Un réseau de neurones frugal pour la détection bioacoustique

Matthieu CARREAU Sébastien FAUCOU Pierre-Emmanuel HLADIK Vincent LOSTANLEN
Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes, France

Résumé – Le développement de capteurs bioacoustiques permet de contribuer à l'étude de populations animales, notamment d'oiseaux, et à la préservation de la biodiversité. Afin de limiter les interventions humaines et d'améliorer l'autonomie des capteurs, nous souhaitons développer des capteurs sans batterie, effectuant le traitement de l'audio sur un microcontrôleur alimenté par une source d'énergie ambiante. Les réseaux de neurones de l'état de l'art permettant la détection de l'activité vocale d'une espèce d'oiseaux nécessitent une mémoire et une capacité de calcul incompatibles avec les ressources d'un tel microcontrôleur. Nous proposons un réseau de neurones frugal en nombre de paramètres pour la détection de l'activité vocale d'une espèce d'oiseaux. Nous montrons que l'architecture BC-ResNet permet de détecter correctement les vocalisations du rossignol philomèle avec seulement quelques milliers de paramètres.

Abstract – The development of bioacoustic sensors is contributing to study animal populations, including birds, and wildlife conservation. In order to limit human interventions and increase the autonomy of sensors, we aim to develop batteryless sensors, processing audio on a microcontroler powered by ambiant energy harvesting. State of the art neural network performing vocal activity detection for a given species require significant memory and computation ressources that are not available on a microcontroler. Therefore, we propose a tiny neural network for bird sounds detection, enabling on-device inference. We show that the BC-ResNet architecture can be trained to correctly detect Common Nightingale vocalizations with only a few thousands parameters.

1 Introduction

Les capteurs bio-acoustiques permettent d'acquérir et de traiter des données sonores issues d'espèces vivantes, qui peuvent contribuer à la préservation de la biodiversité [1]. Ces capteurs sont composés d'un microphone, d'un microcontrôleur et éventuellement d'un système de communication radio. Deux limites à l'autonomie de ces capteurs sont l'énergie et le stockage de données. Ils sont traditionnellement alimentés par une batterie qui doit être régulièrement rechargée et peut être source de pannes. De plus, la quantité de données enregistrée est rapidement importante, pouvant saturer le stockage qui doit alors être remplacé.

Une façon de limiter les interventions humaines nécessaires est de concevoir des capteurs sans batteries [2], effectuant le traitement des enregistrements directement sur le microcontrôleur. Ces capteurs récoltent de l'énergie dans leur environnement (lumière, vibrations mécaniques, ondes radios), et la stocke à court terme dans des supercondensateurs. La dépendance à une source d'énergie extérieure impose de nouvelles contraintes au système, qui doit pouvoir fonctionner de façon intermittente. Cette intermittence est due à la fois à la nature de la source d'énergie qui n'est pas constante, et à sa puissance qui peut être trop faible pour alimenter le système en continu même lorsque la source d'énergie est active. Dans ce cas, le système fonctionnera en alternant des cycles de recharge du condensateur et des cycles de travail.

La puissance de calcul, ainsi que la gestion de la mémoire, sont donc limitées par l'intermittence de l'énergie et sa quantité lorsqu'elle est disponible. Notamment, le coût énergétique important de la communication radio conduit à privilégier le traitement des données audio par le microcontrôleur, afin de limiter la taille des données à transmettre. Éviter l'envoi de données audio est également préférable pour des raisons de vie privée, les enregistrements contenant potentiellement des voix et sons d'origine humaine. L'utilisation de mémoire non volatile permet de s'assurer de la progression de l'inférence d'un réseau de neurones dans un contexte intermittent [3].

De nombreux réseaux de neurones ont été proposés en bioacoustique pour des tâches de détection ou de classification ([4], [5]), mais leur taille de plusieurs millions de paramètres les rend impossible à implémenter sur un microcontrôleur, dont la mémoire est en général limitée à quelques mégaoctets.

