INTERSPEECH 2023
20-24 August 2023, Dublin, Ireland

Adapting a ConvNeXt model to audio classification on AudioSet

Thomas Pellegrini[1,2,3], *Ismail Khalfaoui-Hassani*[1,3], *Etienne Labbé*[2,3], *Timothée Masquelier*[4]

[1]Artificial and Natural Intelligence Toulouse Institute (ANITI), France
[2]Institut de Recherche en Informatique de Toulouse (IRIT), France
[3]Université de Toulouse, CNRS, Toulouse INP, UT3, France
[4]CerCo UMR 5549, CNRS, Université de Toulouse, France

thomas.pellegrini@irit.fr,ismail.khalfaoui-hassani@univ-tlse3.fr,
etienne.labbe@irit.fr,timothee.masquelier@cnrs.fr

Abstract

In computer vision, convolutional neural networks (CNN) such as ConvNeXt, have been able to surpass state-of-the-art transformers, partly thanks to depthwise separable convolutions (DSC). DSC, as an approximation of the regular convolution, has made CNNs more efficient in time and memory complexity without deteriorating their accuracy, and sometimes even improving it. In this paper, we first implement DSC into the Pretrained Audio Neural Networks (PANN) family for audio classification on AudioSet, to show its benefits in terms of accuracy/model size trade-off. Second, we adapt the now famous ConvNeXt model to the same task. It rapidly overfits, so we report on techniques that improve the learning process. Our best ConvNeXt model reached 0.471 mean-average precision on AudioSet, which is better than or equivalent to recent large audio transformers, while using three times less parameters. We also achieved positive results in audio captioning and audio retrieval with this model[1].

Index Terms: audio classification, audio tagging, convolutional neural networks, AudioSet, ConvNeXt

1. Introduction

In the field of audio classification, the so-called "PANNs" for Pretrained Audio Neural Networks, proposed by Kong and colleagues [1], are well-known in the community. Trained on AudioSet [2], they are an off-the-shelf solution to perform sound event detection (SED) with the 527 event categories present in AudioSet. They are very much used also as feature extractors in all kinds of audio downstream tasks, such as SED focused on a specific set of events, automatic audio captioning, language-based audio retrieval, to name a few.

PANNs are vanilla convolutional neural networks (CNNs). CNNs are also still used in vision and interest in them has been rising a lot recently, since modernized CNNs [3, 4, 5] have been shown to outperform vision transformers (*e.g.*, Swin [6]) on ImageNet classification and downstream tasks (semantic segmentation and object detection). These modernized CNNs share multiple features that were not used in PANNs: (a) a stem to substantially reduce the input resolution (typically by a factor 4); (b) depthwise separable convolutions (DSC), which drastically reduce the number of parameters and FLOPS, and therefore allow exploring deeper and wider networks, as well as larger kernel sizes ($> 3 \times 3$); (c) residual skip connections; (d) inverted bottlenecks; (e) downsampling layers that use strided convolutions instead of pooling. We suspected that these modern tricks could also be useful for the field of audio classification. We thus decided to adapt one of the most popular modern vision CNNs, named ConvNeXt [3], to this task.

2. The ConvNeXt architecture/models

The ConvNeXt models [3] were proposed as a fully convolutional alternative to the recent attention-based transformer models in computer vision. ConvNeXt is based on the use of depthwise separable convolutions (DSC) [7], followed by what is known as an inverted bottleneck [8], responsible for the channel mixing operation. Strictly speaking, a depthwise convolution refers to a convolution where each input channel is convolved by its specific kernel, separate from the others. In a broader definition, they refer to "grouped" convolution, where the number of output channels C_{out} (equal to the number of filters) is a multiple of the number of input channels C_{in}. In this situation, if $C_{\text{out}} = K \times C_{\text{in}}$, the integer $K > 1$ is sometimes called the depthwise multiplier.

The ConvNeXt model can be seen as a modernization of the well-known ResNet model [9] aiming to obtain the best compromise between accuracy on the one hand, flops and throughput on the other hand. To achieve this, the ConvNeXt model has introduced several architectural modifications to the previous ResNet architecture at the macro and micro levels. We have retained most of them for our audio classification task, these are listed below:

- A stem, consisting of a strided convolution where the strides are equal to the kernel sizes (patches), is used as the first convolution layer of the model.
- Four stages with ratios 1:1:3:1 regarding the number of blocks within each stage.
- A stage is a sequence of blocks with a depthwise 7×7 convolution followed by an inverted bottleneck, the number of channels and the spatial dimensions are constant within a stage.
- An inverted bottleneck contains a pointwise convolution layer that increases the number of channels by four, followed by a GELU activation function [10], and another pointwise convolution layer that decreases the number of channels back to its value at the bottleneck input. A residual connection sums the resulting feature maps to the block input.
- Separate downsampling layers (regular convolution layers with 2×2 stride and kernels) between the stages.

