# Rapport de Projet GIMA

## Génération de biomolécules anti-inflammatoires et anti-oxydantes

**VELARD Quentin (IM), BOUAOUDA Salma (ID)**

**Ecole des Mines de Nancy**

**Sous la direction de Guenael Cabanes**

# Table des matières

# 1 Introduction

Parmi les objectifs principaux de la recherche en chimie figure la découverte de nouvelles molécules avec des propriétés bien définies. Cependant, l'espace moléculaire étant extrêmement vaste (de l'ordre de $O(10^{60})$), la recherche d'une seule nouvelle molécule peut prendre plusieurs années en laboratoire chimique. Pour résoudre ce problème, de nombreuses méthodes ont émergé grâce à l'évolution technologique et au développement des modèles d'apprentissage profond. En effet, plusieurs travaux de recherche ont exploré cette problématique et ont proposé différents modèles et architectures pour générer automatiquement des molécules. Dans ce rapport, nous allons nous concentrer en particulier sur la génération de molécules anti-inflammatoires (et anti-oxydantes, sous réserve de disposer de données suffisantes), en exploitant des modèles d'apprentissage profond capables d'exploiter l'immensité de l'espace moléculaire et d'accélérer le processus de découverte.

Tous nos codes et recherches sont disponibles sur :**Github**.

## 1.1 Enjeux

### 1.1.1 contexte

Ce sujet se positionne dans le cadre du LUE « Biomolecules 4 Bioeconomy (B4B)» qui s'intéresse à la production de nouvelles biomolécules à destination des marchés agrochimie, biocontrôle, agro-alimentaire, cosmétique, pharmaceutique et médical. Il s'agira dans ce projet de mettre en place des outils basés sur le Deep Learning pour la génération de nouvelles molécules ayant des propriétés anti-oxydantes et anti-inflammatoires. Dans cette optique, nous allons plus précisément étudier la possibilité d'utiliser des Generative Adversarial Networks (GAN) et des Modèle de diffusion, qui sont des d'algorithmes d'apprentissage non-supervisés fréquemment utilisés pour générer de la nouveauté. Nous allons notamment nous baser sur des réseaux pré-entraînés sur d'autres familles de molécules afin d'accélérer le processus d'apprentissage et améliorer la qualité des résultats.

### 1.1.2 Objectifs

Il y a deux axes sur lesquels nous allons nous pencher :

- **Étudier l'état de l'art** sur les approches de deep learning pour la génération de molécules et faire l'inventaire des réseaux pré-entraînés existants.

- **Entraîner une ou plusieurs architectures** de réseaux de neurones pour la génération de biomolécules anti-oxydantes et anti-inflammatoires par "transfert learning" à partir des réseaux pré-entraînés

### 1.1.3 Cadre

Commençons par quelques définitions :

Une **biomolécule anti-inflammatoire** (que l'on notera AI dans la suite) est une molécule d'origine biologique ou synthétique qui a la capacité de réduire ou de moduler

l'inflammation. L'inflammation est une réponse immunitaire de l'organisme face à une infection, une blessure ou une irritation, caractérisée par des symptômes tels que rougeur, gonflement, chaleur, douleur, et parfois une perte de fonction.

- **Caractéristique principale :** *Action sur les médiateurs inflammatoires :* Les biomolécules anti-inflammatoires agissent en inhibant les molécules responsables de l'inflammation, comme les cytokines pro-inflammatoires ou les enzymes.
- **Cibles biologiques :** Ces biomolécules peuvent interagir avec des récepteurs cellulaires ou bloquer des voies de signalisation impliquées dans la réponse inflammatoire.
- **Origines :** Naturelles (Extraits végétaux (curcumine, resvératrol), acides gras oméga-3, peptides bioactifs) ou Synthétiques (Médicaments comme les anti-inflammatoires non stéroïdiens (AINS) ou les corticostéroïdes)

Une **biomolécule antioxydante** (que l(on notera AO dans la suite) est une molécule qui protège les cellules contre les dommages causés par les espèces réactives de l'oxygène (ERO), également appelées radicaux libres. Ces ERO sont produites naturellement dans l'organisme mais peuvent endommager l'ADN, les lipides, et les protéines, contribuant ainsi au vieillissement et à diverses maladies.

- **Caractéristiques principales :**
  - Neutralisation des radicaux libres.
  - Prévention du stress oxydatif.
- **Origines :**
  - *Naturelles :* Vitamines (C, E), Polyphénols (quercétine, flavonoïdes), Caroténoïdes (bêta-carotène), Enzymes (superoxyde dismutase, catalase).
  - *Synthétiques :* Additifs alimentaires (BHT, BHA), Composés pharmacologiques.

Nous allons être ammenés à considérer **plusieurs types de problèmes** au cours de cet étude :

1. Il faut s'assurer du point de vue sens chimique de l'existence des molécules que l'on peut créer.

2. Les propriétés anti-inflammatoire et anti-oxydantes d'une molécule ne peuvent être déterminée qu'à posteriori de leur création (de manière sûre). Les modèles ne saissisent pas toujours la complexité des intéractions biologiques.
3. Il y a extremement peu de données open source sur les molécules AI et/ou AO

Cependant, les avancées en modélisation computationnelle permettent de prédire avec une certaine fiabilité ces propriétés (existence et anti-inflammatoire/anti-oxydante) avant la synthèse expérimentale. Pour discriminer les molécules créées, on s'appuira sur des molécules existantes.

## 1.2 Dataset et SMILES

### 1.2.1 Format SMILES

Le format SMILES (*Simplified Molecular Input Line Entry System*) sera utilisé pour simuler des molécules sur Python.

| Composant | Description | Exemple SMILES |
|---|---|---|
| **Atomes** | Les atomes sont représentés par leurs symboles chimiques (par exemple, `C` pour le carbone, `O` pour l'oxygène). | `C`, `O`, `N`, `Cl` |
| **Liaisons simples** | Les liaisons simples entre atomes sont implicites et ne sont pas représentées par un symbole. | `C-C`, `N-H` |
| **Liaisons doubles** | Représentées par `=` pour une double liaison. | `C=C`, `C=O` |
| **Liaisons triples** | Représentées par `#` pour une triple liaison. | `C#C`, `C#N` |
| **Liaisons quadruples** | Représentées par `$` pour une quadruple liaison (rarement utilisée). | `C$C` |
| **Cycles (anneaux)** | Les cycles sont indiqués par des numéros après un atome, qui se réfèrent à un atome de cycle. | `C1CCCCC1` (cyclohexane) |
| **Branches** | Les branches sont indiquées entre parenthèses `()` et peuvent être ajoutées à n'importe quel atome. | `CC(C)C` (isobutane) |
| **Hydrogènes implicites** | Les hydrogènes liés aux atomes comme le carbone sont implicites et ne sont pas écrits. | `C` (représente `CH4`) |
| **Hydrogènes explicites** | Les hydrogènes peuvent être spécifiés entre crochets, surtout pour les atomes autres que le carbone. | `[CH3]`, `[OH]` |
| **Charges** | Les charges des atomes sont indiquées entre crochets après l'atome avec un signe de charge (`+` ou `-`). | `[NH4+]`, `[O-]` |
| **Stéréochimie** | Les stéréoisomères peuvent être représentés par `@` (configuration) et `/` ou `\` pour les doubles liaisons avec stéréochimie Z/E. | `C[C@H](O)C` (stéréochimie) |
| **Exemple de chaîne linéaire** | Une chaîne simple de carbone avec des hydrogènes implicites. | `CCO` (éthanol) |
| **Exemple de cycle** | Un cycle de six atomes de carbone. | `C1CCCCC1` (cyclohexane) |
| **Exemple d'une molécule avec branche** | Une molécule avec un groupe méthyle attaché à une chaîne principale. | `CC(C)C` (isobutane) |
| **Exemple de molécule plus complexe** | Une molécule avec un groupe carboxyle (acide acétique). | `CC(=O)O` (acide acétique) |

FIGURE 1 – Fonctionnement du format SMILES

### 1.2.2 Dataset

On s'est concentrés sur la recherche de base de données open source ayant des molécules SMILES avec des propriétés AI et/ou AO. Pour cela nous avons exploré differentes

sources de données relatives à la recherche en chimie en ligne. Parmis ces sources, nous pouvons citer :

- *PubChem* : Banque de données américaine de molécules chimiques gérée par le National Center for Biotechnology Information (NCBI). C'est la plus grande base de données chimique open-source.
- *ChEMBL* : Base de données chimique ouverte organisée manuellement de molécules susceptibles de posséder une activité biologique
- *ZINC* : Collection organisée de composés chimiques disponibles dans le commerce.
- *Protein Data Bank (PDB)* : Source de données de biologie structurale et permet en particulier d'accéder à des structures 3D de protéines d'intérêt pharmaceutique.
- *ChemSpider* : Base de données de produits chimiques, propriété de la Royal Society of Chemistry (RSC).
- *MoleculeNet* : Benchmark spécialement conçu pour tester les méthodes d'apprentissage automatique des propriétés moléculaires.
- *DeepChem* : Chaîne d'outils open source de haute qualité qui démocratise l'utilisation de l'apprentissage automatique dans la découverte de médicaments, la science des matériaux, la chimie quantique et la biologie.

