

WaveNet A generative model for raw audio



Projet AS

Lê Jérémy

étudiant : 3606153

Introduction

Présentation du problème

Le projet WaveNet de DeepMind s'introduit dans une problématique Text-To-Speech (TTS). Le TTS, parallèlement à l'Automatic Speech Recognition (ASR), est un problème de mapping entre une série temporelle continue à une séquence de mots discret. L'ASR correspond à trouver le texte qui s'accorde avec un signal sonore généralement vu comme un problème de classification tandis que le TTS correspond à trouver l'onde sonore pour un texte donné (généralement vu comme un problème de régression). Il s'agit de permettre à une machine de comprendre ou générer automatiquement une commande audio.

Le processus d'un système de TTS est généralement divisé en deux parties : l'analyse et le traitement du texte, puis la synthèse du discours. On retrouve plusieurs méthodes connues de Natural Language Processing (NLP) dans l'analyse du texte comme Sentence segmentation, word segmentation, etc. Ce rapport fournit une explication détaillée du projet WaveNet, nous nous intéresserons donc uniquement à la génération du discours. Il existe aujourd'hui trois méthodes principales de génération du discours : Rule-Based, Sample-Based, Model-Based.

Etat de L'art

Dans le but de comparer les ondes sonores créées par WaveNet, DeepMind compare ses générations avec l'état de l'art : synthèse concaténative ou paramétrique.

Approche concaténative

Cette approche consiste à faire la concaténation de fragments d'onde faisant partie d'une grande base de données. On assemble de nombreux fragments pour former le discours finale. Une des limitations de cette approche est qu'il est difficile de modifier une voix sans enrichir la base de données. De plus, le discours produit est dénué d'émotion humaine.

Approche paramétrique

Suite aux limitations de l'approche concaténative, l'approche paramétrique est apparue. L'information nécessaire à la synthèse du discours sont stockés dans un modèle. Cependant, cette approche produit des résultats moins convaincants que l'approche précédente. On y retrouve les HMMs et LSTM.

WaveNet

Explication du modèle

Il est d'usage d'utiliser des RNN ou LSTM lorsque l'on travaille sur des séquences afin de capturer les dépendances temporelles. Cependant, une onde sonore à 16kHz dispose donc de 16 000 échantillons par seconde et donc utiliser ces types de réseaux serait trop lourd. Nous expliquerons par la suite comment le réseau permet d'exprimer les dépendances temporelles sans l'usage de RNN/LSTM.

WaveNet est un réseau de neurones profond permettant de générer des ondes sonores bruts. En effet, WaveNet génère séquentiellement les valeurs en chaque pas de temps. La génération de son s'inscrit principalement dans la problématique du Text-To-Speech. Cependant, on peut y trouver d'autres applications comme la reconnaissance vocale, séparation de source, ré-haussement de la parole... Avant l'apparition de WaveNet, l'état de l'art du TTS utilisait principalement deux systèmes de génération audio : paramétrique et concaténation. Aujourd'hui, WaveNet permet de générer des sons plus proches de l'être humain. Elle peut même copier les mimiques (respiration, pause...) afin d'avoir l'air plus naturel.

La principale différence entre WaveNet et les systèmes de concaténation ou paramétriques s'explique dans la génération d'une onde sonore à chaque pas de temps. Ainsi, WaveNet s'est inspiré des deux modèles de génération d'images, PixelRNN et PixelCNN, où des milliers de prédictions par image étaient effectuées.

En effet, une onde sonore peut être considéré comme un signal dense de valeur avec plus de 16 000 échantillons par secondes. Ainsi l'architecture de WaveNet est basé sur l'architecture présente dans PixelCNN.



Figure 1: A second of generated speech.

