NEURALYZER : UN AGENT CONVERSATIONNEL POUR L'ANALYSE AUDIONUMÉRIQUE

Pierre Guillot
IRCAM - Centre Pompidou
STMS UMR 9912
pierre.guillot@ircam.fr

RÉSUMÉ

Cette contribution présente Neuralyzer, une extension intégrée au logiciel d'analyse sonore Partiels développé à l'IRCAM, prenant la forme d'un agent conversationnel associé à un serveur implémentant le Model Context Protocol (MCP). Neuralyzer vise à assister les utilisateurs dans l'analyse de fichiers audionumériques en facilitant le paramétrage technique des traitements et l'automatisation d'opérations répétitives. L'article examine dans quelle mesure l'utilisation de grands modèles de langage (LLM) peut constituer une couche d'assistance cognitive au sein d'un environnement d'analyse sonore. L'attention est portée sur la stratégie d'inférence locale des modèles, motivée par des considérations de gratuité d'usage, de protection des données et de sobriété numérique pour une approche souveraine. Le texte retrace l'évolution technique du projet, depuis une première phase reposant sur l'utilisation d'API distantes jusqu'à l'intégration de modèles exécutés localement et à l'adoption du MCP permettant une interaction structurée et contrôlée entre le modèle et les fonctionnalités internes du logiciel. Les différentes architectures expérimentées sont analysées de manière critique et les premiers résultats techniques et fonctionnels sont présentés. L'ensemble de ces travaux suggère qu'une intégration maîtrisée de modèles de langage rend l'exploration sonore plus accessible dans des contextes de recherche, de pédagogie et de création musicale.

1. ORIGINES ET ENJEUX DE NEURALYZER

Neuralyzer correspond à un ensemble de fonctionnalités intégrées au logiciel Partiels [7, 8] visant à assister l'utilisateur dans l'interprétation, la configuration et l'enchaînement des analyses. Neuralyzer repose sur l'inférence de grands modèles de langage (LLM) et sur l'utilisation des agents conversationnels [6, 20]. Il ne constitue pas un logiciel autonome, mais une extension fonctionnelle de l'environnement existant. Afin de situer précisément les enjeux scientifiques et techniques et d'éclairer les problématiques auxquelles Neuralyzer entend répondre, il est nécessaire de rappeler brièvement l'architecture et l'évolution du projet Partiels, notamment ses développements récents.

1.1. Aperçu général de Partiels

Partiels est un hôte de plug-ins Vamp, format développé au Centre for Digital Music de Queen Mary University of London spécifiquement destiné à l'analyse de flux audionumériques [4]. L'architecture logicielle repose sur un modèle modulaire : les analyses sont implémentées sous forme de plug-ins dynamiquement chargés. Cette approche permet l'intégration de moteurs d'analyse développés en interne ou par des institutions tierces, l'ajout de nouvelles analyses sans modification du cœur applicatif, ainsi que la coexistence de modèles hétérogènes (algorithmiques et neuronaux).

Comme détaillé dans les publications précédentes [7, 8], le SDK Vamp offre un cadre générique robuste mais présente certaines limitations structurelles :

- impossibilité native de chaîner les analyses ;
- restriction des entrées non audio à des scalaires ;
- restriction des sorties à des types simples (label, scalaire, vecteur).

Ces contraintes deviennent problématiques lorsque les moteurs nécessitent des dépendances inter-analyses — par exemple lorsque les résultats d'une analyse (ensemble de scalaires ou d'étiquettes) doivent être utilisés en entrée d'une autre analyse (transcription puis alignement) — ou lorsque les sorties sont multidimensionnelles, c'est-à-dire lorsqu'une trame de résultat contient plusieurs scalaires représentant des données distinctes (fréquence, amplitude, phase, score de confiance). Ces limitations ont été levées grâce à l'extension Ircam Vamp Extension (IVE), une interface de programmation développée spécifiquement pour étendre les mécanismes d'entrée et de sortie des plug-ins Vamp. IVE permet notamment le chaînage d'analyses via un système d'entrées plus flexible ainsi que la transmission de résultats enrichis.

1.2. Évolution du projet Partiels

Le développement de Partiels a débuté en 2020. Depuis la première publication en 2021, puis la mise à disposition gratuite en 2023 et le passage en open source sous licence GPLv3 en 2024, l'application a connu plusieurs évolutions structurantes. À ce stade, Partiels offre notamment :

- un ensemble étendu d'analyses, incluant la reprise de nombreux outils historiquement disponibles dans l'application AudioSculpt [2, 3] ainsi que de nouvelles analyses reposant notamment sur des modèles neuronaux (tels que Whisper [18] et VAX [5]) ;
- une architecture et un modèle de données assurant la compatibilité ascendante des documents, permettant d'intégrer de nouvelles fonctionnalités et de modifier les outils existants sans rupture de continuité ;
- une organisation structurée en documents, groupes et pistes d'analyse, facilitant la comparaison et la superposition des résultats ;
- un système avancé de paramétrage graphique permettant de modifier les représentations des résultats (couleurs, polices de caractères, positionnement, etc.) ;
- une interopérabilité étendue via l'export vers différents formats (CSV, JSON, CUE, SDIF, PNG, JPEG, etc.) ainsi que via le protocole Open Sound Control (OSC) [21].

