Génération de comportements non verbaux à partir de la parole
M2 IAAA 2022-2023

Yannis Formery
Aix Marseille Université
yannis.formery@etu.univ-amu.fr

Magalie Ochs, Stephane Ayache
Aix Marseille Université
magalie.ochs@lis-lab.fr
stephane.ayache@lis-lab.fr

Abstract

Notre objectif est de générer des comportements non-verbaux d'agents virtuels automatiquement à partir de caractéristiques audios sans s'intéresser à son contenu sémantique. Nous étudions deux modèles de diffusion qui génèrent simultanément les mouvements de têtes, l'orientation du regard et les unités d'action de la méthode FACS : le premier modèle de diffusion intègre les caractéristiques audios par des couches d'attention croisées tandis que le deuxième modèle les intègre en concaténant en entrée avec les données à générer. Les deux modèles sont fondés sur l'architecture U-NET. Nous évaluons notre modèle sur un/deux dataset de deux heures de vidéo, à l'aide de plusieurs métriques de l'état de l'art. Nous évaluerons donc les 2 manières d'intégrer la condition audio ainsi que différentes associations de caractéristiques audio pouvant servir à la génération de nos comportements non-verbaux. Nos expériences montrent que l'intégration de l'audio marche nettement mieux et qu'il est encore trop tôt pour dire quels caractéristiques audios sont les plus utiles pour la génération de comportements non-verbaux.

1 Introduction

La génération de comportements est un sujet de recherche dynamique. La présence d'agents virtuels devient indispensable dans des applications faisant appel à de la réalité virtuelle comme dans les jeux vidéo ou les logiciels de simulation avec environnement virtuel. Ce travail s'inscrit dans le cadre de plusieurs projets : le projet ANR COPAINS qui vise à développer un personnage virtuel persuasif pour inciter les personnes âgées à faire du sport et le projet TRUENESS qui vise à développer une plateforme de réalité virtuelle peuplée de personnages virtuels pour former les individus à lutter contre la discrimination sociale. Dans ce travail, nous nous intéressons à la génération de comportements non-verbaux afin de construire des agents virtuels plus réalistes. Comme de nombreuses tâches impliquant une animation, générer des mouvements d'agents virtuels nécessite la présence d'un expert pour rendre ces derniers crédibles, naturels et réalistes. Cette étape étant fastidieuse et onéreuse, les dispositifs de production automatisés basés sur l'IA générative devraient grandement faciliter la production d'agents virtuels dans de nombreux secteurs.

Bien que des systèmes de générations de mouvements à partir de la parole voient le jour (voir 2) et que ces derniers utilisent des caractéristiques textuelles, prosodiques ou acoustiques, la tache présente la difficulté de pouvoir générer une bonne diversité de mouvement sachant que plusieurs mouvements possibles peuvent être associés à la même parole. Elle présente aussi la difficulté de générer des mouvements fluides et naturels surtout dans les transitions entre pas de temps. Notre travail de recherche se focalisera sur comment intégrer la condition de la parole seulement sur la génération de comportements non-verbaux. Ce travail s'est concentré sur les modèles génératifs les plus récents, basés sur les processus de diffusion. L'objectif est la génération des mouvements de têtes, l'orientation du regard et les expressions faciales conditionnellement à la modalité audio.

Les contributions de ce travail sont :

- Étudier plusieurs modèles de diffusions qui génèrent simultanément les mouvements de tête,l'orientation du regard et les unités d'action (AUs) de la méthode FACS.
- Analyser quels caractéristiques audios sont pertinents pour la génération de mouvement
- Comparer plusieurs manières d'intégrer la

condition de la parole dans nos modèles génératifs.

Définition du problème :
Au vu de notre objectif défini ci-dessus. Le problème peut être posé de la manière suivante :
Soit une séquence de caractéristiques issues d'enregistrement audio (prosodiques, acoustiques et on ignore le contenu de l'audio) que nous noterons $X_{prosody}[0:T]$, l'objectif est de générer une séquence de mouvements et d'expressions du visage. $Y_{behaviour}[0:T]$. Nous remarquerons que la séquence générée est de même longueur que

la séquence issue d'enregistrement audio, car on veut assurer une correspondance réelle, naturelle et crédible entre les comportements d'un agent virtuel et ce qu'il dit. $Y_{behaviour}[0:T]$ comprend 3 groupes de caractéristiques à générer :

- $Y_{head}[0:T]$ qui sont les mouvements de tête exprimés en coordonnée 3D.
- $Y_{gaze}[0:T]$ qui sont les orientations de la tête exprimées également en cordonnée 3D.
- $Y_{AU}[0:T]$ qui sont, cette fois-ci, exprimé par la méthode Facial Action Coding System (FACS) (Ekman and Friesen, 1978)

2 État de l'art

Il est possible de catégoriser les recherches existantes sur la production de comportements en différentes classes, à savoir les méthodes fondées sur des règles et les approches qui se fondent sur les données.

2.1 Méthodes fondées sur des règles

Cassell et al. (1998) fondent le premier système fondé sur les règles, *Animated Conversation*, qui génère de manière automatique des gestes de la main, des expressions faciales et des modèles d'intonation entre plusieurs entités de nature humaine. Pour cela, ils étudient la corrélation sous-jacente entre le langage et le mouvement afin de produire une animation authentique. Contrairement au travail précédent, Kopp and Wachsmuth (2002) ont suggéré une méthode fondée sur un schéma pour produire des énoncés complexes multimodaux (c.-à-d. geste et parole) à partir des indications au format XML de leur forme, plutôt que de se reposer sur des gestes préalablement enregistrés.
Bien que les systèmes de génération de gestes fondés sur des règles soient capables de produire des gestes parfaitement synchronisés avec la parole, la distribution des gestes est souvent peu diversifiée. De tels systèmes sont rigides en ce sens qu'ils ne peuvent produire qu'un petit ensemble d'éléments plausibles de gestes pour une entrée vocale ou un scénario particulier. Par conséquent, l'incapacité à produire des gestes divers de manière non-déterministe signifie que les agents virtuels résultants (ou tout autre mode de réalisation) ne peuvent se comporter de manière expressive et naturelle que pour des exemples limités. Des méthodes fondées sur les données ont été proposées pour tenter de surmonter ces limites.

2.2 Méthodes fondées sur les données

Elles ne dépendent pas d'experts en animation et en linguistique pour fonctionner. Elles apprennent les relations entre la parole et les mouvements.Bergmann and Kopp (2009) proposent une méthode statistique alternative pour représenter la conversion du langage décrivant des objets en gestes iconiques qui leur ressemblent en utilisant les réseaux de décision bayésiens.Bergmann and Kopp (2009) ont suggéré l'utilisation d'un modèle de Markov caché (HMM) pour identifier le clip de mouvement le plus adéquat à partir d'une base de données de capture de mouvement. Pour ce faire, ils ont utilisé des caractéristiques fondées sur la prosodie extraite du discours original. Toujours en conservant les traits prosodiques, Chiu and Marsella (2011) utilisent un générateur de gestes à partir d'un *Hierarchical Factored Conditional Restricted Boltzmann Machine* (HFCRB) en créant tout d'abord une représentation compactée du mouvement par la formation d'une Machine de Boltzmann Restreinte Conditionnelle (CRBM) grâce à un algorithme d'apprentissage non supervisée. Toutefois, ces approches statistiques permettent seulement d'associer une sortie unique pour une entrée précise. Il devient alors nécessaire d'instaurer, en plus de la condition, une génération aléatoire des mouvements, car un discours peut être associé à une multitude de mouvements existants ou pas encore observés.
De nos jours, les systèmes fondés sur les réseaux de neurones ont montré de meilleures performances dans l'apprentissage à partir de grandes quantités de données et la génération de mouvements sous forme de séquences d'images. En se focalisant sur les travaux concernant la génération

