Impact de perturbations internes sur l'entraînement de réseaux profonds pour la détection d'évènements sonores

David PERERA, Slim ESSID, Gaël RICHARD

LTCI, Télécom Paris
19 place Marguerite Perey, 91120 Palaiseau, France
david.perera/slim.essid/gael.richard@telecom-paris.fr

Résumé – L'apprentissage d'invariants est une méthode d'entraînement prometteuse pour les réseaux de neurones profonds, puisqu'elle permet à la fois de pallier le manque de diversité des bases de données disponibles, et de rendre les modèles entraînés plus interprétables. En pratique, l'apprentissage d'invariants passe souvent par l'utilisation d'augmentations de données et de coûts de consistance pénalisant la sensibilité d'un modèle à ces augmentations. Il n'existe cependant pas de consensus concernant la sélection de ces augmentations pour une tâche cible. Cet article étudie l'impact de plusieurs types d'augmentations sur l'entraînement d'un modèle de l'état de l'art, dans le cadre de la détection et de la classification d'évènements sonores. Nous montrons en particulier que la perturbation des représentations internes d'un réseau de neurones profond est bénéfique pour cette tâche.

Abstract – Invariance-based learning is a promising approach in deep learning. Among other benefits, it can mitigate the lack of diversity of available datasets and increase the interpretability of trained models. Practitioners often use a consistency cost penalizing the sensitivity of a model to a set of carefully selected data augmentations. However, there is no consensus about how these augmentations should be selected. This article studies the impact of several types of augmentations on the training of a model. We consider the task of sound event detection and classification for our experiments. In particular, we show that transformations operating on the internal layers of a deep neural network are beneficial for this task.

1 Introduction

L'analyse de sons ambiants est un domaine en plein essor, qui possède plusieurs applications industrielles importantes telles que la maintenance prédictive et la conception de systèmes de surveillance audio. Dans ce cadre, la détection et la classification d'évènements sonores est une tâche consistant à identifier et à délimiter temporellement des évènements sonores dans une scène acoustique complexe. L'apprentissage profond a été appliqué avec succès à cette tâche, améliorant considérablement l'état de l'art. Cependant, cette approche présente deux défauts majeurs [1]. D'abord, elle nécessite l'utilisation d'un large volume de données, et en particulier de données annotées par des opérateurs humains (processus coûteux en temps et en argent). D'autre part, les processus de décision des modèles obtenus par cette approche sont peu interprétables (ce qui pose par exemple un problème pour établir leur fiabilité).

L'augmentation artificielle d'une base de données à l'aide de transformations algorithmiques (bruitage, réverbération, égalisation...) est une technique efficace permettant de pallier le manque de données disponibles et d'exploiter des données non annotées [2]. Cette technique permet du même coup de contraindre les modèles entraînés à développer certains invariants. En sélectionnant des transformations pertinentes, on peut rendre ces modèles plus résilients aux erreurs d'annotations et plus faciles à interpréter [3]. De plus, cette méthode peut s'appliquer facilement à n'importe quel algorithme d'entraînement, et permet d'améliorer ses performances à moindre coût de développement. Ceci explique la popularité de cette approche, notamment dans le domaine de la détection d'évènements sonores. Cependant, il n'existe pas encore de consensus sur le choix des transformations à appliquer aux données.

Cet article propose une étude comparative de l'impact de plusieurs algorithmes d'augmentation artificielle sur une même architecture. Nous utilisons le cadre expérimental offert par la tâche 4 de la compétition DCASE [1] (*Detection and Classification of Acoustic Scene and Events*). En particulier, nous étudions l'avantage de transformations adverses (c'est-à-dire perturbant un modèle de façon maximale), et de transformations internes (c'est-à-dire appliquées directement sur l'espace latent d'un réseau de neurones profond). L'article s'organise de la façon suivante. La section 2 présente un bref état de l'art, puis la section 3 décrit le cadre expérimental utilisé. Enfin, la section 4 propose une discussion des résultats obtenus.

