



Efficient Training of Audio Transformers with Patchout

Khaled Koutini^{1,2}, Jan Schlüter¹, Hamid Eghbal-zadeh^{1,2}, Gerhard Widmer^{1,2}

Institute of Computational Perception¹ & LIT AI Lab², Johannes Kepler University Linz, Austria

first.last@jku.at

Abstract

The great success of transformer-based models in natural language processing (NLP) has led to various attempts at adapting these architectures to other domains such as vision and audio. Recent work has shown that transformers can outperform Convolutional Neural Networks (CNNs) on vision and audio tasks. However, one of the main shortcomings of transformer models, compared to the well-established CNNs, is the computational complexity. In transformers, the compute and memory complexity is known to grow *quadratically* with the input length. Therefore, there has been extensive work on optimizing transformers, but often at the cost of degrading predictive performance. In this work, we propose a novel method to optimize and regularize transformers on audio spectrograms. Our proposed models achieve a new state-of-the-art performance on Audioset and can be trained on a single consumer-grade GPU. Furthermore, we propose a transformer model that outperforms CNNs in terms of both performance and training speed.¹

Index Terms: transformers, audio-tagging, attention models, audio classification

1. Introduction

The transformer architecture [1] has proven very successful in sequence modeling. It allows learning dependencies between different items in the sequence regardless of their positions or their separation in the sequence. Transformers are the state-of-the-art models in different natural language processing tasks [2–4]. More recently, they have been adapted to computer vision [5–7] by extracting small patches from the input image and adding a learnable positional encoding to each patch. The resulting patches form a sequence that can be fed to the transformer. These vision transformer models achieve state-of-the-art performance on image classification tasks, but require large amounts of training data (e.g. in Vision Transformer (ViT) [5]), or heavily depend on extensive data augmentation and knowledge distillation from a CNN model (e.g. in Data-efficient Image Transformers (DeiT) [8]). Gong et al. [9] further adapted vision transformers to audio spectrograms, achieving state-of-the-art performance on Audioset [10] by using pre-trained models from computer vision and using overlapping patches from audio spectrograms for fine-tuning.

The transformer architecture consists of a series of self-attention layers [1]. Each layer relies on calculating a distance between each pair of items from the input sequence. Although this allows each input item to attend to any other item in the sequence, this results in a complexity of $\mathcal{O}(n^2)$ with respect to the input sequence length n , in terms of both memory and computation effort. Reducing the quadratic complexity has been the target of several approaches in natural language processing, the idea being to restrict each input item (token) to attend

¹Source code and pretrained models: <https://github.com/kkoutini/PaSST>

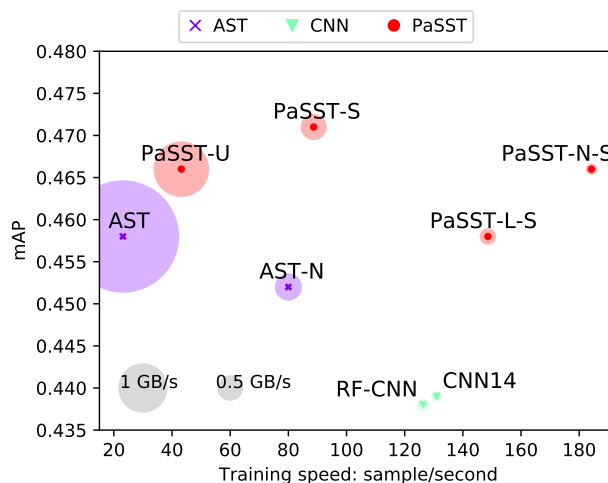


Figure 1: Performance vs training speed on Audioset. The radius of the circle indicates the required GPU memory per sample for training. On the largest publicly available audio dataset, our approach PaSST-S can reach the state-of-the-art performance in less than 2 days on a single consumer GPU. Details are presented in Table 1.

only to a pre-selected subset of input items (tokens). One example is to allow attending only to neighbours inside a sliding window [11, 12]. Additionally, Kitaev et al. [13] use locality-sensitive hashing to approximate attention, reducing the attention complexity to $\mathcal{O}(n \log n)$. BigBird [14] combines sliding windows, global attention, and random interaction between the sequence items. Masking portions of the input sequence has been shown to be an extremely effective method for training encoder/decoder transformers in NLP [2]. In computer vision, the idea of removing patches during *inference* was investigated to assess the vision transformer’s *robustness* against input perturbations [15].