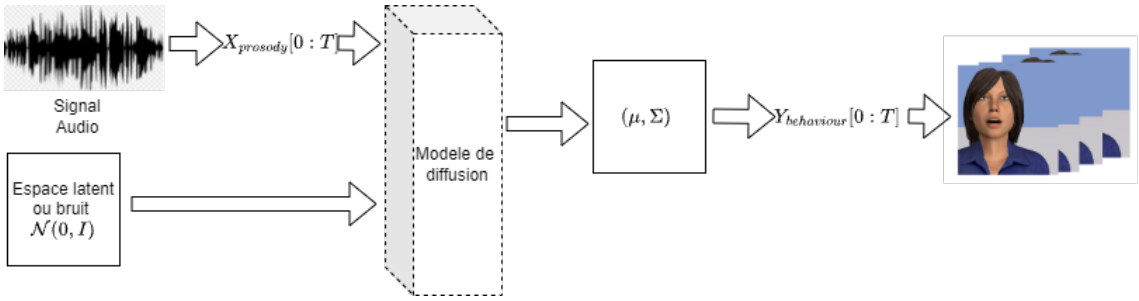

Figure 1: Processus de génération du comportement, $X_{prosody}$

de mouvements à partir de paroles, nous trouvons Hasegawa et al. (2018) qui proposent la génération de mouvements à partir de discours audio. Ils utilisent pour cela un réseau de neurones récurrent LSTM bidirectionnel pour capter les dépendances temporelles longues distances suivies d'un filtre temporel pour lisser les mouvements et limiter les discontinuités dans la séquence générée. Toutefois, ce modèle ne peut générer qu'une seule séquence possible de mouvements pour une entrée quelconque. Kucherenko et al. (2019) améliore le travail précédent en utilisant un autoencodeur sur les mouvements à générer afin de retirer le filtre temporel. Kucherenko (2018) entraînent un variational autoencoder (VAE) sur les mouvements puis ils remplacent la partie encodeur du VAE par un encodeur de parole audio puis entraînent à nouveau de sorte que la distribution engendrée par le remplacement de l'encodeur se rapproche du VAE originel. Les GANs (Generative Adversarial Network), modèles ayant connu de forts progrès ces 10 dernières années (Goodfellow et al., 2014), et les CGAN (Conditionnal Generative Adversarial Network) (Mirza and Osindero, 2014) sont utilisées par Sadoughi and Busso (2018), Ginosar et al. (2019), Ferstl et al. (2020), Rebol et al. (2021) dont les 2 premiers utilisent ces derniers avec des couches récurrentes de type LSTM bidirectionnel.

2.3 Méthodes fondées sur les modèles de diffusion

Bien qu'il n'existe pas de travaux spécifiques sur la tache nous concernant, nous pouvons trouver des travaux focalisés sur des taches proches de la nôtre. Nous pouvons retrouver Tevet et al. (2022) qui génèrent des mouvements de corps à partir de texte, d'actions représentées par des classes ou sans condition apparente. Ils se servent d'un ensemble de données contenant des positions d'os, des vitesses d'os et des coordonnées de jointure associés à leur texte. Ils utilisent un modèle de diffusion basé sur l'architecture d'un Transformers. Nous retrouvons également Dabral et al. (2023a) qui génèrent avec le même type d'ensemble de données des mouvements humain à partir de texte ou de musique. Ils utilisent un modèle de diffusion basé sur une architecture UNet en intégrant la condition de la musique ou du texte par des couches d'attention croisée. Ces 2 travaux cités précédemment se basent sur l'ensemble de données HumanML3D qui est un ensemble de données de langage de mouvement humain 3D. Il couvre un large éventail d'actions humaines telles que les activités quotidiennes (par exemple, « marcher », « sauter »), les sports (par exemple, « nager », "jouer au golf'), les acrobaties (par exemple, 'faire la roue') et l'art (par exemple, « dansant »). Nous citons également le travail de Alexanderson et al. (2022) qui gênèrent aussi des mouvements de corps spécifique à la danse en fonction de l'audio. Ils utilisent des conformers à la place de convolutions dilatées dans leurs modèles de diffusion très complexes et utilisent plusieurs datasets (TSG, Dance, ZeroEggs). Zeng et al. (2023) utilisent les modèles de diffusion pour générer des avatars humains 3D en utilisant du texte. Karunratanakul et al. (2023) font la même tache précédente en rajoutant des contraintes spatiales afin de rendre les mouvements générés plus cohérents. Castillo et al. (2023) utilisent l'ensemble de données AMASS pour générer des séquences de mouvements à l'aide de modèles de diffusion. AMASS est un ensemble de données contenues dans le dataset HumanML3D. Il y aussi le travail de Choi et al. (2022) qui effectuent de l'estimation de pose humaine 3D monoculaire à partir d'image en 2D. Nous finissons par citer le travail de Yuan et al. (2022) qui intègre des contraintes physiques dans le processus de diffusion. Ils proposent

un module de projection de mouvement basé sur la physique qui utilise l'imitation de mouvement dans un simulateur physique pour projeter le mouvement débruité d'une étape de diffusion en un mouvement physiquement plausible.

3 Modèles de diffusion

Comme nous utilisons des modèles de diffusions, cette section se fonde sur les explications de Ho et al. (2020) qui proposent des hypothèses simplifiées que nous allons voir ci-dessous et que nous utiliserons également.

3.1 Définition

Soit q une loi de probabilité issue de notre ensemble d'entraînement telle que $x_0 \sim q(x_0)$,
On fixe la distribution conditionnelle $q(x_{1:T}|x_0)$ à une gausienne. Pour cela, du bruit est ajouté progressivement et chaque état est seulement dépendant de l'état précédent. Tout cela est effectué de la manière suivante:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad (1)$$

$$q(x_{1:T}) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \quad (2)$$

Pour un T assez grand et des bonnes valeurs prises par les $\beta_t \in]0,1[$, x_T approche une distribution gausienne centrée réduite. Ainsi, si on connaît la distribution inverse exacte $q(x_{t-1}|x_t)$, on peut générer $x_T \sim \mathcal{N}(0, I)$ et effectuer le processus inverse pour obtenir un nouvel échantillon selon $q(x_0)$. Mais étant donné que $q(x_{t-1}|x_t)$ dépend totalement de l'ensemble de données et qu'elle n'est pas calculable, Ho et al. (2020) utilisent un principe variationnel et apprennent un VAE pour apprendre les paramètres de la distribution conditionnelle (gaussienne) comme suit :

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad (3)$$

Ainsi, le VAE apprend l'espérance $\mu_\theta(x_t, t)$ et la variance $\Sigma_\theta(x_t, t)$. Il est important de souligner la dépendance de ces deux paramètres par rapport au niveau de bruit t et aux données bruitées courantes pour ce même niveau de bruit.
Par conséquent, on peut écrire la borne inférieure variationelle L_{vlb} comme suivante :

$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T)) \quad (4)$$

L_T est un terme qui rentre dans la borne inférieure variationnelle, mais qui ne dépend pas de l'entraînement. En effet, il s'agit d'une KL-divergence entre $q(x_T|x_0)$, ne dépendant que de la loi q définie précédemment, et $p(x_T)$, ne dépendant pas de θ. Comme $q(x_T|x_0) \approx \mathcal{N}(0, I)$, $p(x_T)$ sera également échantillonnée sur cette même loi. Ainsi $L_T \approx 0$.

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t)) \quad (5)$$

L_{T-1} est aussi un terme qui rentre dans la borne inférieure variationnelle. Il s'agit d'une KL-divergence entre notre loi $p_\theta(x_{t-1}|x_t)$ à approximer et la loi $q(x_{t-1}|x_t, x_0)$ qui est une loi gausienne calculable explicitement.

$$L_{vlb} = \sum_{t=0}^{T} L_t \quad (6)$$

L_{vlb} est notre borne inférieure variationelle. Elle est simplement définie comme la somme des L_t pour t allant de 0 à T.

$$L_0 = -\log p_\theta(x_0|x_1) \quad (7)$$

Pour le terme L_0, Ho et al. (2020) utilisent une vraisemblance seulement adaptée à un type de données (des images prenant des valeurs entre 0 et 255). Vu que nous n'utilisons pas ce type de données, nous proposons une modification discutée plus tard (Voir (4.3)).