We tested several ConvNeXt architectures, and our best results were obtained with the Tiny variant where the numbers of blocks within the four stages are 3-3-9-3, and the numbers of channels are 96-192-384-768. Its adaptation to audio classification is described in details in Section 5.

[1]Our PyTorch source code and checkpoint models are available at https://github.com/topel/audioset-convnext-inf

10.21437/Interspeech.2023-1564

3. AudioSet and evaluation metrics

AudioSet is a large-scale audio classification dataset, comprised of about 2 million 10-second clips downloaded from YouTube [2]. A set of 527 audio event categories were used to annotate the data. Since several events may co-occur in the clips, multiple labels can be active for a single clip. AudioSet tagging is, thus, a multi-label classification task. AudioSet is divided into three subsets: the class-wise unbalanced and balanced subsets with respectively 2,042,985 and 22,176 clips, and an eval set of 20,383 clips. We downloaded the data in 2018, and a number of YouTube links were already broken. Our AudioSet data contains 1,921,982 (unbalanced train set), 21,022 (balanced train set) and 19,393 (eval) clips.

We report the usual evaluation metrics for AudioSet tagging: mean average precision (mAP), mean area under the curve (AUC) and d-prime [2]. They are computed for each class and then averaged (macro-averaging), the higher the better.

4. First experiments towards ConvNeXt

The use of DSC in audio tagging is not new. In [11], for instance, the replacement of regular convolutions by DSCs resulted in large reductions in model complexity, together with performance gains. Nevertheless, we are not aware of this line of work when applied to AudioSet, the largest audio tagging dataset available. In this section, we report two preliminary experiments using slightly modified architectures of PANN's CNN14 and CNN6 [1]. The results, shown in Table 1, motivated us to adapt full ConvNeXt vision models to the task.

In [1], CNN14 and CNN6 were trained on a single V100 GPU, with a 32 batch size and for three days. We used their training scripts to train these models and our depthwise variants on eight V100 GPUs in parallel, for 20 hours, with a batch size of 64 samples per GPU, thus, an effective batch size of 512 samples. This larger batch size is responsible for significant gains in performance, when we compare the CNN6 and CNN14 results reported in [1], and ours, referred to as "CNN14 (rep.)" and "CNN6 (rep.)" hereafter.

Table 1: *First results on the AudioSet test set, reproducing CNN14 and CNN6, and testing variants using depthwise convolutions.*

Model	# params	mAP	AUC	d-prime
CNN14 [1]	80.8M	.431	.973	2.732
CNN14 (rep.)	80.8M	.441	.972	2.971
CNN14Sep	30.5M	.436	.973	2.984
CNN6 [1]	4.8M	.343	.965	2.568
CNN6 (rep.)	4.8M	.350	.967	2.802
CNN6Next	3.3M	.427	.972	2.957

4.1. CNN14Sep: CNN14 with depthwise convolutions

CNN14 consists of six blocks with regular convolution layers with kernels of size 3×3, followed by global pooling and two fully-connected layers. Each convolution block has two convolution layers. The first one doubles the number of input channels, in order to compensate for the 2×2 spatial pooling operation preceding the block. The second convolution layer keeps the number of channels unchanged. We replaced this second layer by a depthwise convolution one. We refer to this CNN14 variant as CNN14Sep. The number of learnable parameters dropped from 80.8 to 30.5 million. CNN14Sep achieved a 0.436 mAP, which is better than the 0.431 value reported in [1], but 0.005 lower than our reproduced CNN14, trained with a large batch size.

