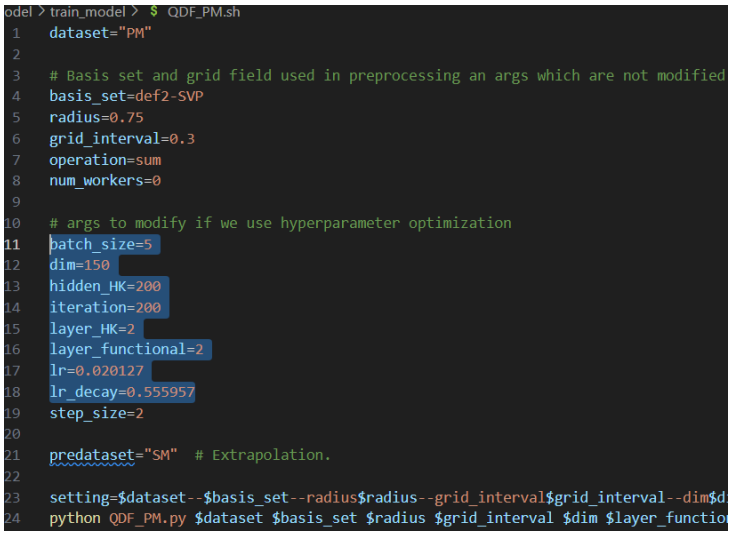

Figure 13: Variable contenu dans QDF_PM.sh

Il suffit juste de faire un copier de la partie surligner dans le fichier des hyperparamètres générés lors de l'entrainement du modèle QDF sur les SM et de le coller dans le fichier QDF_PM.sh. Les modèles d'entrainement sont sauvegardés dans le dossier `datasets/output`.

5.6 Prédiction du PCE

La dernière étape du projet pour le modèle de prédiction consiste à prédire le PCE des PM et SM grâce au modèle qu'on a pu entrainer. Pour cela, on utilise le script `predict.sh` qui se trouve dans le dossier `model/predict`. Avant cela, il y a un prétraitement de données à faire , si les données reçues sont sous forme de SMILES, avec le script `preprocess_predict.sh` se trouvant dans le même répertoire que `model/predict`.

5.7 Courbe d'apprentissage

La courbe d'apprentissage est un outil essentiel pour évaluer la performance d'un modèle au fil du temps. Pour pouvoir la visualiser, il faut exécuter script python `courbe_apprentissage.py` qui se trouve à la racine. Avant de le faire il faudrait s'assurer que le fichier `output_SM.txt ou output_PM.txt` contiennent des données si ce n'est pas le cas il faut aller dans dossier `datasets/output` et faire un copier-coller du fichier `SM-def2-SVP-radius0.75...` dans output_SM.txt ou output_PM.txt.

datasets > output > SM--def2-SVP--radius0.75--grid_interval0.3--dim200--layer_functional1--hidden_HK200--layer_HK3--s

	Epoch	Time(sec)	Loss_PCE	Loss_V	MAE_val(SM)	MAE_test(SM)	Acc_val(SM)	
2	0	1.7773805999999013	116.93929290771484	3.6192049980163574	173.94134521484375	973.0284423828125		
3	1	4.044123499999955	1529.330810546875	3.2063369750976562	148.37893676757812	1016.4052734375	0	
4	2	5.750595599999997	701.3773193359375	1.9248249530792236	122.9163589477539	879.2020874023438		
5	3	7.457661499999972	233.96875	1.3276323080062866	98.09133911132812	675.9686279296875	0.0	
6	4	9.238991199999987	[illegible]		0.8020928502082825	78.78373718261719	333.69622802734375	
7	5	11.156356799999912	[illegible]		1.050991415977478	84.75944519042969	125.65990447998047	
8	6	12.969546899999955	241.84622192382812	1.1403656005859375	55.636688232421875	159.93605041503906		
9	7	14.774868599999999	107.0981674194336	0.7589944005012512	32.60285516479492	192.89100646972656		
10	8	16.625091099999963	148.8047637939453	0.9209372401237488	30.5946044921875	162.35070800781250		
11	9	18.347586099999944	95.59933471679688	0.956247866153717	38.82440948486328	82.00315093994140		
12	10	20.03835159999994	35.67166519165039	0.6742594242095947	59.839439392089844	82.08391571044922		
13	11	21.813433399999894	40.31548309326172	0.8505983948707581	34.44364547729492	107.34072113037109		
14	12	23.521208099999967	22.682964324951172	0.8898800611495972	26.702064514160156	179.11407470703125		
15	13	25.221491899999933	4.090410232543945	0.7223723530769348	40.75896453857422	216.35150146484375		
16	14	26.929018099999894	31.80722427368164	0.7240160703659058	64.1640625	168.348876953125	0.0	
17	15	28.676618299999973	49.98815155029297	0.8390151262283325	79.44902038574219	117.20204925537109		
18	16	30.38933189999989	53.774864196777344	0.7063614726066589	69.64967346191406	115.00826263427734		
19	17	32.114787999999976	42.92646026611328	0.7060925960540771	46.08359909057617	77.39715576171875		
20	18	33.79310129999999	37.15785598754883	0.7863596081733704	32.28407669067383	83.859375	0.0	