Ho et al. (2020) ajoute que grâce au processus de bruitage gaussien défini par l'équation (1), on peut directement échantillonner x_t pour $t \in \mathbb{N}^\star$ en fonction de x_0 sans faire les étapes de bruitages intermédiaires de la manière suivante :

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I) \quad (8)$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad (9)$$

où $\epsilon \sim \mathcal{N}(0, I)$, $\alpha_t = 1 - \beta_t$ et $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.
Il n'est pas possible de calculer $q(x_{t-1}|x_t)$ mais si on rajoute la condition sur x_0, on trouve :

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_t(\tilde{x_t}, x_0), \tilde{\beta}_t I) \quad (10)$$

$$\mu_t(\tilde{x_t}, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha_{t-1}})x_t + \beta_t\sqrt{\bar{\alpha_{t-1}}}x_0}{1-\bar{\alpha}_t} \quad et \quad (11)$$

$$\tilde{\beta}_t = \frac{(1-\bar{\alpha_{t-1}})\beta_t}{1-\bar{\alpha}_t} \quad (12)$$

où les $\beta_1, \cdots, \beta_T$ sont des valeurs dans $]0, 1[$ qui croient selon une fonction mise en place. Les variables $x_1, \cdots, x_T$ sont de mêmes dimensions que x_0, $\alpha_t = 1 - \beta_t$ et $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. La fonction mise en place sur les β_t est croissante de sorte que β_t croit de 0 jusqu'à 1 et la fonction dépend aussi du nombre de niveaux de bruit.
Le fonctionnement des processus de diffusion étant expliqué, nous allons voir comment ces derniers sont entraînés.

3.2 Entraînement

La fonction objective (Équation (6)) à minimiser est une somme de termes (5). Ces derniers sont tous indépendants. De plus, l'équation (9) permet de générer un échantillon x_t bruité pour un pas arbitraire $t \in \mathbb{N}^\star$ lors de la phase en avant. Ainsi, en utilisant la loi a posteriori (Équation (10)) et la loi a priori (Équation (3)), on peut estimer L_{t-1}.
Toutefois, il faut estimer la fonction objective L_{vlb}. Pour cela, on utilise l'espérance $\mathbb{E}_{t,x_0,\epsilon}[L_{t-1}]$ pour estimer cette dernière. Comme (Ho et al., 2020), pour chaque séquence dans chaque petit lot de séquences, on échantillonne t selon une loi uniforme discrète sur $\in \{1, 2, \cdots, T\}$. $\mu_\theta(x_t, t)$ sera prédit directement par notre réseau de neurones. Nous aurions pu aussi demander à notre réseau de prédire x_0 qui peut être utilisé dans l'équation (11). En utilisant l'équation (9) et l'équation (11), on a

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \tag{13}$$

. On remarque alors que notre réseau ne nous prédira plus $\mu_\theta(x_t, t)$ mais le bruit $\epsilon_\theta(x_t, t)$. Ce choix est alimenté par l'argument que (Ho et al., 2020) trouvent que la prédiction du bruit marche le mieux, surtout quand cette prédiction est associée à une fonction de perte repondérée :

$$L_{sim} = E_{t,x_0,\epsilon} \left[||\epsilon - \epsilon_\theta(x_t, t)||^2 \right] \tag{14}$$

Cette fonction de perte correspond à une forme repondérée de L_{vlb} sans prendre en compte les termes dépendant de $\Sigma_\theta(x_t, t)$. De plus, les auteurs trouvent qu'optimiser L_sim donne de meilleurs échantillons qu'en optimisant L_{vlb} directement. Bien qu'en utilisant L_{sim} comme fonction de perte, rien n'est appris sur $\Sigma_\theta(x_t, t)$ mais (Ho et al., 2020) obtiennent de meilleurs résultats en fixant la variance à la valeur $\sigma_t{}^2 I$ où $\sigma_t{}^2 = \beta_t$ ou $\tilde{\beta}_t$. Dans le cadre de notre travail, nous allons aussi tenter l'approche d'apprendre la variance en plus de la fixer. On se fondera pour cela sur le travail de (Nichol and Dhariwal, 2021) (Nichol and Dhariwal, 2021) utilise simplement la fonction de perte suivante :

$$L_{alternative} = L_{simple} + \lambda L_{vlb} \tag{15}$$

où $\lambda = 0.001$ pour ne pas que L_{vlb} prenne le dessus sur L_{simple}. De plus (Nichol and Dhariwal, 2021) proposent de ne pas propager le gradient venant de la prédiction de $\mu_\theta(x_t, t)$ pour L_{vlb}. λ peut être un hyperparamètre, mais dans le cadre de notre travail, nous l'avons fixé comme (Nichol and Dhariwal, 2021).

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta L_{sim}$ or $\nabla_\theta L_{alternative}$
6: **until** converged

Figure 2: Algorithme d'entraînement

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_\theta(x_t, t) z$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 3: Algorithme de génération

Même si l'entraînement d'un processus de diffusion est plus stable que celui d'un GAN ou un CGAN, l'entraînement est plus long, car au vu de l'algorithme d'entraînement, le réseau doit apprendre à comprendre chaque niveau de bruit surtout si on fixe une valeur de T élevée. La génération de nouvelles données est longue lorsque un processus de diffusion est utilisé par rapport à un GAN ou à un CGAN, car il faut remonter tous les niveaux de bruits de T jusqu'à 0.

4 Conditionnement audio des modèles de diffusion

Suite aux recherches menées durant l'état de l'art sur des tâches similaire à la nôtre et sur les modèles de diffusions, nous proposons de travailler avec 2 architectures de réseaux de neurones afin d'étudier 2 types d'intégration de la condition audio dans notre tache de génération. Bien évidemment, nous les comparerons expérimentalement par la suite.

4.1 1er modèle : condition audio avec attention

Ici, nous nous inspirons d'un type de modèles de diffusion fondé sur l'article (Dabral et al., 2023b). Une implémentation du modèle n'étant pas disponible, nous avons construit le modèle selon l'explication donnée par cet article. Comme le réseau intègre les niveaux de bruit issu de la phase de bruitage (les t), il est important de trouver un bon encodage de ces derniers afin que le réseau comprenne à quel pas de temps le bruit fonctionne. Nous utiliserons un encodage de position sinusoïdale inspiré de (Vaswani et al., 2017).

Fonctionnement du réseau :
Les données bruitées passent par une couche linéaire puis passent par une couche convolutive 1 dimension puis par un double bloc ResNet (Zagoruyko and Komodakis, 2016), les niveaux de bruits sont encodés puis ajoutés par somme à la sortie du double bloc ResNet précédent. Nous avons ensuite 3 étapes de sous-échantillonnage. L'étape est constituée d'un double bloc ResNet (Zagoruyko and Komodakis, 2016) suivie d'un sous-échantillonnage de facteur 2 et d'un bloc d'attention croisée qui reprend celui de (Vaswani et al., 2017). Après ces 3 dernières, nous appliquons juste un double bloc ResNet. Nous appliquons par la suite 3 étapes de sur-échantillonnage dont chaque étape est la même que les étapes de sous-échantillonnage à la seule différence que nous effectuons un sur-échantillonnage de facteur 2 avant le double bloc résiduel à la place du sous-échantillonnage. Nous notons que dans chacune de ces étapes, le double bloc résiduel est appliqué sur la concaténation entre la sortie de l'opération de sur-échantillonnage et la sortie d'une étape de sous-échantillonnage au début. Nous finissons par un double bloc résiduel ainsi qu'une couche convolutive une dimension.

4.2 2eme modèle : condition audio en concaténant au début

Ici, nous optons également pour une architecture U-Net (Ronneberger et al., 2015). Bien que ce réseau à la base est utilisé pour de la segmentation d'image médicale, nous allons l'adapter en passant en dimension 1 plutôt que 2 à travers les couches convolutives.
Comme le réseau intègre les niveaux de bruit issu de la phase de bruitage (les t), il est important de trouver un bon encodage de ces derniers afin que le réseau comprenne à quel pas de temps le bruit fonctionne. Nous utiliserons également un encodage de position sinusoïdale inspiré de (Vaswani et al., 2017).