Les développements récents ont en outre permis l'introduction d'un système de préréglages (presets) facilitant l'adaptation des analyses à des contextes d'usage spécifiques, la mise en place d'un système de traduction de l'interface favorisant la contribution communautaire, ainsi que l'amélioration des mécanismes d'exportation vers des environnements tels que Max [1], Pure Data [14] et Reaper [2].

À ce stade, Partiels peut être considéré comme un outil présentant un niveau de maturité fonctionnelle élevé, utilisé dans des contextes de composition, d'enseignement et de recherche.

1.3. Verrous fonctionnels et techniques

Malgré cette maturité, plusieurs verrous techniques et fonctionnels persistent. Parmi les enjeux actuels de Partiels figurent l'élargissement de son accessibilité à des utilisateurs moins experts, l'amélioration des interfaces et l'intégration de nouvelles couches d'interaction. Deux situations récurrentes illustrent ces limites : la configuration de la transformée de Fourier discrète et l'analyse de multiples partiels harmoniques.

1.3.1. La configuration de la transformée de Fourier discrète (DFT)

La transformée de Fourier discrète (TFD) est un outil fournissant une représentation fréquentielle du signal sonore, à la base de nombreuses analyses telles que le barycentre spectral, l'estimation de la fréquence fondamentale ou le sonagramme. Le choix de la taille de fenêtre, du type de fenêtrage, du chevauchement entre fenêtres et du suréchantillonnage requiert une compréhension précise des mécanismes en jeu. Le vocabulaire technique mobilisé ne correspond pas nécessairement aux catégories de pensée de tous les utilisateurs. Certains paramètres pourraient être reformulés en termes de résolution ou de précision fréquentielle et temporelle afin d'en faciliter la compréhension. D'autres pourraient être décrits au moyen d'un vocabulaire plus directement interprétable, comme la complétion de zéros (zero-padding) dans le cas du suréchantillonnage.

1.3.2. L'analyse de multiples partiels harmoniques

Les analyses de partiels harmoniques issues de moteurs tels que PM2 [19] produisent des données multidimensionnelles (fréquence, amplitude, phase) pour chaque partiel. Afin d'offrir une représentation adéquate des résultats, chaque partiel est associé à une piste d'analyse distincte. Toutes les pistes partagent les mêmes paramètres, à l'exception de l'index du partiel. Par ailleurs, afin d'obtenir des résultats robustes, ces analyses requièrent la fréquence fondamentale en entrée. L'analyse de vingt partiels harmoniques implique donc la création d'un groupe, l'analyse préalable de la fréquence fondamentale, la création de vingt analyses de partiels, la modification de l'index du partiel pour chacune d'elles (et, si possible, l'adaptation du nom de la piste en conséquence), puis l'association de l'analyse de fréquence fondamentale à l'ensemble des analyses de partiels. Il s'agit d'un ensemble d'opérations répétitives et fastidieuses, susceptibles de générer des erreurs de configuration.

Ces deux situations soulèvent alors deux questions :

- Comment paramétrer des analyses nécessitant un niveau élevé de technicité ou mobilisant un vocabulaire variable selon les usages ?
- Comment automatiser la création de multiples analyses en ne faisant varier que certains paramètres ?

C'est dans ce contexte que s'inscrit le développement de Neuralyzer : non comme un remplacement des mécanismes existants, mais comme une couche d'assistance destinée à réduire la charge cognitive et les redondances associées à la création et à la configuration des analyses.

2. SPÉCIFICATIONS ET PROTOTYPES DE NEURALYZER

Au-delà des enjeux strictement fonctionnels, ce projet constitue un terrain d'expérimentation critique de différentes approches d'intégration de grands modèles de langage (LLM), notamment au regard de sa souveraineté qui inclue des contraintes éthiques, économiques et écologiques propres aux domaines d'application de Partiels.