Les données

Comme décrit dans l'introduction, les données entrantes du réseau de neurones sont des ondes sonores $\mathbf{x} = \{x_1, \dots, x_T\}$. Ainsi, la probabilité de la i ème composante ne dépend que des composantes précédentes. Ainsi la probabilité d'avoir une séquence (ondes sonores) est :

$$p(\mathbf{x}) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1})$$

La valeur d'un échantillon ne dépend donc que des valeurs précédentes.

Le modèle permet de trouver les probabilités de chaque pas. Or celle-ci dépendent uniquement du son produit précédemment décrit. Dit autrement, le son suivant ne dépend que des sons émis précédent. Dans WaveNet, cette probabilité conditionnelle est modélisé par un empilement de couche de convolution.

Le modèle

Causal convolutions

WaveNet utilise principalement des « causal convolutions ». Cela permet de conserver l'ordre de prédiction de chaque pas de l'onde sonore. Voici un schéma décrivant le mécanisme d'un empilement de causal convolution :

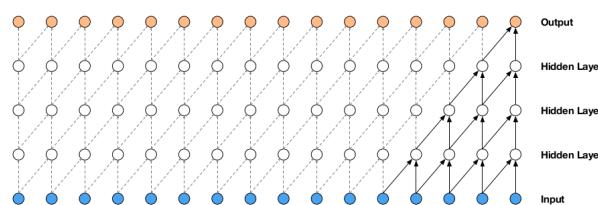
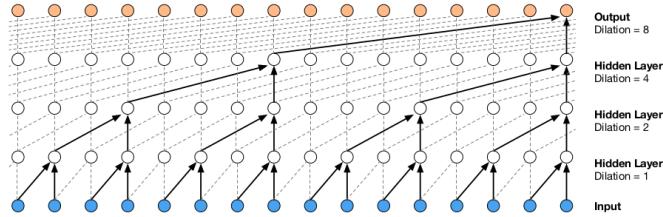


Figure 2: Visualization of a stack of causal convolutional layers.

Lors de la génération d'un son avec ce modèle, chaque prédiction est faite à chaque pas de temps puis la prédiction est renvoyé dans le réseau. Cependant, dans un empilement avec ce type de couche, le champ de réception du filtre dépend du nombre de couche utilisé. Dans la figure ci-dessus, le champ de réception est de 5 pas de temps. Ainsi pour palier ce problème sans considérablement augmenté le coût des calculs, WaveNet a proposé d'utiliser des convolutions à trous.

Dilated causal convolutions



Cette convolution consiste à sauter un nombre de pas prédefinie afin d'augmenter la portée du champ de réception. Le schéma ci-dessus présente un empilement de couche de convolution à trous pour des dilatations de 1, 2, 4, 8. Ces couches permettent deux principes :

- un accroissement exponentielle du facteur de dilatation permet une croissance exponentielle du champ de reception.
- De même, empiler de tels couches permet d'augmenter la capacité et champ de reception du modèle.

Ainsi ces convolutions permettent de capturer des dépendances temporelles sur de longue durée.

Softmax

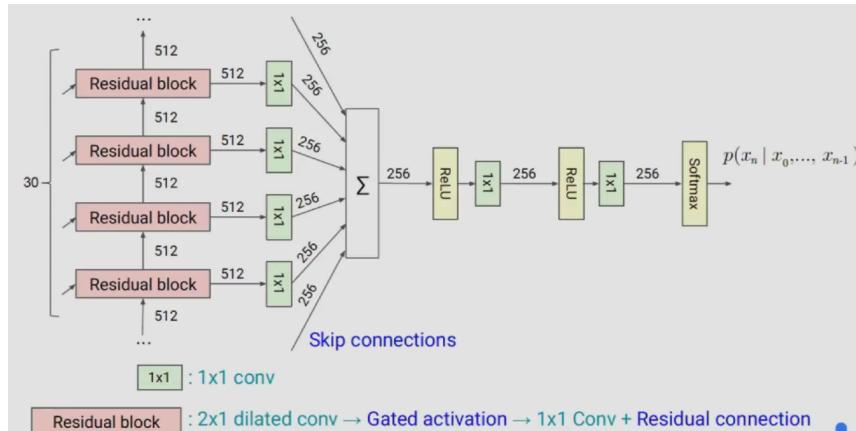
Afin de classifier la sortie du réseau, on utilise un soft max pour définir la valeur des probabilités de l'échantillon au pas de temps en question. En effet, les fichiers audio sont généralement stockées dans des valeurs 16-bit pour chaque pas de temps ce qui engendrera un softmax sur 65 536 valeur différentes...Afin de réduire le coût de calcul, on effectue au préalable une « mu-law companding transformation » pour le mettre à 256 valeurs possibles.