Cet article vise à contribuer à la création de capteurs bioacoustiques sans batteries et alimentés par panneaux solaires, permettant l'étude de l'activité vocale des oiseaux. Nous proposons un détecteur mono-espèce, basé sur un réseau de neurones à convolution (CNN pour *Convolutional Neural Network*), utilisant seulement quelques milliers de paramètres. [1]

2 Efficacité paramétrique du BC-ResNet

L'architecture BC-ResNet (pour *Broadcasted Residual Network*) est une architecture de CNN à connection résiduelle qui a été présentée pour la détection sonore de mots-clés [6]. La principale nouveauté de ce réseau de neurones est l'utilisation de *BC-Blocks*, contenant une couche convolutive fréquentielle puis une couche convolutive temporelle, qui utilisent les paramètres de façon efficace.

La convolution fréquentielle par canal (FDC pour *frequential depthwise convolution*) opère sur une représentation temps–fréquence multicanal $\boldsymbol{x}$ avec un filtre $\boldsymbol{\phi}$ de taille $L = 3$ sur la variable fréquentielle f, d'où une sortie tridimensionnelle $\mathrm{FDC}(\boldsymbol{x})[c, f, w] = \sum_l \boldsymbol{\phi}[c, l]\boldsymbol{x}[c, f - dl, w]$. De plus, la normalisation sous-spectrale (SSN pour *sub-spectral normalization*, [7]) estime la valeur moyenne ($\boldsymbol{\mu}$) et la variance ($\boldsymbol{v}$) des

[1] Répertoire contenant le code : `https://gitlab.univ-nantes.fr/carreau-m-1/bc-resnet-for-bioacoustics`.

coefficients temps–fréquence après filtrage. Ces estimateurs sont localisés selon $S = 5$ sous-bandes disjointes, d'où :

$$\mathrm{SSN}(\boldsymbol{x})[c, f, w] = \frac{\boldsymbol{x}[c, f, w] - \hat{\boldsymbol{\mu}}[c, s]}{\sqrt{\varepsilon + \hat{\boldsymbol{v}}[c, s]}} \times \boldsymbol{\gamma}[c, s] + \boldsymbol{\beta}[c, s], \tag{1}$$

où s est la sous-bande de la fréquence f et les constantes $\boldsymbol{\beta}[c, s]$ et $\boldsymbol{\gamma}[c, s]$ sont des paramètres entrainables. On en déduit un opérateur neuronal $f_2 = \mathrm{SSN} \circ \mathrm{FDC}$ dont le nombre total de paramètres est $C(L + 4S)$, où C est le nombre de canaux.

De même, la convolution temporelle par canal (TDC pour *temporal depthwise convolution*) opère sur un signal temporel multicanal $\boldsymbol{z}$ de longueur $L = 3$ avec un filtre $\boldsymbol{\phi}$, d'où une sortie bidimensionnelle $\mathrm{TDC}(\boldsymbol{z})[c, w] = \sum_l \boldsymbol{\phi}[c, l]\boldsymbol{z}[c, w - dl]$ où d est un facteur de dilatation pour la convolution. La normalisation par batch (BN pour *batch normalization*) est un cas particulier de la SSN avec une seule sous-bande ($S = 1$). On en déduit un opérateur neuronal $f_1 = \mathrm{conv}_{1\times1} \circ \mathrm{swish} \circ \mathrm{BN} \circ \mathrm{TDC}$, où swish est la fonction d'activation $x \mapsto \frac{x}{1+e^{-x}}$ et $\mathrm{conv}_{1\times1}$ est un opérateur linéaire mélangeant les différents canaux en chaque point de la représentation temps-fréquence.

Enfin, on définit avgpool et BC, qui sont des fonctions sans paramètres entrainables modifiant la dimension des tenseurs. avgpool calcule la moyenne d'une représentation temps–fréquence multicanal sur toutes les fréquences : $\mathrm{avgpool}(\boldsymbol{x})[c, w] = \frac{1}{F}\sum_{f=1}^{F} \boldsymbol{x}[c, f, w]$. BC (pour *Broadcast*) permet de passer d'un signal temporel multicanal à une représentation temps-fréquence multicanal en recopiant le signal pour chaque fréquence : $\mathrm{BC}(\boldsymbol{z})[c, f, w] = \boldsymbol{z}[c, w]$.

