# Rapport de projet

# Data augmentation

## ARTICLE : LEARNING AUGMENTATION STRATEGIES FROM DATA

AUTEURS : EKIN D.CUBUK , BARRET ZOPH, DANDELION MANE, VIJAY VASUDEVAN, QUOC V.

Anis BEKRI et Abdel rahim YEHYA | Apprentissage profond

Réfèrent : Dominique FOURER
Date de la soutenance : 01 décembre 2022

# Table des matières

# Contexte

La précision des modèles d'apprentissage en profondeur (Deep Learning) dépend en grande partie de la qualité, de la quantité et de la signification contextuelle des données d'entrainement. Cependant, la rareté des données est l'un des défis les plus courants dans la construction de modèles d'apprentissage en profondeur. Dans les cas d'utilisation en production, la collecte de telles données peut s'avérer coûteuse et chronophage. Une des réponses à ce défis d'envergure est l'augmentation des données d'entrainement (Data augmentation).

L'augmentation des données est un processus qui consiste à accroître artificiellement la quantité de données en générant de nouveaux points de données à partir de données existantes. Cela comprend l'ajout de modifications mineures aux données ou l'utilisation de modèles d'apprentissage automatique pour générer de nouveaux points de données dans l'espace latent des données originales afin d'amplifier l'ensemble de données. A ne pas confondre avec les données synthétiques, ou ces dernières sont générées artificiellement et sans l'utilisation d'images du monde réel.

L'augmentation des données améliore grandement les performances des modèles de Machine Learning, elle est largement utilisée dans pratiquement toutes les applications d'apprentissage profond de pointe, telles que la détection des objets, la classification des images, la reconnaissance des images, la compréhension du langage naturel, etc. Et comme La collecte et l'étiquetage des données peuvent être des processus longs et coûteux pour les modèles d'apprentissage profond, les techniques de Data Augmentation peuvent être un excellent moyen de réduire les couts d'exploitation.

Cependant, cette méthode apporte son lot de challenges, notamment : Trouver une stratégie d'augmentation optimale pour les données. C'est ce à quoi l'article qu'on a étudié va tenter de répondre.

# ARTICLE

L'article « AutoAugment : Learning Augmentation Strategies from Data » va tenter de répondre à une des problématiques de l'augmentation des données, qui est de trouver une stratégie optimal propre à un dataset qu'on va appeler ici une "politique d'augmentions des données" (policy en anglais). Car en effet, les méthodes de data augmentation ne sont pas transférable, Même lorsque des améliorations de l'augmentation ont été trouvées pour un jeu de données particulier, elles ne sont souvent pas transposables sur d'autres ensembles de données. Ex : Le retournement horizontal des d'images est une méthode efficace d'augmentation sur CIFAR-10, mais pas sur MNIST, en raison des différentes symétries présentes dans ces ensembles de données.

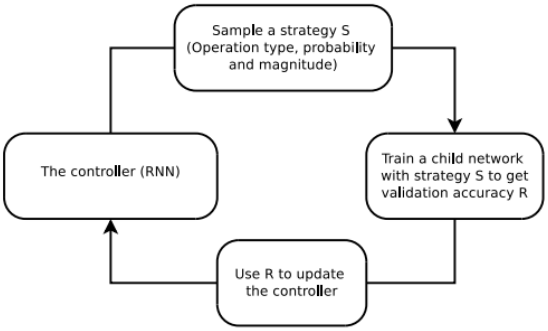
Les auteurs nous décrivent une procédure appelée **AutoAugment** pour rechercher automatiquement des politiques d'augmentation des données. Dans l'implémentation, un espace de recherche a été conçu où une politique consiste en de nombreuses sous-politiques, dont une est choisie aléatoirement pour chaque image dans chaque mini-lot.

