

# **Ingénieur Machine Learning**

## **Projet 8**

Participez à une compétition Kaggle

SETI Breakthrough Listen – E.T. Signal Search

Pelletier Gaëtan

# Table des matières

I. Introduction.....	3
II. Présentation de la compétition.....	3
1. Projet SETI.....	3
2. Traitement des ondes.....	4
3. Dataset et métrique.....	5
a. Jeux de données.....	5
b. Metrique AUROC.....	6
III. Traitement des données.....	6
1. Prétraitement.....	6
2. Partitionnement des données en plis.....	7
3. Data augmentation.....	7
4. Dataloaders.....	8
IV. Choix du modèle.....	8
1. Comparaison de différentes solutions.....	8
2. Hyperparamétrage.....	9
3. Soumission de modèles.....	10
V. Pour aller plus loin.....	11
1. Noyau Kaggle mis à disposition de la communauté.....	11
2. Limites de cette compétition Kaggle.....	13
a. Le temps.....	13
b. Dataleakage.....	13
VI. Conclusion.....	15

# I. Introduction

Pour le dernier projet de la formation Openclassrooms Ingénieur Machine Learning, il est demandé de participer à une compétition Kaggle.

J'ai choisi la compétition intitulée : SETI Breakthrough Listen – E.T. Signal Search. Le but de ce projet est d'étudier des signaux fournis par le projet SETI, afin de développer une IA capable d'isoler des signaux intéressants pour les chercheurs. Étant donné qu'aucun signal extraterrestre ne peut être étudié, les signaux considérés comme « intéressants » sont créés par les chercheurs du centre SETI.

Au cours de ce projet, nous verrons comment classifier des signaux sonores. Puis, après avoir étudié le jeu de données disponible, nous mettrons en place un algorithme de classification d'ondes. Enfin, nous évoquerons les limites de ce projet, ainsi que les difficultés rencontrées durant cette compétition.

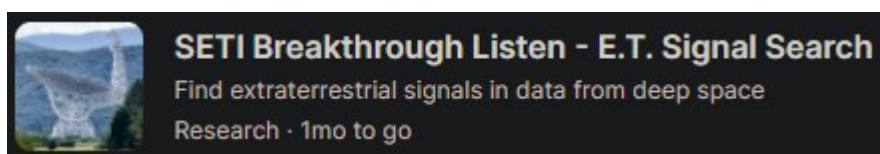
## II. Présentation de la compétition

### 1. Projet SETI

Le centre de recherche SETI (Search for Extra-Terrestrial Intelligence) regroupe différents sujets sur l'étude de l'espace et d'une vie extraterrestre. Le centre SETI s'est associé à la plateforme Kaggle afin de proposer une compétition sur l'étude de potentielles ondes extraterrestres (*illustration 1*).

Le but est d'entraîner une IA afin d'étudier des signaux et de ne garder que ceux potentiellement intéressants. Le gain de temps pour les chercheurs pourrait être significatif avec l'aide de ce genre d'IA.

Afin d'y parvenir, il faut entraîner un modèle à classifier des ondes. L'une des difficultés rencontrées par les chercheurs du centre SETI est que les techno-signatures peuvent être « parasitées » par du bruit terrestre. En effet, les stations de radio, les routeurs wifi, les téléphones, ainsi que l'électronique dans sa globalité, fournissent des ondes radio. Notre IA doit pouvoir identifier des signaux « intéressants » qui ne sont pas du bruit terrestre.