Finalement, nous avons décidé d'utiliser les données provenant de **PubChem**, ce dernier permettant de faire une sélection de molécules souhaitées avec des mots-clés. Cela a facilité notre tâche, car aucune des sources ne fournissait de filtres automatiques pour choisir les molécules avec les propriétés souhaitées. Nous avons créé notre jeu de données contenant 1883 molécules anti-inflammatoires sous format *SMILES* décritent par 6 variables :

- **mw** : Poids moléculaire de la molécule.
- **xlogp** : Logarithme du coefficient de partage octanol-eau, indiquant l'hydrophobicité de la molécule.
- **polararea** : Surface polaire accessible de la molécule, liée à sa capacité à interagir avec l'eau et d'autres molécules polaires.
- **rotbonds** : Nombre de liaisons rotables dans la molécule, ce qui peut influencer sa flexibilité.
- **hbonddonor** : Nombre de donneurs de liaisons hydrogène, important pour les interactions avec d'autres molécules.
- **hbondacc** : Nombre d'accepteurs de liaisons hydrogène, un autre paramètre clé pour les interactions avec des récepteurs ou d'autres molécules.
- **Anti-inflammatoires (AI) et Anti-oxydants(AO) :** à ces 6 variables deux autres existent, nos variables d'intérêt. Elles sont binaires (1 si la molécule possède la propriété).

# 2 Etat de l'art scientifique

## 2.1 Catalogue des méthodes d'entraînements

Plusieurs méthodes d'apprentissage ont été testées par des chercheurs dans le but de générer des molécules.

Sur notre Github, nous avons pris chaque méthode de machine learning individuellement. L'intérêt de chaque méthode pour la génération de biomolécule est évalué par rapport aux autres. Il y a peu de papiers scientifiques à l'heure actuelle combinent ces méthodes et encore moins qui se concentrent sur des bases de données avec des anti-oxydants et anti-inflammatoires(il n'y en a pas).

Voir nos études des différents modèles sur **Github**
Nous ne détaillerons pas plus ces méthodes car nous les avons seulement étudiées comme piste de recherche. La plupart des méthodes de générations se concentrent autour des LLM.

## 2.2 Les GAN : Generationnal of Adversial Network

Les réseaux antagonistes génératifs (GAN) [1] sont des modèles d'apprentissage automatique composés de deux réseaux de neurones en compétition : un générateur, qui crée des échantillons de données, et un discriminateur, qui évalue si ces échantillons sont réels ou générés. Cette dynamique permet au générateur de produire des données de plus en plus réalistes au fil du temps.

## 2.3 Du transformer au LLM

### 2.3.1 Qu'est-ce qu'un transformer ?

La puissance de l'architecture du transformer [2] réside dans sa capacité à apprendre la pertinence et le contexte de tous les mots d'une phrase. L'architecture du transformer est divisée en deux parties distinctes, **le codeur et le décodeur**. Ces composants fonctionnent conjointement et partagent un certain nombre de similitudes.

**Les tokenizers :**

Les modèles d'apprentissage automatique ne sont que de grosses calculatrices statistiques et ils travaillent avec des nombres, pas avec des mots. Par conséquent, avant de transmettre des textes au modèle pour qu'il les traite, on doit d'abord symboliser les mots (ou d'autres types d'input) à l'aide de **tokenizers**.Cette opération convertit l'input en nombres, chaque nombre représentant une position dans un dictionnaire de tous les mots possibles avec lesquels le modèle peut travailler. Par exemple, le tokenizer des LLM détaillé plus-bas prend en input des SMILES qu'il convertit au format numérique.

**Multi-head Causal Self-attention :**

Chaque token génère une représentation basée sur l'attention qu'il porte sur les autres tokens de la séquence.Dans le self-attention, chaque token regarde (attend) les autres tokens dans la séquence pour déterminer quels sont les plus pertinents pour prédire le suivant.Le terme causal indique que l'attention est unidirectionnelle (les tokens futurs ne peuvent pas influencer le traitement des tokens précédents). A noter que BERT (plusbas) n'est pas causal. Multi-head signifie que l'attention est effectuée en plusieurs "têtes"

parallèles, permettant au modèle de capturer différents aspects de la relation entre les tokens (par exemple, relations chimiques).

**L'output :**

L'output est obtenu après un passage dans un feed-forward network. La sortie de cette couche est un vecteur de logits proportionnels au score de probabilité pour chaque token dans le dictionnaire du tokenizer. On met ensuite une couche softmax finale, où chaque output est normalisé en unscore de probabilité. Le mot avec le plus grand score est l'élement prédit le plus probable.

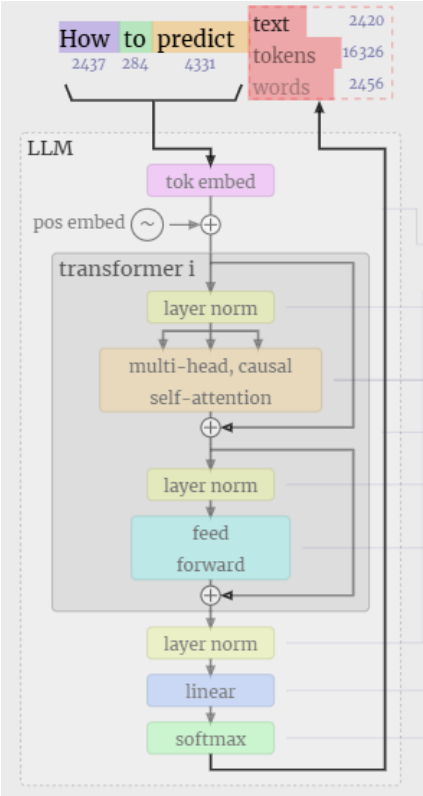


FIGURE 2 – Schéma d'un LLM

### 2.3.2 La pertinance de l'utilisation des LLM pour la biologie computationnelle

Les LLM sont surtout connus pour la génération à partir d'un input de texte. Ils existent également avec d'autres types d'input plus spécique comme le format SMILES pour la génération de molécules. Ils sont aussi très utilisés dans ce domaine pour plusierus raisons. [3]

- D'abord, les transformers sont conçus pour le traitement de données séquentielles. C'est idéal pour le format SMILES qui est une chaîne de caractères séquentielles.
- Les tranformers captent le contexte des inputs, autrement dit la dépendence à longue portée dans les séquences, ce qui est crucial pour comprendre la structure globale des molécules.
- Les transformers peuvent apprendre des motifs complexes dans les données et avoir appris à générer des molécules qui respectent les règles chimiques.

- Ils sont formés sur de large base de données ce qui est utile pour améliorer la diversité des molécules générées (principal défaut de l'architecture GAN).
- Enfin, ces modèles peuvent être affinés ou fine-tuner pour améliorer leur génération de molécules avec des propriétés spécifiques.

## 2.4 Modèles LLM pré-entraînés issu de la littérature adaptés à la génération de molécules

On se concentre ici sur des modèles de génération pour la biologie computationelle. Tous sont en open source sur *Github* ou *HuggingFace*. Notre choix reposent sur des modèles robustes et éprouvés : **ChemBERTa et GPT-2_zinc_87m**. Ce sont des fine-tuning de BERT et de GPT-2. Ils ont été adaptés pour prendre en charge le format SMILES.

