Latent-Playground – Documentation

v1.1.1

Introduction

Latent-Playground est un effet audio Max for Live qui vous permet d'explorer la transformation et la génération sonore via la manipulation de modèles de deep learning. En insérant cet effet sur une piste dans Ableton Live, vous pouvez encoder l'audio entrant dans un ensemble de variables internes, puis le décoder en un audio sortant transformé, tout en modulant ces variables en temps réel. Le résultat est un effet de morphing sonore original, contrôlé par des modulateurs intégrés qui génèrent des variations aléatoires et synchronisées avec le transport d'Ableton. Ce device vous permet de visualiser, modifier, et contrôler directement les paramètres internes des réseaux de neurones pour détourner leur signature sonore.

Cette documentation détaille le fonctionnement de Latent-Playground. Vous y trouverez une description de chaque module interne, une explication claire des concepts fondamentaux sous-jacents, ainsi que des instructions d'installation et de dépannage. Deux schémas explicatifs sont également inclus pour illustrer l'architecture globale du système et le fonctionnement des modulateurs. Certains aspects plus techniques sont développés à la fin du document. Un détail de la fonction de chaque paramètre est disponible dans la fenêtre aide de Live lors du survol de ceux-ci.

Table des matières

Installation

1. **Téléchargement** : Récupérez la release la plus récente du repo GitHub ici. Vous devez télécharger le code source et l'objet *mcs.nn~* en choisissant l'architecture correspondante. Assurez-vous de disposer d'une version compatible d'Ableton Live (Live 11 ou supérieure).

2. **Gestion des externals** : ouvrez un effet audio Max for Live vide dans Ableton et lancer le mode edit. Dans Max, ajouter le dossier *externals* de Latent-Playground aux *File Preferences* (Option -> File Preferences -> +).

3. **Ouverture** : dans Live, ajouter le dossier Latent-Playground à votre bibliothèque utilisateur. Glissez-déposez le device sur une piste audio pour le charger. Vous devriez voir l'interface de Latent-Playground apparaître.

4. **Obtention du modèle** : téléchargez des modèles entraînés à partir des dépôts comme ceux de l'*Ircam* ou bien de l'*Intelligent-Instruments-Lab*. Vous trouverez aussi des modèles créés par la communauté dans le Discord *ACIDS*.

5. **Chargement du modèle** : sur l'interface de Latent-Playground, cliquez sur le bouton Import-Folder qui permet de sélectionner et charger le dossier contenant les modèles pré-entraînés puis sélectionnez le modèle dans le menu. L'activation des boutons power *encode* et *decode* indique la fin du chargement du modèle. Le traitement audio n'est pas permis pendant la durée du chargement.

6. **Configuration et activation**: assurez-vous que la piste sur laquelle est inséré le device reçoit le signal audio que vous souhaitez traiter et que sa sortie est bien routée. Activez l'encodage et le décodage grâce aux boutons power *encode* et *decode*.

7. **Vérification** : vous devriez entendre le son traité par Latent-Playground. Au premier chargement, si le modèle n'est pas correctement trouvé ou compatible, aucun son ne sort — dans ce cas, reportez-vous à la section de dépannage.

Note : L'effet étant fondé sur un modèle de deep learning, il peut induire une latence de traitement entre 100 et 900ms. Latent-Playground compense automatiquement ce délai en interne via un alignement pour garder le son traité synchronisé avec le reste du projet Live.

Concepts fondamentaux

Deep Learning

Le deep learning est une branche de l'intelligence artificielle qui utilise des réseaux de neurones artificiels à plusieurs couches pour apprendre à partir de données. Dans le contexte audio, ces réseaux peuvent apprendre les caractéristiques sonores et les reproduire.

Auto-encodeur

Un auto-encodeur est un type de réseau neuronal qui apprend à compresser des données dans un espace réduit (espace latent) puis à les reconstruire. La partie qui compresse est appelée "encodeur" et celle qui reconstruit est le "décodeur".

Espace latent

L'espace latent est une représentation compacte et abstraite des données originales. C'est un espace de dimensions réduites où chaque point représente une combinaison unique de caractéristiques sonores. Explorer cet espace permet de découvrir de nouveaux sons qui conservent les caractéristiques essentielles du modèle appris.