2 Etat de l'art

Deux stratégies principales existent pour pallier le manque de données d'entraînement [1] : exploiter des données supplémentaires (qui ne sont pas directement utilisables pour la tâche

1. https://dcase.community/

cible, mais qui sont plus faciles à obtenir), et guider l'apprentissage en utilisant des connaissances du domaine. Les méthodes d'apprentissage d'invariants exploitent ces deux idées, et proposent de pénaliser les variations d'un modèle f au voisinage des données d'entraînement. Plus précisément, on cherche à réduire la quantité $\|f(x) - f(\tau(x))\|$ pour un point d'entraînement x, une norme $\|\cdot\|$ et une perturbation τ donnés. Cette approche force le modèle f à développer certains invariants, qu'il est possible de définir explicitement. On peut ainsi obtenir certaines garanties sur le comportement de f, ce qui le rend plus interprétable. On retrace ici brièvement l'historique des principales évolutions dans ce domaine.

L'apprentissage par ensembles temporels (*Temporal Ensembling*, TE en abrégé) [4] compare $f(x)$ avec une moyenne des prédictions $f^{(n)}(x)$ calculées aux différentes époques n de l'entraînement. Cette méthode s'inspire de l'état de l'art en optimisation stochastique [5]. L'idée est d'accélérer la convergence du modèle f en atténuant les phénomènes stochastiques liés aux perturbations aléatoires du modèle et à la sélection aléatoire des lots d'entraînement (notés *mini-batchs* dans la suite).

L'apprentissage dirigé (*Mean Teacher*, MT en abrégé) [6] est une amélioration de TE : au lieu de calculer une moyenne des prédictions à chaque époque de l'entraînement, on maintient une moyenne des paramètres du modèle f. De plus, on actualise cette moyenne après chaque *mini-batch* au lieu de le faire après chaque époque, ce qui accélère l'entraînement du modèle.

En suivant une piste similaire, l'apprentissage adverse virtuel (*Virtual Adversarial Training*, VAT en abrégé) [7] encourage aussi l'invariance d'un modèle à de faibles perturbations en entrée. On utilise cette fois des perturbations adverses (qui maximisent la variation de f) au lieu d'utiliser des perturbations aléatoires. La couche de bruitage adverse (*Adversarial Noise Layer*) [8] généralise l'idée de perturbation adverse aux couches intermédiaires d'un modèle.

Certaines méthodes présentées ci-dessus ont déjà été appliquées à la tâche 4 de DCASE. Depuis 2019, la méthode MT est utilisée comme système de référence (ou *baseline*) pour cette tâche, et la plupart des algorithmes publiés dans le cadre de cette tâche reposent également sur cette méthode. Les auteurs de [9] proposent une étude approfondie de l'application de MT à cette tâche. La méthode VAT a également été appliquée à cette tâche, avec trois architectures différentes : un réseau convolutionnel récurrent (*Convolutional Recurrent Neural Network*, CRNN en abrégé) [10], un réseau récurrent à portes (*Gated Recurrent Neural Network*) [11] et un réseau convolutionnel récurrent à portes (*Gated Convolutional Recurrent Neural Network*) [12]. Plus récemment, [13] étudie une variante dans laquelle le bruit n'est plus adverse mais aléatoire.

Il manque cependant une étude comparative de l'impact de ces méthodes d'entraînement sur une même architecture.

3 Définition de la tâche

La tâche 4 de DCASE propose d'utiliser la base de données DESED (*Domestic Environment Sound Event Detection*), qui est composée d'enregistrements audio de 10 secondes effectués dans des environnements domestiques. DESED se divise en trois bases de données distinctes : une base de données non annotées $\mathcal{D}_u$, une base de données faiblement annotées $\mathcal{D}_w$ et une base de données synthétiques fortement annotées $\mathcal{D}_s$. Les enregistrements réels sont issus d'*AudioSet*, [2] et les enregistrements synthétiques sont construits en combinant des évènements sonores issus d'*AudioSet* avec des arrière plans sonores issus de FSD50k. [3] Il y a dix classes possibles pour l'annotation des enregistrements sonores. Chaque enregistrement de $\mathcal{D}_w$ est annoté avec l'ensemble des labels des évènements sonores qu'il contient. Pour $\mathcal{D}_s$, chaque évènement est localisé temporellement en plus d'être identifié par un label.

Deux métriques sont utilisées pour évaluer les modèles [14]. Afin de mesurer la précision temporelle des prédictions, on utilise un score F1 macro basé sur les évènements (*event-based score*), qu'on note $macro$. Afin de mesurer la précision de labellisation, on utilise le score de détection de sons polyphoniques pénalisant les erreurs de prédictions croisées (*PSDS cross-trigger*), qu'on note $psds$. Pour le calcul de ces deux métriques, nous reprenons les paramètres proposés pour l'édition 2020 de la compétition DCASE. Nous utilisons le score $macro$, calculé sur la base d'évaluation publique [4] proposée lors de cette même édition, pour évaluer et comparer les modèles.