4.2. CNN6Next: CNN6 with ConvNeXt-like blocks

Table 2: *Architecture of PANN's CNN6 and our variant CNN6Next. DW: depthwise, K: depthwise multiplier, BN and LN: Batch and Layer Normalization.*

<table>
<tr><th>CNN6</th><th>CNN6Next</th></tr>
<tr><td colspan="2">Log-Mel spectrograms
1000 frames × 64 mel bins
BN on the 64 mel features</td></tr>
<tr><td>5 × 5@64
BN, ReLU</td><td>7 × 7@64
LN, 1 × 1@256
GeLU, 1 × 1@64</td></tr>
<tr><td colspan="2">Avg Pooling 2 × 2</td></tr>
<tr><td>5 × 5@128
BN, ReLU</td><td>7 × 7@128 DW-K=2
LN, 1 × 1@512
GeLU, 1 × 1@128</td></tr>
<tr><td colspan="2">Avg Pooling 2 × 2</td></tr>
<tr><td>5 × 5 @ 256
BN, ReLU</td><td>7 × 7@256 DW-K=2
LN, 1 × 1@1024
GeLU, 1 × 1@256</td></tr>
<tr><td colspan="2">Avg Pooling 2 × 2</td></tr>
<tr><td>5 × 5 @ 512
BN, ReLU</td><td>7 × 7@512 DW-K=2
LN, 1 × 1@2048
GeLU, 1 × 1@512</td></tr>
<tr><td colspan="2">Global Pooling</td></tr>
<tr><td colspan="2">FC 512, ReLU
FC 527, sigmoid</td></tr>
</table>

We ran experiments with the smaller PANN architecture CNN6, in which we replaced the regular convolution blocks with simplified ConvNeXt convolution blocks, as shown in Table 2. We needed to slightly adapt these blocks, in order to cope with the number of channels that is doubled at each convolution/block layer in CNN6. Thus, instead of having a fully depthwise 7×7 convolution layer, we used a depthwise layer with a depthwise multiplier of two ($K = 2$). We kept the inverted bottlenecks and removed the residual connections (not possible to keep them since the number of channels is doubled at each block). We also simplified the blocks by using 2×2 average pooling instead of the ConvNeXt downsampling convolution layers. A 0.427 mAP value was obtained with CNN6Next, although it has 3.3M parameters only. This shows that a large gain can be obtained by modernizing the vanilla CNN6. Nevertheless, CNN6Next is not scalable, for the two following reasons: residual connections are not possible in this architecture and adding more blocks doubles the number of channels at each new one, leading to strong overfitting.

5. Adapting a ConvNeXt architecture to audio classification

We adapted ConvNeXt to audio classification on AudioSet by changing the stem and the head of the model. This allows us to eventually take advantage of the available checkpoints pretrained on ImageNet, as bootstrap initializations.

The so-called stem layer, a term borrowed from transformers, is the first convolution layer in the ConvNeXt models. It is responsible for reducing the spatial dimensions of the input, by outputting "patches", *i.e.* feature maps obtained from disjoint subparts of the input images. In the original vision ConvNeXts, the size of these output patches was set to 56×56, in order to be consistent with the 7×7 convolution layers that follow, and also with the three downsampling layers that reduce the spatial dimensions by a factor 8 in total. In our task, the input spectrograms are not at all squared, they are much lengthier in time than in frequency, 1000×64 in the case of the PANN's CNNs. With ConvNeXt, our best results were obtained with 1000×224-sized spectrograms, reduced to 252×56 feature maps with our audio-adapted stem. We tested three other resolutions: the original squared 56×56, 122×122 and 504×56, they all obtained worse results than 252×56. For the classification head, we replaced the 1000-dimensional fully-connected layer by a 527-d one, adapted to the 527 AudioSet target categories.

6. Experiments on AudioSet

6.1. Audio augmentation methods

We used three audio augmentations when training ConvNeXt models: SpecAugment [12] and mixup [13], both as implemented in PANN, and we added *Speed Perturbation* [14, 15]. SpecAugment is used to randomly drop one or several time or frequency stripes from the spectrograms, meaning that the values in the selected stripes are artificially set to zero. In PANN, the spectrograms have 1000 time frames and 64 log mel frequency bins. During training, the number of dropped stripes is randomly drawn between zero and two for each minibatch. The maximum stripe width is 64 frames in time, and 8 bins in frequency. We kept this setting with ConvNeXt, except that the maximum stripe width was increased to 28 bins in frequency, since we chose ConvNeXt's spectrograms to have 224 frequency bins reduced to 56 by the stem. Mixup in PANN operates a convex linear summation of two samples and their labels, with a weight drawn for each example in a minibatch, using a beta probability distribution of parameters $\alpha = \beta = 1$. Finally, speed Perturbation refers to resampling the raw audio signals either up (nearest-neighbor upsampling) or down (decimation) according to a rate chosen randomly within $[0.5, 1.5]$. The resulting waveform is, thus, shorter or longer than the original one. Padding or cropping is randomly applied at the start and the end of the stretched signal in order to keep the signal duration constant.

