Table des matières

1 Introduction et objectifs

La dynamique moléculaire (MD) permet de simuler l'évolution atomique de systèmes biomoléculaires mais l'exploration des transitions conformationnelles séparées par de hautes barrières d'énergie libre reste coûteuse lorsque l'on intègre les équations du mouvement à des pas de temps femtosecondes (fs). Pour contourner cette limite, des méthodes d'échantillonnage améliorées comme le Replica Exchange (RE) ont été développées. Le principe consiste à faire évoluer plusieurs copies du système à différentes températures et à autoriser des échanges entre elles, ce qui permet de franchir plus efficacement les barrières énergétiques. Cependant, le nombre de répliques nécessaires croît rapidement avec la taille du système, rendant l'approche très coûteuse, voire irréaliste pour de grands systèmes comme les ribosomes [1].

Dans ce contexte, des approches apprises se développent. Timewarp combine un normalising flow conditionnel et une correction MCMC de Metropolis-Hastings pour générer des propositions de grands pas temporels. La correction assure que l'échantillonnage final respecte la distribution de Boltzmann, condition nécessaire pour rester fidèle à la physique statistique. Une propriété clé rapportée par les auteurs est la transférabilité du modèle : entraîné sur certains peptides, il peut être appliqué à de nouveaux petits peptides avec un gain notable en effective sample size per second. De plus, l'approche fonctionne directement en coordonnées cartésiennes tout-atome, ce qui facilite son extension à d'autres systèmes [5].

L'objectif de ce projet est d'explorer des stratégies permettant de découper un grand système tel que le ribosome en sous-ensembles, d'entraîner des modèles sur ces blocs, puis de recoller les dynamiques obtenues afin de reconstruire le paysage conformationnel global [2, 3, 4]. Dans ce cadre, j'ai mis en œuvre Timewarp, en réussissant à exécuter l'ensemble du pipeline, à obtenir des résultats d'entraînement et d'exploration, et à réaliser un premier niveau de comparaison avec des simulations de dynamique moléculaire classique pour valider le modèle. Au-delà de ces premiers essais, une autre possibilité intéressante est d'exploiter la propriété de transférabilité rapportée par les auteurs : on peut envisager que l'apprentissage réalisé sur certains blocs puisse être réutilisé ou adapté pour explorer efficacement d'autres blocs, avant de recoller leurs dynamiques afin de reconstruire le paysage conformationnel global.

2 Matériels et méthodes

2.1 Conditional normalising flows

Les flows normalisants conditionnels (CNFs) permettent de modéliser directement une distribution conditionnelle complexe $p(y|x)$, plutôt que de se limiter à une prédiction ponctuelle. En effet, les modèles de régression classique par MSE apprennent essentiellement la moyenne de la distribution cible, ce qui conduit souvent à des solutions non-physiques ou floues dans les tâches de prédiction structurée par example ma prédiction de conformations 3D d'un dipeptide. Au contraire, les CNFs introduisent une variable latente z tirée d'une loi conditionnelle $p(z|x)$ et un mapping bijectif $f_\phi(y, x)$ reliant y et z. La vraisemblance conditionnelle est donnée par la formulation classique [6] :

$$p_{Y|X}(y|x) = p_{Z|X}(z|x) \left| \det \frac{\partial f_\phi(y,x)}{\partial y} \right|, \quad z = f_\phi(y, x). \tag{1}$$

où $p_{Z|X}(z|x)$ est une distribution de base simple (par ex. une gaussienne diagonale conditionnée sur x), et le Jacobien $\det \frac{\partial f_\phi(y,x)}{\partial y}$ corrige le changement de variable afin de conserver la masse de probabilité. La variable x joue uniquement le rôle de conditionnement et n'est donc pas dérivée. Ainsi, l'équation (1) exprime directement la densité conditionnelle $p_{Y|X}(y|x)$ à partir de la densité simple $p_{Z|X}(z|x)$ et de la transformation bijective f_ϕ.

Lien avec notre cadre (Timewarp). Dans notre cas, la variable y correspond à la conformation future $x(t+\tau)$, tandis que la variable x correspond à la conformation actuelle $x(t)$. Le mapping f_ϕ apprend donc une transformation bijective entre une variable latente gaussienne z et la configuration future $x(t+\tau)$, conditionnellement à l'état présent $x(t)$ [5] :

$$p_\theta(x(t+\tau) \mid x(t)) = p(z \mid x(t)) \left| \det \frac{\partial f_\phi(x(t+\tau), x(t))}{\partial x(t+\tau)} \right|, \quad z = f_\phi(x(t+\tau), x(t)). \tag{2}$$

Cela justifie naturellement le choix de nos données sous forme de paires $(x(t), x(t+\tau))$: chaque paire représente une transition élémentaire de la dynamique moléculaire, que le CNF cherche à modéliser de manière explicite. Ainsi l'entraînement revient à apprendre la vraisemblance conditionnelle $p_\theta(x(t+\tau) \mid x(t))$ via la transformation f_ϕ, comme illustré dans la Figure 1.

