Rapport de projet Digital.e
Démêlage de représentations neuronales pour l'audio et la voix
Sous la tutelle de M. Artières

Par Killian Le Goff

Dans le cadre d'un projet de recherche du LIS en lien avec l'équipe de neurosciences de la Timone, on cherche à pouvoir altérer des données vocales. Pour ce faire, ce projet propose d'explorer l'espace latent d'un réseau de neurones, afin d'altérer certains attributs de la donnée (essentiellement l'âge, le sexe, l'accent).

Plan :

1. Notions de machine learning

La discipline du machine learning repose sur la recherche des paramètres optimaux d'un modèle donné, par rapport à un jeu de données. L'approche usuelle consiste à choisir une famille de modèles (ex : modèle linéaire entre l'entrée et la sortie, etc…), et à déterminer de façon calculatoire les paramètres qui permettent d'interpoler des données existantes, et d'extrapoler de la façon la plus exacte possible sur de nouvelles données.

Le deep learning est une sorte de machine learning, qui exploite les **réseaux de neurones**. A la différence des autres approches, le modèle n'est pas explicite (explicabliité / interprétabilité), mais possède une forte adaptabilité (pas besoin de définir une seule famille de modèle).

Les réseaux de neurones convolutionnels (**CNN**) sont particulièrement adaptés à certaines thématiques (computer vision, reconnaissance vocale, etc…). Une fois l'architecture définie, on apprend les poids des couches de convolution.

Les réseaux de neurones sont inspirés du cerveau humain,et la discipline du deep learning (ou apprentissage des représentations) fait le lien avec la neuroscience.

Autoencodeur

L'approche basique repose sur l'utilisation de réseau de neurones auto encodeurs (AE). On entraîne un couple (encodeur, décodeur) à reproduire une donnée et apprendre à la **compresser** sans trop perdre d'information. Les deux parties du réseau sont entraînées ensemble : l'encodeur apprend à générer une **représentation intermédiaire**, et le décodeur à reconstruire la donnée initiale. Lors de l'apprentissage, l'erreur utilisée pour l'apprentissage des poids est l'erreur de reconstruction : on compare la donnée reconstruite à la donnée initiale.

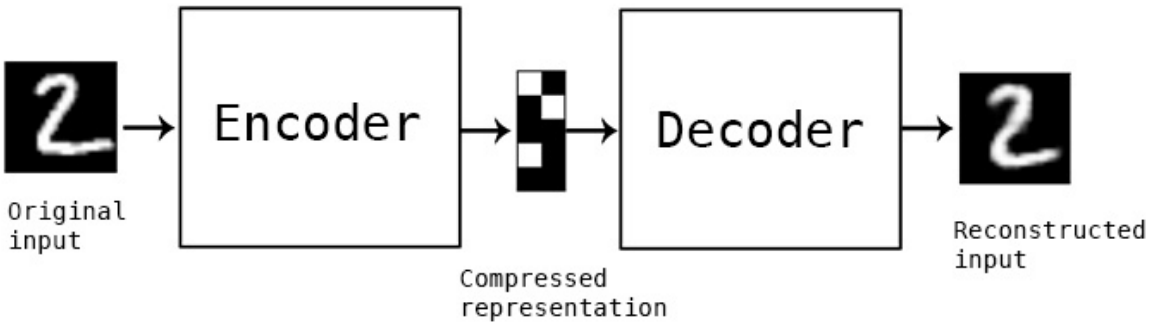


fig.1 Illustration du fonctionnement d'un autoencodeur
source : https://blog.keras.io/building-autoencoders-in-keras.html

GAN

Le GAN (**Generative Adversarial Network**) est une approche assez récente pour la génération de données. Elle consiste à entraîner 2 réseaux en concurrence, le premier est entraîné pour créer de la donnée à partir d'une variable aléatoire (**générateur**), et le second à distinguer si une donnée est vraie ou générée.