BERT utilise une architecture de transformateur bidirectionnelle. Cela signifie qu'il prend en compte le contexte des inputs à la fois à gauche et à droite de chaque position dans une séquence.

GPT-2 utilise une architecture de transformateur unidirectionnelle (ou autoregressive). Cela signifie qu'il génère du texte en prenant en compte uniquement le contexte des inputs à gauche de chaque position dans une séquence.

Au niveau de l'entraînement, BERT est pré-entraîné sur du **Masked Language Modeling (MLM)** : certains mots dans la séquence d'entrée sont masqués, et le modèle doit prédire ces mots masqués en utilisant le contexte des mots environnants. De plus, il a été entraîné aussi sur du Next Sentence Prediction (NSP) : il prédit si une phrase suit une autre phrase dans le texte.

Au niveau de l'entrainement, GPT-2 est entraîné sur une tâche de modélisation de langage non supervisée, où le modèle prédit le mot suivant dans une séquence donnée tout en prenant en compte les mots précédents.

### 2.4.1 ChemBERTa

ChemBERTa [4] explore l'application de modèles basés sur des transformers, inspirés par les progrès du NLP, pour la prédiction des propriétés moléculaires. Le modèle exploite les chaînes SMILES pour le pré-entraînement auto-supervisé et les tâches en aval.

- **Architecture Transformer :**
  - ChemBERTa a 12 têtes d'attention et 6 couches.
- **Dataset :**
  - Utilise un dataset de **77 millions de SMILES uniques** provenant de **PubChem**.
  - Le pré-entraînement a été réalisé sur des sous-ensembles de données allant de **100 000 à 10 millions de SMILES**, avec le plus grand sous-ensemble entraîné en 48 heures sur un seul GPU NVIDIA V100.
- **Tâches d'évaluation :**
  - Des tâches en aval de prédiction des propriétés moléculaires ont été évaluées sur les jeux de données de **MoleculeNet**

— Le fine-tuning a impliqué l'ajout d'une couche de classification linéaire et l'entraînement jusqu'à 25 époques avec arrêt précoce basé sur le ROC-AUC.

- **Comparaison des performances :**
  - ChemBERTa a montré des performances compétitives par rapport à des modèles solides comme les **Directed Message Passing Neural Networks** (D-MPNN) et les modèles basés sur des forêts aléatoires.
  - Les performances se sont améliorées à mesure que la taille du dataset augmentait, atteignant des améliorations notables du ROC-AUC (+0.11) et du PRC-AUC (+0.059) entre 100K et 10M de SMILES.

### 2.4.2 GPT-2_zinc_87m

Le modèle **GPT-2_zinc_87m** [5] est une variante du modèle GPT-2, spécifiquement entraînée pour générer des structures moléculaires en utilisant des chaînes SMILES (Simplified Molecular Input Line Entry System). Il a été développé à partir d'environ 480 millions de chaînes SMILES issues de la base de données *ZINC*, une ressource publique contenant des structures chimiques.

- **Taille** : Environ 87 millions de paramètres, ce qui en fait une version plus légère adaptée à des tâches spécifiques.
- **Entraînement** :
  - Le modèle a été entraîné sur 175 000 itérations avec une taille de lot de 3 072.
  - La perte de validation atteinte est d'environ 0,615.

- **Utilisations principales**
  - **Génération de molécules** : Le modèle peut générer de nouvelles structures moléculaires en produisant des chaînes SMILES, utiles pour la découverte de médicaments ou la conception de nouveaux composés chimiques.
  - **Génération d'embeddings** : Il permet de créer des représentations vectorielles (*embeddings*) à partir de chaînes SMILES, facilitant des tâches telles que la classification de molécules ou la prédiction de propriétés chimiques.
- **Considérations importantes**
  - **Tokens spéciaux** : Le tokenizer utilisé n'ajoute pas automatiquement les tokens spéciaux de début (`<|beginoftext|>`) et de fin (`<|endoftext|>`). Ces tokens doivent être ajoutés manuellement lors de la préparation des données.
  - **Température de génération** : La température influence la diversité des molécules générées. Une température plus élevée produit des structures plus variées, tandis qu'une température plus basse génère des structures plus conservatrices.

— **Performance du modèle** Des tests de génération ont montré que, pour une température de 1,0 :

- Environ 99,97% des molécules générées sont uniques.
- Parmi elles, 99,92% sont valides.

Ces résultats montrent une capacité robuste du modèle à produire des structures moléculaires plausibles.

# 3 Architecture de notre modèle

L'enjeu de cette section est de proposer des méthodes pour augmenter le nombre d'instances (initialement de 1883) de notre jeu de données pour générer des molécules anti-inflammatoires(AI) ou anti-oxydantes(AO) crédibles. Les bases de données ayant des molécules avec des propriétés AI ou AO sont extrêmement rares et peu standardisées. Cette pipeline est le résultat de la lecture de plusieurs papiers, notamment [6] pour l'architecture simple, [7] et [8], et de connaissances personnelles.

## 3.1 Architecture simple

### 3.1.1 Fine-tuning

On rappelle d'abord le schéma classique d'un fine-tuning de modèle de type GPT avant d'expliquer comment nous allons intégrer ce type de modèles pré-existants dans un GAN.

— L'idée est d'injecter notre dataset pour fine-tuner un LLM.

— Le LLM fine-tuné sera alors doté d'une capacité de prédiction sur les molécules pour savoir si elles peuvent avoir des propriétés AI ou AO.

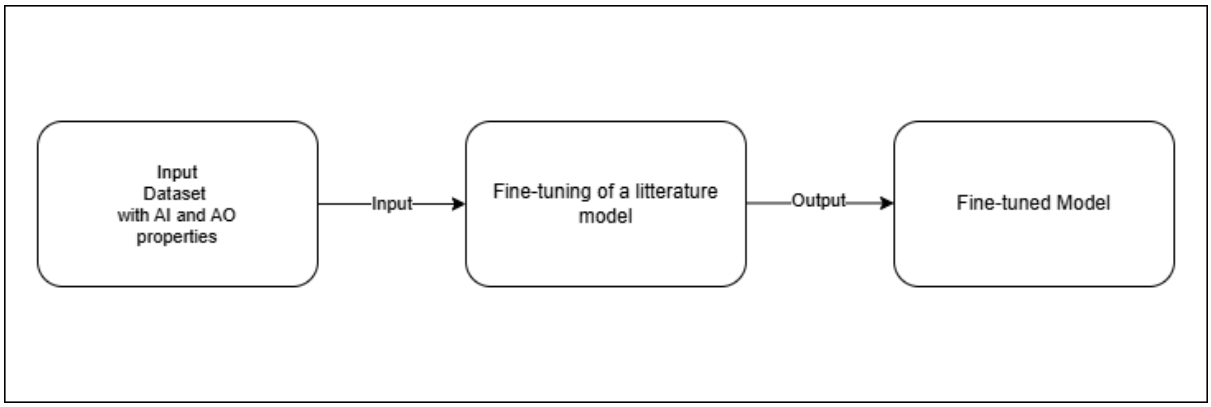


FIGURE 3 – Architecture naive de fine-tuning

### 3.1.2 Architecture GAN + LLM

On va se servir du LLM GPT-2_zinc_87m pour l'intégrer dans une pipeline plus large : Le LLM entraîné va servir de générateur de molécules SMILES et l'on va entraîner le discriminateur sur ces molécules générées. Le discriminateur a pour but de séparer les SMILES crédibles des SMILES non crédibles.

Un **LLM seul** comme GPT-2 peut manquer de mécanismes pour garantir que les molécules générées possèdent des propriétés spécifiques. Par ailleurs, un **GAN seul** est puissant pour générer des données réalistes et peut être entraîné pour optimiser certaines propriétés, mais il peut avoir du mal à capturer la complexité des séquences SMILES sans un modèle de langage puissant. De plus, un GAN fait rapidement preuve de manque de

diversité car son générateur et son discrimineur sont calibrés sur la base de données et n'en bouge pas. Son architecture favorise naturellement l'overfitting.