In this paper, we focus on applying transformers to *audio processing*. We address the shortcomings of current audio transformers from the aspect of computational complexity and memory requirements by introducing a new simple yet effective method for training transformers on spectrograms. In summary, the main contributions of our work are as follows:

- We propose *Patchout*, a method that significantly reduces the computation and memory complexity of training transformers for the audio domain. Patchout also functions as a regularizer, improving the generalization of the trained transformers.
- We disentangle the transformer’s positional encoding [5, 8, 9] into time and frequency positional encoding, allowing for straightforward inference on audio snippets of

variable length without the need for fine-tuning or interpolating positional encodings.

- We investigate additional methods for reducing training complexity and demonstrate how they affect performance on the larger general-purpose Audioset as well as domain-specific downstream tasks.

Our proposed models can achieve state-of-the-art performance on several audio tagging and classification tasks using a single consumer GPU, in a relatively short time (see Figure 1). When different complexity reduction methods are combined, the models outperform CNNs in terms of training speed and memory requirements, in addition to generalization.

2. The Patchout faSt Spectrogram Transformer (PaSST)

The Vision Transformer (ViT) [5] works by extracting small *patches* from an input image and projecting these patches linearly onto a sequence of embeddings. The sequence is then augmented by adding trainable positional encodings as biases to the input sequence. A special classification embedding (classification token) is then appended to the sequence, which is connected to a classifier after the self-attention layers. In Data-efficient Image Transformers (DeiT) [8] another special embedding for distillation (distillation token) is added. Gong et al. [9] showed that *overlapping* the extracted patches improves the performance when training ViT on spectrograms. On the other hand, patch overlapping increases the total number of patches, i.e., the input sequence length. Therefore, overlapping greatly increase the memory and compute requirements for training the transformers. We propose a new method called *Patchout* (Section 2.2) to overcome these issues.

Figure 2 summarizes the proposed transformer architecture: The pipeline starts at the upper left with an audio spectrogram being fed into the model as input. (1) is the patch extraction and linear projection steps as explained in [5]. In (2), frequency and time positional encodings are added, as discussed below. In (3), we apply *Patchout* as explained in Section 2.2, and add the classification token. In (4), we flatten the sequence and pass through d layers blocks of self-attention (d is the depth of the transformer). Finally, a classifier operates on the mean of the transformed C and D tokens.

In Addition to *Patchout* (step 3 in Figure 2), the second main difference between our work and previous work [5, 8, 9] is that we disentangle the positional encoding for the time and frequency dimensions, and as a result, we have two positional encodings: one representing the frequency, and one for time (step 2 in Figure 2). This makes inference and the tuning of the pre-trained models on downstream tasks with shorter audio length simpler. When fine-tuning or inference on shorter audio clips, we simply crop the time positional encoding parameters, without changing the frequency encoding parameter.

2.1. Complexity Analysis

Multi-head attention layers [1] rely on computing a distance between each pair of positions in the input sequence (in the form of an attention matrix), therefore having a complexity of $\mathcal{O}(n^2)$ where n is the input sequence length. As the sequence length grows – for example, when overlapping input patches or for longer audio clips – the compute and memory requirements quickly become problematic. More specifically, given an input of b samples of the dimension $(n \times e)$, where b is the batch size,

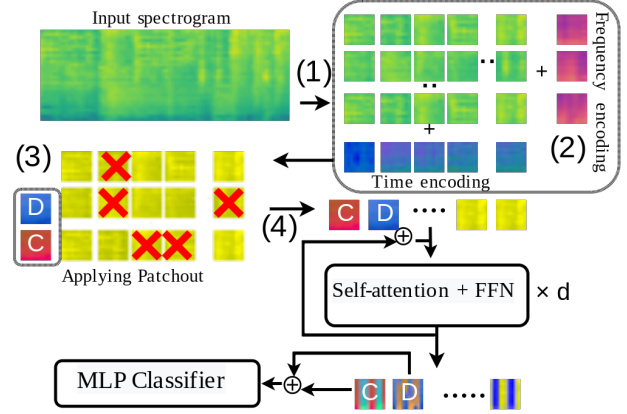


Figure 2: The Patchout transformer (PaSST) architecture as explained in Section 2. The Self-attention layer + FFN (Feed-forward network) is explained in detail in [5]. C: classification token. D: distillation token (only for models based on DeiT).