Une sous-politique est constituée de deux opérations, chaque opération étant une fonction de traitement d'image comme la translation, la rotation ou le cisaillement, ainsi que les probabilités et les amplitudes avec lesquelles les fonctions sont appliquées. Ensuite un algorithme de recherche qui se base sur de l'apprentissage par renforcement est utilisé pour trouver la meilleure politique telle que le réseau neuronal donne la plus grande précision de validation sur un ensemble de données cible.

Selon les auteurs **AutoAugment** apporte d'excellentes améliorations dans deux cas d'utilisation :

1. S'il est appliqué directement sur le jeu de données qui nous intéresse pour trouver la meilleure politique d'augmentation (AutoAugment-direct).
2. Lorsque les politiques apprises sont transférées à de nouveaux ensembles de données.

L'algorithme de recherche (implémenté comme un contrôleur RNN) échantillonne une politique d'augmentation des données S, qui contient des informations sur l'opération de traitement d'image à utiliser. La clé de la méthode réside dans le fait que la politique S sera utilisée pour former un réseau neuronal avec une architecture fixe, dont la précision de validation R sera renvoyée pour mettre à jour le contrôleur. Comme R n'est pas différentiable, le contrôleur sera mis à jour par des méthodes de gradient de politique.




La figure ci-dessus nous montre comment le contrôleur RNN prédit une politique d'augmentation à partir de l'espace de recherche. Un réseau enfant avec une architecture fixe est entraîné jusqu'à convergence pour atteindre une précision R. La récompense R sera utilisée avec la méthode du gradient de politique pour mettre à jour le contrôleur (Apprentissage par renforcement) afin qu'il puisse générer de meilleures politiques au fil du temps.

# Travaux effectués

Dans le contexte de notre projet, nous allons tenter de reproduire les résultats présentés dans l'article de référence sur un nouveau jeu de données. Et ainsi montrer l'efficacité de l'augmentation des données dans un nouveau problème de classification.

## 1. JEUX DE DONNEES CHOISIS

### a. Objectif :

L'objectif de notre projet est de lire les sons de test fournis par le dataset et d'interpréter à quelle classe de bruit les données reçues appartiennent en élaborant deux modèles d'apprentissage profond dont un comporte de l'augmentation de données appliquées à ses données d'entrainement pour prouver l'efficacité de la méthode grâce à des résultats.

### b. Description du jeu de données :

Pour notre projet nous avons décidé de travailler sur le dataset : **Acoustic Event Dataset**. Composé de données sonores ainsi que de 28 classe de sons ( acoustic_guitar, airplane, applause, bird, car, cat, child, church_bell, crowd, dog_barking, engine, fireworks, footstep, glass_breaking, hammer, helicopter, knock, laughter, mouse_click, ocean_surf, rustle, scream, speech, squeak, tone, violin, water_tap, whistle).

Acoustic Event Dataset est composé de 2 Dossier ; un contenant des données d'entrainement (train) contenant 3831 audios classifier dans des dossiers portant le nom de leurs classes respectives, et un autre dossier de test (test) contenant 1212 audios. Tous les enregistrements audios n'ont pas forcément la même taille et ils sont nommés selon leurs classes respectives.

### c. Filtrage et importation des données :

Dans un souci de performances, nous avons décidé de travailler sur uniquement 5 classes : acoustic_guitar, applause, knock, dog, violin. Et en conséquence nous avons donc aussi filtré les enregistrements présents dans le dossier de test pour en garder seulement ceux appartenant à une des 5 classes choisis, le dossier contenant ces données test s'appellera : « track ».

Avant de commencer à construire le modèle d'apprentissage en profondeur, nous devront créer une liste d'extraction. Nous chargerons dans cette liste les données obtenus à partir des enregistrements audios des 5 classes choisis du dossier « train » et

qui seront traiter par la fonction de prétraitement (preprocess) qu'on décrira plus tard dans ce rapport.