La combinaison de ces opérateurs permet de définir le BC-Block réalisant l'opération suivante :

$$\boldsymbol{y} = \boldsymbol{x} + f_2(\boldsymbol{x}) + \mathrm{BC}(f_1(\mathrm{avgpool}(f_2(\boldsymbol{x})))) \tag{2}$$

Oú l'on applique d'abord f_2 à une représentation temps–fréquence multicanal avant de réduire la dimension du résultat à l'aide de avgpool pour appliquer f_1 et revenir à la dimension initiale via BC. On ajoute les connexions résiduelles $\boldsymbol{x}$ et $f_2(\boldsymbol{x})$ au résultat. Ce bloc de base est répété plusieurs fois et forme la partie centrale de l'architecture BC-ResNet, comprise entre un bloc initial de convolution 2D, et un bloc final de classification.

Afin de dimensionner le réseau de neurones, un paramètre multiplicatif τ, commun à tous les blocs, permet de faire varier le nombre de canaux de chaque couche par rapport à une valeur de référence.

3 Pliage et compression pour l'inférence

Les couches BN utilisées permettent d'accélérer l'entraînement mais ne sont pas nécessairement utiles à l'inférence. En effet, elles réalisent des fonctions affines, tout comme les couches de convolution, et leur composition peut être représentée par un unique bloc de convolution. Cette étape est appelée "pliage" des couches BN ([8]) et permet de réduire le nombre de paramètres à stocker ainsi que le nombre d'opérations nécessaire à l'inférence. Elle est illustrée en figure 1 dans le cas des BN qui suivent les couches de convolution temporelles ou fréquentielles par canal. Pour ces deux cas, les opérateurs de convolution sont initialement sans biais, et il est nécessaire d'en introduire pour représenter ceux issus des couches de BN.

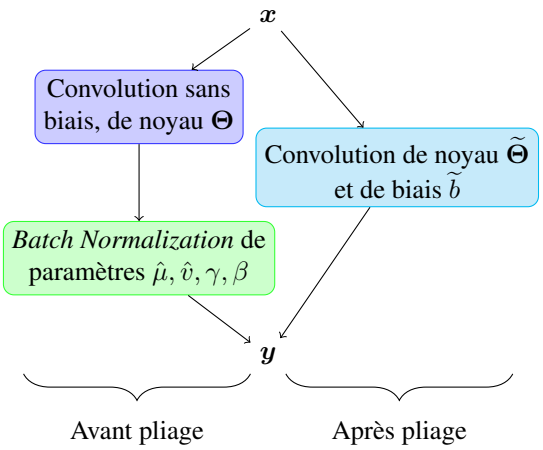

FIGURE 1 : Pliage d'une couche de normalisation avec une couches de convolution. Le chemin «après pliage» (droite) est équivalent au chemin «avant pliage», grâce à la modification des paramètres de la couche de convolution selon les équations (3) et (4) qui permettent de se passer de la couche de *Batch Normalization*.

Après pliage, la couche BN est supprimée et le noyau ϕ et le biais b de la convolution sont mis à jour en fonction des paramètres de BN $\hat{\mu}$, $\hat{v}$, γ et β. Pour des convolutions séparées par couches (*depthwise separable*) comme toutes celles du BC-ResNet, les nouveaux noyaux et biais sont donnés pour chaque couche c par :

$$\widetilde{\boldsymbol{\phi}}[c] = \frac{\gamma[c]}{\sqrt{\hat{v}[c] + \varepsilon}}\boldsymbol{\phi}[c] \tag{3}$$

$$\widetilde{b}[c] = b[c] - \frac{\hat{\mu}[c]\gamma[c]}{\sqrt{\hat{v}[c] + \varepsilon}} + \beta[c] \tag{4}$$

où $b[c] = 0$ dans le cas où la convolution initiale est sans biais. Le pliage permet alors d'économiser le stockage de $4C$ paramètre par couche BN supprimée, au coût éventuel de C paramètres supplémentaires pour le biais $\tilde{b}$.

Un tel pliage n'est en revanche pas possible pour les couches SSN. En effet, les coefficients des fonctions affines de SSN diffèrent selon les sous-bandes spectrales, ce qui ne peut pas être représenté en modifiant les paramètres d'une couche de convolution. Il est cependant possible de compresser la couche SSN et de diviser par deux son nombre de paramètres en posant :

$$\widetilde{\boldsymbol{\gamma}}[c] = \frac{\gamma[c]}{\sqrt{\hat{v}_c + \varepsilon}} \tag{5}$$

$$\widetilde{\beta}[c] = -\frac{\hat{\mu}[c]\gamma[c]}{\sqrt{\hat{v}_c + \varepsilon}} + \beta[c] \tag{6}$$

Puis en remplaçant l'équation (1) par :

$$\mathrm{SSN}(\boldsymbol{x})[c, f, w] = \boldsymbol{x}[c, f, w]\boldsymbol{\gamma}[c, s] + \boldsymbol{\beta}[c, s] \tag{7}$$

Nous économisons ainsi le stockage de $2CS$ paramètres par couche SSN.