L'échantillon d'entraînement est donc au préalable transformé pour être dans ces 256 valeurs possible.

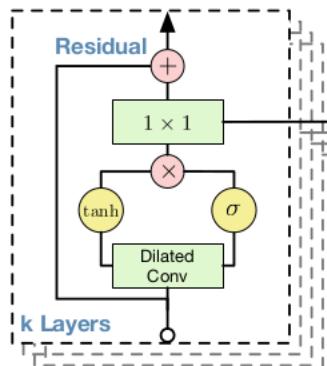
$$f(x_t) = \text{sign}(x_t) \frac{\ln(1 + \mu |x_t|)}{\ln(1 + \mu)},$$

Architecture complète

Dans le but d'intégrer une non-linéarité, les auteurs ont introduits dans l'architecture du modèle des gated convolution, des couches résiduelles et des skip-connections. De plus, comme indiqué précédemment, la sortie est calculé suite à une couche de softmax. Ainsi, le problème de synthèse d'un discours n'est plus vu comme un problème de régression mais un problème de classification. Voici l'architecture complète du réseau :



Plus en détails, on rentre dans un bloc résiduelle !



Avantages et Limitations de WaveNet

WaveNet possède un avantage sur ses concurrents dans la synthèse de discours. En effet, une valeur est générée en chaque pas de temps (échantillon par échantillon). Le modèle exprime également des dépendances non-linéaires entre les échantillons. WaveNet peut également prendre des features supplémentaires permettant de contrôler la sortie (choix du speaker, choix d'une émotion ?).

Une des limitations du modèle est que la génération d'une seconde peut être conséquente... Il faut de nombreuses minutes pour générer une seconde (16 000 échantillons). Le coût en calcul n'est donc pas négligeable...

Plan Experimentale

DeepMind

En effet, l'équipe de WaveNet a proposé plusieurs expériences :

- **Multi-Speaker Speech Generation :** A partir du corpus Voice Cloning ToolKit (VCTK), les auteurs ont entraîné leur modèle sur cette ensemble en conditionnant les échantillons avec l'identité du speaker. Ainsi lors de la génération, on pourrait générer différentes voix selon le conditionnement.
- **Text-To-Speech :** Dans leur seconde expérience, il entraîne le réseau avec ensemble de données d'une durée moyenne de 25 heures en conditionnement localement par les transcriptions.
- **Music :** Dans la troisième expérience, l'entraînement a lieu sur des échantillons de musiques classiques.

Obama et chanson d'oiseau

WaveNet étant ma première implémentation d'un réseau de neurones deep, seul la génération audio m'intéressera dans ce projet. Nous n'appliquerons pas les conditionnements locales ou globales nécessaires pour modifier la sortie générée par le modèle.