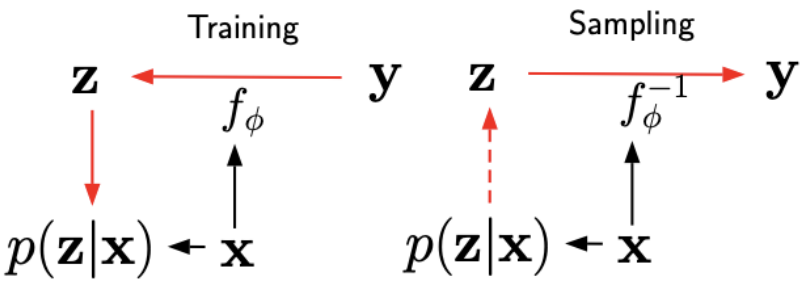


FIGURE 1 – Schéma du Conditional Normalizing Flow (CNF). Pendant l'entraînement (à gauche), la configuration future $y = x(t+\tau)$ est projetée dans l'espace latent $z = f_\phi(y, x(t))$ et la vraisemblance est évaluée sous $p(z|x(t))$. Pendant l'échantillonnage (à droite), on génère $z \sim p(z|x(t))$ puis on obtient $y = f_\phi^{-1}(z, x(t))$.

Cette approche permet de capturer explicitement les dépendances structurelles et la multimodalité des données. Contrairement aux méthodes de diffusion, qui génèrent les échantillons par un processus multi-étapes de débruitage, les CNFs offrent une estimation directe et explicite de la vraisemblance, rendant le modèle plus adapté aux tâches de prédiction structurée en dynamique moléculaire.

2.2 Préparation des données

Sur la base du formalisme décrit ci-dessus, le modèle *Timewarp* est entraîné à partir de paires de configurations moléculaires consécutives extraites de simulations de dynamique moléculaire (MD). Chaque paire $(x^p(t), x^p(t+\tau))$ est enrichie par des vitesses auxiliaires $(x^v(t), x^v(t+\tau))$ tirées d'une loi normale standard $\mathcal{N}(0, I)$. Le modèle f_θ apprend ainsi à prédire la conformation suivante à partir d'un état initial et d'un bruit gaussien, ce qui formalise l'approximation de la distribution conditionnelle $p_\theta(x(t+\tau) \mid x(t))$. Ainsi, chaque paire consécutive est indispensable pour l'entraînement du modèle.

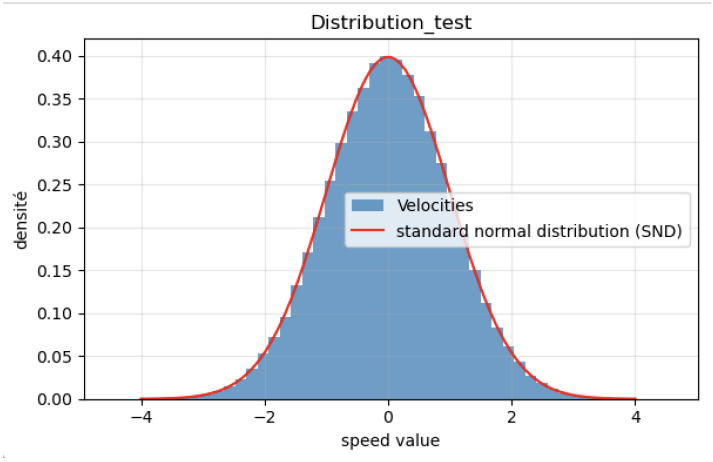


FIGURE 2 – Distribution des vitesses auxiliaires générées pour l'alanine dipeptide. L'histogramme (bleu) coïncide avec la densité théorique $\mathcal{N}(0, I)$ (rouge), validant la préparation des données.

Les trajectoires de référence ont été obtenues pour l'alanine dipeptide par simulation de dynamique moléculaire classique à 310 K avec un thermostat de Langevin ($\gamma = 0.3$ ps^{-1}), en utilisant un pas de temps de 1 fs et une durée totale de 0.752 ns par trajectoire. Quatre trajectoires indépendantes ont été générées, représentant environ 752 000 pas de simulation au total. À partir de ces trajectoires, des paires de configurations $(x^p(t), x^p(t+\tau))$ ont été extraites toutes les 2 ps, ce qui fournit environ 376 points par trajectoire. Chaque

paire est complétée par des vitesses auxiliaires $(x^v(t), x^v(t+\tau))$ échantillonnées selon $\mathcal{N}(0, I)$, conformément à la définition du modèle *Timewarp*. Le jeu de données final contient environ 1500 paires utilisées comme entrées du modèle.

La distribution des vitesses générées a été vérifiée expérimentalement (Figure 2) : l'histogramme observé coïncide avec la densité théorique d'une loi normale standard, garantissant ainsi la cohérence physique du jeu de données.

2.3 Architecture du modèle Timewarp

L'architecture de Timewarp s'appuie sur trois briques principales : (i) des couches de couplage de type RealNVP, (ii) des modules *Atom Transformer* chargés de paramétrer les transformations affines, (iii) une attention locale par noyau multi-têtes permettant de modéliser les interactions entre atomes. Nous détaillons ci-dessous chacun de ces composants, en commençant par les couches de couplage RealNVP. Cette architecture complète est illustrée sur la Figure. 3, qui montre la complémentarité entre les trois briques principales.