1. https://cycling74.com/products/max
2. https://www.reaper.fm/

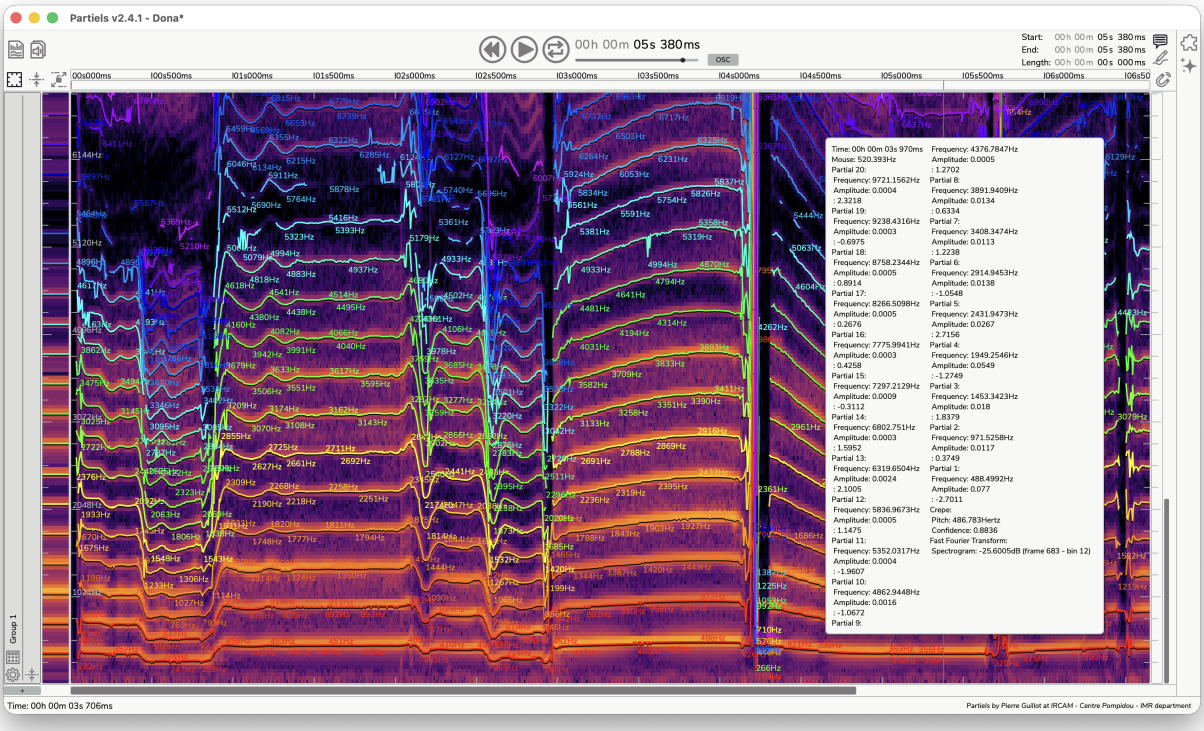

Figure 1. Aperçu de l'application Partiels avec un sonagramme, une analyse de la fréquence fondamentale et les analyses des vingt premiers partiels harmoniques produites par inférence d'un LLM et de l'utilisation du MCP à partir de la requête "*Perform the analysis of twenty partial harmonics*".

- La gratuité du service est en effet indissociable des usages éducatifs, universitaires et de recherche de Partiels. Il n'est pas envisageable, dans ces contextes, d'imposer aux étudiants ou aux utilisateurs le paiement de requêtes afin d'accéder à une assistance intégrée.
- La protection des données personnelles constitue également un enjeu central prévu par la loi en France.
- Enfin, la prise en compte de l'impact écologique de l'outil représente un critère important. L'objectif est, autant que possible, de proposer une solution frugale, limitant les ressources computationnelles et les échanges réseau inutiles.

L'un des objectifs majeurs du projet est également d'explorer l'intégration des LLM dans un cadre applicatif ouvert et gratuit. Au-delà de la qualité des réponses produites, il importe à l'utilisateur de comparer les approches, les modèles et leurs paramètres d'inférence dans un contexte réel d'usage.

Plusieurs approches ont été envisagées et testées. Les choix retenus tiennent compte des contraintes techniques liées au développement et à la distribution de Partiels, des considérations ergonomiques et de l'expérience utilisateur, ainsi que de l'intérêt scientifique et technologique de certaines solutions. Bien que présentées ici selon un ordre logique d'intégration, certaines approches ont été explorées sans perspective immédiate d'intégration, tandis que d'autres ont été développées dès les premières phases du projet.

2.1. Utilisation d'une API distante

Une première approche consiste à utiliser des services existants de grands modèles de langage, tels que ChatGPT (OpenAI) [12] ou Le Chat (Mistral) [3]. Cette solution, largement adoptée dans de nombreuses applications, présente l'avantage de ne pas requérir de compétences techniques approfondies en matière d'inférence de LLM.

Dans cette configuration, Partiels permet de définir la clé privée d'un compte personnel associé à un service en ligne (des tests ont été réalisés avec OpenAI et Mistral). Cette clé autorise l'envoi d'invites (prompts) au nom de l'utilisateur via l'API du service. Une fenêtre de discussion intégrée à Partiels permet d'écrire et de visualiser les messages [4]. Lorsqu'une invite est validée, une requête est transmise au service distant et la réponse du modèle est affichée dans l'interface en distinguant explicitement les rôles (utilisateur ou assistant).