Modèle

Un modèle est un réseau de neurones entraîné sur des données spécifiques. Dans Latent-Playground, vous pouvez charger différents modèles pré-entraînés pour explorer leurs espaces latents uniques.

Génération procédurale

La génération procédurale désigne la création automatique de contenu selon des règles et algorithmes. Ici, elle s'appuie sur des algorithmes de génération de textures visuelles.

Concepts approfondis

Encodeur / Décodeur : Un encodeur est un réseau de neurones (ou autre algorithme) qui compresse un signal d'entrée en une représentation plus compacte. Un décodeur fait l'inverse : il prend une représentation compacte et génère un signal (idéalement proche de l'original). Ensemble, encodeur et décodeur forment un auto-encodeur lorsqu'ils sont entraînés conjointement pour reproduire les données d'entrée. Pour l'utilisateur, l'encodeur/décodeur agit comme une « boîte noire ».

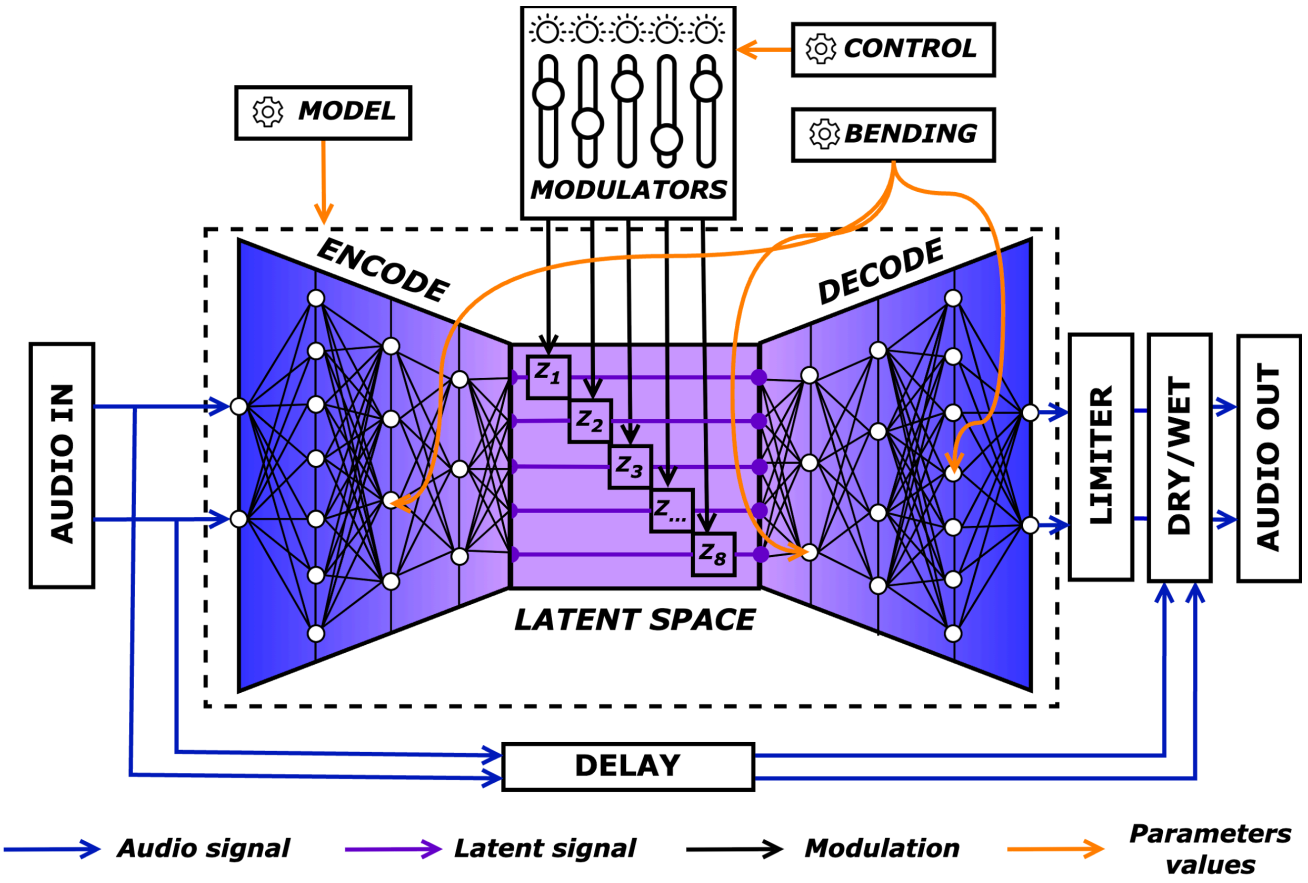
Espace latent & variables latentes : En apprentissage automatique, un espace latent est un espace de caractéristiques cachées, non observables directement, dans lequel un modèle projette des données complexes. Les variables latentes sont les coordonnées dans cet espace. Dans un auto-encodeur audio, les latents sont les nombres produits par l'encodeur et résumant le son. On peut voir cela comme les « caractéristiques essentielles » du signal en entrée que le modèle a appris à extraire pendant sa phase d'entraînement. Manipuler ces variables latentes revient à demander au modèle de reconstruire un son comme si les caractéristiques extraites avaient été différentes.