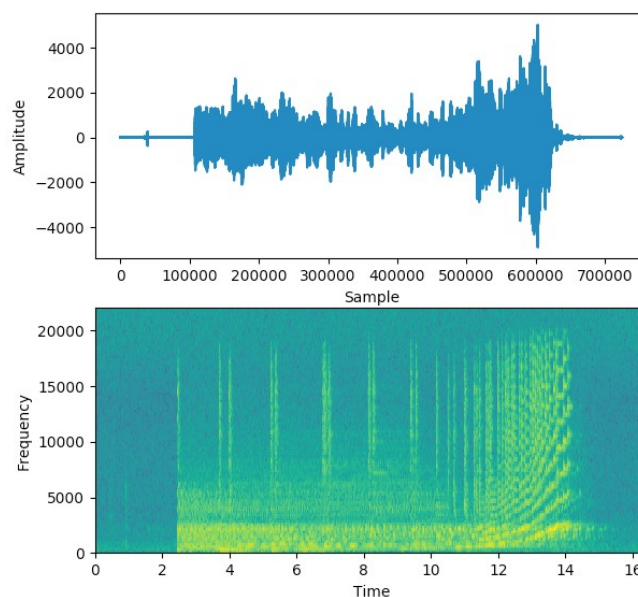


*Illustration 1: Compétition Kaggle - SETI*

## 2. Traitement des ondes

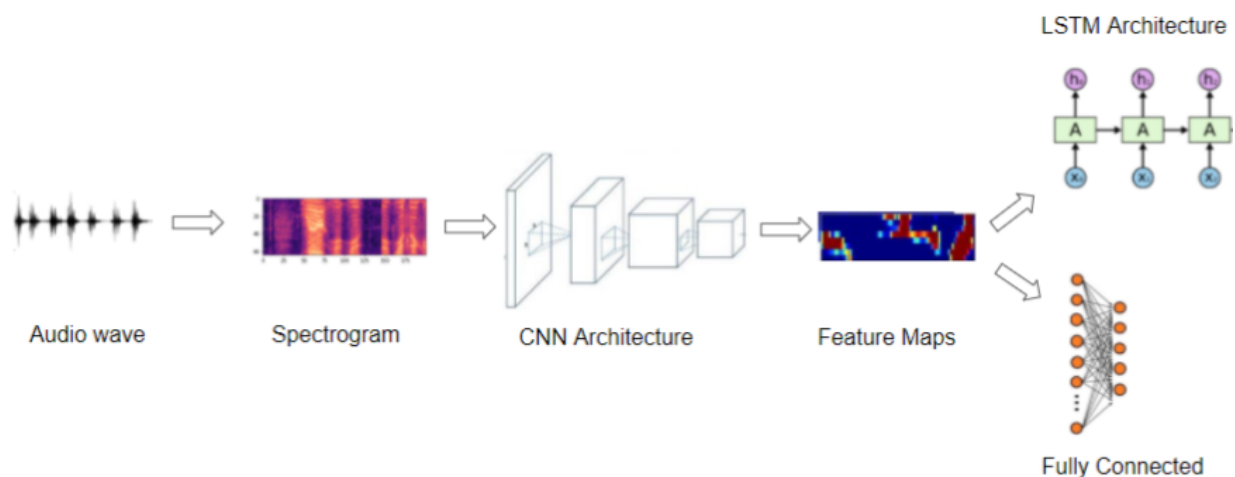
Une onde est la propagation d'un signal à travers l'espace, au cours du temps. Elle est caractérisée par son amplitude, sa fréquence, ainsi que sa forme. La digitalisation d'une onde se fait en l'échantillonnant. On parle alors de fréquence d'échantillonnage.

Afin de classifier une onde, il est possible d'utiliser des réseaux de neurones de type CNN. Cependant, il faut pouvoir transformer une onde en une « image ». Pour se faire, il est possible d'appliquer la transformation de Fourier. Ainsi, on obtient un spectrogramme représentant les fréquences d'une onde au cours du temps. De plus, les intensités de chaque fréquence sont représentées par un jeu de couleurs sur l'image obtenue (*illustration 2*).



*Illustration 2: Une onde et son spectrogramme*

À présent, il est possible d'étudier une onde selon sa problématique (*illustration 3*). Dans notre cas, pour de la classification d'ondes, nous utiliserons un CNN suivi d'un réseau entièrement connecté afin d'étudier les spectrogrammes du jeu de données SETI.



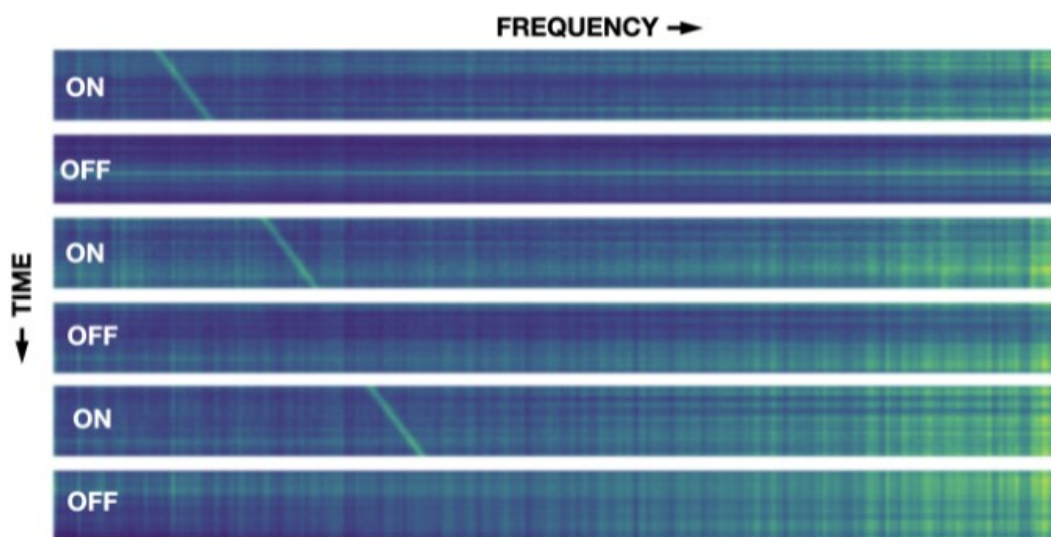
*Illustration 3: Analyses d'une onde à l'aide de réseaux de neurones*

### 3. Dataset et métrique

#### a. Jeux de données

Kaggle nous met à disposition deux jeu de données. Le premier est celui pour l'entraînement qui est constitué de spectrogrammes (*illustration 4*), ainsi que de leurs classes. La classe 1 correspond à un signal potentiellement intéressant.

Le second jeu de données ne contient que des spectrogrammes. Il est le jeu de données test à classifier. Les résultats sont à soumettre à la plateforme Kaggle afin d'obtenir un score et d'être classé sur la compétition.



*Illustration 4: Spectrogramme d'une onde de la classe 1*

Avant de continuer ce projet, il est intéressant d'étudier si la répartition des classes est équilibrée ou non. Rapidement, nous pouvons constater que nous sommes face à un jeu de données déséquilibré (*illustration 5*). Pour gérer cela, il est possible de calculer des poids qui seront indiqués à notre réseau de neurones lors de l'entraînement. Ainsi, il s'entraînera en ayant conscience du déséquilibre présent dans les données qu'il étudie. De plus, nous utiliserons la perte focale au lieu de la perte d'entropie croisée binaire.