On combine ainsi l'architecture d'un GPT et d'un GAN pour tirer partie des avantages des deux architectures :

- Le LLM capture les dépendances séquentielles et contextuelles dans les séquences SMILES, assurant la cohérence et la validité chimique des molécules générées.
- Le discriminateur du GAN fournit un retour d'information au générateur, permettant l'optimisation des propriétés spécifiques des molécules.
- La compétition entre le générateur et le discriminateur pousse le générateur à explorer un espace de solutions plus large, augmentant ainsi la diversité des molécules générées.
- Le GAN aide à améliorer la qualité des molécules en filtrant les solutions non réalistes ou non optimales, grâce au retour d'information du discriminateur.
- Le LLM excelle dans la génération de séquences réalistes, tandis que le GAN optimise ces séquences pour des propriétés spécifiques, combinant ainsi les forces des deux approches
- L'architecture combinée est plus robuste aux erreurs et aux incohérences, car le discriminateur du GAN peut identifier et corriger les molécules non valides générées par le LLM.

Cette première idée est une innovation de notre part et n'est pas, à notre connaisance, déjà présente dans un papier de recherche sur de la biologie générative.

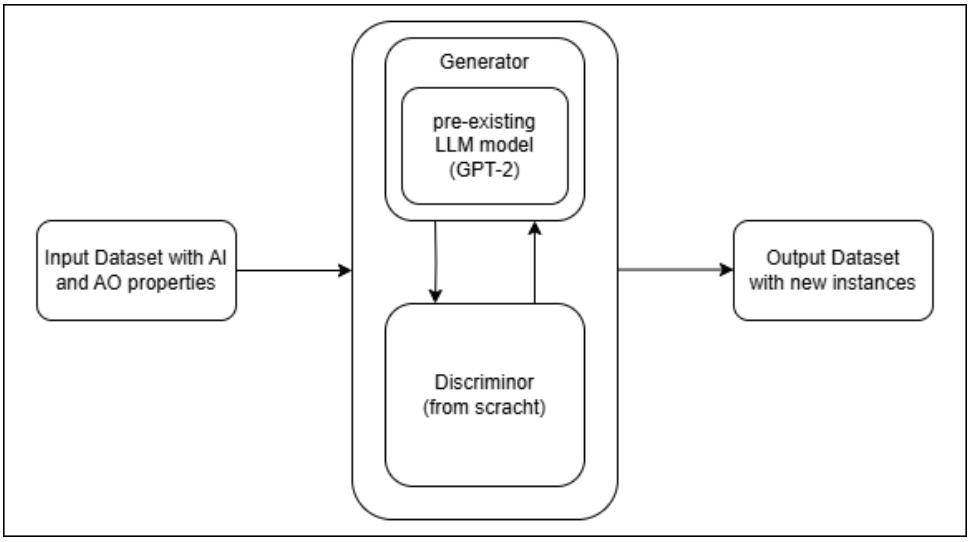


FIGURE 4 – Architecture GAN + LLM = GANLLM

**Justification du choix du modèle [6] :**

Le choix GPT-2 apparaît comme étant le plus naturel car 99,97% des molécules générées sont uniques et parmi elles, 99,92% sont valides. Le modèle (et plus particulièrement le discrimineur) peut alors uniquement se concentrer sur la génération de molécules avec des propriétés AI ou AO et on a un taux de confiance élévée en la propositon de génération du modèle.

## 3.2 Pipelines

A partir des deux architectures précédentes, on met en plus 2 pipelines différentes pour englober nos transformations dans un ensemble plus robuste.

### 3.2.1 Pipeline Data Generation

L'idée est ici d'inscrire l'architecture de la première architecture dans une pipeline data plus vaste pour raffiner les effets de la génération. Elles est composée de trois phases. D'abord, on enrichit la base de données initiales pour différentes transformations chimiques simples qui ne modifient pas les propriétés physico-chimiques de la molécule. Ensuite, filtre les données à la sortie du GAN pour ne garder que les molécules AI ou AO. Enfin, on fait une analyse discriminante linéaire (LDA en anglais) pour ne retenir pour chaque instance que les données pertinentes pour les deux variables cibles (AI ou AO).

- **Etat initial :**
  On a initialement une base de donnée, nommé D avec des molécules ayant des propriétés AO et/ou AI.
- **Etape 1 : Data Augmentation Layer**
  On commence par augmenter le nombre d'instances de la base de données en appliquant certaines transformations chimiques aux données initiales SMILES qui préservent les propriétés AI et A0 des nouvelles molécules générées. Elles sont intégrés à un nouveau dataset qui incluent les molécules initiales. Ces transformations sont : la génération de permutations et de tautomères. Les tautomères d'une molécue ne change pas ses propriétés physico-chimiques la plupart du temps. Le dataset devient D'
- **Etape 2 : Architecture GAN + LLM**
  On injecte le dataset augmenté D' dans le GAN de la figure 2 qui génère de nouvelles molécules qui sont discriminées par le discriminateur du GAN (réseau de neurones concurrents). Les formats SMILES sont triées vis-à-vis de leur crédibilité physico-chimique et non de leur propriété AI et/ou AO. Les molécules qui passent le filtre post-GAN sont injectées dans le nouveau dataset de l'étape précédente. On a une confiance suffisante sur leurs propriétés
- **Etape 3 :Apprentissage par renforcement du dataset de l'étape 2**
  C'est une idée d'étape supplémentaire que nous avons pour régler les molécules générées avec une fonctin de gain. Nous ne l'avons pas implémentée et la laissons ici pour donner des idées à d'éventuels futurs relecteurs.
- **Etape 4 : ADL (Analyse discriminante linéaire) avec en variables d'intérêt AI et AO**
  L'idée est de garder uniquement les variables pertinentes de chaque instance, *id est*, les variables qui aident à déterminer si la molécule est AI et/ou AO pour avoir une base de données avec moins d'informations mais des informations pertinentes relativement à l'enjeu. On utilise une ADL plutot qu'une PCA car les variables cibles sont binaires (AI et/ou AO ou non) et on veut maximiser leur séparation. Idée supplémentaire : on pourrait utiliser les résultats de cette ADL pour améliorer le discriminateur de l'étape 2 en présélectionnant mieux ces molécules.
- **Etat final :**
  On aboutit avec notre dataset final (pour cette partie) augmenté post-LDA que

l'on nomme D'. Plus le nouveau dataset va être grand par rapport à l'initial, plus le risque d'erreur va augmenter et la précision va diminuer vis-à-vis du nombre de molécules AI + AO.