Les GANs ont un espace latent un peu différent des autoencodeurs. En effet, l'espace latent d'un GAN est situé en amont du réseau, et permet de donner un caractère aléatoire à la génération de données. Dans l'article **Unsupervised Discovery of Interpretable Directions in the GAN Latent Space (1)**, un méthode est proposée pour déterminer un l'espace latent d'un GAN dans lequel les directions sont représentatifs des attributs de nos données.

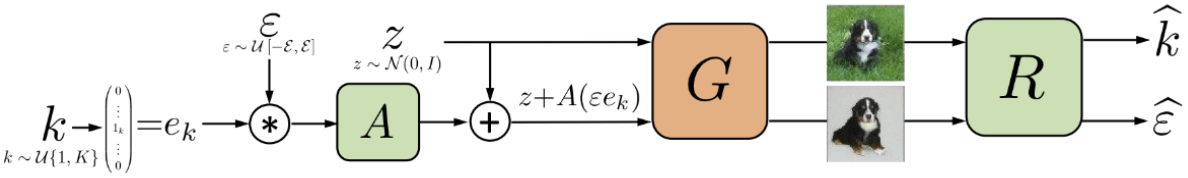


fig.2 Illustration du fonction d'un GAN dans la méthode de l'article (1)
source : https://arxiv.org/pdf/2002.03754.pdf

VAE (Variational AutoEncoder)

Le VAE est une variante de l'autoencodeur traditionnel, dans lequel on réalise des opérations arithmétiques dans l'espace des représentations. Pour le jeu de données "digit" de la librairie scikit-learn, un espace latent de dimension 2 ressemblerait à la figure ci-dessous.

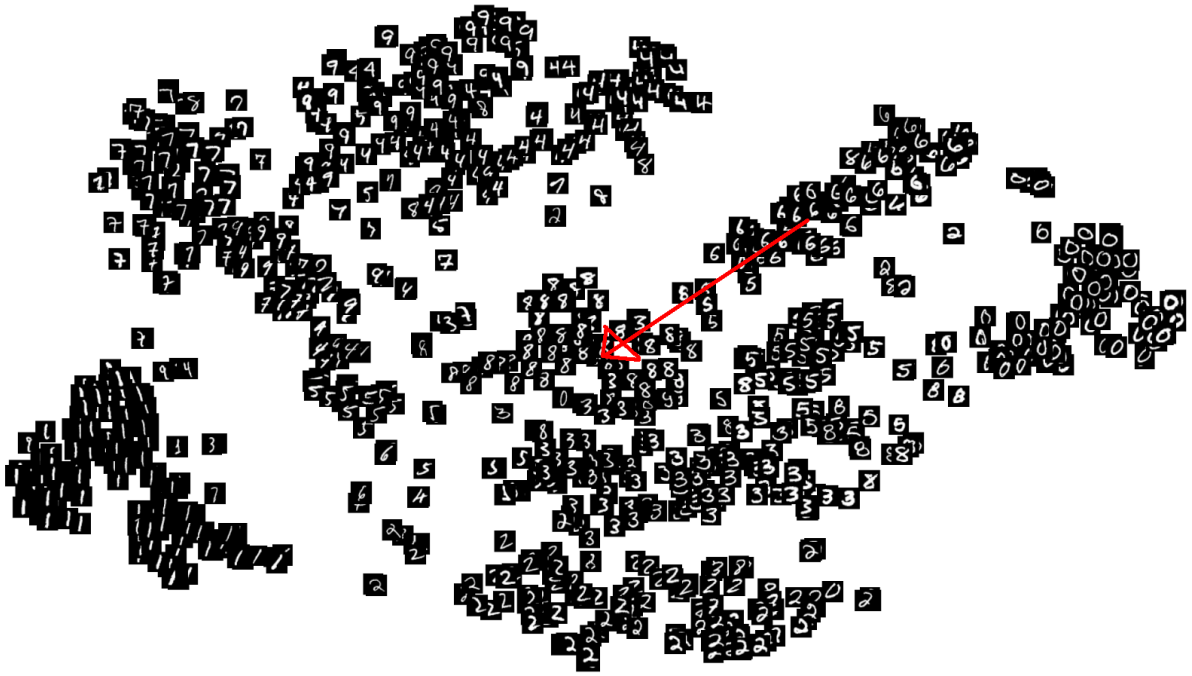


fig.3 Représentation apprise du dataset “digit” de scikit-learn
source : https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