n is the input sequence length, e is the embeddings size, each *multi-head* attention layer projects each input sample to h query Q , key K , and value V matrices, where h is the number of attention heads [1]. Each of Q , K and V has a shape of $(n \times \frac{e}{h})$. The *attention matrix* A is then computed by the matrix multiplication $Q \times K^T$, scaling, and applying the soft-max activation function $A = \text{Softmax}(\frac{Q \times K^T}{\sqrt{d}})$. A has a shape of $(n \times n)$ and is multiplied with V giving the attention output: $A \times V$ resulting in a new sequence with the same shape as the input $(n \times e)$. As a result, the computation complexity (and memory requirements) for all the operations on the attention matrix A grow quadratically $\mathcal{O}(n^2)$ with sequence length n , while the operations in the rest of the network have a linear complexity relationship with n [1, 5, 8]. In short, reducing the sequence length would have a large impact on the computational complexity of these models.

2.2. Patchout

Motivated by (a) the impact of reducing the sequence length on the computation complexity of training transformer models; (b) the fact that audio events are expected to be spread out in time and frequency in an audio clip; (c) the insight that CNNs can benefit from having a small receptive field during training for different audio tasks, as shown in [16], we propose *Patchout*, a method to efficiently train transformer models on audio spectrograms. The idea is to drop parts of the transformer’s input sequence when training, encouraging the transformer to perform the classification using an incomplete sequence. We first extract small overlapping patches from the input spectrograms and linearly project them to vectors, forming the transformer input sequence. We augment the patches with both frequency and time encoding. When training, we randomly drop parts of the sequence, reducing the sequence length, and effectively regularizing the training process. Similar to Dropout [17], during inference, the whole input sequence is presented to the transformer. We distinguish between different types of Patchout as follows:

Unstructured Patchout is the basic form of Patchout,

where we select the patches randomly regardless of their position. We refer to models trained with this method as *PaSST-U*.

Structured Patchout: We randomly pick some *frequency bins/time frames* and remove a whole column/row of extracted patches. This structure is inspired by SpecAugment [18]. We refer to models trained with this method as *PaSST-N*.

2.3. Further Complexity Reduction Methods

2.3.1. Reducing the extracted patches’ overlap

Reducing the overlap between patches results in a lower number of extracted patches, and therefore a smaller transformer input sequence length. However, Gong et al. [9] showed that reducing the overlap (or training without overlapping) degrades the performance of the transformer on Audioset. Patchout can also be used even when there is no overlap between patches. We refer to the system without patch overlapping as *PaSST-N*.

2.3.2. Reducing the depth of the transformer

The depth of the transformer is the number of successive self-attention blocks (d in Figure 2). The depth has a linear relationship with the overall training and inference complexity and influences the total number of parameters of the model. Since we are starting the training from models pre-trained on Imagenet [19] (as explained in Section 3.2), we remove every other self-attention block. This allows us to benefit from the pre-training, compared to removing consecutive blocks, since the residual activations will have a less sudden change. We will refer to the model with the removed blocks as *PaSST-L*. It has $d = 7$ self-attention blocks and 50M parameters compared to 87M in the full model.

3. Experiment Setup

We train our models on Audioset [10], the largest publicly available audio dataset, consisting of around 2 million audio clips from Youtube. The task is to tag the audio clips with labels from 527 possible classes. Furthermore, we fine-tune the models trained on Audioset on various audio classification and tagging tasks, namely, instrument recognition, environmental audio classification, and acoustic scene classification.