Partiels étant une application open source disponible sur GitHub, certains modèles disposent d'une connaissance partielle de son existence. Toutefois,

3. https://mistral.ai/

4. La description détaillée de l'interface est présentée dans la troisième partie du document.

les réponses obtenues restent généralement superficielles. Les modèles ne disposent pas d'une connaissance opérationnelle du fonctionnement interne de l'application ni de la structure exacte des documents. Afin de permettre une modification effective d'un document, il est nécessaire de fournir un contexte explicite au modèle : documentation du logiciel, structure générale d'un document Partiels, puis description de l'état courant du document avant chaque requête. Partiels dispose pour cela de mécanismes d'introspection permettant de convertir un document, ou une partie de celui-ci, au format JSON ou XML. Le modèle peut alors renvoyer une description modifiée, qui est interprétée par l'application pour mettre à jour l'état du document.

En théorie, cette approche est robuste, dans la mesure où les entreprises proposant ces services se situent à la pointe des développements technologiques. Les performances initiales sont généralement élevées et l'accès via API est standardisé. Toutefois, plusieurs limites majeures apparaissent. Le coût pour les utilisateurs constitue un obstacle important : l'abonnement ou la facturation à la requête peut devenir rapidement prohibitif, en particulier lorsque la documentation et les descriptions de documents sont volumineuses. La représentation textuelle d'un document Partiels est en effet riche et détaillée. L'envoi de résultats d'analyse n'a pas été envisagé, leur volume rendant l'opération encore plus coûteuse. Par ailleurs, l'inférence sur des serveurs distants implique un impact écologique non négligeable. Enfin, la protection des données dépend des politiques propres à chaque service, dont la transparence n'est pas toujours garantie.

2.2. Inférence des modèles en local

Une solution plus conforme aux exigences économiques et écologiques consiste à inférer les modèles localement. De nombreux outils permettent aujourd'hui l'inférence de LLM sur une machine individuelle. Sans prétendre à un état de l'art, nécessairement rapidement obsolète, le projet s'est orienté vers la bibliothèque llama.cpp, bibliothèque libre et open source (licence MIT) d'inférence en C++ développée par Georgi Gerganov. Ce choix repose sur plusieurs éléments :

- maintenance active par une communauté étendue ;
- écosystème reposant sur GGML, bibliothèque de tenseurs également développée par Gerganov ;
- cohérence avec Whisper.cpp [18] (utilisé pour le plug-in Vamp Whisper), facilitant l'intégration multiplateforme (Windows, macOS, Linux) ;
- possibilité d'intégration directe dans Partiels, sans exécutable externe, simplifiant la distribution et la configuration [5] ;
- large disponibilité de modèles open source (Mistral [9], Qwen [15], LLaMA [11], Phi [13], etc.) ;
- contrôle fin des paramètres d'inférence et gestion manuelle du contexte.

Dans cette approche, Neuralyzer dispose toujours d'une fenêtre de discussion dédiée. L'utilisateur peut sélectionner le modèle, la taille du contexte [6] et la taille de batch [7]. Par défaut, Partiels propose plusieurs modèles Mistral 3 et Qwen 3.5 [15], en différentes tailles (9B, 14B, 27B, 35B) et quantifications (Q4, Q5). Le téléchargement et l'installation du modèle dans le répertoire approprié sont automatisés. Il est également possible d'installer manuellement d'autres modèles.

Lors de l'initialisation, Neuralyzer transmet au modèle une requête système définissant son rôle, la liste des plug-ins disponibles et les règles de formatage attendues (balises `<response>` et `<document>`). Un extrait de cette requête système est présenté ci-dessous :

```
Your goal is to assist users in analyzing
audio files, managing analysis tracks and
groups, and interpreting results inside
Partiels with maximum technical accuracy,
reliability, and workflow safety.
Available Plugins:
...

Return your response strictly following
this format:
<response>
[Your text in natural language]
</response>
<document>
[The XML description of the modified
document]
</document>
```

À chaque requête utilisateur, l'état courant du document est sérialisé en XML et injecté dans le contexte. La réponse du modèle est ensuite interprétée : le contenu entre `<response>` est affiché dans l'interface, celui entre `<document>` est converti en XML afin de modifier le document.

L'intérêt principal de cette approche réside dans la gratuité, la maîtrise des données et l'absence de requêtes vers des serveurs distants. L'intégration technique reste relativement directe et compatible avec l'architecture existante et ne nécessite pas de modifications complexes de l'architecture de Partiels. Bien que demandant des ressources relativement importantes, les modèles peuvent être inférés sur des ordinateurs personnels. Cependant, malgré ces avantages, les performances observées sont insuffisantes. Les modèles locaux ont une forte tendance à ne pas respecter strictement la syntaxe imposée pour la modification des documents, ce qui peut entraîner des erreurs critiques.

2.3. Utilisation du Model Context Protocol (MCP)

Un document Partiels peut être converti en représentation textuelle (XML ou JSON), et inversement. Cette capacité est utilisée notamment lors de la sauvegarde et du chargement des fichiers. La représentation textuelle

5 . L'utilisation d'une application autonome d'inférence (par exemple LM Studio), potentiellement exécutée sur un serveur local plus puissant, constitue une piste de développement futur.

6 . Nombre maximal de jetons que le modèle peut traiter simultanément.