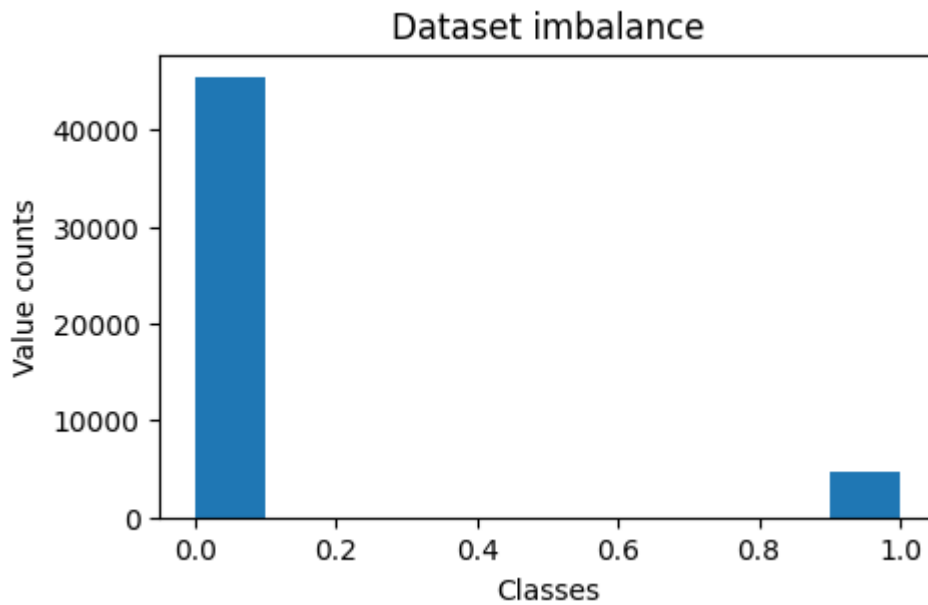


Illustration 5: Déséquilibre du jeu de données

## b. Metrique AUROC

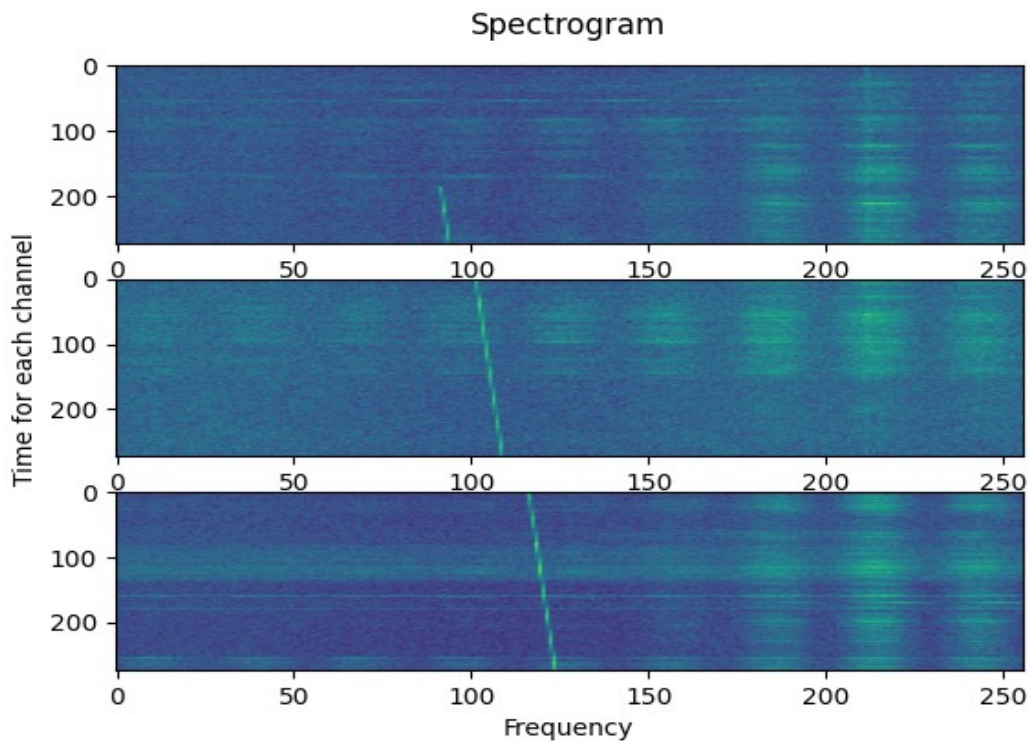
L'aire sous la courbe ROC est la métrique imposée pour cette compétition Kaggle. Une fois notre modèle entraîné, il faut fournir à la plateforme un fichier csv contenant, d'une part, les identifiants des spectrogrammes du jeu de test, d'autre part, les probabilités d'appartenance à la classe 1.

# III. Traitement des données

## 1. Prétraitement

Les spectrogrammes sont composés de 6 parties (cela forme une *cadence*). Nous utiliserons les premières, troisièmes et cinquièmes parties de ces spectrogrammes. Elles correspondent à l'observation d'une unique étoile (*on-target*). Les autres parties (*off-target*) servent à étudier d'autres étoiles et, ainsi, de justifier la présence d'un signal inconnu dans les spectrogrammes obtenus par l'observation de l'étoile principale. Nous gardons donc, pour toutes les images, seulement les spectrogrammes correspondant aux phases *on-target* (illustration 6).