Modulation : En synthèse sonore, la modulation consiste à faire varier un paramètre au cours du temps grâce à un autre signal. Ici, les modulateurs font varier les variables latentes en fonction de signaux aléatoires à basse fréquence.

Bruit : En audio, on parle de bruit pour désigner un signal aléatoire. Ici, le bruit est utilisé de façon créative pour générer des trajectoires pseudo-aléatoires. Le terme pseudo-aléatoire indique que ce bruit est produit par un algorithme déterministe initialisé par une graine (Seed) — il est aléatoire d'un point de vue de l'utilisateur, mais reproductible si besoin.

Modèle RAVE : RAVE (Realtime Audio Variational autoEncoder) est le type de modèle de deep learning utilisé dans Latent-Playground. Développé par Antoine Caillon et Philippe Esling (2021), il s'agit d'un auto-encodeur entraîné pour la synthèse audio de haute qualité en temps réel. Son encodeur réduit le son en un vecteur latent à ~23 Hz, et son décodeur reconstruit le signal audio original avec une latence faible, ce qui le rend idéal pour une utilisation en tant qu'effet temps-réel. Latent-Playground exploite ces propriétés en insérant des modulations dans l'espace latent de RAVE pour en détourner le comportement et créer de nouveaux sons. Pour plus de détails techniques sur RAVE, vous pouvez consulter les publications de recherche associées ou le code source du modèle sur le dépôt du projet.

Architecture générale



Comme illustré ci-dessus, Latent-Playground est composé de plusieurs modules internes qui interagissent pour transformer le son :

Entrée Audio : correspond au signal audio brut de la piste Ableton. Il est acheminé vers le module d'encodage du modèle, et aussi éventuellement directement vers la sortie via le contrôle Dry/Wet.

Encodeur : c'est la première partie du modèle de réseau de neurones. L'encodeur reçoit le son en entrée et le convertit en une représentation compressée interne, appelée vecteur latent, notée généralement Z (ici constitué de 8 composantes $z_1 \dots z_8$). Ce vecteur latent est une version compacte du signal audio, capturant ses caractéristiques essentielles.

Espace latent : il s'agit de l'espace mathématique où vit le vecteur Z. Dans Latent-Playground, l'espace latent est modifié par les modulateurs avant d'être renvoyé vers le décodeur. En temps réel, le vecteur latent évolue sous l'influence de ces modulateurs, créant ainsi une trajectoire latente dynamique.

Modulateurs : ce sont des générateurs de signal aléatoire à basse fréquence qui viennent perturber ou modifier les composantes du vecteur latent. Ils fonctionnent un peu comme des LFO, sauf qu'ils génèrent des séquences pseudo-aléatoires plutôt que des formes d'onde régulières. Chaque dimension latente z_i possède son propre modulateur. Les modulateurs peuvent être activés ou désactivés selon qu'on souhaite introduire ces fluctuations ou garder les latents générés par l'encodeur comme tels.

Décodeur : c'est la seconde partie du modèle neuronal, il prend en entrée le vecteur latent et reconstruit en sortie un signal audio. Si le vecteur latent est inchangé (pas de modulation), le décodeur restituera un son proche de l'original (idéalement identique si le modèle est parfaitement entraîné). Si en revanche les latents sont modifiés, le son de sortie sera transformé en conséquence.

Limiteur : en sortie, un limiteur de sécurité peut être appliqué au signal audio transformé. Le processus de décodage et la modulation peuvent engendrer des crêtes imprévues ou amplifier certaines fréquences, risquant de saturer le signal.