3.1. Preprocessing and Training Setup

We use mono audio with a sampling rate of 32 kHz. We extract Mel features from a window of 25 ms with a hop length of 10 ms, resulting in 128 mel bands, similar to [9]. Kong et al. [20] showed the importance of balancing Audioset; therefore, we balance our training data using importance sampling. We assign a sampling weight to each sample proportional to the inverse frequency of its label $\frac{1}{freq(label)+100}$. We train on 1,893,693 (approx. 2M) training segments, and evaluate on 18,951 audio clips. For each epoch, we sample 200k samples from the full 2M Audioset without replacement. We use the AdamW [21] optimizer with weight decay of 10^{-4} , with a maximum learning rate of 10^{-5} . We use a linear learning rate decay from epoch 50 to 100, dropping the learning rate to 10^{-7} and fine-tune the model for a further 20 epochs.

3.2. ImageNet Pretraining

Gong et al. [9] showed that using pre-trained models on Imagenet significantly improves their performance on Audioset. Therefore, we will use pre-trained models in all our experi-

	mAP	Speed	(\times AST)	Mem	Seq
Baseline [10]	.314	-	-	-	-
PANNs [20]	.439	131.0	(5.7 \times)	.213	-
AST [9]	.459	23.1	(1 \times)	2.33	1212
AST [9] \star	.459	23.1	(1 \times)	2.33	1190
AST-N [9] \star	.454	80.	(3.4 \times)	.534	498
CNN [23] \star	.438	126.3	(5.5 \times)	.213	-
PaSST-B \star	.462	23.1	(1 \times)	2.33	1190
PaSST-U \star	.466	43.2	(1.9 \times)	1.14	790
PaSST-S \star	.471	88.7	(3.8 \times)	.513	474
PaSST-S-L \star	.459	148.6	(6.4 \times)	.311	474
PaSST-S-N \star	.466	184.2	(8 \times)	.202	254

Table 1: *Single-model results on Audioset. \star indicates our run. mAP is the mean average precision (also referred to as precision/recall area under curve). Speed: training throughput in spectrograms per second on an Nvidia Titan RTX GPU (showing the speedup compared to AST [9]). Mem: the required GPU memory to train per sample. Seq: the training sequence length. B: Baseline without Patchout. U: Unstructured Patchout. S: Structured Patchout. N: no-overlap of the extracted patches. L: lighter model with reduced depth=7.*

ments. Our base model is DeiT B \uparrow 384 [8]. We also achieve a comparable performance using computationally more complex ViT models such as stripped-down ViT-hug224 [5]; by removing half of the self-attention blocks, its depth was reduced to only 16 blocks (with the methods explained in Section 2.3.2); this will not be further explored in this paper.

3.3. Data Augmentation

The transformer models are very prone to overfitting, therefore data augmentation plays an essential role in the training process [8]. In our experiments, the following augmentation strategies are used:

Two-level Mix-Up: We use Mix-up [22] since it has been shown to improve performance [9, 23]. We mix both the raw waveforms randomly from the dataset as well as the final spectrograms.

SpecAugment: We use SpecAugment [18] by masking up to 48 frequency bins and 192 time frames similar to [9].

Rolling: We roll the waveforms randomly over time.

Random Gain: We multiply the audio waveforms to change the gain by ± 7 dB.

4. Results

4.1. Audio Tagging on Audioset

Table 1 shows the mean average precision mAP (also referred to as precision-recall area under-curve) results on Audioset [10]. As can be seen, the proposed model *PaSST* achieves a new state-of-the-art performance on the largest available audio tagging dataset. The proposed model outperforms AST [9] and significantly outperforms CNNs. Using Patchout not only improves the performance of the transformer architecture, but also increases the training speed approximately *4 times*, and reduces the required GPU memory to *less than 25%*. As a result, it is possible to train PaSST on a single Nvidia RTX 2080ti (consumer GPU), achieving state-of-the-art performance in 50 hours. Furthermore, *PaSST-L-S* (with a scaled down depth of $d = 7$) and *PaSST-S-N* (without patch overlap) significantly

outperform CNNs while maintaining a higher training throughput, and with similar GPU memory requirements. *PaSST-S-L* and *PaSST-S-N* can be trained on a single GPU to reach .459 and .466 mAP in approximately 25 hours. Applying Patchout on the transformer without overlap (*PaSST-S-N*) outperforms the baseline *PaSST-B* (without Patchout) and *AST* [9] while being up to 8 times faster, and requiring less than 10% of the GPU memory for training. The results are also illustrated in Figure 1. The only difference between the baseline *PaSST-B* and the *AST* model is the positional encoding. *AST* [9], like vision transformers [5, 8], employs grid positional encoding. *PaSST-B*, on the other hand, utilises disentangled time and frequency positional encoding (see Section 2).