En plus de supprimer une partie des spectrogrammes (non utile), nous redimensionnons ces images. De base, toutes les images ont la même taille : 273x256 pixels. Par la suite, nous imposerons une taille de 250x250 pixels à toutes les images. Ce dimensionnant peut être adapté afin de gagner en temps d'exécution, au détriment de la performance du modèle.



*Illustration 6: Spectrogramme on-target*

## 2. Partitionnement des données en plis

Afin d'améliorer la généralisation de notre modèle, nous séparons notre jeu de données en 5 plis. Nous imposons une stratification des données suivant la répartition des classes, afin de conserver la même proportion de ces dernières dans chaque pli. Nous obtenons alors, pour chaque pli, un jeu de données d'entraînement et un jeu de données de validation.

## 3. Data augmentation

Dans un premier temps, et empiriquement, nous effectuons de la data augmentation sur la couleur des spectrogrammes. De plus, nous appliquons un « flip » vertical aléatoire aux données. Bien évidemment, ces transformations ne sont appliquées qu'au jeu de données d'entraînement.

Dans un second temps, nous testons un modèle qui utilise du *mixup*, en nous inspirant d'un tutoriel rédigé sur <https://keras.io/examples/vision/mixup/> (*illustration 7*). Cette technique a été discutée et encouragée par d'autres participants de cette compétition. Succinctement, le *mixup* consiste à mélanger deux images, dans le but d'obtenir un modèle moins catégorique sur les relations entre features et labels. Cette technique améliore la généralisation d'un modèle. De plus, elle est efficace lorsqu'il est difficile de connaître quel type de data augmentation serait nécessaire pour la problématique rencontrée.

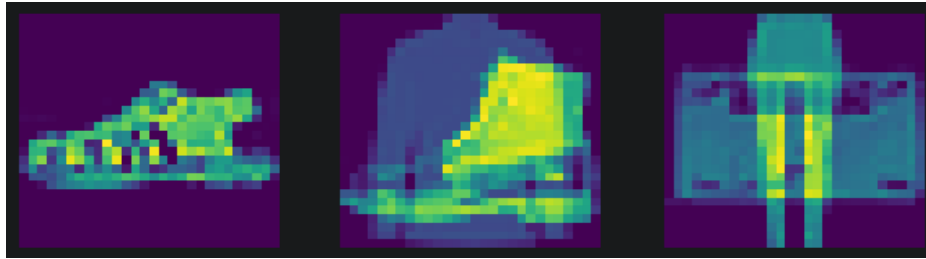


Illustration 7: Mixup avec dataset FashionMNIST

## 4. Dataloaders

Afin d'utiliser les transformations évoquées précédemment, nous devons fournir à notre modèle des *dataloaders*. Ces derniers permettent de présenter des données prêtes à l'emploi pour nos modèles, tout en effectuant, en amont, plusieurs opérations.

Ici, pour le jeu d'entraînement, nous appliquons un mélange des données. Le but de ce mélange est d'obtenir des instances du jeu d'entraînement indépendantes et distribuées de façon identique. Cela améliore la descente de gradient. Puis, nous appliquons les transformations précédentes (*on-target*, redimensionnement, data augmentation). Nous répartissons les données dans des mini-lots. Enfin, la prélecture des lots est configurée.

Pour les jeux de validation et de test, les étapes sont identiques, excepté le fait qu'aucune data augmentation n'y est appliquée. Cette technique ne concerne que le jeu d'entraînement.

Dans le cas spécifique du *mixup*, il faut définir deux *dataloaders* pour le jeu d'entraînement. Puis, nous demandons de mixer deux images provenant de ces derniers, afin de constituer un nouveau *dataloader*. Ce *dataloader* final sera alors composé d'images mixées. C'est lui qui sera utilisé pendant la phase d'entraînement d'un modèle.

## IV. Choix du modèle

### 1. Comparaison de différentes solutions

Afin d'accélérer les temps d'entraînement, nous effectuons ces comparaisons avec seulement 10 % des données du jeu global. Nous comparons 4 modèles, issus du *Transfer Learning*. Les modèles sont les suivants :

- Xception,
- EfficientNet B0



- EfficientNet B4,
- et EfficientNet B7.

Les résultats sont visibles sur l'*illustration 8*. Nous constatons que le modèle EfficientNet B7 possède la plus basse perte, ainsi que la plus haute valeur AUC, toutes deux sur le jeu de validation. Or, EfficientNet B7 est bien plus complexe à entraîner. Nous utilisons seulement 10 % des données ; avec le jeu de données total, ce temps d'entraînement risque d'être pénalisant. De ce fait, nous n'utiliserons pas EfficientNet B7.

Le modèle qui s'impose alors est EfficientNet B4. C'est ce modèle que nous allons utiliser, en tentant d'améliorer sa généralisation.