7 . Nombre de jetons traités en parallèle pendant le traitement de la requête.

contient l'ensemble des informations nécessaires à la restauration complète du document, y compris les identifiants uniques des pistes et des groupes ainsi qu'un grand nombre de paramètres et métadonnées. Cette structure hiérarchique complexe s'avère difficile à manipuler de manière fiable par les modèles.

La solution consiste alors à ne plus exposer directement la représentation complète du document, mais à définir des méthodes spécifiques, clairement typées, permettant de modifier des éléments précisément définis. Chaque méthode possède des paramètres attendus et un schéma d'entrée explicite (signature), ce qui permet de détecter les erreurs et d'éviter toute modification incohérente. Le Model Context Protocol (MCP) [8], introduit fin 2024 par la société Anthropic, permet de décrire de telles méthodes (tools) avec leurs schémas d'entrée et de sortie. Les modèles récents (Claude, Qwen, GPT, Mistral) intègrent un support explicite de ce protocole.

Dans Neuralyzer, le support MCP est implémenté via une classe dispatcher communiquant au format JSON. Cette classe remplit deux fonctions principales :

1. exposer la description formelle des méthodes disponibles (nom, description, schéma d'entrée et de sortie) ;
2. traduire les appels de méthodes effectués par le modèle en actions internes spécifiques à Partiels.

Un exemple de description JSON d'une méthode MCP est présenté ci-dessous. Cette méthode, nommée `get_track_names`, permet de récupérer les noms de certaines pistes d'analyse dans Partiels :

```
{
  "name": "get_track_names",
  "description": "Query the names of one
                  or more tracks",
  "inputSchema": {
    "type": "object",
    "properties": {
      "identifiers": {
        "type": "array",
        "items": { "type": "string" },
        "description": "The unique identifiers of the
                        tracks"
      }
    },
    "required": [ "identifiers" ]
  }
}
```

Par exemple, pour la méthode `get_track_names`, le dispatcher vérifie que les identifiants fournis correspondent à des pistes existantes, puis renvoie une structure associant identifiants et noms. En cas d'erreur (identifiant inexistant, paramètre mal typé), une description explicite de l'erreur est renvoyée.

Afin de tester cette interface, Neuralyzer peut communiquer avec l'application Claude Desktop [9] via un serveur MCP intégré à Partiels. Les premiers résultats se sont révélés très encourageants : Claude, développé par Anthropic, offre un support robuste du protocole et parvient à décrire et modifier le document à partir de la description des méthodes (Figure 1).

L'étape suivante a consisté à intégrer le protocole MCP dans le moteur d'inférence locale. La bibliothèque llama.cpp fournit des interfaces facilitant la déclaration de méthodes et l'interprétation des appels effectués par les modèles compatibles. Seuls certains modèles récents prennent actuellement en charge cette fonctionnalité, mais le protocole tend à devenir un standard. Les résultats obtenus sont encourageants, bien que les modèles locaux, plus limités en nombre de paramètres et en taille de contexte, restent moins performants que Claude. Des erreurs d'appel persistent, mais Neuralyzer peut fréquemment les corriger en renvoyant au modèle une description précise de l'erreur détectée.

3. RÉSULTATS ET PERSPECTIVES DE NEURALYZER

En février 2026, les premières versions bêta de Partiels intégrant Neuralyzer ont été publiées sur GitHub [10]. Cette version permet l'inférence locale de modèles compatibles avec le protocole MCP, ainsi que la communication avec Claude Desktop, comme présenté dans la section précédente. Nous présentons ici les modalités d'usage de Neuralyzer, les premiers résultats observés et les perspectives de développement.

3.1. Modalités d'usage de Neuralyzer

Les interactions avec Neuralyzer s'inscrivent dans des séquences de requêtes successives au cours desquelles le modèle doit maintenir une cohérence explicite avec l'état courant du document. L'ensemble des opérations accessibles transite par les méthodes exposées via MCP, qui couvrent l'inspection et la modification des plug-ins, du document, des fichiers audio, des groupes et des pistes d'analyse. Les méthodes exposées par Neuralyzer via MCP couvrent les domaines suivants :

- Plug-ins : lister les plug-ins Vamp installés (identifiants et descriptions).
- Document : consulter l'état global (groupes, pistes) et configurer l'agencement des fichiers audio et des canaux.
- Fichiers audio : obtenir les métadonnées (format, fréquence d'échantillonnage, durée, canaux).
- Groupes : consulter ou modifier les noms et l'organisation des pistes ; créer ou supprimer des groupes.
- Pistes d'analyse : consulter le type, le nom, la description, les paramètres, les entrées et les propriétés graphiques ; modifier les paramètres, les entrées et l'affichage ; créer des pistes à partir de plug-ins ; supprimer des pistes.