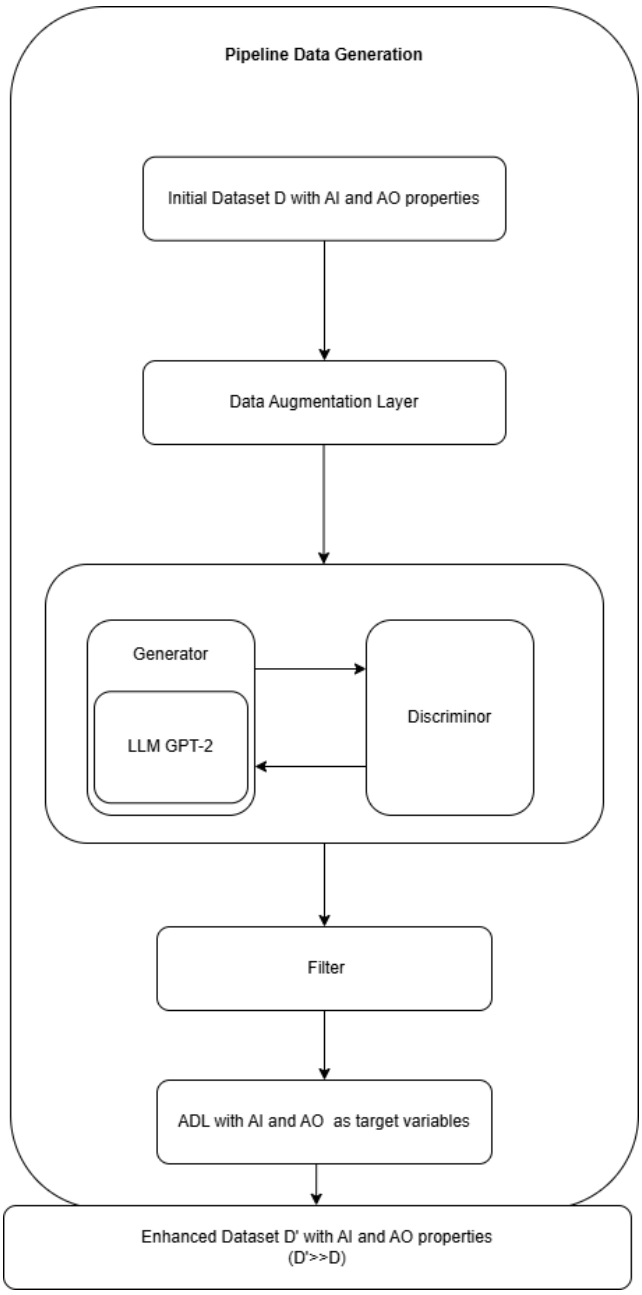


FIGURE 5 – Pipeline Data Generation

### 3.2.2 Pipeline Data Prediction

- **Fine-Tuning for Prediction :** [8] [9] On va fine-tuner le LLM ChemBERTa pré-existant déjà entraînés sur des molécules (sans propriétés AI et AO), il en sort un nouveau modèle fine-tuné capable de prédire les propriétés AI et AO des molécules. On utilise alors ce modèle fine-tuné pour augmenter la base de données de ses prédictions sur un set de molécules on l'on ignore ces propriétés AI et/ou AO.
  - **Etat initial :**
    On dispose des bases de données D et D'.
  - **Etape 1 : Création d'un prédicateur**
    On fine-tune le modèle sur la base de données D initial ou sur la base de données D' pour donner une capacité de prédiction sur les propriétés AO et AI. Il se pose la question de savoir si ce prédicateur est plus pertinent sur la base de données D ou D'.
  - **Etape 2 : Test de ce prédicateur**
    Le modèle, une fois fine-tuner, peut prédire les caractéristiques physiques AI et/ou AO sur la nouvelle base de données. On sélectionne une fraction de la base de données ChemPub inférieure en nombre d'instances au nombre d'instances sur lequel le modèle Boltz-1 a été fine-tuner.
  - **Etat final :** On augmente le dataset D' du nombre d'instances N pour lequel le prédicateur a ressorti un résultat positif (AI et/ou AO) avec N«D' ou N«D.
- **Prediction on a biggger dataset :** Ce modèle est utilisé pour prédire sur un nouveau dataset si les molécules qu'il traite possède des proriétés AO et AI.
  - **Etape 7 :**
    On se tourne vers une plus grande base de données open source de biomolécules SMILES, on choisit une portion de ce dataset et on prédit ces propriétés AI et/ou AO
  - **Etape 8 :**
    On conserve les molécules qui ont été prédites comme possédant des propriétés AI et/ou AO. On les injecte dans le dataset de la fin de la partie 1, on obtient un dataset final que le nomme *final_ dataset*

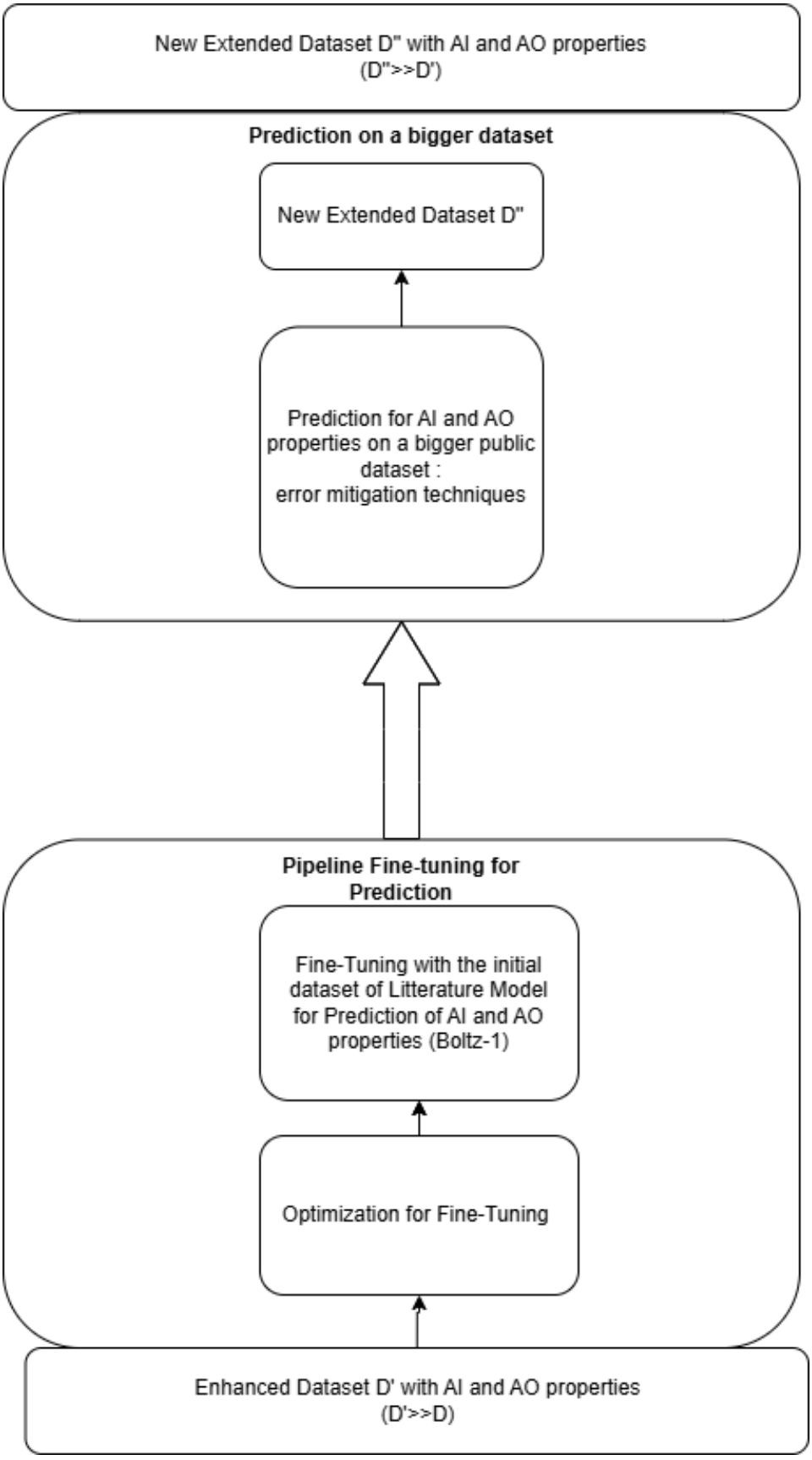


FIGURE 6 – Pipeline Data Generation

## 3.3 Architecture de la pipeline finale

*Ici, l'architecture est décrite telle que nous aurions aimé la faire. Par manque de temps, elle n'a pas été implémentée.*
L'idée est de combiner la **Pipeline Data Generation** et la **Pipeline Data Prediction** dans le but de profiter d'une contrainte de bouclage itérative : Le dataset final obtenu à la fin de la **Pipeline Data Generation** est injecté comme dataset intial dans la **Pipeline Data Prediction**. En d'autres termes, le dataset D" devient le dataset D et on itère le processus pour encore augmenter le nombre d'instances de la base de données.