Sortie Audio : c'est le signal audio final produit par le décodeur, éventuellement mixé avec le signal direct via le Dry/Wet.

Délai d'alignement : compense la latence du modèle. Le délai nécessaire dépend du modèle chargé.

Modulation de l’espace latent

La section Modulation de Latent-Playground permet de faire évoluer automatiquement le vecteur latent du modèle, en suivant des trajectoires générées de façon procédurale. Au lieu de rester figé à un emplacement défini par les offsets manuels, le point dans l’espace latent se déplace continuellement selon un motif pseudo-aléatoire. Concrètement, cela signifie que le son va changer dynamiquement au fil du temps, sans intervention manuelle, en explorant différentes zones de l’espace latent. Lorsque la modulation est activée (*Enable-Mod* sur On), les variations générées viennent s’ajouter aux offsets latents définis par l’utilisateur et les valeurs générées par l’encodeur si celui-ci est activé.

Trajectoires latentes procédurales

Une fois la modulation activée, le module génère huit trajectoires latentes coordonnées, c’est-à-dire des évolutions synchronisées sur les huit dimensions du vecteur latent. Ces trajectoires sont produites par des séquences pseudo-aléatoires basées sur des fonctions de bruit procédural. L’utilisateur peut choisir le type de séquence utilisé via le menu déroulant *Basis*. Chacune propose un style de mouvement différent dans l’espace latent :

- ***cell*** – Bruit produisant des variations par blocs. Le vecteur latent prend des valeurs relativement stables pendant un certain temps puis saute brusquement à d’autres valeurs. Ce motif en “cellules” crée des changements saccadés et peut servir pour des effets rythmiques ou glitch.

- ***simplex*** – Bruit Simplex générant une trajectoire douce et continue. Le vecteur latent dérive progressivement dans l’espace, avec des fluctuations fluides et organiques.

- ***perlin*** – Bruit à gradients qui, comme le Simplex, produit des changements relativement continus mais avec un caractère légèrement différent. Les trajectoires en mode gradient restent relativement douces et subtiles.

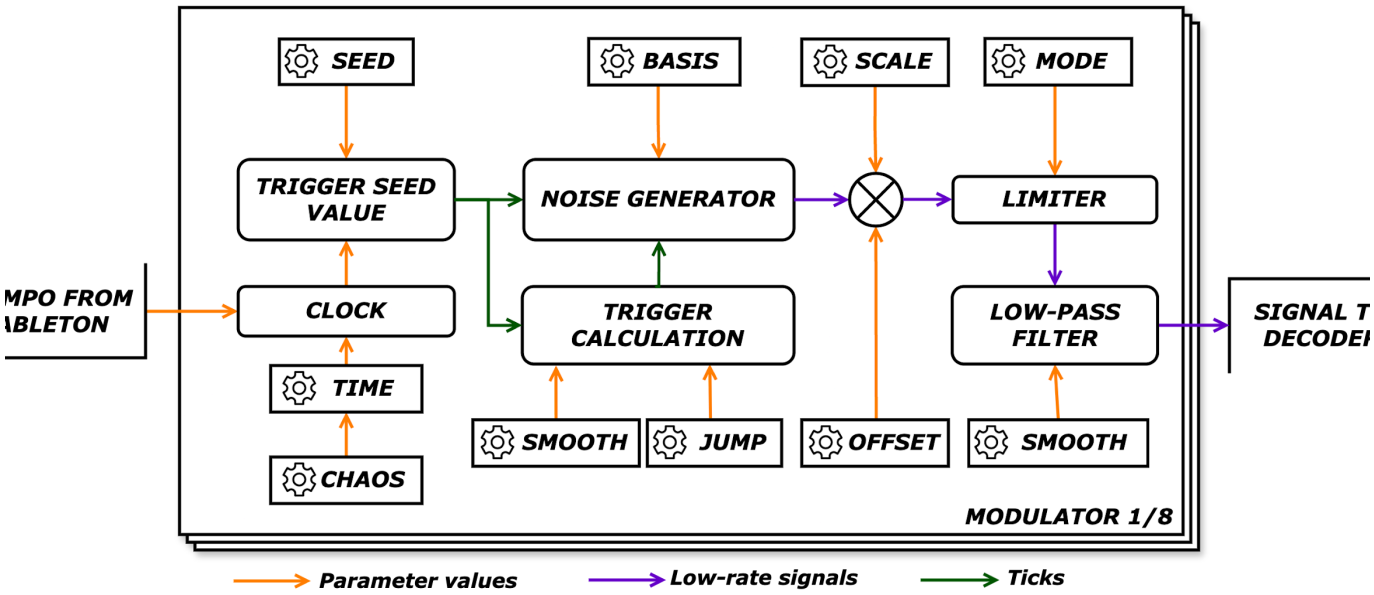
- ***distorted*** – Bruit combinatoire distordu, qui crée des trajectoires plus chaotiques en déformant l’espace de variation. Le mouvement latent devient moins prévisible, avec des fluctuations complexes. Ce mode donne des variations riches et aléatoires, pouvant aller de mouvements lents irréguliers à des oscillations très chaotiques selon le réglage de vitesse.