Lorsqu'une requête est adressée à Neuralyzer, le modèle doit en premier lieu récupérer l'état courant

8. https://modelcontextprotocol.io/
9. https://docs.anthropic.com/
10. https://github.com/orgs/Ircam-Partiels/discussions/348

du document, ainsi que, si nécessaire, la liste des plug-ins disponibles. Sur cette base, il élabore une stratégie d'action consistant soit à répondre directement à l'utilisateur ou à l'utilisatrice, soit à invoquer une ou plusieurs méthodes MCP afin de modifier le document. Chaque appel est validé par le dispatcher interne, qui renvoie soit une description d'erreur, soit un compte rendu structuré des modifications effectuées. Ce mécanisme permet au modèle d'ajuster sa stratégie de manière itérative jusqu'à l'obtention du résultat souhaité. L'intégration de Neuralyzer s'appuie également sur le mécanisme natif d'undo/redo de Partiels. Toute modification effectuée via MCP est encapsulée dans une commande réversible. Ainsi, même en cas de stratégie inadéquate ou d'interprétation erronée par le modèle, l'utilisateur ou l'utilisatrice conserve la possibilité de revenir à un état antérieur du document.

3.1.1. Inférence locale du modèle

En inférence locale, il est possible de réinitialiser le modèle afin de nettoyer le contexte conversationnel et de repartir d'un état "vierge". Cette opération peut s'avérer nécessaire lorsque le modèle accumule des hallucinations ou lorsque l'on change de document [11] . Il convient de souligner que la mémoire conversationnelle ne contient pas l'intégralité de l'état effectif du document. Ce choix permet de limiter la taille du contexte transmis au modèle, mais impose en contrepartie que celui-ci réinterroge régulièrement l'état réel du document. Si cette étape n'est pas effectuée, des incohérences peuvent apparaître, ce qui nécessite parfois de rappeler explicitement au modèle la nécessité de synchroniser ses informations avec l'état interne de Partiels. Lorsque la capacité maximale de la fenêtre de contexte est sur le point d'être atteinte, Neuralyzer procède à un résumé de l'historique conversationnel en omettant les informations détaillées relatives à l'état du document, tout en conservant intégralement les messages les plus récents. Cette stratégie vise à stabiliser le comportement du modèle et à limiter les effets de saturation du contexte.

3.1.2. Neuralyzer comme serveur MCP

Neuralyzer peut également fonctionner comme serveur MCP indépendant, permettant à l'application Claude Desktop d'interagir directement avec l'instance de Partiels en cours d'exécution. Une option dédiée dans les propriétés de Partiels active ce mode, et la déclaration du serveur MCP est automatiquement configurée par l'application. Dans cette configuration, le modèle agit comme un orchestrateur externe invoquant les méthodes exposées par Partiels. La logique de raisonnement est assurée par le modèle distant, tandis que la logique d'exécution et de validation demeure strictement interne à Partiels. Cette dissociation renforce la séparation entre mémoire conversationnelle et état effectif du document, et permet d'expérimenter différentes stratégies d'inférence sans modifier l'architecture logicielle de l'application.

Des travaux récents ont commencé à explorer les applications du MCP dans le domaine audio. Fan [16] a présenté MCP2OSC, un serveur permettant le contrôle paramétrique de systèmes audio compatibles avec Open Sound Control (OSC) via des commandes en langage naturel, démontrant ainsi comment Claude peut générer des messages OSC et gérer des schémas de paramètres. De même, le framework AudioFab [17] utilise le MCP comme protocole de base pour la gestion des outils dans un écosystème de traitement audio intelligent, prenant en charge diverses techniques audio dans les domaines de la parole, du son et de la musique grâce à une architecture modulaire d'outils conteneurisés.

3.2. Résultats d'usage de Neuralyzer

À ce stade, aucune méthodologie objective d'évaluation des performances n'a encore été formalisée. Les versions intégrant Neuralyzer étant récentes, nous nous appuyons principalement sur des observations empiriques issues de scénarios de test représentatifs. Les modèles utilisés étant multilingues, les expérimentations ont néanmoins été conduites en anglais afin de limiter les variations liées à la langue. Les tests portent notamment sur deux requêtes en lien direct avec les problématiques exposées en première partie :

1. Améliore la précision fréquentielle tout en préservant la précision temporelle du sonagramme (*Improve frequency precision while preserving the temporal precision of the spectrogram*)
2. Effectue l'analyse de vingt partiels harmoniques (*Perform the analysis of twenty partial harmonics*)

Dans le cas de l'utilisation de Neuralyzer comme serveur MCP avec Claude Desktop, les résultats sont globalement très encourageants. La première requête conduit à une augmentation cohérente de la taille de fenêtre et à un ajustement corrélatif du facteur de chevauchement afin de maintenir la résolution temporelle. La seconde requête nécessite davantage d'étapes intermédiaires : création d'une piste d'estimation de la fréquence fondamentale, création de multiples pistes de partiels à partir du plug-in PM2, modification du paramètre "partialid" pour chaque piste, connexion de l'entrée adéquate et renommage des pistes. Le modèle parvient généralement à élaborer une stratégie pertinente, bien qu'il requière parfois des précisions supplémentaires de l'utilisateur ou de l'utilisatrice. Évidemment, les contraintes écologiques, éthiques et monétaires ne permettent pas d'envisager cette approche comme une solution viable pour Partiels mais elle peut être une référence pour comparer et améliorer l'inférence de modèles locaux.