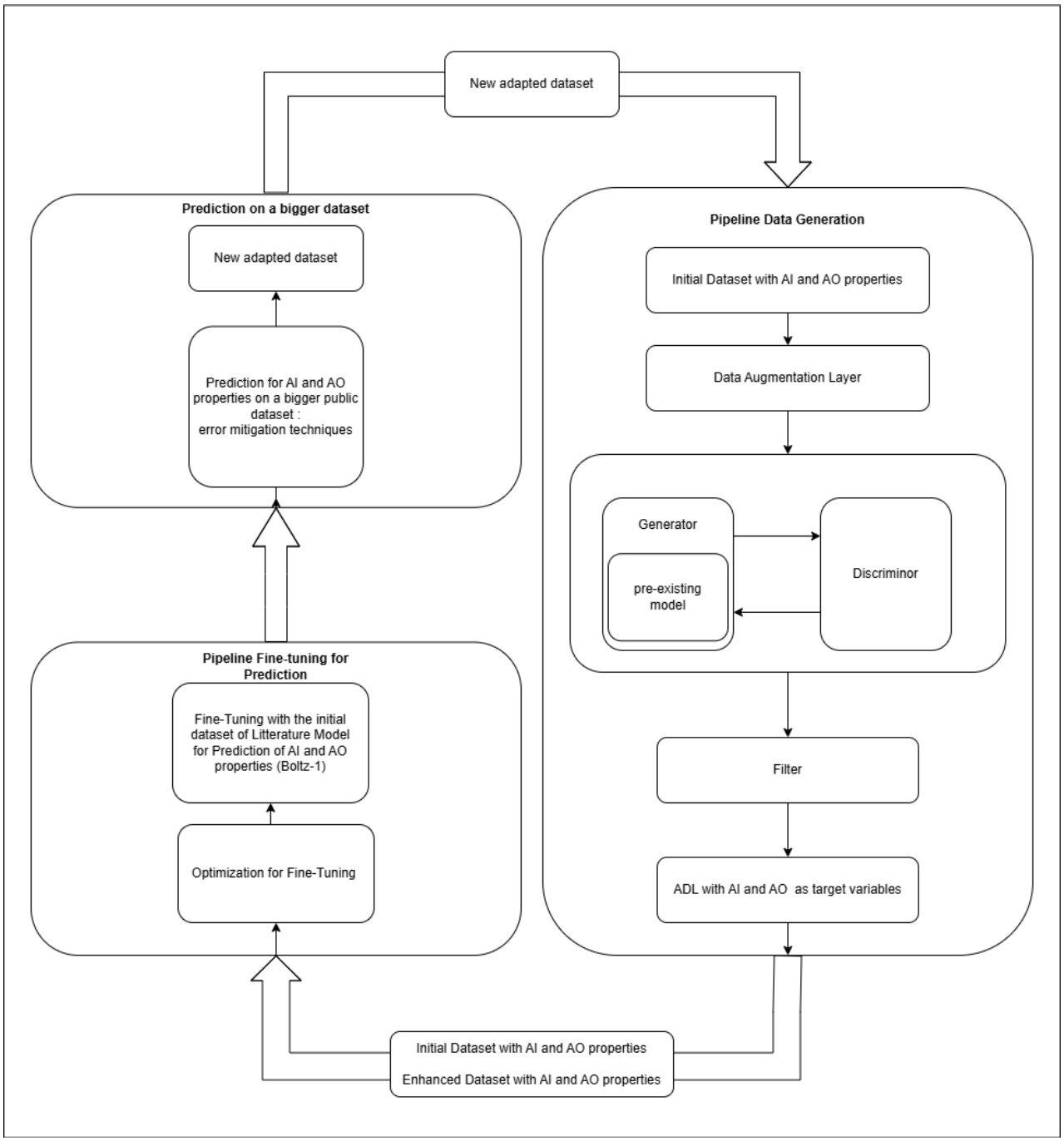


FIGURE 7 – Pipeline Finale : Architecture

# 4 Métriques utilisés pour la génération de molécules

Lors de la génération de biomolécule ayant des propriétés spécifiques (anti-inflammatoires ou antioxydantes), les métriques suivantes permettent d'évaluer la répartition statistique des molécules générées après transformation.

## 4.1 Validité

La **validité** mesure la proportion de molécules générées qui respectent les règles de la chimie (par exemple, pas de structures impossibles ou de graphes chimiques invalides). Cette métrique peut être calculée en utilisant des outils comme `RDKit`.

$$\text{Validité} = \frac{\text{Nombre de molécules valides}}{\text{Nombre total de molécules générées}}$$

## 4.2 Originalité

L'**originalité** évalue le pourcentage de molécules générées qui sont nouvelles et ne figurent pas dans le jeu de données initial.

$$\text{Originalité} = \frac{\text{Nombre de molécules nouvelles}}{\text{Nombre total de molécules générées}}$$

## 4.3 Diversité

La **diversité** mesure la variabilité structurelle entre les molécules générées. Une méthode courante est d'utiliser la similarité de Tanimoto appliquée aux empreintes moléculaires (*fingerprints*).

$$\text{Diversité} = 1 - \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \text{Sim}_{\text{Tanimoto}}(M_i, M_j)$$

où $\text{Sim}_{\text{Tanimoto}}(M_i, M_j)$ est le score de similarité entre les molécules $M_i$ et $M_j$, et $N$ est le nombre total de molécules générées.

Avant de calculer la similarité de Tanimoto, il faut convertir les chaînes SMILES des molécules en empreintes moléculaires. Cela a été fait à l'aide de la bibliothèque RDKit. Les empreintes moléculaires sont des vecteurs binaires (ou parfois des vecteurs numériques) qui représentent les caractéristiques structurelles de la molécule, telles que les groupes fonctionnels, la connectivité des atomes, ou les motifs chimiques récurrents.

Une fois que les molécules sont représentées par leurs empreintes moléculaires, le calcul de la similarité de Tanimoto entre deux molécules se fait comme suit :

$$Tanimoto(A, B) = \frac{\text{Nombre de bits communs (1 dans les deux empreintes)}}{\text{Nombre de bits totaux (1 dans l'une ou l'autre des empreintes)}}$$

En d'autres termes, on compare les deux vecteurs binaires (ou numériques) obtenus à partir des empreintes moléculaires des molécules, et on calcule la proportion de caractéristiques communes (bits à 1) par rapport à l'ensemble des caractéristiques présentes dans au moins une des deux molécules.

## 4.4 Drug-likeness (Pharmacocinétique)

Le **drug-likeness** évalue combien des molécules générées respectent des critères pharmacologiques spécifiques, tels que les règles de Lipinski :

- Poids moléculaire $\leq 500\,\text{Da}$,
- LogP $\leq 5$,
- Nombre de donneurs de liaison hydrogène $\leq 5$,
- Nombre d'accepteurs de liaison hydrogène $\leq 10$.

$$\text{Drug-likeness} = \frac{\text{Nombre de molécules respectant les règles}}{\text{Nombre total de molécules générées}}$$

Ces règles ont été formulées par Christopher A. Lipinski en 1997 et reposent sur des propriétés physico-chimiques des molécules qui influencent leur absorption, leur distribution, leur métabolisme et leur élimination (les propriétés ADME)

# 5 Résultats

Les calculs suivant ont été réalisés sur un CPU intel core i7 de 2013 avec 2,4Ghz, 16Go de RAM ou sur Google Collab gratuit (GPU :Tesla K80/P100/T4, 12Go de RAM). Tous les scores affichés ici sont des proportions de 0 à 1 (voire partie 4)

## 5.1 Résultats de la Data Augmentation Layer (DAL)

On évalue ici les indicateurs avancées de la partie 4 sur différents dataset avant même d'entraîner le modèle. Les différents dataset sont (de gauche à droite) :

- Le dataset initial (1 883 instances scrappé de PubChem)
- Le dataset initial + les tautomères des molécules initiales (3 359 instances)
- Le dataset initial + les permutations des molécules initiales (10 015 instances)
- Le dataset initial + les permutations + les tautomères (11 503 instances)

| Critères | Données initiales | Tautomères seuls | Permutations seules | Tautomères + Permutations |
|---|---|---|---|---|
| Nombre de molécules | 1883 | 3359 | 10015 | 11503 |
| Validité | 1.00 | 1.00 | 1.00 | 1.00 |
| Originalité | 0.00 | 0.44 | 0.81 | 0.84 |
| Diversité | 0.91 | 0.90 | 0.88 | 0.89 |
| Druglikeliness | 0.79 | 0.74 | 0.72 | 0.72 |

FIGURE 8 – Résultats Data Augmentation Layer ( à $10^{-2}prs$)

On note ici que les molécules générées sont toutes générées à partir de leur propriété anti-inflammatoires (AI=1). On passe donc de 1883 instances à 11 503, toutes valides au sens de RDKit. De plus,

- Le score d'originalité est mesuré par rapport au dataset initial. On note que l'augmentation du score d'originalité en fonction de la taille du dataset est liée aux permutations (la plupart des permutations générées n'existaient pas auparavant). De même, le dataset tautomères + permutations n'est pas la somme stricte des deux précédents dataset car on filtre les redondances.
- Le score de diversité reste stable ce qui appuie l'intérêt de notre démarche avant tout entraînement du modèle car cet indicateur est conservé.
- l'indicateur Drug-likeliness nous permet ici de jauger de la pertinence de notre couche d'augmentation dans le cadre de l'enjeu (générer des molécules pour l'industrie pharmaceutiques). Il est également conserver en augmentant la taille du dataset.