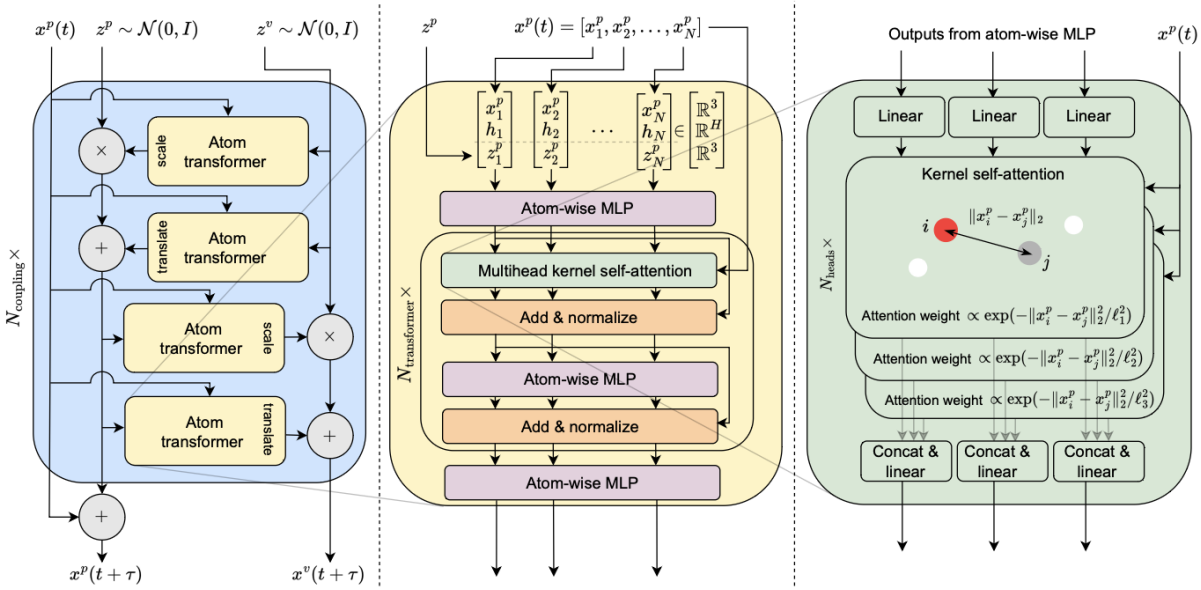


FIGURE 3 – Schéma de l'architecture du modèle *Timewarp*. *Gauche* : une couche de couplage RealNVP conditionnelle. *Centre* : un module *Atom Transformer*. *Droite* : le module d'attention locale multi-têtes (kernel self-attention). [5]

2.3.1 RealNVP Coupling Flow

Le flow normalisant utilisé suit l'architecture RealNVP [8] avec des couches de type *coupling layers*. À chaque étape, une partie des variables est maintenue fixe et sert de condition pour transformer l'autre partie, via une transformation affine composée d'un facteur de mise à l'échelle (*scale*) et d'un facteur de translation (*shift*). Cette structure rend la transformation bijective et permet de calculer efficacement le Jacobien associé, condition nécessaire pour estimer les densités de probabilité.

Formellement, dans la ℓ-ième couche de couplage, les mises à jour s'écrivent :

$$z^p_{\ell+1} = s^p_{\ell,\theta}(z^v_\ell; x^p(t)) \odot z^p_\ell + t^p_{\ell,\theta}(z^v_\ell; x^p(t)), \tag{3}$$

$$z^v_{\ell+1} = s^v_{\ell,\theta}(z^p_{\ell+1}; x^p(t)) \odot z^v_\ell + t^v_{\ell,\theta}(z^p_{\ell+1}; x^p(t)), \tag{4}$$

où l'opérateur $\odot$ désigne le produit élément par élément. Les fonctions $s^p_{\ell,\theta}, t^p_{\ell,\theta} : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ produisent respectivement les facteurs de *scale* et de *shift* appliqués aux variables de position, tandis que $s^v_{\ell,\theta}, t^v_{\ell,\theta} : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ jouent un rôle analogue pour les variables auxiliaires de vitesse.

Dans notre cas, les positions x et les vitesses auxiliaires v sont mises à jour de manière alternée : les vitesses servent de condition pour transformer les positions, puis les positions mises à jour servent à leur tour de condition

pour transformer les vitesses. Les facteurs de transformation des positions dépendent uniquement des vitesses, et réciproquement pour les vitesses, ce qui rend le Jacobien triangulaire inférieur et permet un calcul efficace de son déterminant [5]. Les paramètres de ces transformations sont produits par un réseau de type *Atom Transformer*, ce qui permet d'intégrer l'information structurale et locale des configurations moléculaires dans le processus de transformation. Cette conception garantit à la fois la réversibilité mathématique du flow et la cohérence physique des échantillons générés. *Notons enfin la présence d'une connexion résiduelle : la sortie du flow prédit le changement $x(t+\tau) - x(t)$, plutôt que $x(t+\tau)$ directement (cf. Fig. 2).*

2.3.2 Atom transformer

Les paramètres de transformation (s_θ, t_θ) sont produits par un réseau de type *Atom Transformer*, dont l'objectif est de générer dynamiquement les facteurs de *scale* et de *shift* à partir de l'information locale de la configuration moléculaire. Ce module agit comme un réseau auxiliaire qui fournit les coefficients nécessaires aux transformations affines dans les couches de couplage RealNVP.