> *Assistant : The frequency precision of the spectrogram has been improved while preserving temporal precision by*

11 . Pour le moment, l'opération réinitialise le modèle, ce qui prend du temps. Les versions futures devraient simplement effacer le contexte.

adjusting the parameters of the "Fast Fourier Transform" track. Specifically, the window size was increased to 50.0 ms, and the window overlapping was set to 4x. These changes enhance the frequency resolution without significantly compromising the temporal resolution.

En inférence locale, les résultats dépendent fortement du modèle, du nombre de paramètres et de la taille de la fenêtre de contexte. Les modèles de taille inférieure à 30 milliards de paramètres produisent fréquemment des erreurs d'appel ou des interprétations incorrectes de l'état du document. Une fenêtre de contexte d'environ 32768 tokens semble constituer un compromis acceptable. Les tests ont notamment été réalisés sur un MacBook Pro M4 Max doté de 36 Go de RAM et avec le modèle Qwen3.5 [1] (Qwen3-Coder-30B-A3B-Instruct-Q4_K_M) [12] qui offre des performances sensiblement supérieures à celles observées avec Qwen2.5 [15] ou Mistral 3. Pour la première requête, il ajuste correctement la taille de fenêtre et le chevauchement. Pour la seconde, des corrections automatiques ou manuelles sont souvent nécessaires, notamment en cas d'erreurs liées à la sélection des plug-ins ou à la synchronisation de l'état du document. Dans l'ensemble, avec le modèle Qwen3.5 (35B Q4), le taux de réussite des requêtes est très élevé. Il est rare qu'un appel de l'outil MCP échoue, et le temps d'exécution est de l'ordre de quelques secondes. Les erreurs sont principalement liées à un manque de contexte pouvant entraîner des interprétations erronées des opérations. Malgré ces limites, les résultats apparaissent encourageants et justifient de poursuivre les expérimentations dans cette direction. Les progrès observés entre différentes générations de modèles laissent envisager une amélioration progressive de la robustesse sans augmentation proportionnelle des ressources matérielles, même si la contrainte d'une machine relativement puissante demeure.

3.3. Perspectives de développement de Neuralyzer

Les perspectives de développement de Neuralyzer s'articulent autour de quatre axes principaux : l'enrichissement du support MCP, l'amélioration du paramétrage des modèles, l'optimisation des stratégies d'inférence et l'évolution des modalités de déploiement.

3.3.1. Enrichissement du support MCP

Un premier axe concerne l'enrichissement des données accessibles via MCP. À ce stade, le modèle interagit principalement avec la structure du document et les paramètres des analyses, mais n'a pas accès de manière détaillée aux résultats numériques produits par les moteurs d'analyse. Une telle évolution suppose cependant de maîtriser la quantité de données transmises afin d'éviter une saturation du contexte. Elle implique également de structurer ces résultats de manière exploitable par le modèle, éventuellement sous forme résumée ou agrégée.

Dans la même perspective, l'intégration d'informations plus riches sur les plug-ins constitue un levier important. Ces informations pourraient être associées directement aux plug-ins via l'IRCAM Vamp Extension (IVE) et rendues accessibles au modèle afin d'améliorer la pertinence des stratégies proposées.

L'accès à la documentation de Partiels et à des exemples d'usage structurés renforcerait également la qualité du raisonnement du modèle, en réduisant la part d'inférence implicite.

3.3.2. Amélioration du paramétrage des modèles

Un deuxième axe concerne l'amélioration du paramétrage des modèles en inférence locale. L'interface actuelle pourrait être étendue afin de permettre l'utilisation de modèles personnalisés, au-delà de ceux proposés par défaut. Le choix de la quantification, de la taille de contexte ou de paramètres tels que la température influence directement la stabilité, la précision et le degré de variabilité des réponses. Offrir un contrôle plus fin de ces paramètres permettrait d'adapter Neuralyzer à différents contextes d'usage, qu'ils soient pédagogiques, exploratoires ou strictement techniques.

3.3.3. Optimisation des stratégies d'inférence

Un troisième axe porte sur l'optimisation des stratégies d'inférence. Des approches de fine tuning spécialisées sur des corpus liés à l'analyse audionumérique pourraient améliorer la compréhension de concepts spécifiques tels que le fenêtrage, le suréchantillonnage ou l'analyse de partiels harmoniques. Par ailleurs, l'intégration de mécanismes de type Retrieval-Augmented Generation (RAG) [10], reposant sur l'indexation de la documentation de Partiels et des plug-ins, permettrait de réduire la dépendance à la mémoire conversationnelle brute et d'améliorer la pertinence des réponses tout en maîtrisant la taille du contexte transmis.