Fonctionnement du réseau : :
Les données bruitées et les caractéristiques audio sont concaténées puis passe par une couche convolutive, et les niveaux de bruit sont encodés. Le réseau est suivi d'un nombre d'étape de sous-échantillonnage. L'étape est constituée à la suite de 2 bloc ResNet (Zagoruyko and Komodakis, 2016), une normalisation de groupe, un bloc d'attention, une connexion résiduelle et l'opération de sous-échantillonnage. Le réseau passe ensuite par 2 bloc ResetNet (Zagoruyko and Komodakis, 2016) dont une couche d'attention entre les 2. Ensuite, on a un nombre d'étapes de sur-échantillonnage dont chaque étape est similaire aux étapes de sous-échantillonnage sauf qu'on a une opération de sur-échantillonnage cette fois-ci. Le réseau termine par un bloc ResNet (Zagoruyko and Komodakis, 2016) ainsi qu'une couche convolutive.

4.3 Entraînement et modification par rapport à la définition

Les 2 modèles présentés précédemment seront entraînés d'une part avec la fonction de perte L_{sim} (14) et d'autre part par la fonction de perte $L_{alternative}$ (15).
Le terme L_0 (7) sera remplacé par une simple MSE entre la prédiction de x_0 à partir de x_1 et la vraie valeur de x_0.
De plus, on utilise un procédé appelé Exponential Moving Average. Il renforce la stabilité de la convergence d'un schéma et l'assiste à parvenir à une solution globale supérieure en empêchant la convergence vers un minimum local. Afin d'éviter des modifications abruptes dans les

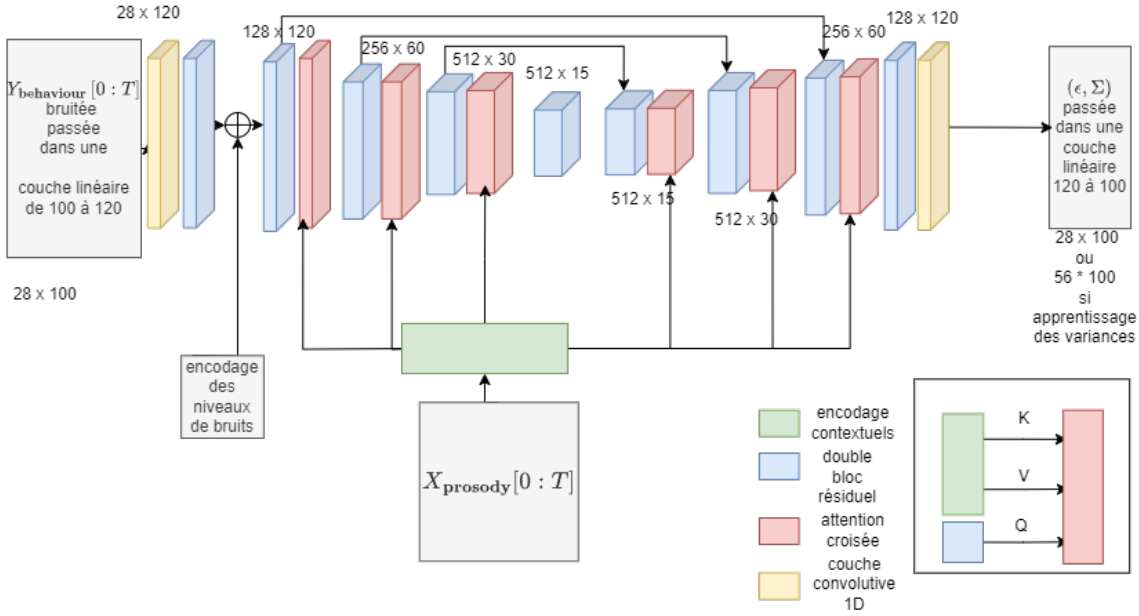

Figure 4: Schéma de notre premier modèle. Nous faisons remarquer que les encodages contextuels sont calculés seulement en utilisant une couche linéaire et qu'à chaque bloc d'attention croisée, les encodages contextuels sont différents car les dimensions diffèrent.

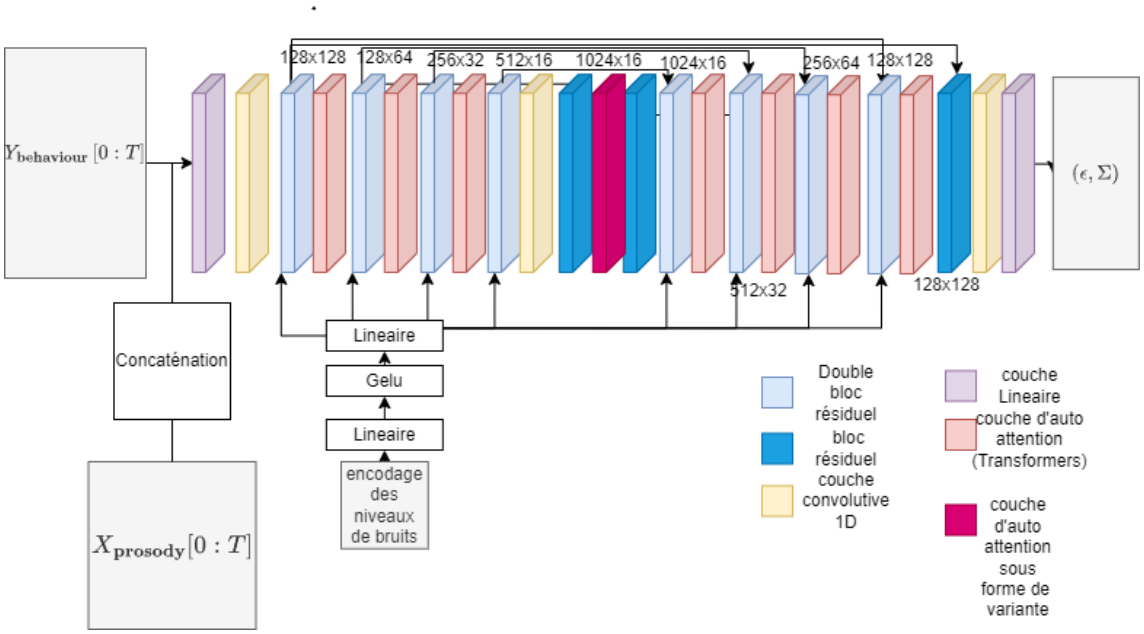

Figure 5: Schéma de notre deuxième modèle. Les dimensions des 2 premières couches et des 2 dernières ne sont pas indiquées car elle dépendent de la taille des caractéristiques audio choisies

pondérations du schéma durant la formation, une duplication des pondérations actuelles est réalisée avant la mise à jour des pondérations du schéma. Ultérieurement, les pondérations du schéma sont mises à jour pour être la moyenne pondérée des pondérations actuelles et celles de l'étape post-optimisation.

$$EMA_t = \begin{cases} \theta_0 \text{ si } t = 0 \\ \alpha\theta_t + (1-\alpha)EMA_{t-1} \text{ sinon} \end{cases} \quad (16)$$

où $\alpha \in]0, 1[$ est le taux

5 Expériences

Dans cette partie, nous allons présenter le dataset utilisé ainsi que les expériences réalisées. Nous commencerons par détailler comment nos données ont été obtenues puis nous étudierons 3 types de comparaisons. (apprentissage de variance ou pas, type d'intégration de condition et caractéristiques audio choisies)

Données :
Nous avons à notre disposition un ensemble d'enregistrement de vidéos audio. Initialement, obtenir un ensemble de données en bonne quantité et de bonne qualité dans la tache qui nous concerne est un processus coûteux et demandant du temps. Malheureusement, l'acquisition d'un bon ensemble de données est une étape indispensable et responsable de bonnes performances de nos futurs apprentissages. Pour rendre nos données interprétables par nos modèles, nous utilisons 2 boites à outil. La première boite à outil nous permettra d'extraire les caractéristiques prosodiques et acoustiques, il s'agira de OpenSmile (Eyben et al., 2010). La deuxième boite à outil aura pour objectif de nous extraire les caractéristiques visuelles. Il s'agira du logiciel OpenFace (Baltrusaitis et al., 2016).

OpenSmile est une boite à outil prenant en entrée un signal audio et nous donne les caractéristiques présentées ci-dessus sous forme de séquences temporelles selon une fréquence. Dans notre cas, on extrait à 50fps. Elles sont au nombre de 25. Nous expliquerons le détail de ces 25 caractéristiques plus tard et nous choisirons quelles sont les caractéristiques à choisir avec ou non les dérivées premières et secondes de ces dernières.