## 2. PREPARATION DE JEU DE DONNEES

Tout d'abord, nous allons avoir besoin d'un moyen de représenter numériquement les audios, ceci sera possible en convertissant une forme d'onde brute des données audio sous forme de spectrogramme grâce aux bibliothèques de traitement audio de Tensorflow.

### a. Conversion des données en spectrogrammes :

Dans cette étape, nous allons créer la fonction pour compléter les étapes de prétraitement requises pour l'analyse audio (fonction preprocess). Nous convertirons les formes d'onde précédemment acquises grace la bibliothèque librosa sous forme de spectrogrammes. Ces signaux audio visualisés sous forme de spectrogrammes seront utilisés par notre modèle d'apprentissage en profondeur dans les prochaines étapes pour analyser et interpréter les résultats en conséquence. Dans le bloc de code ci-dessous, nous acquérons toutes les formes d'onde et calculons la transformée de Fourier à court terme des signaux avec la bibliothèque TensorFlow pour acquérir une représentation visuelle, comme indiqué dans l'image fournie ci-dessous.

```
def preprocess(file_path):
    wav , sr = librosa.load(file_path,sr=16000)
    l=len(wav)
    wav = wav[:50000]
    zero_padding = tf.zeros([50000] - tf.shape(wav), dtype=tf.float32)
    wav = tf.concat([zero_padding, wav],0)
    spectrogram = tf.signal.stft(wav, frame_length=320, frame_step=32)
    spectrogram = tf.abs(spectrogram)
    spectrogram = tf.expand_dims(spectrogram, axis=2)
    return spectrogram , l
```

Comme nos données audio n'ont pas toutes la même tailles nous avons du convenir d'une taille moyenne (50000 cycles : 50000/16000 = 3.25 secondes car nous sommes sur une fréquence de 16khz), supprimer les cycles qui dépassent dans les données de plus grande taille, et compléter avec des zéros pour les données de taille inférieure à la moyenne.

### b. Répartition des données :

Ensuite nous avons créé un dataframe pour séparés le spectrogramme de son label, et à partir de cela deux listes : X : « features » contenant le spectrogramme et y : « class » contenant le label associé.

Nous avons ensuite encodé les labels pour les avoir en vecteur de nombre en convertissant les étiquettes de classe en un vecteur d'encodage one-hot (one-hot encoding vector). La fonction « to_categorical » généré une colonne booléenne pour chaque catégorie ou classe, une seule de ces colonnes peut prendre la valeur 1 pour chaque échantillon. Ensuite nous avons mélanger ces données en utilisant les fonctionnalités intégrées de TensorFlow (shuffle) pour avoir des données hétéroclites. Et enfin nous avons séparés les données en données d'entrainement et de test à hauteur de 80% et 20% dans cet ordre.

## 3. MODELE D'APPRENTISSAGE PROFOND

Nous avons choisi un modèle CNN (Convolutional Neural Network), car après plusieurs recherches il nous a semblé que ce dernier était le plus adaptés à notre jeu de donnés audio.

### a. Couches que comporte le model :

Nous avons procédé à l'ajout des couches convolutives avec la forme respective de l'étiquette d'échantillon (1491, 257, 1) pour construire deux blocs de couches convolutives avec 16 filtres et une taille de noyau de (3, 3). La fonction d'activation ReLU est utilisée dans la construction des couches convolutionnelles et nous avons ajouter une couche de MaxPooling après chacune de ces couches pour réduire la taille des données et ainsi gagner en rapidité et précision.