3.3.4. Évolution des modalités de déploiement

Enfin, l'évolution des modalités de déploiement constitue un enjeu structurant. L'inférence locale requiert des ressources matérielles importantes, ce qui peut restreindre certains usages. La mise en place d'un serveur d'inférence dédié, par exemple au sein de l'IRCAM, permettrait de mutualiser les ressources et de proposer un service centralisé. Une telle architecture pourrait également être adaptée à des contextes pédagogiques, notamment pour équiper une salle de classe avec un serveur local configuré pour plusieurs postes clients. Cette approche soulève toutefois des questions liées à la charge, à la maintenance, aux coûts énergétiques et aux considérations éthiques associées à l'usage de modèles de grande taille.

12. https://huggingface.co/unsloth/Qwen3-Coder-30B-A3B-Instruct-GGUF

4. CONCLUSION

Le projet Neuralyzer, bien qu'encore à un stade expérimental, apporte d'ores et déjà des éléments de réponse aux problématiques initiales et ouvre de nouvelles perspectives de recherche. Les premières expérimentations montrent que l'usage de l'inférence locale de modèles de langage peut contribuer à réduire la complexité de prise en main d'un outil d'analyse sonore et à améliorer l'expérience utilisateur, notamment en limitant les opérations répétitives, tout en respectant des contraintes éthiques, écologiques et économiques.

Les résultats obtenus et les perspectives identifiées montrent que Neuralyzer ne constitue pas un simple ajout fonctionnel, mais s'inscrit dans un chantier de recherche au croisement de l'ingénierie logicielle, de l'interaction humain–machine et de l'intelligence artificielle appliquée à l'analyse sonore.

5. REMERCIEMENTS

Je remercie Jean-Louis Giavitto, Serge Lemouton et Miller Puckette pour leurs échanges constructifs autour de l'intelligence artificielle, de l'usage de LLM et du développement informatique de façon générale. Nous remercions aussi les utilisateurs et utilisatrices du projet Partiels pour leur soutien technique et leurs retours d'expérience précieux lors des phases de développement et de test. Enfin, nous remercions Milo Guillot Le Naire pour ses retours originaux et constructifs sur les différentes versions de ce document.

6. REFERENCES

[1] Bai, J. et al. "Qwen Technical Report" *arXiv :2309.16609*, 2023.

[2] Bogaards, N. et Röbel, A. "An interface for analysis-driven sound processing" *in AES 119th Convention*, New York, USA, 2004.

[3] Bogaards, N., Roebel, A., Rodet, X. "Sound analysis and processing with AudioSculpt 2" *Proceedings of the International Computer Music Conference*, Miami, USA, 2004.

[4] Cannam, C., Landone, C., Sandler, M. "Sonic visualiser : An open source application for viewing, analysing, and annotating music audio files", *Proceedings of the ACM Multimedia 2010 International Conference*, University of Firenze, Italy, 2010.

[5] Doras, G., Teytaut, Y., Roebel, A. "A linear memory ctc-based algorithm for text-to-voice alignment of very long audio recordings" *Applied Sciences, 13(3), 1854.*, 2023

[6] Brown, T. B. et al. "Language Models are Few-Shot Learners" *arXiv :2005.14165*, 2020.

[7] Guillot, P. "Partiels - Exploring, Analyzing and Understanding Sounds" *Proceedings of the 28-th Int. Conf. on Digital Audio Effects (DAFx25)*, Ancona, Italy, 2025.

[8] Guillot, P. "Partiels : un logiciel pour analyser et explorer le son" *Proceedings of the 32nd Journees d'Informatique Musicale*, Lyon, France, 2025.

[9] Jiang, A. et al. "Mistral 7B" *arXiv :2310.06825*, 2023.

[10] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-T., Rocktäschel, T., Riedel, S., Kiela, D. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" *arXiv :2005.11401*, 2021.

[11] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G. "LLaMA : Open and Efficient Foundation Language Models" *arXiv :2302.13971*, 2023.

[12] OpenAI et al. "GPT-4 Technical Report" *arXiv :2303.08774*, 2024.

[13] Abdin, M. et al. "Phi-3 Technical Report : A Highly Capable Language Model Locally on Your Phone" *arXiv :2404.14219*, 2024.

[14] Puckette, M. "Pure Data : another integrated computer music environment" *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.

[15] Yang, A. et al. "Qwen2 Technical Report" *arXiv :2407.10671*, 2024.

[16] Fan, Y. "MCP2OSC : Parametric Control by Natural Language" *arXiv preprint arXiv :2508.10414*, 2025

[17] Zhu, C., Han, J., Xue, Q., Wang, K., Zhao, H., Zhang, Z. "AudioFab : Building A General and Intelligent Audio Factory through Tool Learning" *rXiv preprint arXiv :2512.24645*,2025

[18] Radford, Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I. "Robust Speech Recognition via Large-Scale Weak Supervision" *https ://arxiv.org/abs/2212.04356*, 2022

[19] Röbel, A. "Estimation of partial parameters for non stationary sinusoids" *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.

[20] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. "Attention Is All You Need" *arXiv :1706.03762*, 2023.

[21] Wright, M. et Freed, A. "Open SoundControl : A new protocol for communicating with sound synthesizers" *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.