OpenFace est une boite à outil prenant en entrée une vidéo et nous donne les caractéristiques visuelles présentées dans la définition du problème. (Voir 1). Nous avons donc au total 28 caractéristiques visuelles extraites. Ces dernières sont aussi extraites sous forme de séquences temporelles selon une fréquence qui sera de 30 fps dans notre cas. Ces caractéristiques seront à générer par notre processus de génération de comportements.

Ces caractéristiques, générées par notre processus de génération, seront finalement fournies à un logiciel d'animation afin de rendre compte du mouvement construit. Le logiciel Greta (Niewiadomski et al., 2009) est alors choisi, car il permet d'animer les comportements de personnages visuels selon des caractéristiques visuelles qui sont exactement celles que nous générerons. Il lit un fichier csv contenant les séquences de ces caractéristiques visuelles. Ce logiciel est alors très adapté à la tâche qui nous incombe.

Bien que nous sommes en possession de caractéristiques liées aux audios et aux vidéos, il est nécessaire d'aligner les caractéristiques audio et vidéos afin de rendre compte de la correspondance entre ces 2 derniers. En effet, au vu des extractions expliquées précédemment, les segments audio ou vidéo extraite ne correspondent pas forcément au vu de la fréquence d'extraction choisi et au fonctionnement de nos 2 boîtes à outils. Un échantillonnage est alors effectué sur la fréquence 25fps afin d'obtenir un alignement correct. On possède alors un alignement audio-vidéo à la fréquence 25fps.

Ces étapes d'extractions sont effectuées sur le corpus TRUENESS (Ochs et al., 2023). Il s'agit d'un corpus composé de 36 vidéos enregistrées dans le Technosport d'Aix-Marseille Université à Luminy pour un total de 2 h 01 d'enregistrement. Dans chaque vidéo, les vidéos contiennent des échanges recueillis comprenant des représentations de situations courantes de discrimination sexiste et raciste interprétées par divers acteurs dans diverses circonstances. Chaque vidéo contient les échanges entre 2 interlocuteurs, mais se focalise sur le visage d'un des 2 acteurs. Nous divisons ces données en ensemble d'entraînement et de tests, de durée respective 84 min et 7 sec, 36 min et 53 sec. Pour entraîner le modèle, les vidéos sont divisées en segments de 100 séquences (4 secondes environ). Nous obtenons 3 016 segments pour le jeu d'entraînement et 1397 segments pour le jeu de test. Avant d'effectuer l'alignement cité précédemment, nous avons effectué un prétraitement des données vidéos avec une suppression de valeurs aberrantes, une meilleure estimation de valeurs extraites avec peu de confiance par OpenFace et un centrage de ces caractéristiques vidéos. Pour les caractéristiques vidéos, on ajoute les dérivées, nous alignons en fonction de notre fréquence de 25 fps.

Maintenant que les données ont été clairement expliquées, nous allons comparer nos expériences sur 3 problématiques : Nous commencerons par comparer le type d'intégration de la condition audio, suivie de l'intérêt d'apprendre la variance et enfin les caractéristiques audio choisies. Avant d'énoncer nos résultats, il est important de rappeler nos méthodes d'évaluation ainsi que les détails de nos entraînements.

Normalisation des données :
Avant d'entraîner nos modèles de diffusion, il faut passer par une phase de pré-traitement des données. Les données sont normalisées entre -1 et 1. Nous utilisons pour ça le MinMaxScaler de scikit-learn. Cette normalisation est justifiée par la phase inverse du modèle de diffusion. En effet, Ho et al. (2020) disent que cette normalisation garantit que le processus inverse du réseau neuronal fonctionne sur des entrées mises à l'échelle de manière cohérente à partir de $p(x_T) = \mathcal{N}(x_T; 0, I)$.

Comme dit précédemment, notre travail et donc nos expériences vont consister à voir quelles sont les meilleures caractéristiques permettant de générer les comportements les plus naturels et crédibles. Nous regarderons aussi quel est la meilleure manière d'insérer ces dernières dans nos modèles. Dans le cadre de ce travail, nous choisissons d'étudier 4 ensembles de caractéristiques audio. Pour chacun d'entre eux, nous entraînons nos 2 modèles sur ce dernier à la fois avec apprentissage et non-apprentissage de la variance comme indiqué précédemment. Nous avons alors au total 16 expériences à notre disposition.

Détails de l'entraînement :
Tous nos modèles sont entraînés avec 2500 epochs. Afin de fixer les meilleurs ensembles d'hyperparamètres pour chacune de nos expériences, nous entraînons un nombre conséquent de modèles en faisant varier le taux d'apprentissage, la taille du noyau de nos couches convolutives, la taille de nos lots, le nombre maximum de bruitage gaussien lors de la phase avant de notre modèle de diffusion. Nous utilisons une optimisation Adam et afin d'éviter toute explosion de gradient, on utilise un planificateur de taux d'apprentissage linéaire. Vu que nous utilisons l'Exponential Moving Average, nous entraînons nos modèles avec plusieurs taux afin de déterminer le meilleur. Par manque de métriques d'évaluations subjectives à travers des évaluations d'utilisateurs, nous nous contentons de métriques objectives. Nous rajoutons une métrique appelée FVD (Frechet Video Distance) pour pouvoir compenser le fait que l'ACP ne nous permet pas précisément de différencier les comportements générés par nos modèles.

Métriques d'évaluations objectives
Afin d'entraîner un modèle, il est important de définir des métriques objectives pour avoir un aperçu des performances du modèle. Nous avons à notre disposition les métriques objectives suivantes.

- **La fonction de perte :** Cette métrique objective ne peut pas être ignorée. En effet, elle est le seul moyen de pouvoir entraîner un modèle par le biais d'une optimisation de cette dernière fonction. Elle permet aussi d'observer s'il y a ou pas du sur-apprentissage. Même s'il s'agit d'une métrique universelle dans le cadre de l'apprentissage automatique, elle reste peu fiable dans la tâche qui nous incombe. Bien qu'utilisant des GAN ou des CGAN, Wu et al. (2021) nous expliquent qu'une fonction de perte suffisamment proche de 0 n'est pas une raison suffisante pour sélectionner un modèle. En effet, cette technique de mesure a également la propension à négliger les infimes différences de comportements qui, pourtant, altèrent complètement la perception d'un être humain.

- **L'analyse en composantes principales (ACP) :** Elle favorise la diminution des dimensions visuelles de nos échantillons, ensuite elle projette sur un repère bidimensionnel diverses séquences issues de notre ensemble de tests initial et de notre ensemble de tests généré. Nous pouvons ainsi étudier la distribution de ces deux ensembles dans l'espace 2D. Une distribution optimale ne garantit pas des résultats fiables, mais une distribution inadéquate est sûre de produire des résultats erronés.

- **Frechet Video Distance :** Il s'agit d'une métrique récente pour les modèles génératifs de vidéo basés sur FID (Frechet Inception Distance) . Dans notre cas, on utilisera cette métrique sur les vidéos générées par

0 = Speak	1 = Loudness_sma3
2 = alphaRatio_sma3	3 = hammarbergIndex_sma3
4 = slope0-500_sma3	5 = slope500-1500_sma3
6 = spectralFlux_sma3	7 = mfcc1_sma3
8 = mfcc2_sma3	9 = mfcc3_sma3
10 = mfcc4_sma3	11 = F0semitoneFrom27.5Hz_sma3nz
12 = jitterLocal_sma3nz	13 = shimmerLocaldB_sma3nz
14 = HNRdBACF_sma3nz	15 = logRelF0-H1-H2_sma3nz
16 = logRelF0-H1-A3_sma3nz	17 = F1frequency_sma3nz
18 = F1bandwidth_sma3nz	19 = F1amplitudeLogRelF0_sma3nz
20 = F2frequency_sma3nz	21 = F2bandwidth_sma3nz
22 = F2amplitudeLogRelF0_sma3nz	23 = F3frequency_sma3nz
24 = F3bandwidth_sma3nz	25 = F3amplitudeLogRelF0_sma3nz

Table 1: Les différentes caractéristiques audio de OpenSmile sauf Speak qui est rajouté lors du prétraitement qui vaut 0 ou 1 en fonction de si la personne filmée parle ou pas