Nous avons ensuite ajouter une couche flatten qui va procéder à l'aplatissement de la sortie acquise des couches convolutives pour la rendre adaptée à un traitement ultérieur. Enfin, nous pouvons ajouter les couches entièrement connectées avec la fonction d'activation softmax avec un nœud de sortie pour recevoir la sortie de classification multi-classes. L'extrait de code et le résumé du modèle construit sont présentés ci-dessous.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , Dropout ,Activation,Flatten , Conv2D , MaxPooling2D
from tensorflow.keras.optimizers import Adam
from sklearn import metrics

model = Sequential()
model.add(Conv2D(16, (3,3), activation='relu', input_shape=(1491, 257,1)))
model.add(MaxPooling2D((2,2)))
model.add(Conv2D(16, (3,3), activation='relu'))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(11, activation='softmax'))
```

## 4. METHODES D'AUGMENTATION DE DONNEES

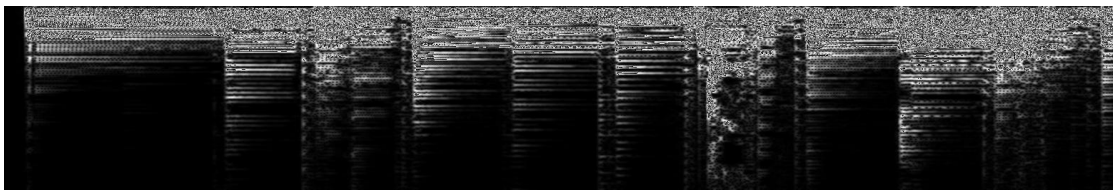
Les opérations que nous avons utilisées dans nos expériences proviennent de PIL, une bibliothèque d'images Python. Ce sont les mêmes opérations utilisées par les auteurs de l'articles : ShearX/Y, TranslateX/Y, Rotate, AutoContrast, Invert, Equalize, Solarize, Posterize, Contrast, Color, Brightness, Sharpness, Cutout, Sample Pairing. A l'exception de celles qui agissent sur le coloris de l'image, étant donné que nos données sont sous forme de spectrogrammes bicolor, utiliser de telles méthodes serait vain.

Nous avons choisi de regrouper 3 méthodes d'augmentation de données basés sur certains attributs générés de façon aléatoire dans une fonction qu'on appellera « AutoAugment » :

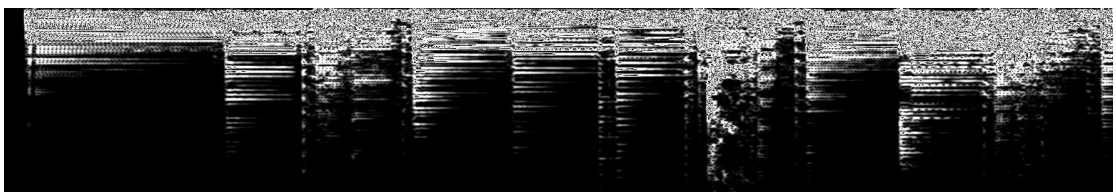
```
def autoAugment(x,n=1):
    img =Image.fromarray(np.uint8(tf.transpose(x)[0] * 255) , 'L')
    for i in (0,n):
        angle=np.random.randint(0,360)
        factor=np.random.randint(-10,10)
        enhancer = ImageEnhance.Contrast(img)d