Les nombres totaux de paramètres à stocker avant et après le pliage et la compression sont comparés en figure 2, pour différentes tailles de BC-ResNet. Le gain le plus important

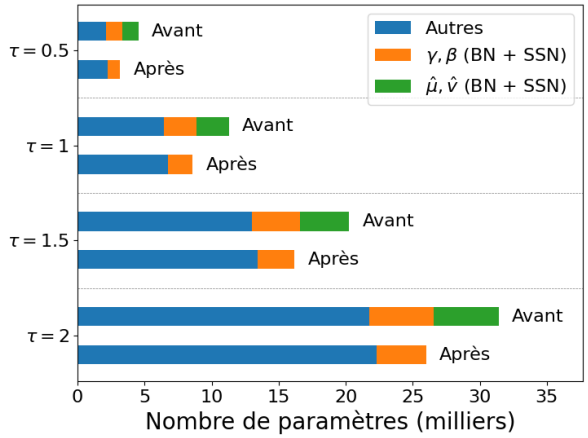

FIGURE 2 : Nombre de paramètres du BC-ResNet avant ou après pliage et compression du modèle pour l'inférence, pour différentes tailles de BC-ResNet ($\tau \in \{0.5, 1, 1.5, 2\}$). Les parties «$\gamma, \beta$ (BN + SSN)» et «$\hat{\mu}, \hat{v}$ (BN + SSN)» désignent respectivement les paramètres entraînables et les estimateurs de moyenne et variance des couches de BN et SSN. Ces derniers sont supprimés lors de la compression.

en paramètres correspond à la compression des couches SSN qui supprime les estimateurs $\hat{\mu}$ et $\hat{v}$. Cela est du au fait que le nombre total de paramètres pour les couches SSN est plus important que pour les couches BN, car des coefficients différents sont stockés sur chaque bande. On observe une légère augmentation de la partie "Autres", correspondant à l'ajout des biais dans les convolutions lors du pliage. On observe également que le gain relatif en paramètres stockés décroît avec τ, allant de 30% ($\tau = 0.5$) à 17% (pour $\tau = 2$). En effet, le nombre de paramètres économisés est proportionnel aux nombre de canaux, donc linéaire en τ, alors que le nombre total de paramètres comporte une partie quadratique en τ, due notamment aux couches $\text{conv}_{1\times1}$, ayant C^2 paramètres.

L'ordre de grandeur du nombre de paramètres du BC-ResNet est cohérent avec des implémentations existantes de réseaux de neurones sur microcontrôleurs [9].

4 Application à la détection en bioacoustique

Après avoir été introduite pour le traitement de la parole, l'architecture BC-ResNet a été adaptée et utilisée pour la classification de paysages sonores [10] ou de signaux médicaux [11], mais n'a pas été appliquée en bioacoustique à notre connaissance. Nous proposons dans cette section d'évaluer les performances de cette architecture en considérant la tâche de détection de l'activité vocale d'une espèce d'oiseaux : le rossignol philomèle *(Luscinia megarhynchos)*.

Pour cela, nous avons créé un jeu de données à partir de la base de données iNaturalist [12], qui contient des enregistrements de durées variables avec un étiquetage faible, indiquant seulement l'espèce la plus audible. Nous avons découpé ces enregistrements en sections de 5 secondes, puis utilisé le modèle *Perch* [4], pré-entraîné pour la classification multi-espèce, afin de confirmer les étiquetages faibles. Ainsi, nous conservons en tant qu'échantillons positifs les sections de 5 secondes issues d'un enregistrement étiqueté «rossignol philomèle» si nous y détectons l'activité d'un rossignol via *Perch*. De même, nous conservons en tant que négatifs les sections issues d'un enregistrement sans étiquette «rossignol philomèle», sur lesquelles l'activité d'un rossignol n'est pas détectée via *Perch*. Les sections sauvegardées sont ré-échantillonnées à 16 kHz pour être compatibles avec les capacités d'un microcontrôleur. Nous avons obtenu une base d'entraînement (respectivement, de validation) contenant 4748 (resp. 1252) clips audios de 5 secondes dont 2098 (resp. 370) positifs et 2650 (resp. 882) négatifs.