Cette Data Augmentation Layer permet de pallier à notre problème initial de manque de datas sur les molécules anti-inflammatoires et anti-oxydantes en générant nous même des molécules artificiellement sur lesquelles on a un bon taux de confiance ( les sont valides et les indicateurs avancés sont conservés, voire améliorés)

## 5.2 Résultats sur différentes architectures

Le LLM utilisé (seul ou dans l'architecture GAN+LLM) est **GPT2 Zinc**, voire pourquoi plus haut. Le nombre de molécules générées est artificiellement limité à 100 car la puissance de calcul à notre disposition ne disposait pas de suffisamment de RAM pour faire tourner un modèle plus grand. C'est une voie évidente d'amélioration de génération de molécules que nous aurions aimé poursuivre.

| Critères | LLM seul | GAN + LLM |
|---|---|---|
| Nombre de molécules générées | 100 | 100 |
| Validité | 0.16 | 1.0 |
| Originalité | 1.00 | 1.00 |
| Diversité | 0.56 | 0.85 |
| Druglikeliness | 0.16 | 0.95 |
| Temps d'éxécution | 10min | 9min52s |
| Accuracy | 0.09 | 0.62 |

FIGURE 9 – Résultats des différentes architectures sans Data Augmentation Layer

Quelques remarques :

- Le LLM seul a un mauvais score de validité car il n'y a pas de réseaux de neurones antagonistes qui filtrent les sorties du LLM. Cela montre qu'une architecture plus classique serait caduque pour notre projet. En plus d'avoir un rôle de filtre, le discriminateur du GAN permet au modèle d'apprendre de ses erreurs, on se rapproche d'une architecture d'apprentissage par renforcement (ou reinforcement learning) en vogue dans la recherche actuellement pour les LLM.
- Le subset est petit (uniquement 100 instances générées à partir des 1883 initiales) mais le modèle arrive dans les eux cas à notre proposer que des molécules orginiales
- La diversité n'est pas un critère pertinent dans ce cas, étant donné le mauvais score de validité
- Le mauvais score de Druglikeliness pour le LLM seul est lié au mauvais score de validité (une molécule n'a aucune chance d'être un médicament si elle n'existe pas physico-chimiquement). C'est cohérent
- Le temps d'éxécution est similaire pour les deux architectures
- Un nouvel indicateur est introduit car ici on a entraîné le modèle. C'est l'accuracy (ou précision globale) post-filtre (voire schéma Pipeline Generation) :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}} \tag{1}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

où :

- $TP$ (True Positives) : Prédictions positives correctes

- $TN$ (True Negatives) : Prédictions négatives correctes
- $FP$ (False Positives) : Faux positifs (erreurs de type I)
- $FN$ (False Negatives) : Faux négatifs (erreurs de type II)

Ici, l'accuracy va prédire le nombre de molécules générées et étiquetées comme AI qui sont effectivement AI. Le score est ici de 0.62 pour deux classes : AI ou non-AI. C'est un score un peu mieux que ce que le hasard ferait (0.5 sur un grand nombre) mais cela reste faible.

| Critères | LLM seul | GAN + LLM |
|---|---|---|
| Nombre de molécules générées | 100 | 100 |
| Validité | 0.11 | 1.0 |
| Originalité | 0.86 | 1.00 |
| Diversité | 0.95 | 0.85 |
| Druglikeliness | 0.10 | 0.90 |
| Temps d'éxécution | 118min33s | 124min54s |
| Accuracy | 0.08 | 0.76 |

FIGURE 10 – Résultats des différentes architectures avec Data Augmentation Layer

On note deux changements majeurs en injectant la Data Augmentation Layer plutot que le dataset initial dans les modèles :

- Le temps d'éxécution est plus long, quasiment proportionnellement
- La précision du modèle GAN+LLM est plus importante. On renforce sa capacité de prédiction en multupliant par 6 sa base de données initiales en y ajoutant des tautomères ainsi que des permutations.

D'autres pistes pourraient être envisagées pour améliorer cette précision :

- Augmenter la taille du dataset initial si lorsque que des bases de données plus grande avec des propriétés AI ou AO seront disponibles.
- Essayer de filtrer les tautomères produits. En effet, on sait que la plupart des tautomères produit conservent pour la plupart les propriétées chimiques des molécules (donc les propriétés AI) mais il est possible que dans une faible proportion, les tautomères produits perdent leurs propriétés AI ou AO.

## 5.3 Résultat Analyse discriminante linéaire

On fait, **dans la figure 10**, une LDA sur la Data augmentation Layer après le passge dans le modèle GAN+LLM. La LDA est une méthode de réduction de dimension qui cherche à maximiser la séparation entre les classes tout en minimisant la variance intra-classe. C'est pertinent ici car on veut garder uniquement les informations importantes pour que notre modèle classifie correctement les molécules SMILEs AI des molécules non-AI. Ici, tous les points sont regroupés sur la composante LD1. L'analyse est la suivante :

- Quelques points en cyan sont éloignés à droite, indiquant que certaines molécules anti-inflammatoires sont distinctes de la majorité. Elles vont être compliquées à déterminer.
- Les molécules sont discriminées ici sur un seul axe et la sépaation entre les deux classes est nette. Le modèle arrive bien à discriminer les molécules AI et AO et de manière linéaire

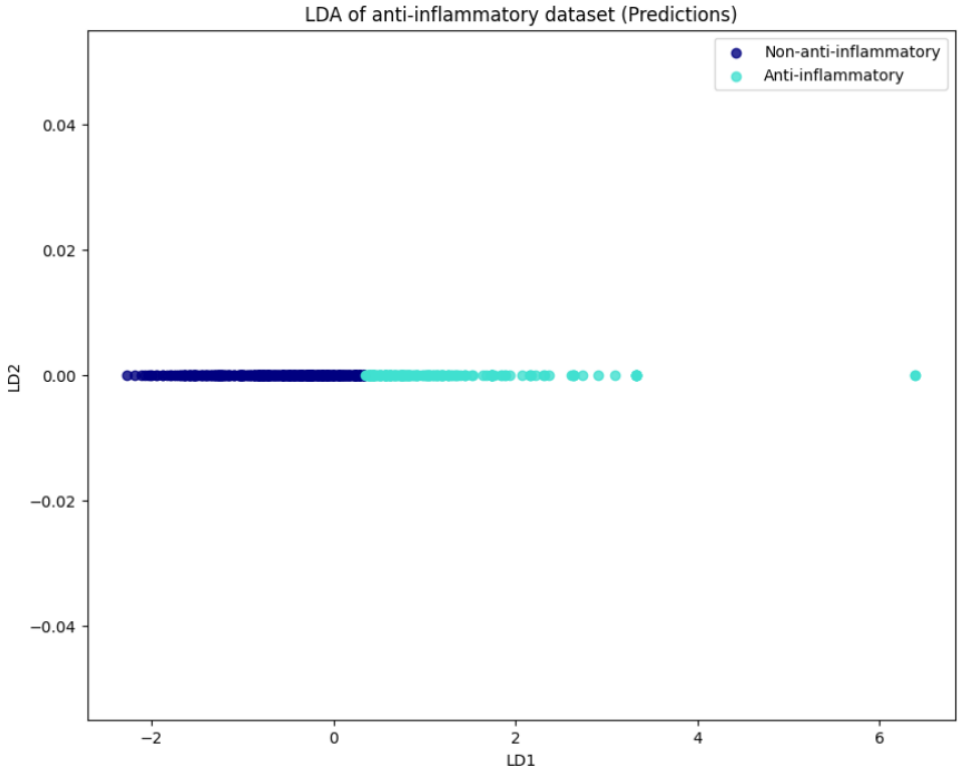


FIGURE 11 – LDA sur D"

On affiche, **dans la figure 11**, les poids accordés par la LDA sur le dataset initial D et les poids accordés sur le dataset final de la pipeline data generation.