Nous concentrons notre étude sur la *baseline* [5] de cette édition de la tâche 4. Cette *baseline* exploite une architecture CRNN, qui est communément utilisée en audio. Elle obtient des performances élevées sur la tâche. De plus, l'impact de ses divers composants a été étudié de façon exhaustive. La *baseline* prend en entrée des mel spectrogrammes avec 128 bandes Mels, construits avec une fenêtre d'analyse de taille 2048 et un pas de 255. Les signaux d'entrée sont échantillonnés à 16kHz. Le bloc convolutionnel (CNN) du CRNN est composé de 7 couches avec des filtres de taille (16, 32, 64, 128, 128, 128, 128) et un noyau de taille 3x3. Chaque convolution est suivie d'une couche de normalisation par lot (*Batch Normalization*), d'une unité linéaire à portes (*Gated Linear Unit*), d'une couche d'extinction de neurones (noté *Dropout* dans ce qui suit) avec probabilité 0.5, et d'une couche de sous-échantillonnage (*Max-pooling*). Le bloc récurrent (RNN) est composé de deux unités récurrentes à portes (*Gated Recurrent Unit*) comportant chacune 128 couches. Il est suivi par une couche linéaire avec une fonction d'activation exponentielle normalisée (*Softmax*), et d'une couche linéaire avec une fonction d'activation sigmoïde. Un filtrage médian est appliqué en post-traitement. Au total, cette architecture possède 11 millions de paramètres. Le modèle est entraîné sur 200 époques avec l'optimiseur *Adam*.

2. https://research.google.com/audioset/
3. https://annotator.freesound.org/fsd/release/FSD50K/
4. https://zenodo.org/record/3588172
5. https://github.com/turpaultn/dcase20_task4/

4 Expériences

On compare plusieurs objectifs d'entraînement. Chaque fois, l'objectif d'entraînement peut se décomposer en trois fonctions de coût : une métrique pénalisant les erreurs de prédictions $\mathcal{L}_{class}$, une métrique encourageant l'invariance à certaines perturbations $\mathcal{L}_{const}$ et une métrique de régularisation $\mathcal{L}_{reg}$.

Nous effectuons un entraînement par *mini-batch*, et nous segmentons chaque *mini-batch* en trois parties, correspondant chacune à une des bases de données $\mathcal{D}_u$, $\mathcal{D}_w$ et $\mathcal{D}_s$. Nous utilisons les proportions respectives $(1/2, 1/4, 1/4)$.

On note f la *baseline* et f_{ema} sa moyenne glissante exponentielle, recalculée à chaque itération. Si on note $f^{(n)}$ le modèle obtenu après l'itération n (notation analogue pour f_{ema}) et $\alpha^{(n)}$ un coefficient ajustable à chaque itération, alors f_{ema} est défini par

$$f_{ema}^{(n+1)} = \alpha^{(n)} f^{(n)} + (1 - \alpha^{(n)}) f_{ema}^{(n)}. \tag{1}$$

En notant x un enregistrement de la base DESED, y le label correspondant lorsqu'il existe, d un bruit gaussien, $\mathcal{L}_{BCE}$ l'entropie croisée binaire (*Binary Cross-Entropy*) et $\mathcal{L}_{MSE}$ l'erreur quadratique moyenne (*Mean Square Error*), on peut résumer l'objectif d'entraînement *baseline* de la façon suivante :

$$\mathcal{L}_{class} = \begin{cases} \mathcal{L}_{BCE}[f(x),\ y] & \text{si } (x,y) \in \mathcal{D}_w \cup \mathcal{D}_s \\ 0 & \text{sinon} \end{cases}, \tag{2}$$

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[f(x),\ f_{ema}(x+d)], \tag{3}$$

$$\mathcal{L}_{reg} = 0. \tag{4}$$

Au cours des expériences réalisées, nous conservons la métrique de classification utilisée par la *baseline*. Nous utilisons une régularisation $\mathcal{L}_2$ sur l'ensemble des paramètres du modèle, que des expériences préliminaires ont prouvé nécessaire pour obtenir des performances comparables à celles de la *baseline*.