4.2. Fine-tuning and Transfer to Downstream Tasks

We fine-tune the pre-trained (on Audioset) models on several downstream audio tagging and classification tasks with different dataset sizes, Table 2 summarizes the results. *PaSST-(B,U,S)* models use the pre-trained *PaSST-S* on Audioset, but for fine-tuning, we use no Patchout, unstructured Patchout, and structured Patchout respectively. It is worth noting that the transformer models can be fine-tuned using a small number of epochs. The results suggest that researchers and practitioners can use pre-trained PaSST and fine-tune them on downstream tasks without the need for large computational resources.

In summary, fine-tuning the transformer model outperforms state-of-the-art CNNs on all tasks. Patchout results in significant speedups and, in many cases, improved generalization. When combined with Structured Patchout (-S), reducing complexity by removing patch overlap (-N) performs better than reducing transformer depth (-L) and enables faster fine-tuning.

We only replace the MLP classifier in the pre-trained models for fine-tuning. When we use Patchout, we randomly remove roughly half of the input sequence. Each experiment was repeated three times, and the average results are reported. The speedup in Table 2 is relative to *PaSST-B* and is rounded up to the nearest integer. Details on the setup of each task can be found in our github repository.

Polyphonic Musical Instrument Recognition: The task here is to detect all the instruments present in an audio clip. The *OpenMIC* dataset [24] consists 20,000 audio clips. Each clip is 10 seconds long and can be assigned multiple tags out of 20 classes. The metric for the task is the mean average precision. The state-of-the-art methods for this task are CNNs with restricted receptive fields [23]. *PaSST-S-N* reaches the state-of-the-art performance in less than 30 minutes on a single consumer GPU.

Environmental Sound Classification: The *ESC50* dataset [25] consists of 2,000 environmental 5-second audio clips. The task is to classify each clip into one out of 50 possible classes. We report the accuracy averaged over the 5 official folds [25]. All PaSST variants (with Patchout) can be fine-tuned on this dataset in less than 5 minutes on a single GPU. The state-of-the-art performance was achieved using the *AST* transformer model [9]. The difference between *AST* and *PaSST-B* is in the positional encoding, as explained in Section 2.

Acoustic Scene Classification: The task is to recognize the acoustic scene of 10-second audio clips. We use the TAU Urban Acoustic Scenes 2020 Mobile dataset [26] as used in the DCASE 2020 challenge (*DCASE20*). The audio clips are recorded with different devices and further simulated devices are introduced. The performance is measured using accuracy on a dataset including unseen devices. The first place in the

	OpenMIC	ESC50	DCASE20	FSD50K
Baseline	.795 [24]	76.9 [25]	54.1 [26]	.434 [28]
SOTA	.831 [23]	95.6 [9]	73.7 [27]	.558 [29]
-B *	.837 1×	96.3 1×	76.3 1×	.649 1×
-U *	.843 4×	96.5 2×	75.6 4×	.639 4×
-S *	.843 4×	96.8 2×	75.6 4×	.653 4×
-S-L *	.841 6×	95.5 2×	73.7 6×	.584 6×
-S-N *	.840 8×	96.4 4×	73.9 8×	.637 8×

Table 2: Results in performance and speedup compared to the base model *PaSST-B* for the downstream tasks: polyphonic instrument tagging using *OpenMIC* [24] dataset (mean average precision), Environmental Sound Classification *ESC50* [25] (accuracy), Cross-device Acoustic Scene Classification *DCASE20* [26] (accuracy). Sound Event Recognition (Tagging) in *FSD50K* [28](mean average precision). The second part of the table compares different PaSST variants (Table 1).

challenge used CNNs [27]. Patchout accelerates training on this task, reaching state-of-the-art in less than an hour. Patchout also allows for fine-tuning on a single consumer GPU. It does, however, lead to a decrease in accuracy.