- ***voronoi*** – Bruit de type Voronoï engendrant des motifs segmentés par régions. Le vecteur latent tient une valeur jusqu’à atteindre une frontière puis bascule soudainement vers une autre valeur stable, un peu comme un changement d’état. Cela produit des variations en paliers irréguliers qui peuvent rappeler des motifs rythmiques non linéaires (par exemple des alternances imprévues après des durées

variables).

- ***fractal*** – Bruit fractal turbulent combinant plusieurs échelles de variation. La trajectoire latente obtenue est complexe et agitée, mélangeant des oscillations rapides et lentes. On obtient ainsi des changements avec des détails à petite échelle (variations rapides) superposés à des dérives à plus grande échelle. Ce mode est propice à des modulations très animées, occupant l'espace latent de façon erratique tout en restant cohérentes.

Schéma d'architecture du modulateur



Paramètres de modulation

Plusieurs paramètres permettent de contrôler finement la génération des huit trajectoires latentes.

- ***Reset*** – Ce bouton permet de réinitialiser tous les paramètres des modulateurs mis à part *mode* et *basis*.

- ***Jump*** – Ce paramètre détermine la rapidité du parcours dans l'espace latent. Il influe sur l'ampleur du saut du vecteur latent d'une étape à l'autre. Une vitesse élevée fera varier le son très rapidement, en provoquant de grands changements à chaque nouvelle étape de la séquence (trajectoire plus "nerveuse"). Au contraire, une valeur de *Jump* faible ralentit le déplacement : les positions latentes successives seront plus proches les unes des autres, ce qui se traduit par une évolution lente et progressive du son. En pratique, utilisez un *Jump* bas (p. ex.

100–200) pour des modulations douces, et un *Jump* plus haut (vers 800–1000) pour des variations énergiques ou chaotiques.

- ***Mode*** : Chaque modulateur comprend aussi un limiteur de plage qui s'assure que la valeur modulée reste dans la plage [-3, 3]. *Mode* détermine alors la manière dont ce limiteur gère les valeurs excédant cette plage. *None* désactive le clipping. *Clip* écrête le signal abruptement. *Warp* replie le signal. *Fold* replie le signal en changeant sa polarité.

- ***Seed*** : Afin que les séquences pseudo-aléatoires soient reproductibles et éventuellement bouclées, chaque modulateur utilise une graine (nombre initial pour le générateur aléatoire). Pour plus de lisibilité et moins de surcharge de paramètres, les 8 valeurs de *seed* sont prélevées dans un nuage de points qui ordonne ces valeurs par rapport à des caractéristiques statistiques (voir footnote 6).

- ***Trigger*** : les huit générateurs peuvent être relancés de manière synchronisée en utilisant le bouton *trigger* et les valeurs de *seed* actuellement établies. L'horloge envoie régulièrement un signal au générateur de bruit indiquant qu'il est temps de calculer une nouvelle valeur aléatoire pour la modulation. Ce trigger peut également être utilisé pour déclencher un renouvellement de *seed.*

- ***Time*** – Ce paramètre permet de synchroniser la modulation sur le tempo du projet lorsque le transport de Live est activé. Lorsqu'une *Time-Mode* est activé, les étapes de la trajectoire latente seront quantifiées sur le transport de Live avec la durée rythmique spécifiée . Par exemple, en choisissant une subdivision à la noire (1/4), les séquences alimentant le vecteur latent seront déclenchées à partir des valeurs *seed* à chaque temps, créant ainsi des trajectoires bouclées à la noire. Si *Time-Mode* est désactivé alors les trajectoires vivent selon une horloge propre, spécifiée en ms.