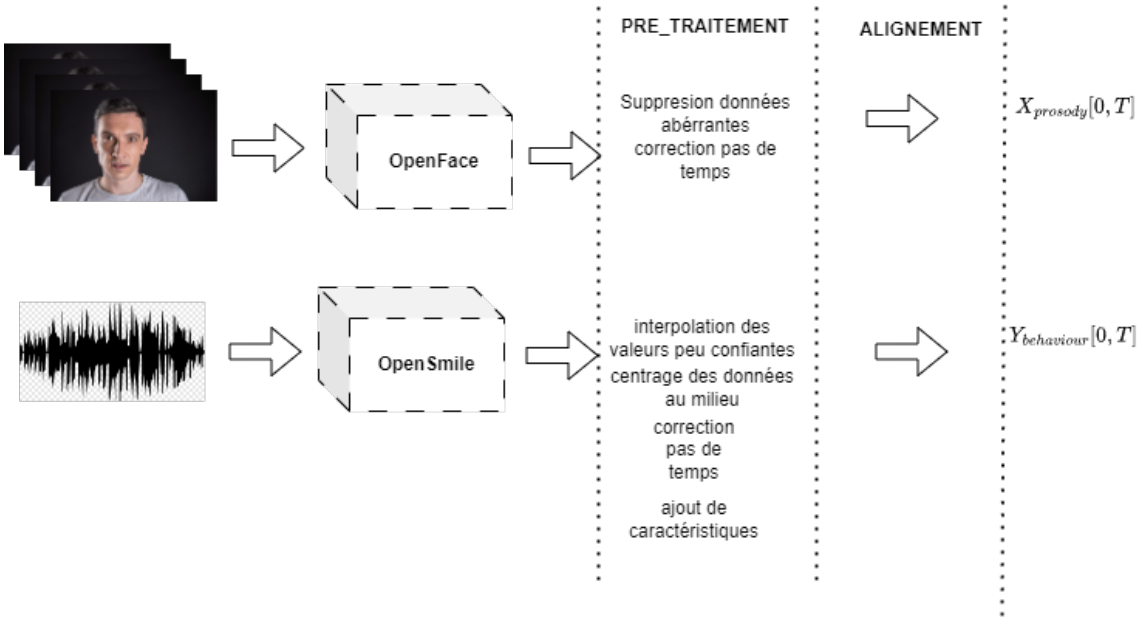

Figure 6: Schéma qui explique comment obtenir nos données

Greta sur l'ensemble de tests et sur les vidéos que nous générerons grâce à nos modèles. Cette métrique est fondée sur l'article de Unterthiner et al. (2019). Par souci de mémoire, nous nous contentons de calculer la FVD sur les 50 premières frames de nos vidéos générées par rapport aux 50 premières de nos vidéos de tests de référence. Les frames seront dimensionnées puis passées dans la procédure de calcul de cette métrique.

5.1 Étude du conditionnement audio

Dans cette sous-section, nous nous servons de quelques expériences pour comparer les moyens d'intégrer la condition audio. Le modèle 1 correspond à l'intégration par attention (voir (4.1)) tandis que le modèle 2 fait référence à l'intégration par concaténation (voir (4.2)). Nos résultats sont indiqués dans le tableau (2). Nous voyons que malgré de très petits écarts de fonction de perte entre ces 4 expériences choisies, les valeurs de FVD ont de très grands écarts de valeurs. Ces dernières sont nettement plus basses pour l'intégration audio par concaténation avec des caractéristiques audios fixées et un type de choix de variance fixée. Une petite étude statistique sera effectuée à la fin de la section afin de savoir s'il est vraiment judicieux de ne dire qu'une manière d'intégrer l'audio marche mieux que l'autre.

5.2 Impact de l'apprentissage de la variance

Dans cette sous-section, nous nous servons de quelques expériences pour comparer si fixer la variance améliore les comportements générés par nos processus de diffusion. Le modèle 1 fait référence à notre modèle fondée sur l'attention (voir (4.1)) tandis que le modèle 2 fait référence à notre modèle qui intègre la condition audio par concaténation (voir (4.1)). Nos résultats sont indiqués dans le tableau (3). Nous voyons également que malgré de très petits écarts de fonction de perte entre ces 7 expériences choisies pour étudier cette problématique, les valeurs de FVD sont nettement plus basses pour les entraînements apprenant la variance pour des caractéristiques audio fixées et un type de modèle fixé. Une petite étude statistique sera effectuée à la fin de la section afin de savoir s'il est vraiment judicieux de dire que l'apprentissage de la variance améliore grandement la génération de nos comportements.

5.3 Choix des caractéristiques audio

Dans cette sous-section, nous nous servons de nos expériences restantes pour comparer les performances de nos modèles au regard de nos caractéristiques audios choisies. Le modèle 1 fait référence à notre modèle fondée sur l'attention (voir (4.1)) tandis que le modèle 2 fait référence à notre modèle qui intègre la condition audio par concaténation (voir (4.1)). Nos résultats sont indiqués dans le tableau (4). Dans cette problématique et au regard de ces 9 expériences, il est difficile d'avoir une première conclusion sur quelles sont les caractéristiques à utiliser même si certains ensembles de caractéristiques audios choisies donnent de meilleures FVD . Nous pouvons seulement remarquer que l'ajout de la colonne $Speak$ peut rendre meilleurs nos comportements générés. Cela est peut-être lié à l'apprentissage d'un comportement d'humain qui ne parle pas en rajoutant cette dernière caractéristique.

5.4 Étude statistique

Nous séparons en deux distributions les FVD obtenues par nos deux types de modèles sur nos 16 expériences. On effectue un test de Student et nous obtenons une p-value de 0.07 donc le choix de l'intégration est un peu significatif. On effectue pareil en séparons en deux distributions les FVD obtenues sur nos modèles apprenant la variance ou ne l'apprenant pas et on obtient une p-value de 0.3921551879391997 donc le choix de l'apprentissage de la variance n'est pas significatif statistiquement.

6 Conclusions et perspectives

Nous présentons, à notre connaissance, les premiers modèles de diffusion qui génèrent conjointement les mouvements de tête, l'orientation du regard et les unités d'action (AUs) de la méthode FACS automatiquement à partir de la parole. Les modèles de diffusion présentent des résultats prometteurs mais très difficiles à paramétrer. Il reste alors à ce jour une grande quantité de paramètre à tester et d'architecture possible. Il se rajoute également d'autre ensemble de caractéristiques audio à tester potentiellement. Malgré l'absence de métriques d'évaluation subjectives à travers des évaluations utilisateurs, nous nous sommes contentés de métriques objectives comme des fonctions de pertes, d'ACP et surtout de FVD. Bien que les métriques objectives soient

Caractéristiques utilisées	fonction de perte	L_{sim}	L_{vlb}	FVD	variance apprise	modèle
Toutes , dérivées 1ère et 2nd	0.0402	0.0382	0.0021	59.65	Oui	modèle 2
Toutes , dérivées 1ère et 2nd	0.0426	0.0413	0.0013	406.42	Oui	modèle 1
0,2,3,7,8,9,11,15 + dérivées	0.0361	0.0334	0.0027	58.91	Oui	modèle 2
0,2,3,7,8,9,11,15 + dérivées	0.0422	0.0393	0.0029	183.75	Oui	modèle 1

Table 2: Tableau récapitulatif de certaines expériences utilisées pour la comparaison de type d'intégration audio, le modèle 1 fait référence à notre modèle fondée sur l'attention tandis que le modèle 2 fait référence à notre modèle qui intègre la condition audio par concaténation. Les caractéristiques audio utilisées sont indiquées par leur numéro présent dans le tableau des caractéristiques audios plus haut. Chaque résultat présent dans ce tableau est le meilleur résultat obtenu en termes de fonction de perte pour nos combinaisons d'hyperparametres tentées.