        img_aug=img.rotate(angle)
        img_aug=enhancer.enhance(factor)
        img_aug=ImageChops.invert(img_aug)
        pix = np.array(img_aug,dtype='float32')
        pix/=255
        pix=tf.transpose(pix)
        pix=tf.expand_dims(pix, axis=2)
        #pix=tf.convert_to_tensor(pix)
    return pix
```

Voici des exemples de résultat à la suite de l'applications des 3 méthodes séparément sur une donnée :

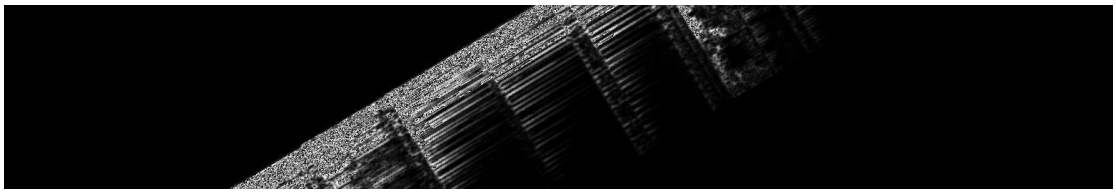
1. Echantillon de base :



2. Application de l'opération « Contrast » :



3. Application de l'opération « Rotate » :

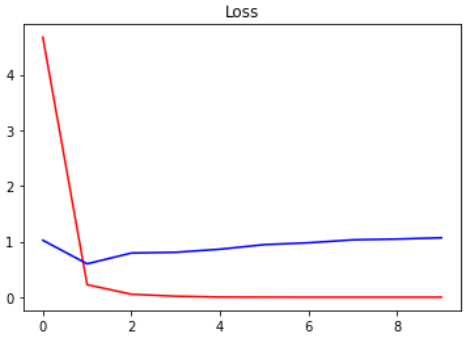


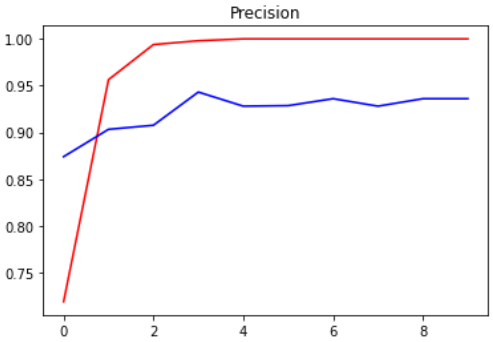
4. Application de l'opération « Invert » :

## Résultats :

Comparaison entre les deux modèles :

### 1. RESULTAT DE LA BASELINE

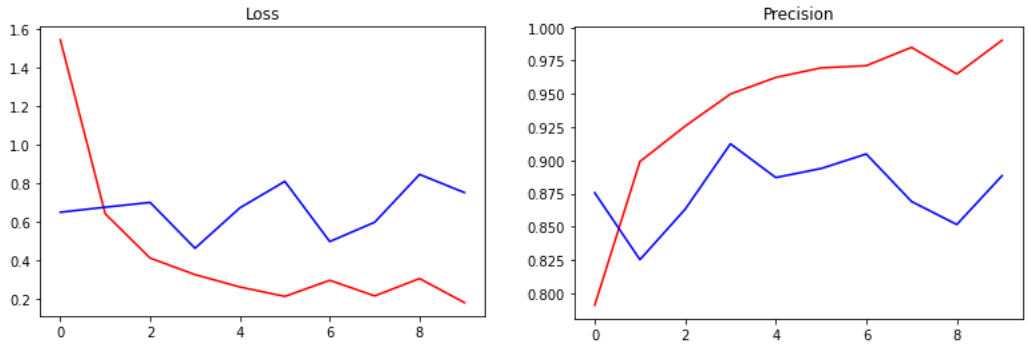



Nous avons obtenu une précision de : 0.9189189076423645

Ainsi qu'un rappel de : 0.9107142686843872

## 2. RESULTAT SUITE A L'AUGMENTATION DES DONNEES




Nous avons obtenu une précision de : 0.9196428656578064

Ainsi qu'un rappel de : 0.9196428656578064

Soit une amélioration 0.009 pour la valeur de rappel qui est rappelons la proportion de résultats positifs réels qui a été identifiée correctement. Et une amélioration de 0.001 de la précision qui quant à elle représente la proportion d'identifications positives qui était effectivement correcte.

# Conclusion :

Nous avons réussi à reproduire les résultats de l'article de référence sur notre jeu de données, et ainsi prouver l'efficacité des méthodes d'augmentation des données sur notre modèle d'apprentissage profond.

Malgré la nature de notre dataset qui n'est pas adapté aux méthodes d'augmentation des données de la bibliothèque PIL citées dans l'article, après certaines modifications apportées aux données, les méthodes se sont avérées efficace pour améliorer les résultats de notre modèle.