Pour chaque atome i, on construit un vecteur d'entrée $a_i = [x_i^p(t),\, h_i,\, z_i]^\top \in \mathbb{R}^{H+6}$, où $x_i^p(t) \in \mathbb{R}^3$ est la position de l'atome i dans l'état de conditionnement, $h_i \in \mathbb{R}^H$ est un embedding appris du type atomique, et $z_i \in \mathbb{R}^3$ est une variable latente auxiliaire (soit z_i^v, soit z_i^p selon le contexte). Notons que z_i^p est exclu lorsqu'on calcule s_θ^p, t_θ^p afin de préserver la bijectivité du flow.

Les vecteurs a_i sont ensuite projetés dans un espace de dimension D via un perceptron multi-couche ($\phi_{in} : \mathbb{R}^{H+6} \to \mathbb{R}^D$), puis traités par $N_{\text{transformer}}$ couches empilées, combinant *kernel self-attention*, normalisation et connexions résiduelles. Le mécanisme d'attention repose uniquement sur les positions $x^p(t)$, ce qui reflète la contrainte physique selon laquelle les interactions locales d'une molécule à un instant donné dépendent uniquement de la configuration spatiale.

Enfin, chaque atome produit une nouvelle représentation enrichie, projetée dans $\mathbb{R}^3$ par un dernier MLP ($\phi_{out} : \mathbb{R}^D \to \mathbb{R}^3$), qui fournit les paramètres (facteurs de *scale* ou de *shift*) utilisés dans les couches RealNVP. Ainsi, l'Atom Transformer agit comme un module intermédiaire : il agrège l'information structurelle et spatiale, puis délivre des coefficients qui guident la transformation du flow, tout en respectant les contraintes de permutation et de localité [5].

2.3.3 Kernel self-attention

Dans une molécule, les interactions physiques sont *locales* : les atomes voisins s'influencent fortement, tandis que l'influence des atomes éloignés est négligeable. Pour capturer cette localité, l'*attention* n'est pas définie par la similarité des vecteurs caractéristiques comme dans les Transformers classiques, mais directement en fonction de la distance euclidienne entre paires d'atomes, au moyen d'un noyau de type RBF [7], dont la forme est $K(x_i, x_j) = \exp\big(-\gamma \|x_i - x_j\|^2\big)$.

Les poids d'attention w_{ij}, représentant l'influence de l'atome j sur i, sont définis à l'aide d'un noyau RBF :

$$w_{ij} = \frac{\exp\big(-\|x_i^p - x_j^p\|_2^2/\ell^2\big)}{\sum_{j'=1}^{N} \exp\Big(-\|x_i^p - x_{j'}^p\|_2^2/\ell^2\Big)}, \tag{5}$$

où ℓ est un paramètre de portée spatiale (*lengthscale*). Plus deux atomes sont proches, plus leur similarité (et donc leur influence réciproque) est forte.

Une fois les poids calculés, la représentation de chaque atome i est mise à jour par une moyenne pondérée des représentations de tous les autres atomes j :

$$r_{\text{out},i} = \sum_{j=1}^{N} w_{ij}\, V \cdot r_{\text{in},j}, \tag{6}$$

où $V \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ est une matrice linéaire apprise, et $r_{\text{in},j}$ désigne la représentation en entrée de l'atome j.

Afin de capturer des échelles spatiales multiples, une version *multi-head* est utilisée : plusieurs têtes calculent simultanément des poids w_{ij} avec des valeurs de ℓ différentes, puis agrègent les informations locales à différentes portées spatiales. Pour chaque tête, on utilise un *lengthscale* distinct pour calculer les poids d'attention selon

l'équation 5, puis on met à jour les caractéristiques atomiques conformément à l'équation 6. Le mécanisme multi-head consiste à répéter ce processus en parallèle avec plusieurs valeurs de ℓ, ce qui permet d'agréger l'information à différentes échelles spatiales.

2.4 Mécanisme d'utilisation

Le modèle *Timewarp* repose sur un flow normalisant conditionnel, structuré en deux processus complémentaires : **a) Phase inverse (Entraînement)** : La conformation future $x(t+\tau)$ est projetée dans l'espace latent gaussien $z = f_\phi(x(t+\tau), x(t))$. La vraisemblance conditionnelle $p(z|x(t))$ est évaluée, ce qui permet d'optimiser les paramètres θ du modèle en maximisant la log-vraisemblance. **b) Phase directe (Exploration)** : A partir d'un état courant $x(t)$, un échantillon latent $z \sim p(z|x(t))$ est généré. Celui-ci est transformé par $f_\phi^{-1}(z, x(t))$ pour produire une conformation candidate $x(t+\tau)$. *Nous utilisons une connexion résiduelle* : la sortie du flow représente le déplacement $\Delta x = x(t+\tau) - x(t)$, et la proposition s'écrit donc $\tilde{x}(t+\tau) = x(t) + \Delta x$. Enfin, la proposition est ensuite validée via une étape de Monte Carlo (critère de Metropolis-Hastings), ce qui garantit la conformité avec la distribution de Boltzmann.