Analyse et interprétation :

- Le dataset initial contient 50% de molécules AI et 50% de molécules non-AI tandis que le dataset final n'en contient que 8% par construction (et donc 92% de molécules AI). La différence des poids entre les deux graphiques permet d'expliquer les variables du dataset pertinentes pour décrire les molécules AI. En effet, le graphique révèle que certaines corrélations initiales ne sont perdues après la data augmentation layer et la génération du modèle GAN+LLM. Par exemple, la corrélation positive entre la charge d'une molécule et sa capacité AI est remise en cause. De facto, les

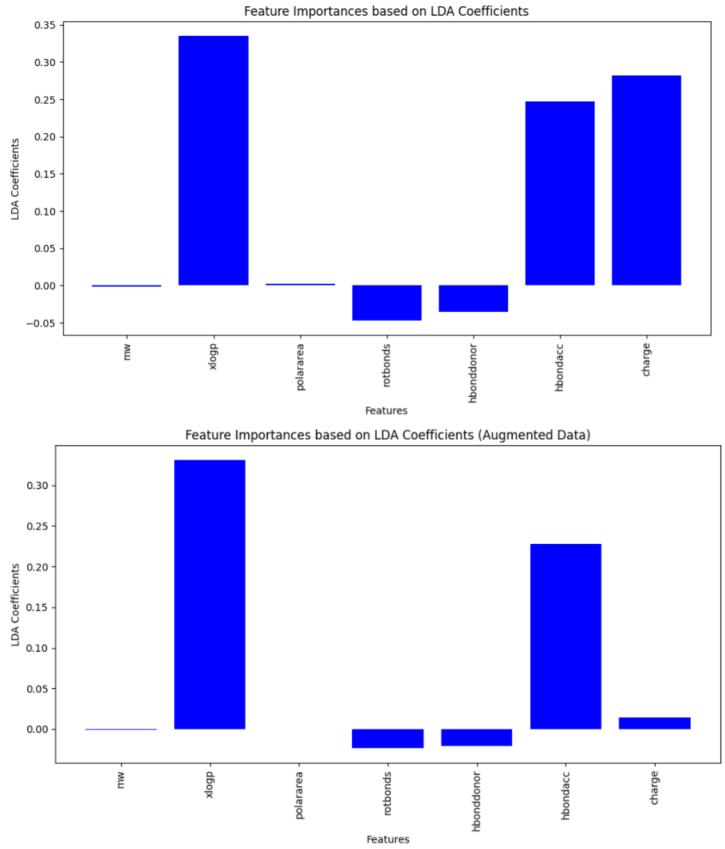


FIGURE 12 – LDA sur D (en haut) et sur D"(en bas)

molécules générées qui sont AI et conservent les mêmes scores sur les indicateurs avancés ne sont plus décrites par la variable *charge*.

- La plupart des poids des variables descriptives pour décrire une molécule AI sont similaires dans les deux graphiques. On peut voir que dans les deux graphes les poids de xlopx, hbonddacc, hbonddonor et rotbonds sont similaires
- On peut interpréter la perte du poids de certaines variables comme une erreur du modèle de génération. Cela peut être également lié a la Data Augmentation Layer, les tautomères et les permutations "effacent" le poids la variable charge en ne générant que des molécules AI qui ne sont pas AI à cause de leur charge. On aurait donc une description des molécules AI sur un subset de la base de données initiale.

## 5.4 Résultats du fine-tuning de ChemBERTa

Le modèle utilisé est **ChemBERTa**. Nous n'avons pas eu le temps de beaucoup travailler sur cette partie mais nous avons un résultat à présenter.

Lorsque l'on fine-tune ChemBERTa sur le dataset initial non augmenté et que l'on essaye ensuite de prédire, le caractère AI ou non sur une base de validation, on obtient un score de 0,62. Ceci démontre, comme expliqué plus haut, que l'on fait mieux que le hasard et que cette méthode est viable. Il resterait maintenant à l'optimiser pour améliorer le score d'accuracy. Cela peut être une piste à envisager pour un prochain projet 3A.

# 6 Conclusion

Dans ce rapport, nous avons exploré l'utilisation des modèles d'apprentissage profond pour la génération de nouvelles biomolécules aux propriétés anti-inflammatoires et antioxydantes. Notre démarche s'est articulée autour de plusieurs étapes clés : après une analyse approfondie de l'état de l'art, nous avons étudié différentes architectures, notamment les *Generative Adversarial Networks* (GAN) et les modèles *Large Language Models* (LLM), en mettant l'accent sur des approches adaptées à la chimie de génération de molécules.

Egalement, une part importante de notre temps investi sur ce projet a servi à la constitution d'un jeu de données spécifique issu de bases de données telles que PubChem ayant des molécules anti-inflammatoires ou anti-oxydantes, la mise en place de pipelines de génération et de prédiction, ainsi que l'évaluation des molécules générées à l'aide de métriques pertinentes (validité, originalité, diversité, et *drug-likeness*).

Les résultats obtenus montrent que les approches basées sur le *fine-tuning* et l'architecture *GAN+LLM* de modèles pré-entraînés, notamment *ChemBERTa* et *GPT-2_zinc_87m*, permet d'améliorer ou de conserver la qualité des molécules générées tout en ayant une capacité de génération mieux que le hasard.

Cependant, plusieurs défis restent à relever. La complexité des interactions biologiques rend difficile la prédiction précise des propriétés des molécules avant leur validation expérimentale. De plus, la performance des modèles pourrait être améliorée en intégrant davantage de données d'apprentissage qui manquent pour le moment dans les bases de données open source. On pourrait également explorer des architectures hybrides combinant plusieurs approches comme ce que nous aurions aimé faire avec la contrainte de bouclage de l'architecture augmentée.

A titre personnel (Quentin Velard), beaucoup des idées que j'ai développé ici m'ont été apporté dans mon deuxième stage de césure où j'étais chargé de développer une architecture sur des ordinateurs quantiques pour des tâches de MLOps et pallier au problème de "catastrophic forgetting" des ordinateurs classiques ou quantiques actuels. Ce stage m'avait permis de contribuer à un article de recherche sur le sujet qui est actuellement en peer-review et va être publier dans le journal IEEE prochainement.

Nous tenons à remercier Guenael Cabanes pour ses conseils et son suivi tout au long de ce projet passionant.

# 7 Bibliographie


**[1]** Aman Chadha *GAN course*, 2024.
https://github.com/amanchadha/coursera-gan-specialization?tab=readme-ov-file. Consulté octobre 2024.

**[2]** Matthieu Labonne *LLM course*, 2024.
https://github.com/mlabonne/llm-course. Consulté octobre 2024.

**[3]** Menua Bredosian, Philipp Guevorgian, Tigran Fahradyan. *Small Molecule Optimization with Large Language Models*, 2024.
https://arxiv.org/abs/2407.18897. Consulté décembre 2024.

**[4]** Seyone Chithrananda, Gabriel Grand, Bharath Ramsundar. *ChemBERTa : Large-Scale Self-Supervised Pretraining for Molecular Property Prediction*, 2020.
https://arxiv.org/abs/2010.09885. Consulté novembre 2024.

**[5]** Ncfrey. *ChemGPT-1.2B*, 2024.
https://huggingface.co/ncfrey/ChemGPT-1.2B. Consulté décembre 2024.

**[6]** Yao Dong, Simiao Yu, Chao Wu, Yike Guo Semantic Image Synthesis via Adversarial Learning,2017 https://arxiv.org/abs/1707.06873. Consulté en octobre 2024

**[7]** Shikun Feng, Yuyan Ni, Yan Lu, Zhi-Ming, Wei-Ying Ma, Yanyan Lan. *UNIGEM : A UNIFIED APPROACH TO GENERATION AND PROPERTY PREDICTION FOR MOLECULES*, 2024.
https://arxiv.org/pdf/2410.10516. Consulté décembre 2024.

**[8]** Bonggun Shin, Sungsoo Park, JinYeong Bak. *Controlled Molecule Generator for Optimizing Multiple Chemical Properties*, 2024.
https://dl.acm.org/doi/10.1145/3450439.3451879. Consulté décembre 2024.

**[9]** Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, Regina Barzilay. *Boltz-1 Democratizing Biomolecular Interaction Modeling*, 2024.
https://www.biorxiv.org/content/10.1101/2024.11.19.624167v1. Consulté octobre 2024

**[10]** Félix Therrien, Edward H. Sargent, Oleksandr Voznyy. *Using GNN property predictors as molecule generators*, 2024.
https://arxiv.org/html/2406.03278v1. Consulté novembre 2024.