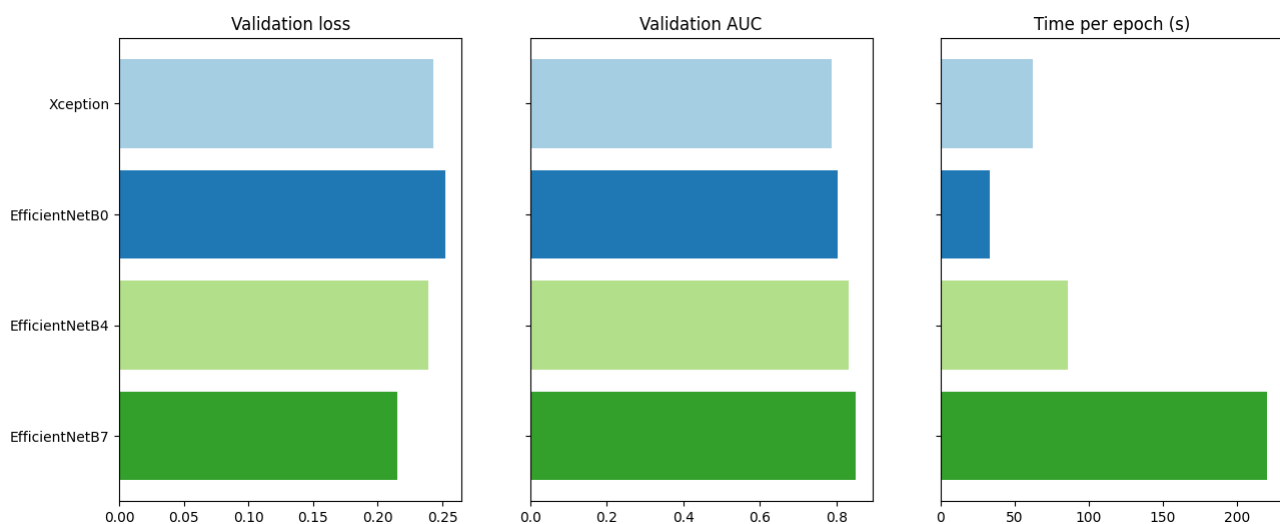


Illustration 8: Comparaison de modèles avec 10 % des données

## 2. Hyperparamétrage

Ayant retenu le modèle EfficientNet B4, nous lui ajoutons une couche de type Dropout. Grâce à la librairie *keras-tuner*, et à une optimisation bayésienne, nous déterminons les valeurs à utiliser pour le coefficient d'extinction et le taux d'apprentissage. Cela se fait toujours avec 10 % des données.

Notre modèle étant enfin prêt, nous pouvons démarrer son entraînement pour les différents plis. Nous n'oublierons pas d'utiliser un rappels de type *early stopping* afin d'éviter tout sur-apprentissage. De plus, un rappel de type *model checkpoint* est mis en place. Son but est d'effectuer des sauvegardes au fil du temps, afin de prévenir tout problème avec Kaggle (le temps d'entraînement étant relativement long, proportionnellement au quota d'heures de GPU offert par Kaggle).

### 3. Soumission de modèles

Notre première approche est d'utiliser un unique modèle. Pour le déterminer, nous comparons les résultats de notre réseau de neurones sur le jeu de validation, pour chaque pli. De plus, pour le meilleur des plis, nous entraînons un réseau de neurones utilisant la technique *mixup*.

Les résultats nous indiquent que le modèle entraîné sur le quatrième pli (*cv\_3*) et celui utilisant le *mixup*, sont les modèles présentant les plus faibles pertes (*illustration 9*). Concernant le score AUC pour le jeu de validation, les résultats sont quasiment équivalents, sauf pour les modèles *cv\_0* et avec *mixup* qui semble légèrement plus faibles.