- ***Chaos*** – Pour plus de lisibilité et moins de surcharge de paramètres, les 8 valeurs de *Time* sont calculées à partir d'une valeur main et un offset aléatoire est appliqué à chaque dimension. Cet offset est de ±*Time.*

- ***Energy-Scale*** – Ce paramètre contrôle l'ampleur des variations autour de la position latente de base. En augmentant *Scale*, on élargit le rayon d'exploration du vecteur latent : les valeurs s'écartent davantage de l'offset central, pouvant aller aux extrémités de l'espace latent. Un Scale important permet donc d'obtenir des contrastes marqués dans le son (exploration de zones latentes très éloignées du point de départ), tandis qu'un *Scale* faible restreint la trajectoire autour du point initial pour des variations plus subtiles. Par exemple, pour ne pas trop s'éloigner d'un timbre de base, gardez Scale vers 0.5–1 ; pour explorer des timbres très différents en continu, vous pouvez pousser vers 2 ou 3.

- ***Deviation-Offset*** – Ce paramètre introduit un décalage des valeurs latentes pour chaque dimension. Il est très imprédictible mais s'associe avec *Mode* pour écarter les valeurs latentes sans sortir de l'espace de timbre appris.

- ***Smooth*** – Ce paramètre règle la fréquence de coupure du filtre de sortie. Il détermine également la fréquence de calcul des valeurs de bruit successives. En d'autres termes, *Smooth* agit comme un paramètre de résolution pour la génération des trajectoires latentes.

Module Control

Ce module permet d'ajouter une couche d'interaction en offrant un moyen intuitif de manipuler les paramètres des modulateurs via un slider de contrôle à deux dimensions. À l'aide d'un petit réseau de neurones (MLP - Multi-Layer Perceptron), ce module est capable de déduire une disposition de tous les paramètres à partir d'un point XY. Pour ce faire, il doit recueillir quelques couples [paramètres | XY] pour pouvoir apprendre un mapping pertinent.

L'idée est de proposer un espace de contrôle réduit et sensible, dans lequel chaque position (X, Y) est mappée vers un ensemble de paramètres complets via un réseau entraînable. Cela permet par exemple de naviguer entre différentes textures ou dynamiques sonores simplement en bougeant un curseur mappable dans Live.

1. Choisissez une zone du pad XY pour spécifier la position d'un point dans l'espace de contrôle.

2. Choisissez les paramètres à mapper parmi *seed, scale, offset* et *time*.

3. Activez le mode ***MAP***, une fois la disposition des paramètres des modulateurs établis, cliquez sur ***ADD***, un point apparaît alors en arrière-plan. Si vous voulez revenir en arrière, vous pouvez choisir un couple de points dans le menu déroulant puis utiliser ***DELETE*** ou ***UPDATE***.

4. Répétez l'étape 3 une dizaine de fois, en dispersant sur le pad les variations sonores selon vos idées de morphing.

5. Cliquez sur ***TRAIN***, le MLP effectue ses calculs et fait apparaître une valeur qui témoigne de la pertinence des résultats. Plus cette valeur est proche de 0 plus le réseau se sera "accroché" aux données d'entrées.

6. Activez le mode ***PREDICT***, la manipulation du pad XY se répercute sur les paramètres des modulateurs. Si le mapping n'est pas pertinent, revenez à l'étape 3.

7. Pour initier un nouveau mapping, appuyez sur ***RESET*** puis revenez à l'étape 1.

8. Vous pouvez sauvegarder votre mapping puis le charger et l'appliquer à nouveau.

Module Bending : Modification directe du modèle

Le module Bending permet de modifier les poids internes du modèle chargé afin d'en altérer profondément le comportement. Cette approche s'inspire du circuit bending en électronique, où l'on détourne un circuit de son fonctionnement original pour obtenir des résultats créatifs inattendus. De même, le bending agit au cœur du modèle en changeant ses paramètres appris, contrairement aux modulateurs qui se contentent de modifier les latents. Le résultat est une forme d'édition avancée du modèle : vous créez de nouvelles variations de comportement à partir d'un modèle pré-entraîné, sans devoir le réentraîner.