On observe qu’il suffit de réaliser certaines translations dans cet espace, afin d’espérer modifier les caractéristiques de nos données ; ici nous pourrions envisager d’obtenir un chiffre 8 en sortie avec un 6 en entrée en réalisant la translation notée en rouge sur la figure ci-dessus.

2. Enjeux et approche du projet

Le projet se place dans le cadre d'un sujet de thèse sur la compréhension de la représentation des données vocales dans le cerveau, qui est réalisé avec l'équipe de neurosciences de la Timone et le LIS (Laboratoire d'Informatique des Systèmes).

Contexte du projet

Un projet de deuxième année a été réalisé dans le cadre de ce projet, et a consisté en l'implémentation de la méthode explicitée dans le papier **Unsupervised Discovery of Interpretable Directions in the GAN Latent Space de Andrey Voynov et Artem Babenk (1)** . Ce papier de recherche propose une méthode de découverte automatique de directions interprétables dans l'espace latent d'un GAN. Ce projet explore une approche alternative à celle-ci.

Données pour le projet

Les données sont issues du jeu de données collaboratif **Mozzila CommonVoice**. Chaque personne voulant contribuer à ce jeu de données peut s'enregistrer en train de lire une phrase, ou valider l'enregistrement proposé par d'autres personnes. Si un échantillon reçoit 2 votes de validation, il rentre dans le dataset. Certains attributs sont notés sur chaque échantillon comme la langue (les données de langues différentes sont séparées dans différents jeux de données)

fig.4 Composition du dataset Common Voice
source : https://commonvoice.mozilla.org/fr/datasets

Prétraitement

A l'obtention du jeu de données, nous avons :
-la phrase prononcée (original_sentence)

-la référence de la personne enregistrée et de l'enregistrement (id)
-les caractéristiques du locuteur (age, gender, accent)
-des fichiers audios, de l'ordre de quelques secondes d'enregistrement

La première étape du prétraitement est de séparer les fichiers audio, en alignant les phrases sur le texte pour réaliser un découpage en phonèmes. On obtient ainsi des enregistrements sur des fenêtres de 30 ms, calés sur des morceaux de phrase (champs "sentence"). La donnée a alors l'allure suivante.

age	gender	accent	original_sentence	sentence	client_id	sound_id	intervals
thirties	male	us	A person on a blue bench under a blue blanket.	a person	0a0f10d1306657ff2506f99a92144d8f01895d0a66b6fe...	common_voice_en_62600_0	([(0.0, 0.040000000000000036, 'padding', ''), ...

fig. 5 Labelling du jeu de données

La seconde étape est de calculer le **spectrogramme** de chacun des morceaux d'enregistrements. Cela consiste à calculer la transformée de Fourier discrète, à chaque date d'acquisition.

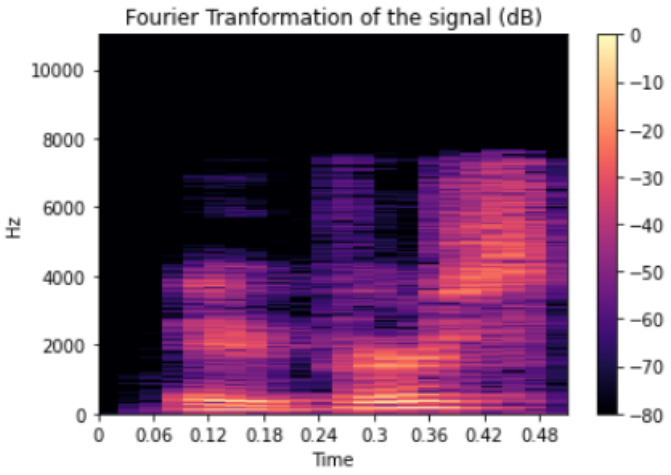


fig.6 Allure d'un spectrogramme calculé sur un enregistrement de voix

Approche du projet

Le projet se concentre sur l'utilisation d'autoencodeurs variationnels. Après avoir récupéré un modèle d'autoencodeur convolutionnel pré-entraîné sur des données de spectrogrammes, on explore l'espace latent pour comprendre la représentation latente et la distribution des différents attributs des locuteurs dans cet espace.