Col 3: Loss_PCE

Figure 14: Contenu des données d'entrainement de datasets/ouput à copier-coller

On peut voir sur le graphe que les métrics utilisés pour ce modèle sont la perte et l'accuracy.

5.8 Demo pour le modèle de prédiction

En ce qui concerne la démo pour le modèle de prédiction, l'utilisateur pourra exécuter les fichiers `.ipynb` qui se trouvent dans les dossiers `data_preprocess` et `model/preprocess` dans l'ordre des étapes décrites précédemment. Une fois traitement des données réalisés une autre notebook présent dans le dossier `model/train_model` sert uniquement à expliquer certaines composantes du projet. Pour avoir une démonstration de l'entrainement, faire du transfert learning et prédire il faudrait alors utiliser les scripts `QDF_SM.sh` et `QDF_PM.sh` qui se trouvent dans le dossier `model/train_model` ainsi que ceux dans le dossier `predict`.

5.9 VAE

Le pipeline d'entraînement est structuré en plusieurs modules opérationnels :

- **Chargement et nettoyage** : lecture des fichiers, suppression des erreurs et normalisation.
- **Encodage vectoriel** : transformation des SMILES en séquences compatibles avec le réseau neuronal.

- **Modélisation** : apprentissage d'une représentation latente des molécules à l'aide d'un modèle génératif.
- **Prédiction** : estimation de la Power Conversion Efficiency (PCE) en sortie du modèle.
- **Analyse et visualisation** : évaluation des performances, affichage graphique, génération de nouvelles structures.

Pour pourvoir exécuter le modèle, il faur se rendre dans le dossier `VAE` et exécuter le script `vae_essaie.py` qui se trouve dans ce dernier depuis un terminal bash avec `python vae_essaie.py` en activant l'environnement virtuel avant. Un notebook est présent pour expliquer les fonctions du modèle en plus des commentaires.

5.10 Conseils d'utilisation

5.10.1 GPU

Lors de l'entrainement du modèle de prédiction, il est crucial d'utiliser un GPU suffisamment puissant et de disposer d'une quantité de mémoire RAM adaptée, que ce soit sur un PC personnel ou sur un serveur distant. En effet, l'entraînement de modèles de deep learning, en particulier sur des jeux de données volumineux ou des architectures complexes, nécessite des ressources matérielles importantes pour garantir des temps de calcul raisonnables et permettre l'utilisation de batchs de grande taille. Pour l'entrainement de notre modèle de prédiction on a utilisé un server en ligne (OVHCloud) avec une **NVIDIA L40 S** avec 45 Go de mémoire GPU.

5.10.2 Basis set

Le choix de la base de fonction est important pour la prédiction du PCE. Pour ce projet on a utilisé la base de fonction **def2-SVP** mais il est possible de le changer en fonction de la préférence. On l'a utilisé dans ce projet, car il couvrait un grand nombre d'élément chimique et est plus précis que le **6-31G**.

6 Axes d'améliorations

6.1 Suggestion modèle de prédiction

6.1.1 GPU

Comme il a été mentionné précédemment, on utilise une **Nvidia L40 S**. Même avec cette configuration, on est limité en ce qui concerne la **batch_size** de notre modèle. La taille du lot (batch size) est un hyperparamètre qui détermine le nombre d'échantillons à traiter avant de mettre à jour les poids du modèle. Pour l'augmenter, il faudrait utiliser plusieurs GPU et faire probablement de l'entrainement distribué pour permette de traiter plusieurs lots et potentiellement améliorer le modèle.