2.4.1 Phase I - Entraînement (Flow inverse)

Dans la phase d'entraînement, notre objectif est d'apprendre la distribution conditionnelle des évolutions futures, en utilisant le flow dans le sens inverse. A l'entraînement, pour chaque paire de structures $(x(t), x(t+\tau))$, on part de la configuration cible (future) $x(t+\tau)$ et on la projette dans l'espace latent gaussien à l'aide du flow inverse f_ϕ^{-1}. Cette projection permet au modèle d'apprendre une représentation compacte et universelle des trajectoires futures, directement exploitable lors de la phase de génération.

La distribution conditionnelle que l'on cherche à approximer s'écrit [5] :

$$p_\theta(x(t+\tau)|x(t)) = \mathcal{N}\Big(f_\theta^{-1}(x(t+\tau); x(t)); 0, I\Big) \left|\det \mathcal{J}_{f_\theta^{-1}(\cdot; x(t))}(x(t+\tau))\right|, \tag{7}$$

où $x(t+\tau)$ est la future configuration moléculaire que l'on souhaite prédire, conditionnée sur l'état initial $x(t)$; f_θ^{-1} désigne la projection inverse par le flow, ramenant la structure future observée dans l'espace latent gaussien ; $\mathcal{N}(\cdot; 0, I)$ est la densité de probabilité dans l'espace latent gaussien ; $\det \mathcal{J}_{f_\theta^{-1}}$ est le déterminant du Jacobien, qui ajuste la probabilité pour prendre en compte la déformation de l'espace induite par le flow.

En pratique, l'apprentissage se fait en maximisant la log-vraisemblance de toutes les trajectoires futures observées. Pour chaque paire $(x^{(k)}(t), x^{(k)}(t+\tau))$ de notre dataset, le modèle prédit la probabilité d'observer l'état futur connaissant l'état initial. On optimise alors :

$$\mathcal{L}_{\text{lik}}(\theta) = \frac{1}{K} \sum_{k=1}^{K} \log p_\theta(x^{(k)}(t+\tau)|x^{(k)}(t)), \tag{8}$$

où K est le nombre total de paires d'entraînement.

Cette formulation correspond à un objectif de *maximum de vraisemblance* classique : Le modèle est entraîné à donner une forte probabilité aux transitions observées, en couvrant l'ensemble des évolutions futures possibles, et pas uniquement la plus probable.

2.4.2 Phase II - Exploration (MCMC avec le flow entraîné)

Une fois le modèle entraîné (Phase I), le flow est utilisé en sens direct comme *générateur de propositions*. L'objectif est désormais de produire des échantillons qui suivent correctement la distribution de Boltzmann, favorisant les conformations de faible énergie. Pour cela, le flow est intégré dans un schéma de type Metropolis–Hastings (MCMC).

1. Génération de propositions À partir d'une structure courante $X_m = (x_m^p, x_m^v)$, le modèle génère une nouvelle proposition $\tilde{X}_m$ selon sa distribution conditionnelle :

$$\tilde{X}_m \sim p_\theta(\cdot \mid x_m^p). \tag{9}$$

où $\tilde{X}$ désigne une proposition issue du flow, à partir de l'état courant. Cette étape exploite directement la capacité du modèle appris à approximer les trajectoires futures plausibles.

2. Critère d'acceptation La proposition n'est pas automatiquement acceptée : elle est validée via la règle de Metropolis–Hastings, qui combine la distribution cible de Boltzmann et la probabilité donnée par le modèle :

$$\alpha(X, \tilde{X}) = \min\left(1, \frac{\mu_{\text{aug}}(\tilde{X})\, p_\theta(X|\tilde{X}^p)}{\mu_{\text{aug}}(X)\, p_\theta(\tilde{X}|X^p)}\right), \tag{10}$$

où μ_{aug} est la distribution cible de Boltzmann, qui attribue une probabilité plus élevée aux conformations d'énergie potentielle plus faible ; p_θ est la probabilité donnée par le modèle flow entraîné ; le rapport compare la vraisemblance de la proposition et celle de l'état courant. Si ce rapport est supérieur à 1, la proposition est acceptée. Sinon, elle est acceptée avec la probabilité $\alpha(X, \tilde{X})$; dans le cas contraire, l'état reste inchangé.

3. Ré-échantillonnage des vitesses Après chaque proposition, les variables auxiliaires (ici, les vitesses x^v) sont ré-échantillonnées par un pas de Gibbs sampling selon une loi normale standard : $x^v \sim \mathcal{N}(0, I)$ assurant ainsi une exploration correcte de l'espace des états.

4. Objectifs d'optimisation L'entraînement est affiné pour maximiser le taux d'acceptation des propositions. On introduit pour cela le ratio d'acceptation dépendant du modèle :

$$r_\theta(X, \tilde{X}) = \frac{\mu_{\text{aug}}(\tilde{X})\, p_\theta(X|\tilde{X}^p)}{\mu_{\text{aug}}(X)\, p_\theta(\tilde{X}|X^p)}, \tag{11}$$

et la fonction objectif correspondante :

$$\mathcal{L}_{\text{acc}}(\theta) = \frac{1}{K}\sum_{k=1}^{K} \log r_\theta(x^{(k)}(t), \tilde{x}_\theta^{(k)}(t+\tau)). \tag{12}$$

Afin d'éviter que le modèle ne se limite à des propositions triviales ($\tilde{x}_\theta^{(k)}(t+\tau) = x^{(k)}(t)$), on ajoute une régularisation par entropie différentielle :

$$\mathcal{L}_{\text{ent}}(\theta) = -\frac{1}{K}\sum_{k=1}^{K} \log p_\theta(\tilde{x}_\theta^{(k)}(t+\tau) \mid x^{(k)}(t)). \tag{13}$$

5. Résultat et justification En répétant le cycle *proposition* → *acceptation* → *ré-échantillonnage*, la chaîne de Markov converge vers la distribution de Boltzmann, conformément à la théorie de Metropolis–Hastings. Dans nos expériences, le taux d'acceptation obtenu est de l'ordre de 10%, ce qui correspond bien aux valeurs rapportées dans la littérature (1–15%) [5].