Le BC-ResNet est entraîné à la classification binaire de spectrogrammes en échelle des mels, représentant 40 bandes de fréquences réparties entre 1500 Hz et 8000 Hz. La résolution temporelle des spectrogrammes est fixée à 313, correspondant à une taille de fenêtre de 512 échantillons avec un recouvrement de 50 %.

Pour mettre en valeur l'intérêt de l'architecture BC-ResNet par rapport à un CNN ordinaire, nous avons également entraîné un autre CNN ayant un nombre de paramètres similaire et une architecture plus simple. Il s'agit d'un CNN dont la première couche de convolution est similaire au bloc initial du BC-ResNet. Elle est suivie de 5 blocs convolution-*maxpool-ReLU*, où chaque convolution est séparée en 2 opérations de la même façon que dans les Mobilenets [13] : une convolution 2D canal par canal avec un filtre de taille (3, 3), puis une convolution (1, 1) permettant de mélanger les canaux. Enfin, les activations sont agrégées par une couche *avgpool* avant de passer par 2 couches denses. Comme pour le BC-ResNet, un paramètre multiplicatif τ permet de faire varier le nombre de canaux par couche.

Pour chaque architecture, plusieurs tailles de modèles ont été comparées, en faisant varier les paramètres τ, choisis de façon à couvrir une même gamme de nombre de paramètres. Chaque modèle a été entraînée 5 fois pour chaque valeur de τ à partir de graines aléatoires différentes. La durée des entraînements a été fixée à 50 *epochs*, avec la fonction de coût d'entropie croisée binaire (*Binary Cross Entropy Loss*), un taux d'apprentissage de 0.001, divisé par 2 lorsque la fonction de coût en validation ne décroît plus. Nous avons utilisé des méthodes d'augmentation de données : mélange de forme d'ondes avec des enregistrements négatifs, translations circulaires des spectrogrammes en temps et en fréquence, et masquage de bandes spectrales ou temporelles [14].

Nous choisissons de mesurer la performance de classification par l'aire sous la courbe précision-rappel (AUPRC pour *Area Under Precision-Recall Curve*) obtenue sur le jeu de validation. Les résultats sont représentés en figure 3, représentant pour chaque modèle le minimum, la médiane et le maximum des AUPRC obtenues pour 5 répétitions de l'entraînement avec des graines aléatoires différentes. On observe que l'AUPRC est significativement plus élevée avec le BC-ResNet qu'avec le CNN ordinaire. L'AUPRC maximale obtenue avec 30000 paramètres pour le CNN ordinaire est inférieure à la plus faible obtenue par le BC-ResNet avec moins de 5000 paramètres. Le BC-ResNet atteint systématiquement un AUPRC de 0.97 pour $\tau = 2$ (soit environ 26000 paramètres). Ces résultats montrent la capacité du BC-ResNet à détecter les vocalisations du rossignol philomèle avec un faible nombre de paramètres.

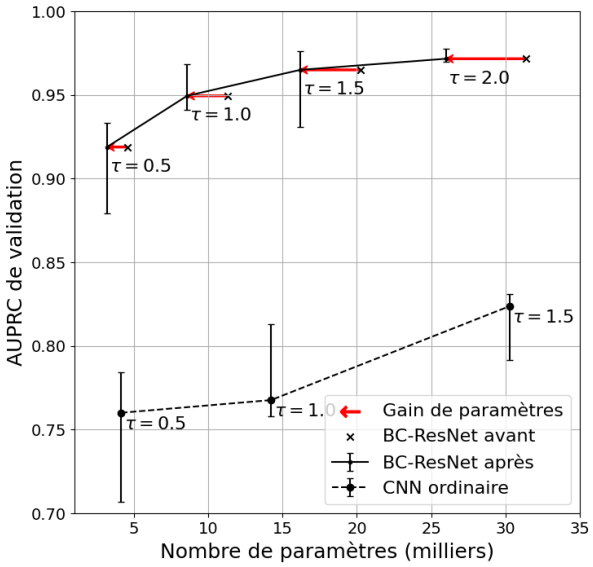

FIGURE 3 : Performances de classification mesurée par l'AUPRC (aire sous la courbe précision–rappel) en fonction de la taille du réseau de neurones, variant via le paramètre τ. Chaque réseau de neurones est entraîné cinq fois avec des graines aléatoires différentes, les barres verticales représentent les extremums obtenus et le point central donne la médiane. Les paramètres du BC-ResNet pour la ligne continue sont comptés après les étapes de pliage et compression décrites en section 3. Les flèches représentent la diminution du nombre de paramètres lors de ces étapes.