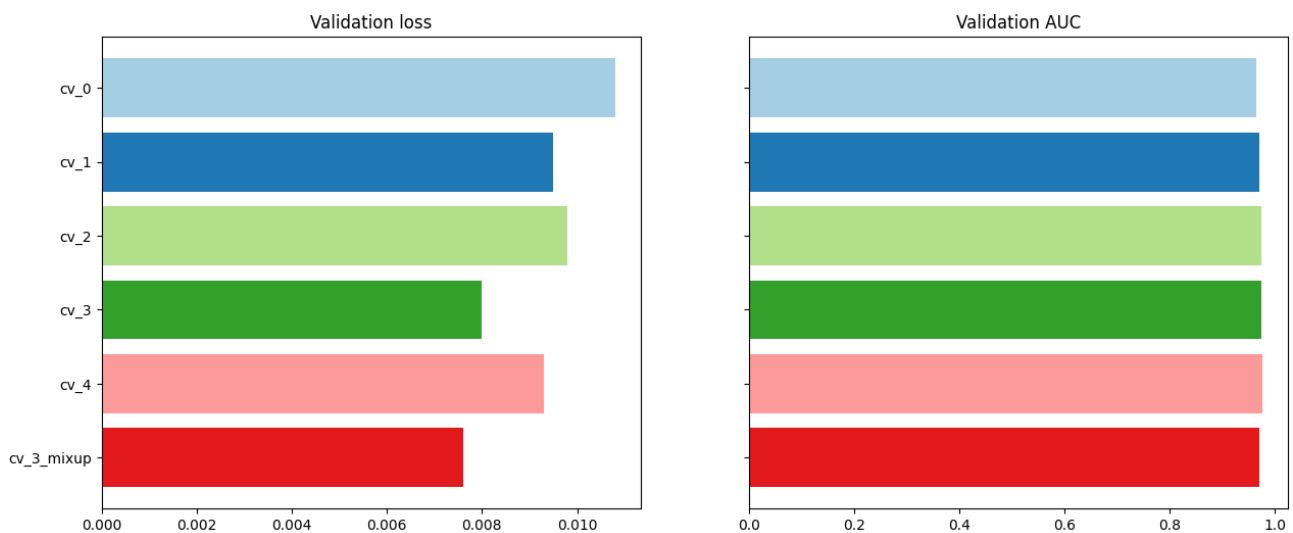
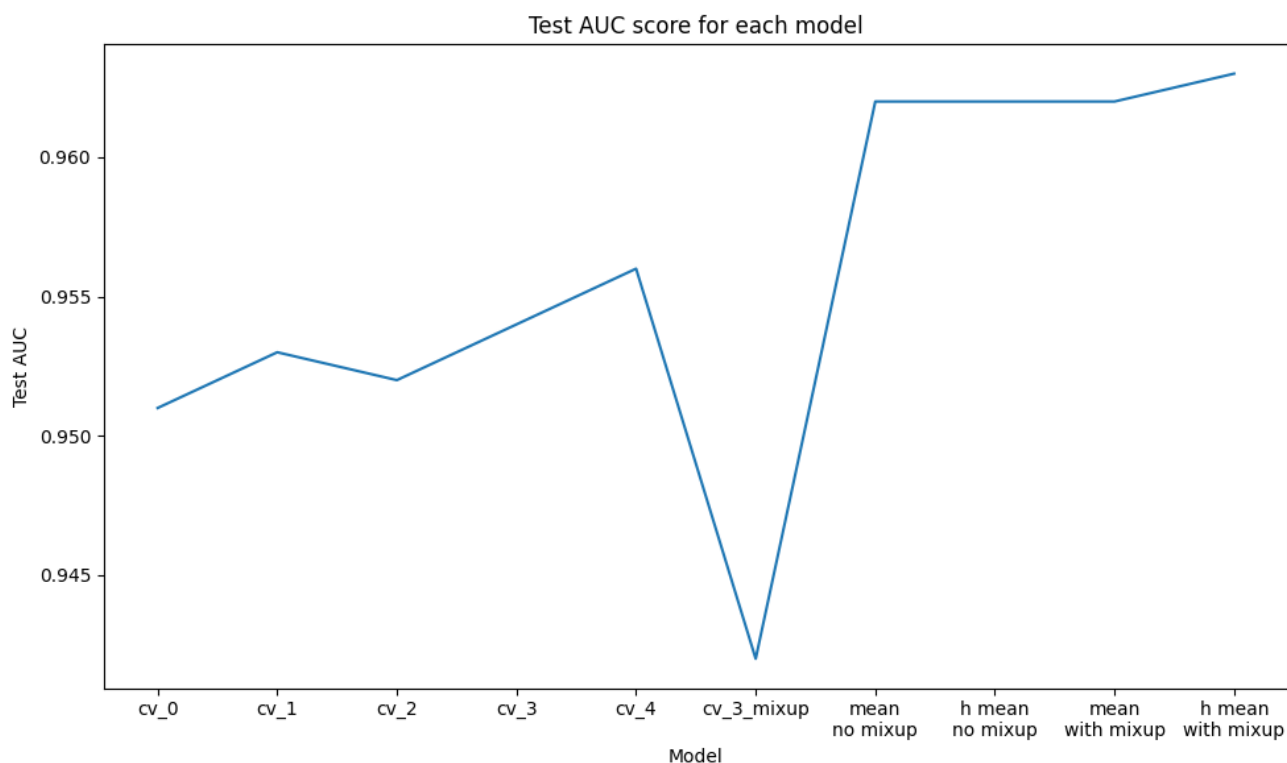


Illustration 9: Recherche du meilleur modèle sur le jeu de données complet

Une seconde approche consiste à effectuer une prédiction commune. Pour garder des probabilités de prédictions, nécessaires pour la soumission sur Kaggle, nous effectuons une moyenne des prédictions avec différents modèles. Puis, nous calculons une moyenne harmonique qui applique, pour chaque prédiction, un coefficient correspondant à la perte sur le jeu de validation divisée par la meilleure perte (i.e. la plus basse). Au total, nous effectuons quatre prédictions :

- moyenne des probabilités avec les modèles issus des plis sauf le modèle utilisant le *mixup*,
- moyenne harmonique avec les modèles issus des plis sauf le modèle utilisant le *mixup*,
- moyenne des probabilités avec tous les modèles dont celui utilisant le *mixup*,
- moyenne harmonique des probabilités avec tous les modèles dont celui utilisant le *mixup*.

En soumettant ces 10 prédictions (6 modèles uniques + 4 prédictions par des moyennes), nous obtenons de la part de Kaggle la valeur AUC pour le jeu de test (*illustration 10*). Le meilleur score provient de notre dernière méthode : moyenne harmonique des probabilités avec tous les modèles dont celui utilisant le *mixup*. Nous obtenons un score AUC sur le jeu de test de **0,963**.



*Illustration 10: Score AUC sur le jeu de test*

## V. Pour aller plus loin

### 1. Noyau Kaggle mis à disposition de la communauté

En plus de cette compétition, Openclassrooms nous demande de fournir à la communauté un noyau Kaggle (i.e. un Notebook hébergé sur Kaggle) permettant de partager un élément intéressant. En explorant les discussions, ainsi que les codes mis à disposition jour après jour, un sujet n'a que peu été exploré : l'utilisation de l'AutoML. Pour rappel, le but de l'AutoML est de proposer un pipeline permettant d'entraîner plusieurs modèles, de les optimiser et de fournir le meilleur algorithme répondant à la problématique. Notre démarche sera donc de fournir un kernel Kaggle mettant en avant l'utilisation de l'AutoML avec le jeu de données SETI.

Pour se faire, nous avons eu recours à AutoKeras. Cet AutoML reprend la même architecture que celles utilisées par les modèles de l'API Keras. Ayant utilisé Keras pour gérer les données via l'API Data, l'utilisation d'AutoKeras se fait très facilement.