3 Résultats

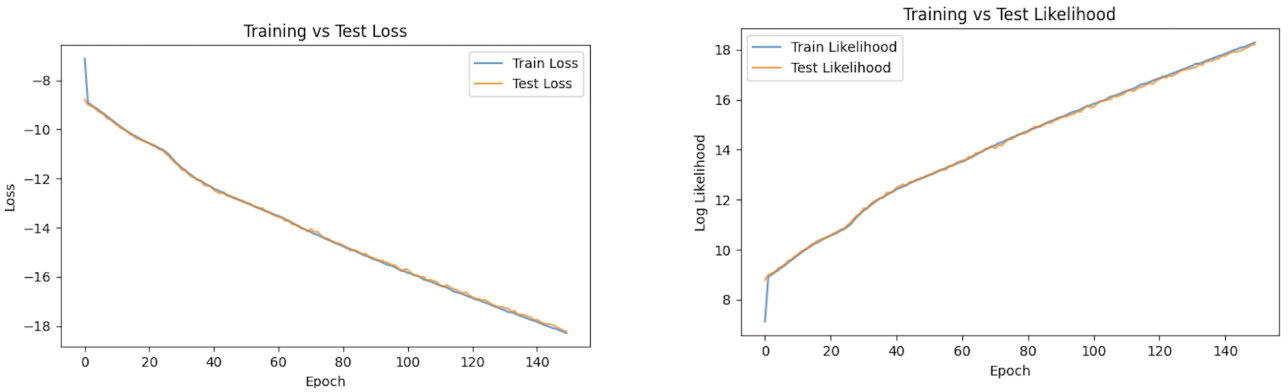
3.1 Résultats d'entraînement

Nous avons implémenté et entraîné *Timewarp* sur notre propre jeu de données ($\sim$ 1500 paires $(x(t), x(t+\tau))$). Conformément à la formule 8, l'apprentissage vise à *maximiser* la log-vraisemblance conditionnelle moyenne $\mathcal{L}_{\text{lik}}(\theta)$ des futurs observés. En pratique, on trace à la fois (i) la courbe de log-vraisemblance (à maximiser) et (ii) la *loss* optimisée par le code, qui n'est autre que son opposé :

$$\text{Loss}(\theta) \;=\; -\,\mathcal{L}_{\text{lik}}(\theta) \;=\; -\frac{1}{K}\sum_{k=1}^{K} \log p_\theta\big(x^{(k)}(t+\tau) \mid x^{(k)}(t)\big),$$

ce qui explique que, sur nos courbes, la loss décroît tandis que la log-vraisemblance augmente de manière fortement corrélée.

Les figures 4a et 4b montrent que les courbes *train* et *test* se superposent presque parfaitement : la convergence est stable, sans écart significatif entre entraînement et validation. Cela indique que le modèle apprend correctement la structure statistique des paires du jeu de données. Étant donné la taille modeste du dataset ($\sim$ 1500 paires), il subsiste toutefois un risque d'adaptation aux exemples spécifiques ; nous interprétons donc cette stabilité comme une bonne optimisation du critère de vraisemblance, tout en gardant à l'esprit que des validations supplémentaires (p. ex. séparation stricte par trajectoires, ou validation croisée) seraient utiles pour évaluer la généralisation.

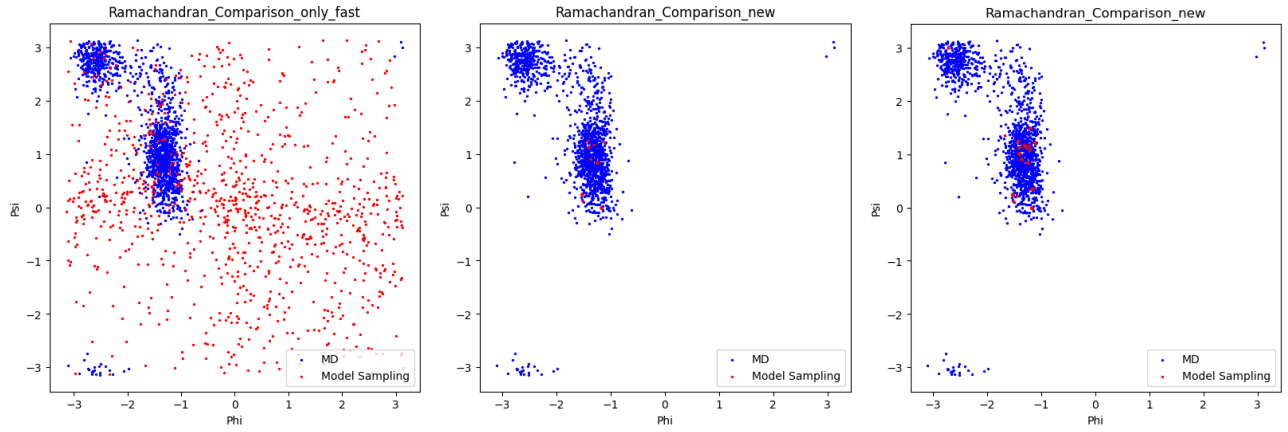


(a) Train vs Test Loss (minimisée).