Le coût de consistance le plus simple que nous ayons considéré est la distance $\mathcal{L}_2$ entre une prédiction $f(x)$ et une prédiction perturbée $f(x+d)$ (avec d un bruit gaussien) :

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[f(x),\ f(x+d)]. \tag{5}$$

Nous avons également considéré un coût de consistance adverse,

$$d = \nabla_d\, \mathcal{L}_{MSE}[f(x),\ f(x+d)]\,|_{d=0}, \tag{6}$$

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[f(x),\ f(x+d)], \tag{7}$$

une variante de VAT,

$$d = argmax_{\|d\| \leq \epsilon}\, \mathcal{L}_{MSE}[f(x),\ f(x+d)], \tag{8}$$

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[f(x),\ f(x+d)], \tag{9}$$

ainsi qu'un coût de consistance par interpolation linéaire (*Mixup*), calculé à partir d'un second échantillon x' de la base de données DESED,

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[Mixup(f(x), f(x')),\ f(Mixup(x, x'))]. \tag{10}$$

Toutes ces déformations peuvent s'appliquer au niveau des représentations internes du modèle. Nous faisons le choix d'appliquer des déformations à la sortie du bloc CNN et à la sortie du bloc RNN de la *baseline*.

Enfin, nous expérimentons deux variantes supplémentaires. La première consiste à combiner VAT et MT (dans ce cas, nous n'utilisons pas de régularisation $\mathcal{L}_2$) :

$$d = argmax_{\|d\| \leq \epsilon}\, \mathcal{L}_{MSE}[f(x),\ f(x+d)], \tag{11}$$

$$\mathcal{L}_{const} = \mathcal{L}_{MSE}[f(x),\ f_{ema}(x+d)], \tag{12}$$

$$\mathcal{L}_{reg} = 0. \tag{13}$$

La seconde consiste à appliquer le coût de consistance uniquement aux données non annotées.

5 Résultats

Pour chaque méthode d'entraînement, nous sélectionnons les meilleurs hyperparamètres et nous ne retenons que le meilleur modèle obtenu au cours de l'entraînement. Les différentes combinaisons étudiées et les scores obtenus sont résumés dans le tableau 1.

TABLE 1 – Comparaison des méthodes d'entraînement. Les meilleurs scores obtenus (valeurs plus élevées) sont indiqués en gras, et les scores dépassant la *baseline* sont soulignés.

Nom	Augmentation	Emplacement		Données		Scores	
		Entrée	Interne	Toutes	$\mathcal{D}_u$	macro	psds
class	Aucun			x		0.291	0.500
noise	Aléatoire	x		x		**0.397**	0.534
deep noise	Aléatoire		x	x		0.388	**0.565**
deep adv	Adverse		x	x		0.374	0.559
vat	VAT	x		x		0.358	0.539
deep vat	VAT		x	x		0.362	0.542
deep mixup	*Mixup*		x	x		0.351	0.507
baseline	MT	x		x		0.381	0.552
deep vat mt	MT + VAT	x		x		0.311	0.461
vat in	VAT	x			x	0.337	0.540

Si on s'intéresse à la métrique $macro$, on observe d'abord que l'objectif MT peut être remplacé avantageusement par un coût de régularisation $\mathcal{L}_2$ et un coût de consistance pénalisant la sensibilité au bruit et à la perturbation *Dropout*. Des expériences supplémentaires ont d'ailleurs montré que cet avantage se conserve même lorsqu'on diminue la quantité de données utilisées pour l'entraînement.

Cet algorithme d'apprentissage est une simplification de l'approche MT. En effet, le modèle f_{ema} n'étant plus nécessaire, on peut diviser par deux le nombre de paramètres maintenus en mémoire pendant l'entraînement. Cette méthode permet également de diminuer la durée d'une époque, puisque l'actualisation des poids de f_{ema} n'est plus nécessaire, et que la norme $\mathcal{L}_2$ est calculée implicitement par certains optimiseurs (c'est le cas de l'implémentation de *Adam* avec *Pytorch*, que nous utilisons ici). Finalement, cette méthode mène à une simplification conceptuelle : il est plus facile d'analyser l'impact individuel des objectifs d'entraînement (classification, régularisation et consistance), et d'ajuster finement leur contribution relative durant l'entraînement (ce qui n'est pas possible avec la méthode MT, qui combine implicitement régularisation et

consistance). Ces avantages viennent cependant au prix d'un hyperparamètre supplémentaire à ajuster.