À cause de la taille du jeu de données, l'entraînement de plusieurs modèles est très long par rapport au quota de GPU Kaggle disponible. Nous n'avons pas pu entraîner cet AutoML sur le jeu de données global. Au lieu de cela, nous avons utilisé 10 % du jeu de données et entraîné seulement 10 modèles automatiquement. Le meilleur modèle obtenu possède une perte d'environ 0,3722. Cela ne

semble pas être en mesure de concurrencer notre modèle EfficientNet B4 et sa perte de 0,2395 sur le jeu de validation (aussi entraîné sur 10 % des données).

Avec plus de temps à consacrer à ce projet, c'est un axe de recherche que nous aurions pu continuer d'explorer. Forcer l'AutoML à utiliser des architectures plus complexes ou augmenter drastiquement le nombre de modèles à entraîner, pourrait permettre de s'affranchir de l'utilisation de modèles « simples » (*illustration 11*).

```
Best val_loss So Far: 0.3722068667411804
Total elapsed time: 03h 50m 56s

63/63 [=====] - 4s 53ms/step - loss: 0.3722 - auc: 0.4758

Model: "model"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 250, 250, 3)]	0
cast_to_float32 (CastToFloat)	(None, 250, 250, 3)	0
normalization (Normalization)	(None, 250, 250, 3)	7
random_flip (RandomFlip)	(None, 250, 250, 3)	0
random_rotation (RandomRotat)	(None, 250, 250, 3)	0
random_contrast (RandomContr)	(None, 250, 250, 3)	0
conv2d (Conv2D)	(None, 248, 248, 128)	3584
max_pooling2d (MaxPooling2D)	(None, 124, 124, 128)	0
dropout (Dropout)	(None, 124, 124, 128)	0
flatten (Flatten)	(None, 1968128)	0
dropout_1 (Dropout)	(None, 1968128)	0
dense (Dense)	(None, 1)	1968129
classification_head_1 (Activ)	(None, 1)	0