(b) Train vs Test Log-likelihood (maximisée).

FIGURE 4 – Évolution conjointe de la loss et de la log-vraisemblance au cours de l'entraînement : La loss correspond ici à l'opposé de la log-vraisemblance moyenne sur le dataset (cf. Phase I) : sa diminution régulière est donc équivalente à l'augmentation de la log-vraisemblance. On observe une convergence stable des courbes train et test, sans écart significatif, ce qui montre que le modèle apprend correctement la distribution conditionnelle à partir d'un dataset de taille modeste ($\sim$ 1500 paires).

3.2 Comparaison des régimes d'échantillonnage



(a) Sans MCMC (flow direct).

(b) MCMC–500 avec `num_steps`=5 000 et `save_interval`=10.

(c) MCMC–500 000 avec `num_steps`=500 000 et `save_interval`=1.

FIGURE 5 – **Comparaison des distributions Ramachandran (ϕ, ψ) : MD (bleu) vs Timewarp (rouge). (a)** Sans correction MCMC — les échantillons restent très diffus. **(b)** MCMC court (500 états sauvegardés) — la distribution commence à se concentrer autour de certains bassins. **(c)** MCMC long (500 000 états sauvegardés) — la concentration s'accentue et se rapproche davantage de la référence MD, mais la couverture reste locale. Cette comparaison illustre le rôle du mécanisme d'acceptation Metropolis–Hastings : des chaînes plus longues tendent à mieux aligner la distribution du modèle avec celle de la référence. Dans nos expériences, le taux d'acceptation est d'environ $\sim$ 10% ; malgré une amélioration notable, la distribution ne converge pas entièrement vers la MD, ce qui s'explique par la taille limitée du jeu d'entraînement ($\sim$ 1500 paires) ainsi que par les dynamiques propres au modèle. Pour assurer une comparaison équitable, toutes les cartes ont été tracées avec les mêmes bornes de coordonnées.

Après validation de la Phase I, nous évaluons la qualité d'exploration (Phase II). La Figure 5 compare la distribution des angles dièdres (ϕ, ψ) entre la dynamique moléculaire de référence (MD) et les échantillons

générés par Timewarp, sous trois régimes distincts : (i) sans correction MCMC, (ii) avec une chaîne courte (500 étapes), et (iii) avec une chaîne longue (500000 étapes).

Cette comparaison permet d'illustrer le rôle du mécanisme d'acceptation Metropolis-Hastings : sans correction, les échantillons sont très diffus et sous-échantillonnent certains bassins ; après un petit nombre d'itérations, la distribution commence à se rapprocher des états métastables, et avec un grand nombre d'itérations, elle se concentre davantage autour des bassins principaux, conformément à la loi de Boltzmann.

Notons toutefois que, dans nos expériences, la convergence reste locale : même avec 5×10^5 itérations, les échantillons ne couvrent pas complètement l'espace conformationnel observé en MD. Nous interprétons cela comme une limite liée à la taille restreinte du dataset d'entraînement ($\sim$1500 paires), ainsi qu'à la dynamique intrinsèque du modèle.

3.3 Validation structurale (Exemple de conformation générée)

Au-delà de l'évaluation statistique (Phase I et Phase II), nous avons souhaité vérifier que les échantillons générés correspondent bien à une structure moléculaire réaliste. Pour cela, nous avons extrait une frame (la 100^{e}) du fichier `trajectory_coords.npy` produit par le pipeline complet (flow inverse pour l'entraînement, suivi du flow direct avec échantillonnage MCMC). Cette configuration a été convertie en format `.pdb` et visualisée à l'aide d'outils de modélisation moléculaire.

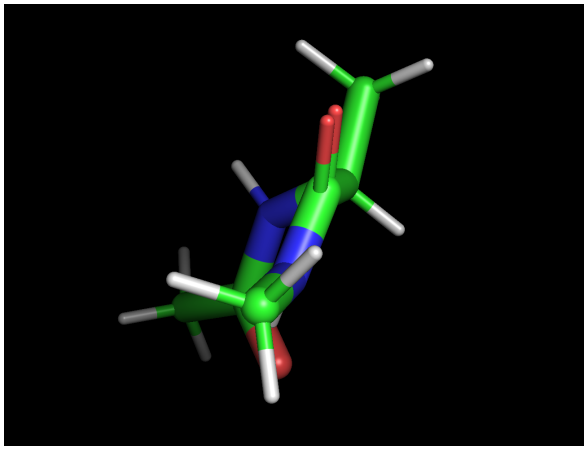


FIGURE 6 – Exemple de structure moléculaire obtenue à partir d'une frame générée par Timewarp (frame 100 du fichier `trajectory_coords.npy`). La structure correspond bien à un alanine dipeptide, observé ici dans une conformation particulière. Cette visualisation illustre concrètement l'un des échantillons dont la statistique globale est représentée dans les cartes de Ramachandran.