Si on s'intéresse à la métrique $psds$, on observe cette fois que le bruitage des représentations internes de f permet d'atteindre les meilleures performances. Cette méthode d'entraînement offre d'ailleurs un bon compromis entre score $macro$ (diminuant légèrement de 0.397 à 0.388 par rapport à un bruitage en entrée seulement) et score $psds$ (augmentant nettement de 0.534 à 0.565). On pourrait expliquer ce résultat de la façon suivante. Les auteurs de [2] suggèrent que bruiter l'entrée d'un classifieur permet d'éloigner ses frontières de décisions des données d'entraînement, et donc d'améliorer ses capacités de généralisation. En suivant cette hypothèse, ajouter du bruit au niveau des couches internes de f améliore ses capacités en tant que classifieur. Ceci se traduit par une amélioration du score $psds$, plus sensible à la précision de labellisation. En revanche, le score $macro$, plus sensible à la précision temporelle, ne change que très peu.

Malgré ces premiers résultats encourageants, les expériences que nous avons réalisées avec des perturbations plus complexes (bruitage adverse, *Mixup*) n'ont pas permis d'obtenir d'améliorations par rapport à la méthode MT. Il n'y a qu'une seule exception : le bruitage interne adverse améliore légèrement le score $psds$. On peut d'ailleurs y voir une confirmation de notre hypothèse sur l'effet du bruitage interne. L'échec des perturbations adverses semble indiquer que les invariants développés avec ces méthodes ne sont pas utiles à la détection et à la classification d'évènements sonores. Ceci motive l'étude de transformations spécifiques au domaine audio (égalisation, réverbération...), plus pertinentes pour cette tâche.

6 Conclusion

Cet article analyse l'impact relatif de différents coûts de consistance sur l'entraînement du système de référence de la tache 4 de la compétition DCASE. Nous proposons une variante simple de la méthode MT, qui permet d'améliorer ses performances. Nous nous sommes limités ici à l'étude d'augmentations indépendantes du type de données utilisées. Les résultats des expériences réalisées suggèrent cependant que ces augmentations ne sont pas alignées avec notre tâche cible, la détection et la classification d'évènements sonores. C'est à une future étude de clarifier cette question et d'examiner l'impact d'augmentations spécifiques au domaine audio. Avec l'accroissement de la dimension de l'espace des augmentations, on peut prévoir que la conception de l'algorithme de sélection des augmentations devient une problématique centrale.

Références

[1] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples : A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3) :1–34, 2020.

[2] J. E. Van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2) :373–440, 2020.

[3] L. Ericsson, H. Gouk, and T. M. Hospedales. How well do self-supervised models transfer ? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[4] S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv :1610.02242*, 2016.

[5] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *Journal on control and optimization*, 30(4) :838–855, 1992.

[6] A. Tarvainen and H. Valpola. Mean teachers are better role models : Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[7] T. Miyato, S. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training : a regularization method for supervised and semi-supervised learning. *Transactions on pattern analysis and machine intelligence*, 41(8) :1979–1993, 2018.

[8] Z. You, J. Ye, K. Li, Z. Xu, and P. Wang. Adversarial noise layer : Regularize neural network by adding noise. In *International Conference on Image Processing (ICIP)*. IEEE, 2019.

[9] N. Turpault and R. Serizel. Training sound event detection on a heterogeneous dataset. *arXiv preprint arXiv :2007.03931*, 2020.

[10] A. Agnone and U. Altaf. Virtual adversarial training system for DCASE 2019 task 4. *Detection and Classification of Acoustics Scenes and Events (DCASE) Challenge*, 2019.

[11] M. Zöhrer and F. Pernkopf. Virtual adversarial training and data augmentation for acoustic event detection with gated recurrent neural networks. In *Interspeech*, 2017.

[12] R. Harb and F. Pernkopf. Sound event detection using weakly-labeled semi-supervised data with GCRNNS, vat and self-adaptive label refinement. *arXiv preprint arXiv :1810.06897*, 2018.

[13] H. Dinkel, X. Cai, Z. Yan, Y. Wang, J. Zhang, and Y. Wang. A lightweight approach for semi-supervised sound event detection with unsupervised data augmentation. In *Proceedings of the 6th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), Online*, 2021.

[14] G. Ferroni, N. Turpault, J. Azcarreta, F. Tuveri, R. Serizel, Ç. Bilen, and S. Krstulović. Improving sound event detection metrics : insights from DCASE 2020. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.