6.2. Training setup

We trained the models with an effective batch size of 512 audio samples, on eight V100-32GB GPUs, for 75k iterations (corresponding to about 20 hours of training). We used the AdamW optimizer and a "one-cycle" learning rate (LR) scheduler with a maximum LR value of 4e-3, reached after 30% of the total number of training steps (22.5k iterations). Weight decay (WD) was crucial to succeed in training ConvNeXt -Tiny models, which strongly overfit otherwise. Several values were tested and a 0.05 WD value was chosen. A 0.4 drop path [16] level was used, also very useful to reduce overfitting.

6.3. Results on the AudioSet test set

Table 3 shows the number of parameters, FLOPs and throughput (samples / s) when available, and the mAP values of our best ConvNeXt model (last line), and of models from the literature: PANN's CNN14, two audio transformers: Audio Spectrogram Transformer (AST, [17]), PaSST-S [18], and a ConvNext-Tiny, implemented in the JAX language in an audio processing framework called Audax[2]. We report results from the literature obtained with single models, without ensembling methods. Our model achieved a 0.471 mAP (0.973 AUC, 3.071 d-prime), outperforming AST and CNN14 (the original and the reproduced one), and is on par with PaSST, although having about three times less parameters and a throughput about twice as fast. Compared to CNN14, the decrease of throughput observed in ConvNeXt is due to the larger size of the input spectrograms processed by the stem (224 bins instead of 64). Finally, the ConvNeXt-Tiny from Audax was reported to only reach 0.402 mAP, a much lower value than ours. This large gap is probably due to a lack of adaptation to the task, in particular they kept the squared 56×56 stem output resolution.

We would like to mention that Schmid and colleagues [19] achieved better results, in particular a 0.483 mAP with a large MobileNetV3 model (MobileNet40_as_ext, 68.4 million parameters). Nevertheless, their models were trained in a completely different way than ours: knowledge distillation used to mimic the predictions of a large ensemble of PaSST models. We would have to use the same learning framework with our models for the sake of a fair comparison.

Finally, we tested the larger model ConvNeXt-Small (49.9M parameters) variant. It differs from ConvNeXt-Tiny only by the number of blocks of the third stage: 3-3-27-3, instead of 3-3-9-3. Although trained with larger values for WD (0.1) and drop-path (0.8), ConvNeXt-Small still suffers from overfitting and achieved 0.458 mAP only.

Table 3: *Mean average precision (mAP) on AudioSet. The throughput was calculated at inference, on a batch consisting of 64 samples of size 320 000 (10 s), using a single V100-32GB gpu.*

Model	# params	FLOPs	Throughput (samples/s)	mAP
CNN14 [1]	80.7M	**21.2G**	**378.2**	.431
AST [17]	88.0M			.459
PaSST-S [18]	87.0M		88.7	**.471**
ConvNeXt-Tiny (JAX, audax)	28.2M			.402
ConvNeXt-Tiny (ours)	**28.2M**	**21.1G**	**153.6**	**.471**

6.4. Influence of pretraining on ImageNet

Our best result on AudioSet, reported in Table 3, was obtained using a model initialization pretrained on ImageNet1K. A randomly-initialized model achieved 0.462 mAP (0.970 AUC, 3.031 d-prime), which is about 1% absolute worse than with pretraining. This result is inline with the AST and PaSST works, in which models pretrained on ImageNet were better initializations than random ones.

[2] https://github.com/SarthakYadav/audax

7. Downstream tasks

In this section, we report very encouraging results in two downstream tasks, obtained with our ConvNeXt-Tiny model, trained on AudioSet, and used frozen.