En parallèle, nous entraînerons des réseaux de neurones convolutionnels à classifier des spectrogrammes selon les différents attributs disponibles dans les données (age, gender, accent).

Avec ces deux éléments, nous appliquerons la méthode d'arithmétique dans l'espace latent dans le but de générer des données avec des attributs modifiés. Nous mettrons alors à l'épreuve les classifieurs d'attribut pour évaluer la qualité de ces altérations.

3. Résultats

Exploration de l'espace latent

Pour explorer l'espace latent, on fait passer les données dans l'encodeur pré-entraîné. On obtient alors des données représentées dans un espace de plus faible dimension que la donnée brute (on passe à dim = 250, soit une diminution d'un facteur 10). En cherchant à visualiser la répartition des attributs dans les plans formés par les axes de l'espace latent, on obtient notamment la figure.7. Avec cette approche, la séparabilité des attributs n'est pas atteignable.

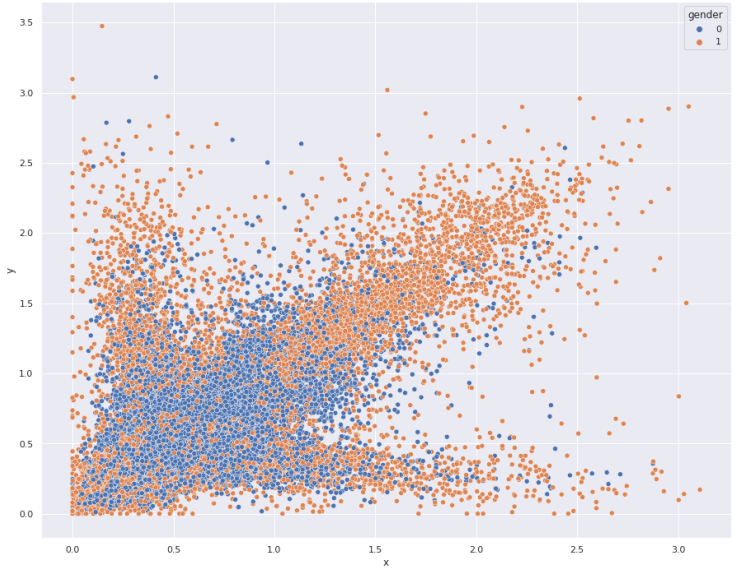


fig.8 Graphique de l'espace latent restreint à 2 axes, avec indication de l'attribut "gender"

Pour contrer la problématique du grand nombre de dimensions, on explore la piste de la réduction de dimension appliquée à l'espace latent. On utilise alors l'ACP (Analyse en Composante Principale), qui revient à trouver une base qui concentre la variance de nos données sur les premiers axes. La variance cumulée de l'ACP est donnée en figure.9. Pour obtenir une variance cumulée significative (par exemple 90%), on observe qu'il faut conserver près de la moitié des axes.

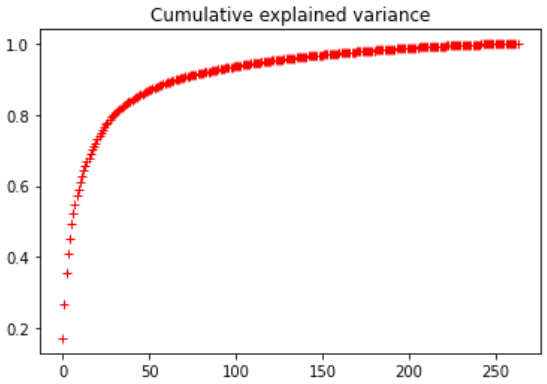


fig.9 Variance explicative cumulée de l'ACP

En traçant les points dans le plan formé par 2 axes de l'ACP, on obtient notamment la figure.10. On observe que la donnée est plus séparable que sur la même approche réalisée sur l'espace latent non réduit. On peut voir se dessiner une délimitation entre les deux modalités de la variable "gender".