Caractéristiques utilisées	fonction de perte	L_{sim}	L_{vlb}	FVD	variance apprise	modèle
Toutes ,dérivées 1ère et 2nd	0.0402	0.0382	0.0021	59.65	Oui	modèle 2
Toutes ,dérivées 1ère et 2nd	0.0320	0.0320		83.44	Non	modèle 2
Toutes ,dérivées 1ère et 2nd	0.0426	0.0413	0.0013	406.42	Oui	modèle 1
Toutes ,dérivées 1ère et 2nd	0.0423	0.0423		454.37	Non	modèle 1
0,2,3,7,8,9,11,15 + dérivées	0.0361	0.0334	0.0027	58.91	Oui	modèle 2
0,2,3,7,8,9,11,15 + dérivées	0.0383	0.0383		98.47	Non	modèle 2

Table 3: Tableau récapitulatif de certaines expériences utilisées pour la comparaison sur l'impact de l'apprentissage de la variance, le modèle 1 fait référence à notre modèle fondée sur l'attention tandis que le modèle 2 fait référence à notre modèle qui intègre la condition audio par concaténation. Les caractéristiques audio utilisées sont indiquées par leur numéro présent dans le tableau des caractéristiques audios plus haut. Chaque résultat présent dans ce tableau est le meilleur résultat obtenu en termes de fonction de perte pour nos combinaisons d'hyperparametres tentées.

Caractéristiques utilisées	fonction de perte	L_{sim}	L_{vlb}	FVD	variance apprise	modèle
0,2,3,7,8,9,11,15 + dérivées	0.0371	0.0371		699.94	Non	modèle 1
7,8,9,10 + dérivées	0.0354	0.0331	0.0023	232.76	Oui	modèle 2
7,8,9,10 + dérivées	0.0328	0.0328		452.15	Non	modèle 2
7,8,9,10 + dérivées	0.0440	0.0426	0.0014	250.05	Oui	modèle 1
7,8,9,10 + dérivées	0.0340	0.0340		252.68	Non	modèle 1
0,7,8,9,10 + dérivées	0.0343	0.0335	0.0007	374.73	Oui	modèle 2
0,7,8,9,10 + dérivées	0.03	0.03		66.82	Non	modèle 2
0,7,8,9,10 + dérivées	0.0401	0.0385	0.0015	170.90	Oui	modèle 1
0,7,8,9,10 + dérivées	0.0452	0.0452		274.33	Non	modèle 1

Table 4: Tableau récapitulatif de certaines expériences utilisées pour la comparaison des caractéristiques choisies, le modèle 1 fait référence à notre modèle fondée sur l'attention tandis que le modèle 2 fait référence à notre modèle qui intègre la condition audio par concaténation. Les caractéristiques audio utilisées sont indiquées par leur numéro présent dans le tableau des caractéristiques audios plus haut. Chaque résultat présent dans ce tableau est le meilleur résultat obtenu en termes de fonction de perte pour nos combinaisons d'hyperparametres tentées.

peu fiables, nous avons pu remarquer numériquement, statistiquement, mais aussi visuellement à travers nos expériences que l'intégration audio par concaténation génèrent de bien meilleurs comportements que l'intégration par attention malgré des fonctions de pertes très proches pour les 2. L'apprentissage de la variance augmente également la qualité de nos comportements, mais n'est pas significatif selon notre test statistique. De plus, à ce jour, il est impossible de conclure quelles caractéristiques audios sont les plus pertinentes dans la tache qui nous incombe. Il est alors judicieux dans des travaux futurs, de travailler avec d'autres caractéristiques audios ou bien de travailler avec ces mêmes caractéristiques, mais sur des corpus différents car nos données et comportements dépendent énormément du corpus, de notre prétraitement et surtout des boîte à outils utilisées pour extraire nos caractéristiques sachant que ces derniers présentent des imperfections d'extraction.

References

Simon Alexanderson, Rajmund Nagy, Jonas Beskow, and Gustav Henter. 2022. Listen, denoise, action! audio-driven motion synthesis with diffusion models.

Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: An open source facial behavior analysis toolkit. pages 1–10.

Kirsten Bergmann and Stefan Kopp. 2009. Gnetic – using bayesian decision networks for iconic gesture generation. volume 5773, pages 76–89.

Justine Cassell, Catherine Pelachaud, Norman Badler, Mark Steedman, Brett Achorn, Tripp Becket, Brett Douville, Scott Prevost, and Matthew Stone. 1998. Animated conversation: Rule-based generation of facial expression, gesture spoken intonation for multiple conversational agents. *Technical Reports (CIS)*.

Angela Castillo, Maria Escobar, Guillaume Jeanneret, Albert Pumarola, Pablo Arbeláez, Ali Thabet, and Artsiom Sanakoyeu. 2023. Bodiffusion: Diffusing sparse observations for full-body human motion synthesis.

Chung-Cheng Chiu and Stacy Marsella. 2011. How to train your avatar: A data driven approach to gesture generation. pages 127–140.

Jeongjun Choi, Dongseok Shim, and H. Jin Kim. 2022. Diffupose: Monocular 3d human pose estimation via denoising diffusion probabilistic model.

Rishabh Dabral, Muhammad Hamza Mughal, Vladislav Golyanik, and Christian Theobalt. 2023a. Mofusion: A framework for denoising-diffusion-based motion synthesis.

Rishabh Dabral, Muhammad Hamza Mughal, Vladislav Golyanik, and Christian Theobalt. 2023b. Mofusion: A framework for denoising-diffusion-based motion synthesis. In *Computer Vision and Pattern Recognition (CVPR)*.

P. Ekman and W.V. Friesen. 1978. *Facial action coding system: A technique for the measurement of facial movement*. Consulting Psychologists Press, Palo Alto, CA.

Florian Eyben, Martin Wöllmer, and Björn Schuller. 2010. opensmile – the munich versatile and fast open-source audio feature extractor. pages 1459–1462.

Ylva Ferstl, Michael Neff, and Rachel McDonnell. 2020. Adversarial gesture generation with realistic gesture phasing. *Computers Graphics*, 89.

Shiry Ginosar, Amir Bar, Gefen Kohavi, Caroline Chan, Andrew Owens, and Jitendra Malik. 2019. Learning individual styles of conversational gesture. *CoRR*, abs/1906.04160.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Dai Hasegawa, Naoshi KANEKO, Shinichi Shirakawa, Hiroshi Sakuta, and Kazuhiko Sumi. 2018. Evaluation of speech-to-gesture generation using bi-directional lstm network.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. 2023. Gmd: Controllable human motion synthesis via guided diffusion models.

Stefan Kopp and Ipke Wachsmuth. 2002. Model-based animation of co-verbal gesture. pages 252 – 257.

Taras Kucherenko. 2018. Data driven non-verbal behavior generation for humanoid robots. pages 520–523.

Taras Kucherenko, Dai Hasegawa, Gustav Henter, Naoshi Kaneko, and Hedvig Kjellström. 2019. Analyzing input and output representations for speech-driven gesture generation. pages 97–104.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Alex Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. *CoRR*, abs/2102.09672.

Radoslaw Niewiadomski, Elisabetta Bevacqua, Maurizio Mancini, and Catherine Pelachaud. 2009. Greta: An interactive expressive eca system. volume 2, pages 1399–1400.

Magalie Ochs, Jean-Marie Pergandi, Alain Ghio, Carine André, Patrick Sainton, Emmanuel Ayad, Auriane Boudin, and Roxane Bertrand. 2023. A forum theater corpus for discrimination awareness. *Frontiers in Computer Science*, 5.

Manuel Rebol, Christian Gütl, and Krzysztof Pietroszek. 2021. Passing a non-verbal turing test: Evaluating gesture animations generated from speech. *CoRR*, abs/2107.00712.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Najmeh Sadoughi and Carlos Busso. 2018. Novel realizations of speech-driven head movements with generative adversarial networks. pages 6169–6173.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. 2022. Human motion diffusion model.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. 2019. Fvd: A new metric for video generation. In *DGS@ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Bowen Wu, Chaoran Liu, Carlos Ishi, and Hiroshi Ishiguro. 2021. Modeling the conditional distribution of co-speech upper body gesture jointly using conditional-gan and unrolled-gan. *Electronics*, 10:228.

Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. 2022. Physdiff: Physics-guided human motion diffusion model.

Sergey Zagoruyko and Nikos Komodakis. 2016. Wide residual networks. *CoRR*, abs/1605.07146.

Yifei Zeng, Yuanxun Lu, Xinya Ji, Yao Yao, Hao Zhu, and Xun Cao. 2023. Avatarbooth: High-quality and customizable 3d human avatar generation.