1. Initialisez la mise en mémoire des paramètres du modèle : une fois le modèle chargé, appuyez sur ***INIT*** dans la fenêtre bending.

2. Sélectionnez la couche à modifier : Le module Bending opère généralement par couche (ou groupe de poids) du réseau. Sélectionnez la couche que vous souhaitez manipuler avec le slider ***LAYER*** en bas. Le device affichera les valeurs actuelles de ces poids pour la couche sélectionnée.

3. Ajustez les paramètres de bending : Trois paramètres principaux vous permettent de définir comment les poids seront altérés :

    - ***FLIP*** : inverse la polarité de tous les poids de la couche sélectionné.e

    - ***SCALE*** : multiplie tous les poids de la couche par un facteur. Une valeur supérieure à 1 amplifie les poids (connections plus fortes), tandis qu'une valeur entre 0 et 1 les atténue (jusqu'à 0 qui annulerait complètement les poids). Ce scaling conserve la structure relative des poids d'origine, mais modifie leur amplitude globale.

    - ***DISTRIBUTION*** : permet de remplacer les poids par de nouvelles valeurs tirées d'une distribution aléatoire contrôlée. Quatre types sont disponibles (sélectionnables via les icônes correspondantes) : Uniforme, Gaussienne, Bêta et Laplace. Le choix de la distribution influence la répartition statistique des nouveaux poids.

    - ***CHAOS*** : contrôle à quel point les nouveaux poids aléatoires s'écartent des valeurs d'origine. En pratique, le module analyse d'abord la distribution des poids actuels de la couche, puis génère de nouvelles valeurs qui en dévient plus ou moins fortement selon le pourcentage de chaos. Ce paramètre réalise une randomisation partielle : à 0% on préserve presque entièrement le pattern appris, à 100% on le remplace quasi complètement par du hasard. Vous pouvez ainsi doser finement l'introduction d'aléatoire dans les poids.

4. Ajustements manuels : le module Bending offre une interface visuelle des poids, vous pouvez directement dessiner ou modifier ces valeurs avant de les appliquer.

5. Revert avant application : si finalement la distribution générée ne vous convient pas avant de l'avoir appliquée, utilisez la fonction ***REVERT***. Cela va simplement annuler les changements en cours pour cette couche et restaurer dans l'interface les poids précédents sans impacter le modèle.

6. Appliquez les nouveaux poids au modèle : quand vous êtes satisfait de la configuration générée pour la couche, cliquez sur ***SET*** pour envoyer ces poids dans le modèle. La couche sélectionnée utilisera alors immédiatement ces nouvelles valeurs en interne.

7. Recall des poids d'origine : si vous avez déjà appliqué les nouveaux poids au modèle mais que le résultat ne vous satisfait pas, le module offre un moyen de revenir en arrière en appuyant sur ***RECALL.***

8. Reload : si nécessaire, vous pouvez à tout moment recharger l'ensemble du modèle à son état initial grâce à l'option ***RELOAD***. Cela va remplacer les poids de toutes les couches par ceux d'origine du modèle pré-entraîné.

9. Sauvegarder une configuration de bending : lorsque vous avez obtenu une altération du modèle qui vous plaît, vous pouvez la sauvegarder pour la réutiliser plus tard. La sauvegarde est constituée d'un fichier .wav contenant tous les paramètres du réseau et d'un fichier .json qui permet de rediriger les poids vers la bonne couche du réseau au chargement.

Live-Mode : lorsque *Live-Mode* est activé, les interventions sur la couche sélectionnée sont directement transmises au modèle.

Dépannage et ressources

Si vous rencontrez des difficultés avec Latent-Playground, voici quelques pistes de dépannage et ressources utiles :

- **Le son reste inchangé (aucun effet audible)** : ouvrez la fenêtre Max (Max Console) via Ableton (clic droit sur le titre du device > Edit) et regardez les messages d'erreur éventuels. Certains externals et modèles peuvent ne pas se charger correctement.