Les résultats fournies par WaveNet prouve que le modèle permet de générer de l'audio assez réaliste. Ainsi, j'ai eu l'idée d'essayer de recréer automatiquement une voix d'une célébrité. De plus, dans le but de rendre hommage au mandat de Barack Obama, j'ai donc souhaité reproduire sa voix artificiellement en utilisant un corpus de données de ses discours présidentielles. L'article décrivant le modèle WaveNet indique que l'entraînement d'un tel modèle fonctionne mieux lorsque l'on utilise plusieurs orateurs différents (l'usage d'autres orateurs sera sûrement nécessaires). Les discours de Barack Obama ont été récupéré sur le site suivant où l'on y trouve plus de 400 discours différents:

<http://www.americanrhetoric.com/barackobamaspeeches.htm>

Dans la troisième expérience décrite par l'article, le modèle parvient également à générer de la musique. Ainsi, pour imiter cette expérience, nous utiliserons un corpus de sons émis d'oiseau. De même, nous pouvons également conditionné le modèle sur l'identité de l'oiseau (race de l'oiseau). Ainsi, nous aimerais que le modèle puissent reproduire des sifflements d'oiseau. Les chants d'oiseau ont été récupéré via deux sources différentes :

- MLSP 2013 Bird Classification:
https://www.kaggle.com/c/mlsp-2013-birds/data?mlsp_contest_dataset.zip
- Birds Songs Online:
<https://data.gov.au/dataset/bird-songs-online>

Expérience réalisé

Afin de tester la performance et l'efficacité de notre implémentation du modèle WaveNet, J'ai d'abord souhaité réalisé un entraînement du modèle sur la base VCTK comme indiqué dans l'article.

Realisation de l'experience

Torch

Au cours de cette UE, nous avons eu le plaisir de découvrir une librairie de Deep Learning faite par Facebook, Torch. Elle permet facilement de créer un réseau de neurones par deux types de modules : nn.sequential et nn.gmodule (nngraph). Ces modules implémentent la méthode de back-propagation nécessaire à l 'entraînement d'un tel réseau.

Librairies utilisées

De nombreuses librairies ont été utilisées au cours de ce projet :

- **nn,nngraph** : il s'agit des librairies de neural networks. On y retrouve toutes les classes nécessaires pour construire notre modèle.
- **Penlight, csvigo, Ifs** : ces trois librairies ont été utilisé pour gérer les connexions avec le système d'exploitation (Path, creation de dossier,...).
- **Audio** : Cette librairie permet d'intégrer des fichiers audio à notre implémentation sous forme de tenseur.
- **Torchnet** : Afin de simplifier mon code, j'ai décidé d'utiliser torchnet qui facilite l'apprentissage d'un modèle.
- **Cutorch,Cunn,Cudnn** : cuda in torch.

GPU acceleration

Pour accélérer la vitesse des calculs, au lieu d'utiliser le CPU pour faire les calculs, on peut utiliser le GPU pour accélérer le traitements des données. J'ai donc utilisé cuda proposé par nvidia sur ma machine. Cependant, le niveau de performances de mon GPU n'était que de 3. J'ai donc eu pour de lourdes architectures, des problèmes de mémoires...

Causal convolution

La plupart des éléments présents dans l'architecture de WaveNet existe déjà dans les bibliothèques utilisés. Cependant les couches de causal convolutions n'existe pas... Il fallait donc les implémenter à la main. Pour créer une causal convolution, on applique au préalable un masque puis une convolution dilatée classique.

Code

Plusieurs choix ont été fait dans l'implémentation pour contrôler l'apprentissage de notre réseau. Afin d'éviter le sur-apprentissage, nous avons divisé notre ensemble VCTK et Obama en trois partie

s : train, valid et test. Lors de l'apprentissage, on calculé l'erreur sur l'ensemble de validation afin d'éviter le sur-apprentissage.

De plus, afin de réduire nos fichiers à 16 kHz, j'ai utilisé un facteur de réduction pour réduire l'échantillonnage des fichiers wav. Afin de vérifier la cohérence de ce facteur, j'ai réécouté un échantillon audio resamplé en 16kHz et celui-ci se retrouve inchangé.

Le choix du nombre de features map pour les convolutions n'ont pas été indiqué par DeepMind. J'ai donc arbitrairement utilisé les valeurs présentes dans mon implémentation (valeur que j'ai fixé par défaut).