Sound Event Recognition (Tagging) on FSD50K: The *FSD50K* dataset [28] consists of 51K audio clips annotated with 200 sound event classes taken from the Audioset ontology [10]. The dataset contains 100 hours of audio and is the second largest publicly available general purpose sound event recognition dataset after Audioset. Furthermore, the *FSD50K* evaluation set is of high quality, with each evaluation label being double-checked and assessed by two to five independent annotators [28]. The reported results are on the official evaluation subset of *FSD50K* using the best model on the validation subset. The state-of-the-art in *PSLA* [29] is achieved through CNN architecture and a collection of performance-improving methods such as ImageNet pre-training, label enhancement, balancing, data augmentation, and weight averaging. On this dataset, our approach significantly outperforms the current state-of-the-art. Fine-tuning *PaSST-S* and *PaSST-S-N* takes less than 2 hours and 1 hour, respectively.

5. Conclusion

We propose a new method for efficiently training transformers on audio spectrograms, achieving state-of-the-art performance on Audioset as well as several downstream tasks. Furthermore, Patchout significantly reduces compute complexity and memory requirements for training transformers. We investigate additional methods for reducing training complexity and propose two models, *PaSST-S-L* and *PaSST-S-N*, that outperform CNNs while having a faster training speed and comparable memory requirements. Our pre-trained models can be fine-tuned on several audio downstream tasks with little resources and little additional training time.

6. ACKNOWLEDGMENT

This work has been supported by the COMET-K2 Center of the Linz Center of Mechatronics (LCM) funded by the Austrian Federal Government and the Federal State of Upper Austria. The LIT AI Lab is financed by the Federal State of Upper Austria. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

7. References

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT Minneapolis, MN, USA, June 2-7, 2019, Volume 1*, 2019, pp. 4171–4186.
- [3] D. R. So, W. Manke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le, "Primer: Searching for efficient transformers for language modeling," *Advances in neural information processing systems*, 2021.
- [4] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "LUKE: deep contextualized entity representations with entity-aware self-attention," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP*. Association for Computational Linguistics, 2020, pp. 6442–6454.
- [5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations ICLR 2021, Virtual, May 3-7, 2021*.
- [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 *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9992–10 002.
- [7] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: deformable transformers for end-to-end object detection," in *International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.
- [8] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *ICML 2021, 18-24 July 2021, Virtual*, vol. 139, 2021, pp. 10 347–10 357.
- [9] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio Spectrogram Transformer," in *Interspeech 2021, Brno, Czechia, 30 August - 3 September, 2021*, pp. 571–575.
- [10] 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 *ICASSP 2017, New Orleans, LA, 2017*.
- [11] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage BERT: A globally normalized BERT model for open-domain question answering," in *EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 5877–5881.
- [12] S. Sukhbaatar, E. Grave, P. Bojanowski, and A. Joulin, "Adaptive attention span in transformers," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019*, pp. 331–335.
- [13] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.
- [14] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontañón, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Advances in neural information processing systems*, 2020.
- [15] M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. S. Khan, and M. Yang, "Intriguing properties of vision transformers," in *Advances in neural information processing systems*, 2021.
- [16] K. Koutini, H. Eghbal-zadeh, M. Dorfer, and G. Widmer, "The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification," in *EUSIPCO 2019, A Coruña, Spain, 2019*.
- [17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [18] D. S. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech 2019, Graz, Austria, 15-19 September, 2019*, pp. 2613–2617.
- [19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference On Computer Vision and Pattern Recognition.*, 2009, pp. 248–255.
- [20] 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 Trans. Audio Speech Lang. Process.*, vol. 28, pp. 2880–2894, 2020.
- [21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.
- [22] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- [23] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1987–2000, 2021.
- [24] E. Humphrey, S. Durand, and B. McFee, "OpenMIC-2018: An open data-set for multiple instrument recognition," in *ISMIR 2018, Paris, France, September 23-27, 2018*, pp. 438–444.
- [25] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, 2015, pp. 1015–1018.
- [26] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in DCASE 2020 Challenge: generalization across devices and low complexity solutions," in *DCASE2020 Workshop*, 2020.
- [27] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing Acoustic Scene Classification Models with CNN Variants," DCASE2020 Challenge, Tech. Rep., 2020.
- [28] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: an open dataset of human-labeled sound events," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 30, pp. 829–852, 2022.
- [29] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.