- **Artefacts indésirables dans le son** : Si vous entendez des bruits parasites, des craquements ou des fluctuations trop abruptes, deux causes possibles : (1) Le CPU de votre ordinateur est surchargé, ce qui peut arriver, car le processus neuronal est gourmand – essayez d'augmenter la taille du buffer audio d'Ableton, 2048 samples est une bonne valeur. (2) Les modèles chargés n'ont pas été exportés en mode *streaming* ou ne contiennent pas les méthodes *encode* et *decode.*

- **Le device ne se charge pas / crash au chargement** : Latent Playground utilise des objets externes de la librairie FluCoMa et un objet mcs.nn~ modifié pour son fonctionnement interne. Si nécessaire, ouvrez Max et chargez un à un les objets externals dans un patch vide. Vérifiez les chemins dans *Files Preferences* et les versions utilisées de Max for Live (>= 8.6.5).

- **Demander de l'aide** : Rendez-vous sur le dépôt GitHub du projet Latent-Playground où vous pouvez ouvrir une nouvelle page d'aide dans *issue*. Concernant Rave plus généralement vous pouvez consulter les Discord ACIDS et RAVE.

Footnotes

1. Le modèle pré-entraîné est généralement fourni sous forme de fichiers de poids au format .ts. Latent Playground attend que vous pointiez vers le dossier contenant ces fichiers.

2. Que représentent les variables latentes ? – Il est difficile de donner une signification simple à chaque z_i. Dans un modèle entraîné sur de la voix, il se peut qu'un z contrôle grossièrement la hauteur du formant, un autre la brillance, etc., mais ce ne sont pas des paramètres explicitement définis, plutôt des composantes abstraites que le réseau a apprises. L'avantage est qu'en les combinant et en les modulant, on explore des transformations complexes du son (par ex. changer z_1 et z_5 peut altérer le timbre différemment que changer seulement l'un des deux). Il s'agit vraiment d'un « espace sonore » dans lequel on peut se déplacer en utilisant les z comme coordonnées.

3. RAVE – Realtime Audio Variational autoEncoder. RAVE utilise un auto-encodeur convolutionnel avec une perte de type adversaire et perceptuelle, permettant un codage efficace du son. Il est variationnel car il apprend non seulement à compresser le son, mais aussi la distribution probabiliste des latents, ajoutant un bruit gaussien pendant l'entraînement pour généraliser. En pratique, dans Latent Playground, l'aspect variationnel signifie qu'il y a une part de bruit inhérente dans l'encodage – ce qui tombe bien puisque nous exploitons également du bruit dans la modulation !

4. Fréquence de modulation vs fréquence latente – Comme mentionné, le modèle RAVE met à jour les latents ~23 fois par seconde. Si vos modulateurs génèrent des changements plus lents, alors pendant plusieurs frames latentes consécutives le décodeur va recevoir la même valeur (si pas de lissage) ce qui crée un palier stable, puis un saut. Cela peut s'entendre comme un léger crénelage ou un saut abrupt dans le son. Pour éviter cela, vous pouvez soit augmenter la fréquence de modulation, soit augmenter le lissage de manière à ce que le changement soit graduel même à basse fréquence.

5. Interpolation des poids du modèle – La manipulation des paramètres d'un réseau de neurones artificiels pose le problème de la taille des données en jeu. Le backend PyTorch qui gère les calculs tensoriels du modèle gère des couches pouvant contenir jusqu'à un million de valeurs flottantes. Dans MaxMsp, les listes sont limitées à 32767 éléments, ce qui nécessite de réduire la taille des couches avant de les mettre en mémoire et donc d'effectuer l'opération inverse lorsque l'on veut modifier le modèle. C'est une opération destructive dans les deux sens pour les couches de taille > 32767.

6. Seed-Map – Les graines sont des nombres entiers qui servent à donner un point de départ à la génération de variables pseudo-aléatoires déterministes. C'est un paramètre essentiel mais très peu intuitif, on pourrait le décrire comme chaotique, deux graines successives *i* et *i+1* donnant des séquences de nombres très

différentes. De plus, la plupart des générateurs utilisés dans les logiciels audio acceptent des nombres entiers signés codés en 32 bits, ce qui donne des valeurs possibles de graine entre -2 147 483 648 et 2 147 483 647. Pour permettre une utilisation plus fluide, Latent-Playground génère des nuages de points à partir de données statistiques issues d'échantillons de textures (buffers de 8192 samples) dont les graines ont été tirées aléatoirement. Un algorithme UMAP est alors utilisé pour réduire ces données (256 échantillons à ~25 dimensions d'analyse) en un nuage de points (256 échantillons à 2 dimensions).