NB : Si il le GPU n'est pas assez puissant lors de l'exéction du modèle, il serai possible que des erreurs liés à la mémoire apparaissent. Souvent des erreurs de type `CUDA out of memory` apparaissent.

6.1.2 Hyperparamètres

Pour ce projet, on a utilisé un algorithme d'optimisation des hyperparamètres (ex : batch_size) via la librairie `hyperopt`. Cependant, on a été limité en termes de temps ce qui a valu une dimunution de l'espace de recherche de ces hyperparamètres. À l'avenir il serait utile d'agrandir l'espace de recherche des hyperparamètres avec des valeurs plus grandes dans l'espace de recherche pour étendre la recherche des meilleures hyperparamètres.

```
print("Commence l'optimization des hyperparamètres.\n'")

# Définir l'espace d'hyperparamètres
space = {
    'dim': hp.quniform('dim', 100, 500, 50),
    'layer_functional': hp.quniform('layer_functional', 1, 5,
    'hidden_HK': hp.quniform('hidden_HK', 50, 300, 50),
    'layer_HK': hp.quniform('layer_HK', 1, 4, 1),
    'lr': hp.loguniform('lr', -10, -3),
    'lr_decay': hp.uniform('lr_decay', 0.5, 0.99),
    'step_size': hp.quniform('step_size', 2, 24, 2),
    'iteration': hp.quniform('iteration', 200, 400, 50),
    'setting': setting,
    'operation': operation,
    'N_orbitals': len(orbital_dict),
    'N_output': N_output,
    'batch_size': hp.quniform('batch_size', 2, 5, 1),
    'data' : {
        'train': dir_dataset + 'train/train_' + field,
        'val': dir_dataset + 'val/val_' + field,
        'test': dir_dataset + 'test/test_' + field
    }
}
```

Figure 15: Espace de recherche des Hyperparamètres

6.1.3 Fonction de pertes du modèle et fontion d'activation

La fonction de perte est un élément clé dans l'entraînement d'un modèle d'apprentissage automatique. Elle mesure la différence entre les prédictions du modèle et les valeurs réelles. Dans le cas de la prédiction de PCE, la fonction de perte utilisée est la **MAE** (Mean Absolute Error). Le MAE est une mesure linéaire de la différence entre les valeurs prédites et les valeurs réelles. Pour espérer mieux extrapoler les données, il serait intéressant d'utiliser une fonction de perte

non linéaire comme la **Huber Loss ou RMSE** qui est moins sensible aux valeurs aberrantes que la MAE.

Pour les fonctions d'activation, il serait utile de les changer pour vérifier l'effet l'obtenu sur la prédiction. Notamment, celle présente à la couche de sortie du DNN pour prédire de PCE, puisque le PCE doit être strictement positif on pourrait envisager uen fonction d'activation comme **ReLU** par exemple.

6.2 Modèle de génération

6.2.1 Traduction SMILES en SELFIES

L'une des principales difficultés rencontrées concerne la conversion des représentations SMILES en SELFIES (Self-Referencing Embedded Strings). Bien que SELFIES soit plus robuste pour la génération de structures valides, sa conversion depuis des SMILES complexes n'est pas toujours triviale. Certains encodages échouent ou génèrent des représentations ambiguës, nécessitant une étape supplémentaire de validation et de correction. Cela introduit une complexité additionnelle dans le pipeline de traitement et peut réduire la qualité des entrées pour le modèle génératif.

6.2.2 Génération de molécules via NO-Operation (NOP)

Lors de la génération de nouvelles structures moléculaires, nous avons observé que certains modèles avaient tendance à produire un grand nombre de caractères de type NO-Operation (NOP), c'est-à-dire des éléments neutres ou répétitifs sans signification chimique. Cela peut résulter d'un mauvais apprentissage des règles syntaxiques moléculaires ou d'un sur-apprentissage sur les motifs fréquents. La présence excessive de NOP réduit la diversité et la qualité des molécules générées, ce qui compromet leur validité et leur pertinence.