Le résultat, illustré en Figure 6, montre clairement un *alanine dipeptide* cohérent : la connectivité atomique et la géométrie locale respectent la structure chimique attendue. Cette observation confirme que le modèle Timewarp ne génère pas seulement une distribution statistique cohérente des angles dièdres, mais est capable de produire des conformations moléculaires physiquement valides.

4 Discussion

Taille et diversité du jeu d'entraînement. Le jeu de données utilisé dans cette étude ne contient qu'environ $\sim$ 1500 paires $(x(t), x(t+\tau))$. Pour un flow conditionnel guidé par un *Atom Transformer*, cette taille reste limitée et ne couvre qu'une partie restreinte de l'espace conformationnel. Nos résultats montrent que l'allongement massif de la chaîne MCMC (500 contre 500 000 étapes) n'apporte qu'une amélioration marginale dans les cartes de Ramachandran : La limitation ne provient donc pas principalement de la phase d'exploration, mais plutôt de l'apprentissage, qui manque de diversité statistique. Ainsi, il serait pertinent d'augmenter la taille du dataset en générant davantage de trajectoires MD et en variant les valeurs de τ, afin d'enrichir la dynamique apprise et de mieux évaluer la généralisation.

Validation structurale encore ponctuelle. Nous avons présenté un exemple qualitatif de structure (frame 100 de la trajectoire générée), qui correspond bien à un alanine dipeptide. Toutefois, cette validation reste ponctuelle : une seule frame isolée ne permet pas d'évaluer la cohérence dynamique des conformations générées. Une piste naturelle consiste donc à comparer plusieurs séquences consécutives de frames, à calculer le RMSD vis-à-vis de conformations de référence et à confronter les distributions de longueurs de liaisons et d'angles internes à celles issues de la MD, ce qui permettrait de vérifier si les trajectoires générées conservent un comportement physiquement réaliste.

« Sans MCMC » vs protocole de l'article. Dans nos expériences, la condition « sans MCMC » correspond à l'utilisation du flow direct sans étape de correction. Cette configuration n'est pas strictement équivalente à celle utilisée dans l'article original, ce qui peut expliquer une dispersion plus marquée des échantillons et une couverture moins fidèle des bassins. Il conviendrait donc de préciser clairement cette différence méthodologique et, dans de futures expériences, d'aligner plus strictement le protocole avec celui de la publication (même nombre d'échantillons, filtrage ou burn-in identiques), afin de rendre la comparaison plus directe et d'évaluer si des améliorations sont possibles même en l'absence de correction MCMC.

5 Conclusion

Ce protocole, testé ici sur un système modèle, pourrait à terme être appliqué à des biomolécules beaucoup plus complexes. Une perspective particulièrement prometteuse serait de découper un système de grande taille, tel que le ribosome, en sous-unités plus petites, de générer des ensembles conformationnels locaux pour chaque fragment à l'aide de flows conditionnels basés sur un *Transformer*, puis d'utiliser le modèle entraîné comme prédicteur flexible des configurations possibles, suivant la distribution de Boltzmann. Mon travail constitue une première tentative dans cette direction : j'ai pu mettre en place l'ensemble du processus et en démontrer la faisabilité, mais les résultats restent limités par la puissance de calcul disponible et les contraintes liées aux environnements de calcul à distance.

Références

[1] Lauren Wickstrom, Alan Okur, and Carlos Simmerling. Multidimensional Replica Exchange Molecular Dynamics Yields a Converged Ensemble of an RNA Tetranucleotide. *Journal of Chemical Theory and Computation*, 5(5) :1347–1352, 2009. https://pmc.ncbi.nlm.nih.gov/articles/PMC3893832/

[2] Joseph Anderson, et al. Cascades of Information with Message-Passing. *arXiv preprint arXiv :2012.06333*, 2020. https://arxiv.org/abs/2012.06333

[3] Grégoire Sergeant-Perthuis, et al. Message Passing and Structure Assembly in Ribosomal Modeling (Abstract). HAL preprint, 2024. https://hal.sorbonne-universite.fr/hal-04527780/file/abstract_ACT_submission.pdf

[4] Mojtaba Shafiei, et al. Probabilistic Message Passing for Molecular Systems. *arXiv preprint arXiv :2201.11876*, 2022. https://arxiv.org/pdf/2201.11876

[5] Leon Klein, Andrew Y. K. Foong, Tor Erlend Fjelde, Bruno Mlodozeniec, Marc Brockschmidt, Sebastian Nowozin, Frank Noé, and Ryota Tomioka. Timewarp : Transferable Acceleration of Molecular Dynamics by Learning Time-Coarsened Dynamics. *arXiv preprint arXiv :2302.01170*, 2023. https://arxiv.org/abs/2302.01170

[6] Michael Winkler and Frank Noé. Timewarp : Transferable Acceleration of Molecular Dynamics by Learning Time-Coarsened Dynamics. *arXiv preprint arXiv :1912.00042*, 2019. https://arxiv.org/pdf/1912.00042

[7] Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. *Journal of Machine Learning Research*, 11 :1471–1490, 2010. https://jmlr.org/papers/v11/chang10a.html

[8] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017. https://arxiv.org/abs/1605.08803