Resultat

Les résultats des générations audio n'ont pas été très concluantes : soit le son produit était un bruit blanc soit celui-ci était un bruit très aigu...

J'ai donc pensé à sur-apprendre mon modèle afin de pouvoir reproduire les données d'apprentissage lors de la génération mais sans succès... Vous pourrez voir les différentes courbes d'apprentissage en annexes.

Je n'ai pas réussi à aboutir à un résultat fiable. En plus du temps de calcul non négligeable, je n'ai pas eu le temps de revoir mon implémentation du modèle... Afin de vérifier si le modèle peut correctement générer des séquences, je pourrais utiliser des données d'entraînements créées à la main et courte.

Une fois le model fonctionnel, je souhaiterai experimenter sur les données de Barack Obama pour reproduire sa voix. Ensuite dans un second temps, j'ai trouvé dans mes recherches qu'il était possible d'implémenter une version de WaveNet rapide nommé « Fast WaveNet ».

Annexes

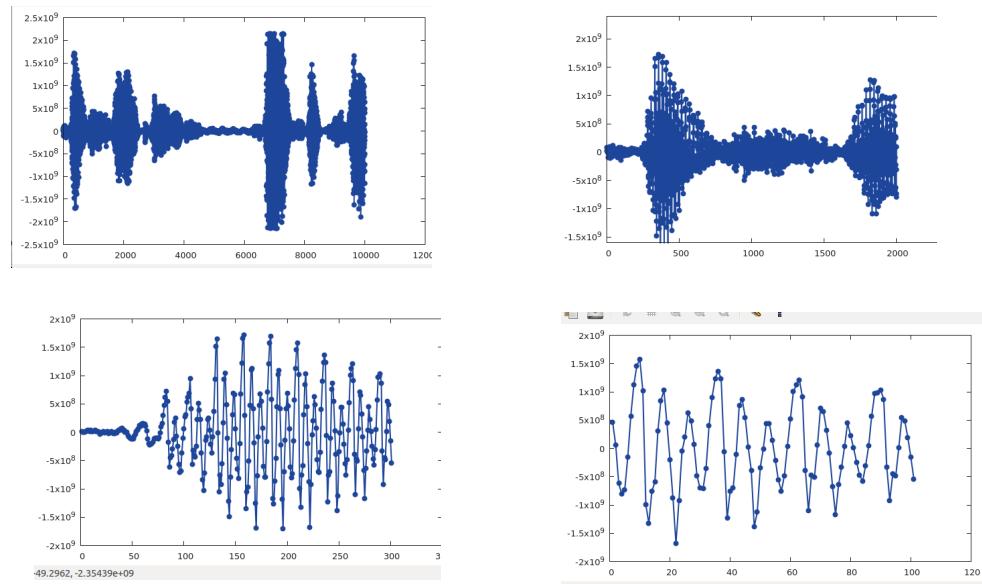


figure : Onde produit par un fichier du corpus VCTK

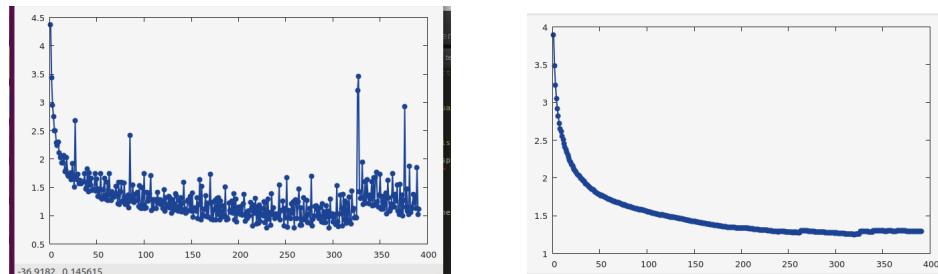


figure : courbe du loss en training (gauche) et valid (droite)