7.1. Automated Audio Captioning (AAC) on AudioCaps

Table 4: *Audio Captioning results on the AudioCaps test set.*

Model	mAP	SPIDEr (CIDEr-D/SPICE)	FENSE
SOTA [20]	N/A	.475 (.769/.181)	N/A
CNN14 [1]	.647	.435 (.697/.174)	.610
CNN14 (rep.)	.727	.451 (.725/.177)	.620
ConvNeXt (ours)	**.749**	**.471** (**.760/.182**)	**.638**

AAC aims to build models that can produce a sentence written in natural language, that describes the content of an audio recording. Most AAC systems employ encoder models pretrained on AudioSet to better recognize and describe sound events. In this work, we use a decoder architecture similar to [21, 22]. We extract a sequence of 31 audio embedding for each audio file, using either CNN14 or our ConvNeXt-Tiny model, without their last pooling and classification layers. The encoder weights are frozen during training. We add a projection layer after the encoders to match the input embedding dimension of the decoder part. The decoder is a standard transformer decoder [23] with 6 layers, 4 attention heads per layer, a global embedding size set to 256, a global dropout probability of 0.2 and GELU. The final model contains 12.4M trainable parameters and 79.7M frozen parameters. We used AudioCaps [24], which is the largest audio-language dataset publicly available. The audio files are 10-second clips extracted from YouTube. Our version of the dataset contains 46 230 audio files in the training subset, 464 in the validation and 912 in the test subsets.

During inference, we use the standard beam search algorithm with a beam size set to two. We report several standard metrics used in AAC. CIDEr-D [25] is the cosine similarity scores between the TF-IDF scores of the n-grams, SPICE [26] computes an F-Score value over the semantic propositions extracted from the sentences and SPIDEr [27] is the mean of CIDEr-D and SPICE. FENSE [28] combines two scores: the first one being the cosine similarity between SBERT embeddings extracted from the sentences, the second one is the binary detection made by a Fluency Error Detector trained to detect the common errors made by AAC systems. We also report the mAP scores of each encoder on the AudioCaps test subset.

The results in Table 4 show a clear improvement in AAC scores when using a stronger pre-trained encoder. The best encoder is ConvNeXt-Tiny, which achieves a 0.471 SPIDEr score, a value very close to the SOTA method which reach a SPIDEr value of 0.475. Moreover, our model contains only 40.6M parameters (28.2M frozen and 12.4M trainable) compared to the 108M trainable parameters used in the SOTA system.

7.2. Language-based Audio Retrieval on Clotho v2.1

This multimodal task is, to some extent, the reverse task of AAC. It aims to retrieve and rank audio recordings based on a queried caption, formulated as a free-form sentence written in natural language. The Clotho v2.1 dataset [29] was used for this task in the Detection and Classification of Acoustic Scenes and Events (DCASE) 2022 challenge. The development set contains 3839 audio clips with 19 195 reference captions for training, and 1045 audio recordings with 5225 captions in the "Development-testing" subset, on which we report our results in Table 5. The main evaluation metric is a standard metric in information retrieval: mean Average Precision at top-10 (mAP@10). We also report recall values at top-10 (R@10). All the systems use two pretrained neural networks: an audio encoder (PANN and PaSST pretrained on AudioSet) and a text encoder (sentence BERT variants, pretrained on sentence similarity tasks). The state-of-the-art system [30] uses PANN's CNN14 and BERT. We were not able to reproduce their result, instead we used another system, called PaSST-MPnet [31], in which PaSST was used to encode the audio recordings using the 527-AudioSet class logits as embeddings. MPnet is a sentence BERT model that provides 768-dimensional embeddings for sentences. The audio logits embeddings are projected onto the MPnet ones with a single linear layer. We used the code provided by the authors, and simply replaced the logit embeddings of PaSST by those predicted with our ConvNeXt-tiny trained on AudioSet. As shown in Table 5, PaSST and ConvNeXt performed similarly, with 0.229 and 0.230 mAP@10, respectively. An enhanced variant named PaSST+tags was proposed in [31], where the MPnet textual embeddings of the AudioSet tags are combined to the audio logit embeddings, in the audio encoding part of the system. When replacing PaSST with ConvNeXt in this second system, we observed a significant improvement in mAP@10, with a 0.240 value for ConvNeXt+tags, compared to 0.234 for PaSST+tags.

Table 5: *Audio retrieval results on the Clotho Dev-test split.*

Audio encoder	# params	mAP@10	R@10
SOTA [30]	195M	.260	.530
PaSST-MPNet [31]	196M	.229	.482
+ tags [31]	196M	.234	.485
ConvNeXt-MPNet (ours)	146M	.230	.488
+ tags (ours)	**146M**	**.240**	**.488**

8. Conclusions

In this work, we wanted to revisit CNNs applied to audio classification under the new light shed by CNNs recently modernized in computer vision. We show that a ConvNeXt-Tiny model, adapted to audio tagging on AudioSet, achieved the same precision as a recent transformer, with about three times less parameters. This model achieved very good results on two downstream tasks, audio captioning and language-based audio retrieval. We plan to continue this line of work, but using other learning paradigms, such as knowledge distillation and self-supervision to further explore the capabilities of this type of models.