```
=====
Total params: 1,971,720
Trainable params: 1,971,713
Non-trainable params: 7
```

*Illustration 11: AutoKeras - entraînement avec 10 % des données*

## 2. Limites de cette compétition Kaggle

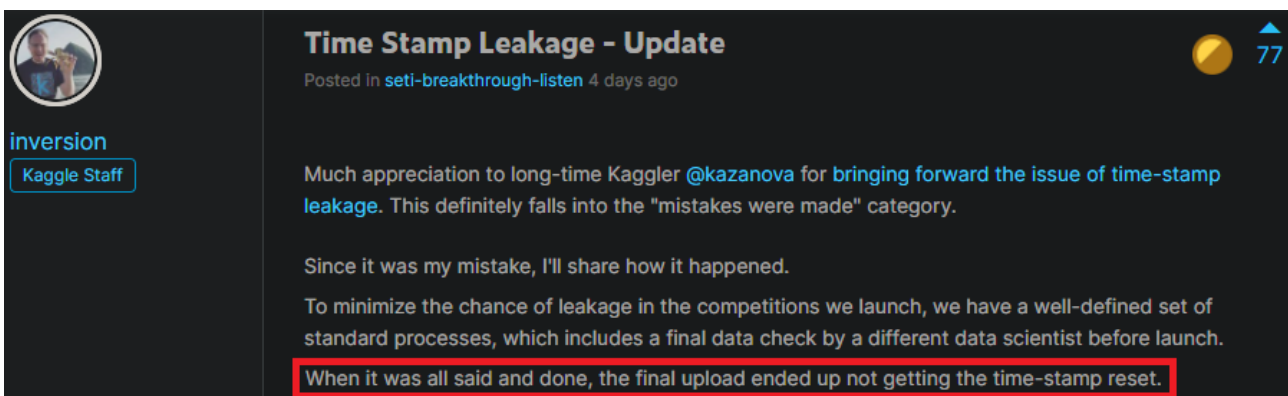
### a. Le temps

La principale limite de cette compétition a été le quota hebdomadaire de GPU offert par Kaggle. Le jeu de données étant conséquent (plus de 60 Go), il était plus agréable de travailler directement avec des kernel Kaggle. Bien que Kaggle permette d'utiliser gratuitement des GPU, il n'est possible de faire cela que 30 heures par semaine. À cause de la quantité de données, de l'utilisation de data augmentation et de modèles de plus en plus complexes à entraîner, ce quota est très rapidement atteint. Cela a donc limité certaines recherches, comme par exemple l'utilisation d'un jeu de données réduit, ou l'utilisation de *mixup* seulement pour un seul pli.

Avec plus de temps, ou l'accès à une solution GPU plus efficace, il serait possible de tester d'autres configurations de pré-traitement des données (e.g. tester différents dimensionnements d'images), de data augmentation, ou de mélanges de modèles (*stacking*, ajout d'algorithmes d'AutoML aux moyennes, etc).

### b. Dataleakage

Une seconde limite est apparue durant ces derniers jours : un problème de dataleakage a été révélé. En téléchargeant les données localement, il est possible d'obtenir la date de création de chaque image. Or, cette date n'a pas été remise à zéro (*illustration 12*). La relation entre cette dernière et la classe d'appartenance des spectrogrammes est une évidence pour certains modèles de machine learning.



**Time Stamp Leakage - Update**  
Posted in [seti-breakthrough-listen](#) 4 days ago

**inversion**  
Kaggle Staff

Much appreciation to long-time Kaggler [@kazanova](#) for bringing forward the issue of time-stamp leakage. This definitely falls into the "mistakes were made" category.

Since it was my mistake, I'll share how it happened.

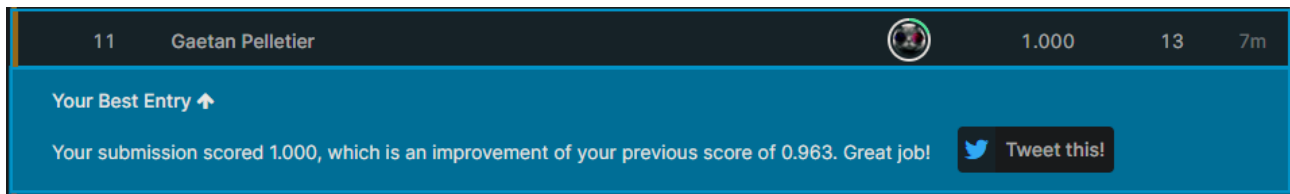
To minimize the chance of leakage in the competitions we launch, we have a well-defined set of standard processes, which includes a final data check by a different data scientist before launch.

**When it was all said and done, the final upload ended up not getting the time-stamp reset.**

Illustration 12: Dataleakage - Time Stamp Leakage

En quelques jours, les scores AUC sur le jeu de test ont tous dépassés **0,99**. Afin de tester cette faille, nous avons appliqué de l'AutoML sur les dates de créations. *AutoGluon* nous a permis

de mettre en place un modèle ensembliste de type LightGBM utilisant du *stacking*. Ce modèle obtient, sur le jeu de test, le score AUC parfait de **1,00**. Cela nous a même placé 11ème de la compétition (*illustration 13*).



*Illustration 13: Score parfait avec dataleakage (time stamp)*

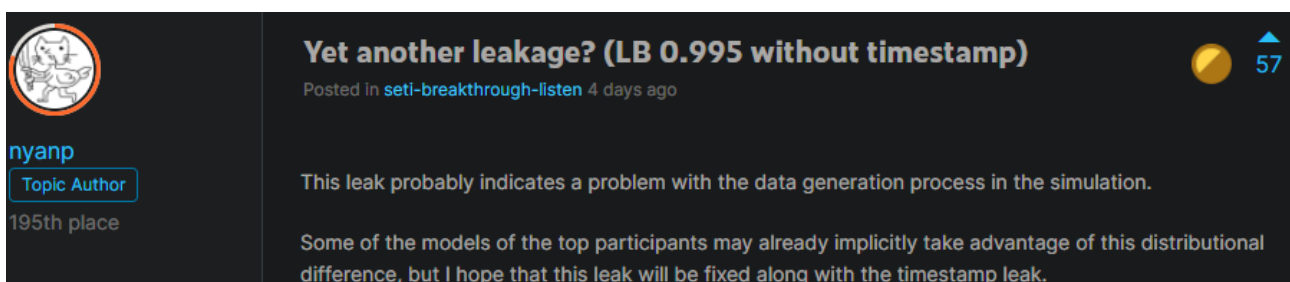
Bien évidemment, le leaderboard ne peut se maintenir. Il a été annoncé que ce dernier sera remis à zéros dans un futur proche. De plus, le jeu de test rejoindra le jeu d'entraînement et Kaggle est en attente d'un nouveau jeu de test de la part du centre de recherche SETI. Ainsi, la compétition pourra redémarrer et le leaderboard sera plus représentatif.

Cependant cette remise à zéro pourrait ne pas pouvoir résoudre tous les problèmes. En effet, certaines personnes de la communauté investiguent un problème de dataleakage présent au sein même des spectrogrammes. Il n'existe, à ce jour, aucun signal extraterrestre capté par l'être humain. De ce fait, les signaux considérés comme « intéressants » ont été créés par le centre SETI. Ce dernier nous assure que cette génération a été bien menée et qu'il n'est pas possible de mettre en place un modèle pouvant détecter cela (*illustration 14*).

After we perform the signal injections, we normalize each snippet, so you probably can't identify most of the needles just by looking for excess energy in the corresponding array. You'll likely need a more subtle algorithm that looks for patterns that appear only in the on-target observations.

*Illustration 14: Génération de spectrogrammes par le centre SETI*

Or, il semblerait qu'il est possible de retrouver cette génération au sein des spectrogrammes, puis de l'exploiter afin d'obtenir un score AUC quasiment parfait (*illustration 15*). Ce potentiel problème de dataleakage au sein des images est toujours en cours d'investigation.



*Illustration 15: Probable dataleakage au sein des spectrogrammes*

## VI. Conclusion

Au cours de cette compétition, nous sommes parvenus à classer des ondes grâce à l'utilisation des prédictions de plusieurs réseaux de neurones de type CNN. De plus, nous avons pu fournir à la communauté un tutoriel sur l'utilisation de l'AutoML avec le jeu de données SETI.

Pour aller plus loin, il faudrait consacrer du temps sur différentes techniques de data augmentation, afin d'améliorer les performances de notre IA. Puisqu'il est fort probable que cette recherche se fasse de manière empirique, nous devrions aussi trouver une solution afin de gérer le quota hebdomadaire d'utilisation de GPU offert par Kaggle. Tester d'autres façon de mettre en commun les prédictions de différents réseaux de neurones pourrait aussi permettre d'améliorer la qualité des prédictions finales.

Cependant, cette partie expérimentale ne pourra se faire qu'une fois le nouveau jeu de données disponibles ; tout en espérant que cette compétition ne souffre plus, à l'avenir, de problème de data leakage.