Annexe A. Les unités d'action faciales

AU	Description
AU01	Élévation des sourcils intérieurs
AU02	Élévation des sourcils externes
AU04	Abaissement des sourcils
AU05	Élévation des paupières supérieures
AU06	Rehaussement des joues
AU07	Serrage de la paupière
AU09	Rétrécissement du nez
AU10	Élévation de la lèvre supérieure
AU12	Étirement du coin des lèvres
AU14	Fossettes
AU15	Abaissement du coin des lèvres
AU17	Élévation du menton
AU20	Étirement des lèvres
AU23	Resserrement des lèvres
AU25	Écartement des lèvres
AU26	Écartement de la mâchoire
AU45	Clignement des yeux

Table 5: Description des unités d'action faciales (AUs) extraites par OpenFace, prédites par nos modèles et animées par Greta. OpenFace peut détecter l'intensité (de 0 à 5) de 17 AUs.

Annexe B. Exemple de PCA entre 2 modèles

Nous nous apercevons qu'il est en effet difficile de se rendre compte si les comportements générées sont naturels, nous pouvons seulement voir si la distribution est crédible. Mais il est surtout compliqué de comparer des modèles comme on peut voir sur la figure (7) et (8).

Annexe C. Répertoire vidéos utilisées

Pour calculer les FVD durant mes expériences, nous avons du générer des vidéos pas très longue de quelques secondes par souci de mémoire. Le lien du dossier les contenant toutes est ci-dessous : https://drive.google.com/drive/folders/19mbxjhQa2uesuedbhSzEcoGU3aDgsRvO?usp=sharing

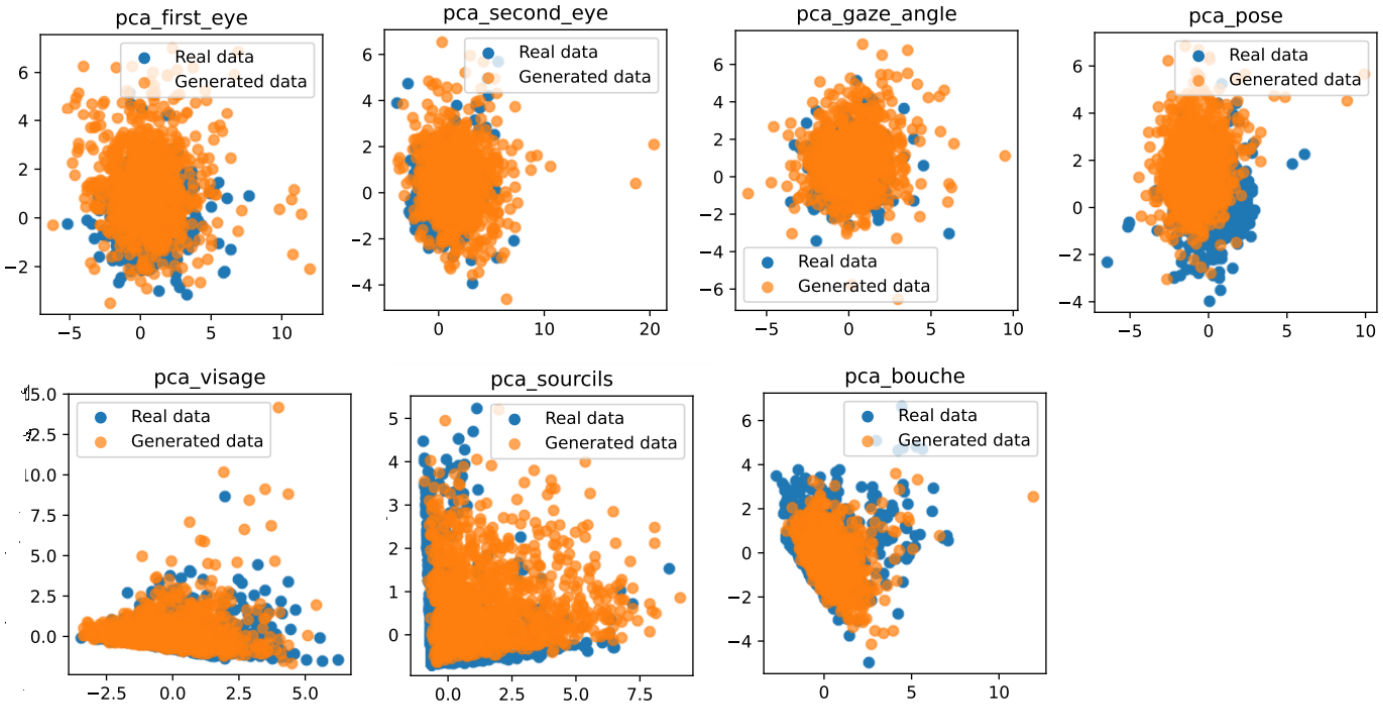

Figure 7: PCA sur l'expérience où on a utilisé toute les caractéristiques audios avec variance apprise et modèle basé sur la concaténation

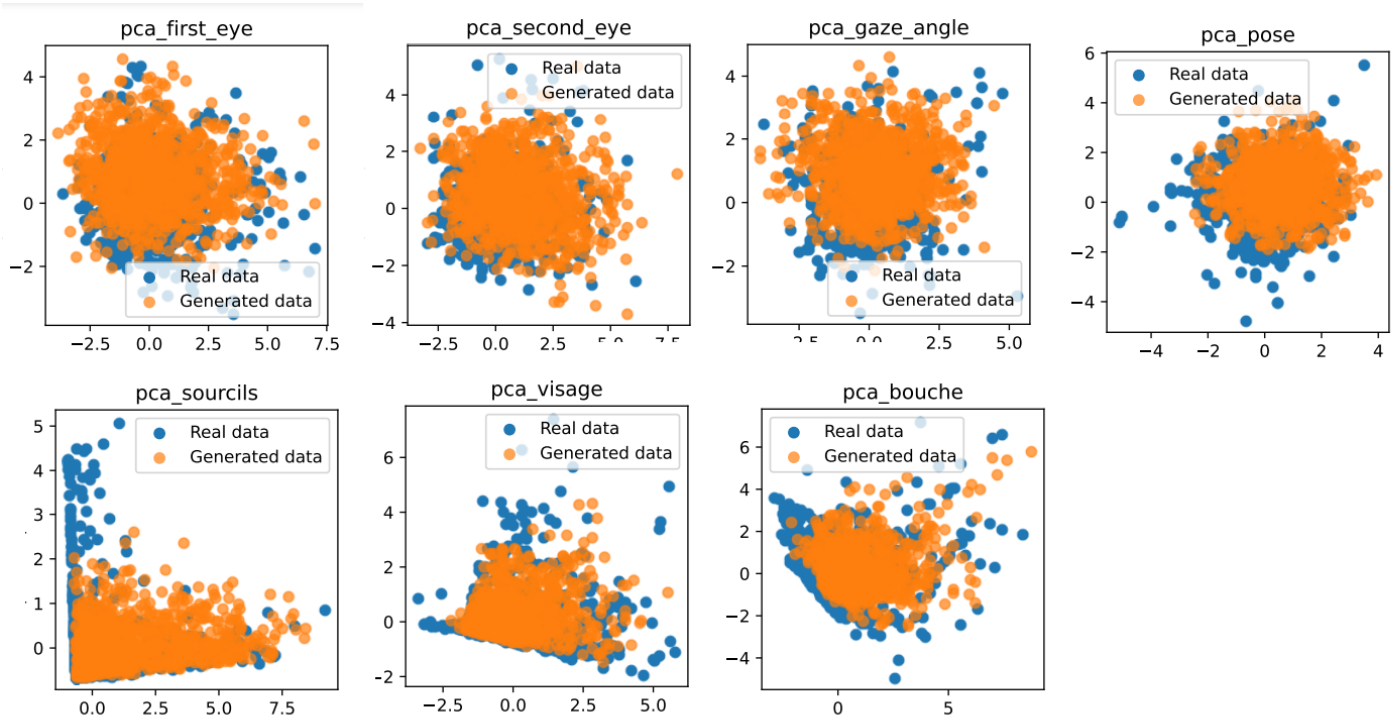

Figure 8: PCA sur l'expérience où on a utilisé toute les caractéristiques audios avec variance non apprise et modèle basé sur la concaténation