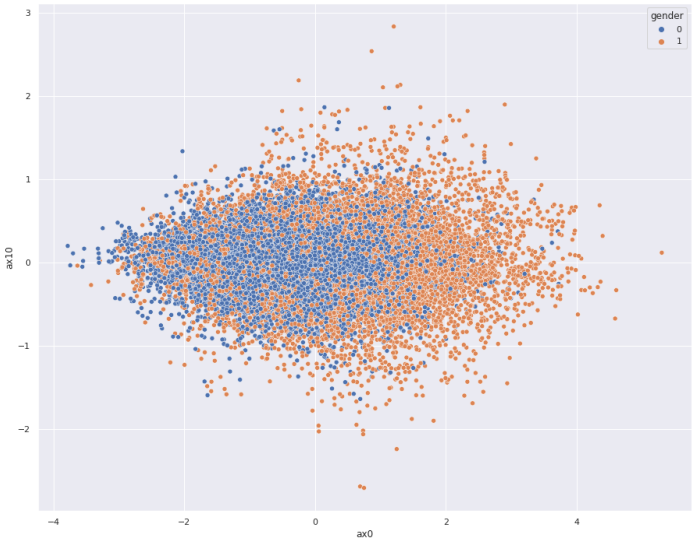


fig.10 Graphique de l'espace latent, sur les 2 axes les plus informatifs au sens de l'ACP

Pour continuer dans cette logique, on réalise la réduction de notre espace latent à 2 dimensions grâce à l'algorithme TSNE (T-distributed Stochastic Neighbour Embedding), qui propose de réduire les dimensions à 2, tout en conservant la proximité entre les points. On obtient alors le nuage de point de la figure 11. Ici encore, les différents attributs présents dans notre jeu de données ne sont pas bien séparables.

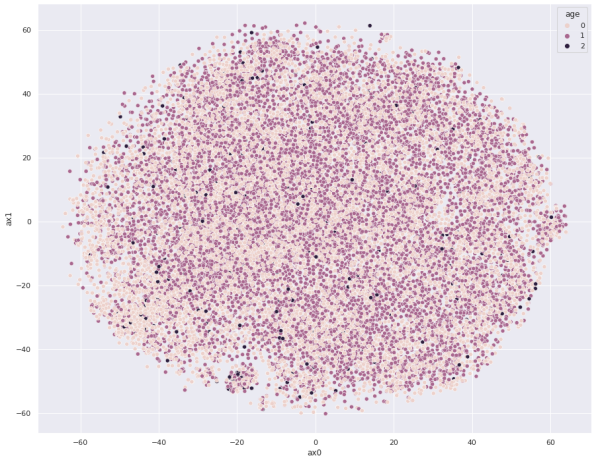
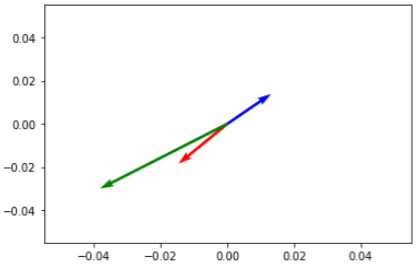


fig.11 Représentation de la réduction de dimension par TSNE

A partir du résultat de l'algorithme TSNE, on calcule les vecteurs moyens des différentes modalités de nos variables. Pour la feature "age", on obtient à titre d'exemple la figure 12. On observe que la différence d'ordre de grandeur entre le vecteur moyen d'une modalité et de sa variance est très important, ce qui est en partie dû à l'architecture d'autoencodeurs