5 Conclusion

Nous avons montré que le BC-ResNet pouvait être entraîné pour effectuer la détection d'activité vocale d'une espèce d'oiseau dans des enregistrements audio. La taille réduite du réseau de neurones le rend compatible avec les capacités d'un microcontrôleur pour réaliser l'inférence sur un capteur bioacoustique déployé en environnement.

Ces résultats ouvrent des perspectives de travail pour l'implémentation de l'inférence sur microcontrôleur dans un contexte d'exécution intermittente, contrainte en mémoire et en puissance de calcul.

Remerciement

Nous remercions Xiran Zhang et Mehdi Latif.

Ce travail a été financé, totalement ou partiellement, par l'Agence Nationale de Recherche (ANR) française, à travers le projet OWL "ANR-23-IAS3-0003-01".

Références

[1] Devis TUIA et al. "Perspectives in machine learning for wildlife conservation". In : *Nature Communications* 13.1 (2022). ISSN : 2041-1723.

[2] Saad AHMED et al. "The Internet of Batteryless Things". In : *Communications of the ACM* 67.3 (2024). ISSN : 0001-0782, 1557-7317.

[3] Graham GOBIESKI, Brandon LUCIA et Nathan BECKMANN. "Intelligence Beyond the Edge : Inference on Intermittent Embedded Systems". In : *24th International Conference on Architectural Support for Programming languages and Operating Systems*. Providence RI USA : ACM, 2019. ISBN : 978-1-4503-6240-5.

[4] Burooj GHANI, Tom DENTON, Stefan KAHL et Holger KLINCK. "Global birdsong embeddings enable superior transfer learning for bioacoustic classification". In : *Scientific Reports* 13.1 (2023). ISSN : 2045-2322.

[5] Stefan KAHL, Connor M. WOOD, Maximilian EIBL et Holger KLINCK. "BirdNET : A deep learning solution for avian diversity monitoring". In : *Ecological Informatics* 61 (2021). ISSN : 15749541.

[6] Byeonggeun KIM, Simyung CHANG, Jinkyu LEE et Dooyong SUNG. "Broadcasted Residual Learning for Efficient Keyword Spotting". In : *Interspeech*. 2021.

[7] Simyung CHANG et al. "Subspectral Normalization for Neural Audio Data Processing". In : *ICASSP*. 2021.

[8] Benoit JACOB et al. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In : *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[9] Mahathir MONJUR, Yubo LUO, Zhenyu WANG et Shahriar NIRJON. "SoundSieve : Seconds-Long Audio Event Recognition on Intermittently-Powered Systems". In : *21st Annual International Conference on Mobile Systems, Applications and Services*. Helsinki, Finland : Association for Computing Machinery, 2023. ISBN : 9798400701108.

[10] Byeonggeun KIM, Seunghan YANG, Jangho KIM et Simyung CHANG. *QTI Submission to DCASE 2021 : Residual Normalization for Device-Imbalanced Acoustic Scene Classification with Efficient Design*. Rapp. tech. DCASE2021 Challenge, 2021.

[11] Jungbeom KO et al. "Efficient Cardiovascular Disease Diagnosis System for an Wearable Device based on Multi Stage BCResNet". In : *21st International SoC Design Conference (ISOCC)*. 2024.

[12] Mustafa CHASMAI, Alex SHEPARD, Subhransu MAJI et Grant VAN HORN. "The iNaturalist Sounds Dataset". In : *Advances in Neural Information Processing Systems* (2024).

[13] Andrew G. HOWARD et al. *MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv : 1704.04861 [cs.CV].

[14] Daniel S. PARK et al. "SpecAugment : A Simple Data Augmentation Method for Automatic Speech Recognition". In : *Interspeech* (2019).