9. Acknowledgments

This work was partially supported by the French ANR agency within the LUDAU project (ANR-18-CE23-0005-01) and the French "Investing for the Future PIA3" AI Interdisciplinary Institute ANITI (Grant agreement ANR-19-PI3A-0004). This work was performed using HPC resources from GENCI-IDRIS (Grant 2022-AD011013587).

10. References

[1] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP*, New Orleans, LA, 2017.

[3] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 11 976–11 986.

[4] Y. Rao, W. Zhao, Y. Tang, J. Zhou, S.-N. Lim, and J. Lu, "HorNet: Efficient High-Order Spatial Interactions with Recursive Gated Convolutions," in *NeurIPS*, 2022, pp. 1–17.

[5] W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, and X. Wang, "MetaFormer Baselines for Vision," *arXiv*, 2022.

[6] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 10 012–10 022.

[7] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," 2016. [Online]. Available: https://arxiv.org/abs/1606.08415

[11] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, "Sound event detection with depthwise separable and dilated convolutions," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[12] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[13] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2018.

[14] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, Dresden, 2015, pp. 3586–3589.

[15] L. Cances, E. Labbé, and T. Pellegrini, "Comparison of semi-supervised deep learning algorithms for audio classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, no. 1, p. 23, 2022.

[16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. ECCV*. Amsterdam: Springer, 2016, pp. 646–661.

[17] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Proc. Interspeech*, Brno, 2021, pp. 571–575.

[18] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," in *Proc. Interspeech*, Incheon, 2022, pp. 2753–2757.

[19] F. Schmid, K. Koutini, and G. Widmer, "Efficient large-scale audio tagging via transformer-to-cnn knowledge distillation," *arXiv preprint arXiv:2211.04772*, 2022.

[20] E. Kim, J. Kim, Y. Oh, K. Kim, M. Park, J. Sim, J. Lee, and K. Lee, "Improving audio-language learning with mixgen and multi-level test-time augmentation," 2022. [Online]. Available: https://arxiv.org/abs/2210.17143

[21] E. Labbé, T. Pellegrini, and J. Pinquier, "Irit-ups dcase 2022 task6a system: stochastic decoding methods for audio captioning," DCASE2022 Challenge, Tech. Rep., July 2022.

[22] X. Mei, Q. Huang, X. Liu, G. Chen, J. Wu, Y. Wu, J. Zhao, S. Li, T. Ko, H. L. Tang, X. Shao, M. D. Plumbley, and W. Wang, "An encoder-decoder based audio captioning system with transfer and reinforcement learning for DCASE challenge 2021 task 6," DCASE2021 Challenge, Tech. Rep., July 2021.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[24] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 119–132.

[25] R. Vedantam, C. L. Zitnick, and D. Parikh, "CIDEr: Consensus-based Image Description Evaluation," *arXiv:1411.5726 [cs]*, Jun. 2015, arXiv: 1411.5726.

[26] P. Anderson, B. Fernando, M. Johnson, and S. Gould, "SPICE: Semantic Propositional Image Caption Evaluation," *arXiv:1607.08822 [cs]*, Jul. 2016, arXiv: 1607.08822.

[27] S. Liu, Z. Zhu, N. Ye, S. Guadarrama, and K. Murphy, "Improved Image Captioning via Policy Gradient optimization of SPIDEr," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 873–881, Oct. 2017, arXiv: 1612.00370.

[28] Z. Zhou, Z. Zhang, X. Xu, Z. Xie, M. Wu, and K. Q. Zhu, "Can Audio Captions Be Evaluated with Image Caption Metrics?" Jan. 2022, number: arXiv:2110.04684 arXiv:2110.04684 [cs, eess].

[29] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *Proc. ICASSP*. IEEE, 2020, pp. 736–740.

[30] X. Mei, X. Liu, H. Liu, J. Sun, M. D. Plumbley, and W. Wang, "Language-based audio retrieval with pre-trained models," DCASE2022 Challenge, Tech. Rep., July 2022.

[31] T. Pellegrini, "Language-based audio retrieval with textual embeddings of tag names," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Nancy, 2022, pp. 151–155.