7 Appendices

7.1 References

References

[1] Jinyu Sun, Dongxu Li, Yue Wang, Ting Xie, Yingping Zou, Hongmei Lu and Zhimin Zhang, *J. Mater. Chem. A*, 2024, DOI: 10.1039/D4TA03944K

[2] Tsubaki, Masashi and Mizoguchi, Teruyasu, *Quantum Deep Field: Data-Driven Wave Function, Electron Density Generation, and Atomization Energy Prediction and Extrapolation with Machine Learning*, Phys. Rev. Lett. **125**(20):206401, 2020, doi: 0.1103/PhysRevLett.125.206401

7.2 Définitions additionnelles ou figures complémentaires

InChI (Identifiant International des Composés Chimiques)

Définition : Une représentation textuelle normalisée d'une molécule chimique, offrant une structure plus rigoureuse que SMILES.

Exemple : Benzène : `InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H`

Orbitales de type gaussien (GTOs)

Définition : Une Orbitale de Type Gaussien (GTO) est une fonction mathématique utilisée comme fonction de base en chimie quantique pour approximer les orbitales atomiques. Au lieu d'utiliser une décroissance exponentielle pure, les GTOs emploient une fonction gaussienne, ce qui simplifie le calcul des intégrales dans les calculs de structure électronique moléculaire.

Formule :

$$\phi(r) = R_l(r)Y_{lm}(\theta, \phi)$$

$\phi(x)$ **:** La fonction orbitale unidimensionnelle, où x est la coordonnée spatiale.

$R_l(r) = Nr^l e^{-\alpha r^2}$ **:** La partie radiale du GTO, où r est la distance au noyau.

N **:** La constante de normalisation.

l **:** Un entier positif ou nul déterminant le type (par exemple, $l = 0$ pour s, $l = 1$ pour p).

α **:** Un paramètre positif contrôlant l'étalement (largeur) de la gaussienne.

$e^{-\alpha r^2}$ **:** La partie gaussienne, assurant une décroissance rapide lorsque $|r|$ augmente.

$Y_{lm}(\theta, \phi)$ **:** Les harmoniques sphériques, qui dépendent des angles θ et ϕ et décrivent la partie angulaire de l'orbitale.

Autoencodeur variationnel : VAE

Définition : Un autoencodeur variationnel est un modèle génératif profond utilisé en apprentissage automatique. Il apprend une représentation latente probabiliste des données d'entrée à l'aide de deux réseaux de neurones :

Un **encodeur** qui projette les données d'entrée vers une distribution latente.

Un **décodeur** qui échantillonne à partir de cette distribution latente pour générer de nouvelles données similaires à l'ensemble d'entraînement.

Les VAE sont particulièrement utiles pour générer de nouvelles structures (molécules, etc.) et explorer l'espace afin de découvrir des motifs intéressants.

Réseau de neurones profond : DNN

Définition : Un réseau de neurones profond (DNN) est un réseau de neurones artificiel composé de plusieurs couches de neurones entre l'entrée et la sortie. Contrairement à un réseau simple avec une ou deux couches cachées, un DNN contient un grand nombre de couches cachées qui lui permettent de modéliser des relations très complexes et de détecter des motifs subtils dans les données. Cela le rend très efficace pour de nombreuses tâches, telles que la reconnaissance d'images, la traduction automatique et la prédiction de comportements.

Quantum Deep Field : QDF

Définition : Le Quantum Deep Field (QDF) est une méthode qui combine la chimie quantique avec des réseaux de neurones profonds pour prédire les propriétés de molécules ou de matériaux. Elle utilise des informations issues de calculs quantiques (telles que les distributions électroniques) et les intègre dans un modèle d'apprentissage profond, permettant des prédictions plus précises et fiables que les approches traditionnelles.

PCE : Rendement de conversion de puissance

Définition : Le rendement de conversion de puissance (PCE, pour Power Conversion Efficiency) est une mesure de l'efficacité avec laquelle un dispositif (par exemple, une cellule solaire) convertit l'énergie incidente (comme la lumière) en énergie utilisable (comme l'électricité).

Fonctionnelles

Définition : En chimie quantique, une fonctionnelle est une fonction qui prend une fonction (par exemple, la densité électronique) comme argument et retourne une valeur scalaire. Les fonctionnelles sont essentielles dans la théorie de la fonctionnelle de la densité (DFT) pour approximer l'énergie totale d'un système.

Base de fonctions (Basis-set)

Définition : Un ensemble de base (basis-set) est un ensemble de fonctions mathématiques utilisées pour décrire les orbitales électroniques dans les calculs de chimie quantique. Le choix de l'ensemble de base influence la précision et le coût des calculs.