utilisée, mais également à la méthode TSNE qui peut déformer quelques peu la donnée. Toutefois, cette approche est encourageante, car on entrevoit de la séparabilité entre les modalités des différentes variables.




```
Age 0 mean vector: [-2.0073302, -1.6413031] +/- [927.06366, 777.2335]
Age 1 mean vector: [1.7862424, 1.2266225] +/- [797.69507, 755.3502]
Age 2 mean vector: [-5.2060485, -2.6586773] +/- [848.37604, 839.395]
```

fig.12 : Vecteurs moyens des modalités de la variable âge, dans l'espace obtenu par TSNE

Trucage des données

L'objectif de cette partie est d'arriver à modifier les attributs de notre données. Par exemple, on veut être capable de vieillir une voix. Pour se faire, on reprend l'idée de vecteurs moyens proposée précédemment. On va donc calculer pour chaque modalité un vecteur moyen. Par exemple, pour la variable "gender" nous aurons le vecteur moyen de la modalité 0 et celui de la modalité 1.

Ensuite nous décalons les données de l'espace latent selon les différences entre deux modalités d'une même variable, ce qui se résume à la formule suivante:

*shifted data = latent data + eps * (end mod - start mod)*

Une fois nos données décalées selon les vecteurs moyens, nous les reconstruisons avec le décodeur pré-entraîné, puis nous mettons au défi les classifieurs de nos différents attributs.

Classifieurs d'attributs

Pour évaluer la qualité de nos opérations arithmétiques dans l'espace latent, nous étudions le résultat de la classification des attributs de la donnée modifiée. L'idée est qu'il faut déterminer si les opérations arithmétiques arrivent à duper les classifieurs de la variable qu'on propose de truquer.

L'architecture de nos classifieurs est donnée en figure.13. On utilise une seule architecture pour obtenir des modèles de classifieurs pour les variables "age" et "gender", toutefois ils seront entraînés séparément.

Les performances de chaque classifieur sont présentées en figure.14.

```python
class spectrogram_model(nn.Module):
    def __init__(self, n_out):
        """
        n_out = number of values for the chosen variable (ex : for age, n_out=3)
        """
        super(spectrogram_model, self).__init__()
        # conv layers
        self.conv1 = nn.Conv2d(1, 8, 5)
        self.conv2 = nn.Conv2d(8, 16, 5)
        self.conv3 = nn.Conv2d(16, 32, 5)

        # max pooling
        self.pool = nn.MaxPool2d(2, 2)

        # activation
        self.act1 = nn.Linear(32 * 19 * 6, 120)
        self.act2 = nn.Linear(120, 60)
        self.act3 = nn.Linear(60, n_out)

        # dropout
        self.drop = nn.Dropout(p=0.1)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x))) # shape = 10, 8, 90, 39
        x = self.drop(x)
        x = self.pool(F.relu(self.conv2(x))) # shape = 10, 16, 43, 17
        x = self.drop(x)
        x = self.pool(F.relu(self.conv3(x))) # shape = 10, 32, 19, 6
        x = self.drop(x)
        x = x.view(-1, 32 * 19 * 6)
        x = F.relu(self.act1(x))
        x = F.relu(self.act2(x))
        x = self.act3(x)
        return x
```

fig.13 Architecture des classifieurs d'attribut

gender_prediction	0	1
gender		
0	2779	400
1	219	1613

Accuracy = 0.8765

age_prediction	0	1
age		
0	213	1999
1	179	2558
2	3	72

Accuracy = 0.5516

fig.14 Matrice de confusion et précision des modèles de classification d'attributs

Qualité du trucage des données

(1) **Unsupervised Discovery of Interpretable Directions in the GAN Latent Space de Andrey Voynov et Artem Babenk**

(2) **Definition de “disentangled representation”,** https://deepai.org/machine-learning-glossary-and-terms/disentangled-representation-learning

(3) **Algorithme TSNE, article wikipedia** https://fr.wikipedia.org/wiki/Algorithme_t-SNE