ATGNN: Audio Tagging Graph Neural Network

Shubhr Singh, Christian J. Steinmetz, Emmanouil Benetos, *Sr. Member, IEEE*, Huy Phan and Dan Stowell

***Abstract*—Deep learning models such as CNNs and Transformers have achieved impressive performance for end-to-end audio tagging. Recent works have shown that despite stacking multiple layers, the receptive field of CNNs remains severely limited. Transformers on the other hand are able to map global context through self-attention, but treat the spectrogram as a sequence of patches which is not flexible enough to capture irregular audio objects. In this work, we treat the spectrogram in a more flexible way by considering it as graph structure and process it with a novel graph neural architecture called ATGNN. ATGNN not only combines the capability of CNNs with the global information sharing ability of Graph Neural Networks, but also maps semantic relationships between learnable class embeddings and corresponding spectrogram regions. We evaluate ATGNN on two audio tagging tasks, where it achieves 0.585 mAP on the FSD50K dataset and 0.335 mAP on the AudioSet-balanced dataset, achieving comparable results to Transformer based models with significantly lower number of learnable parameters.**

***Index Terms*—Audio tagging, Graph Neural Networks, Computational sound scene analysis**

I. Introduction

Environmental sounds carry a rich and complex mixture of information, which is organised and categorised by the human auditory system into distinct concepts known as *sound events* (e.g. dog bark, door slam, car passing by). The advent of deep learning models such as convolutional neural networks (CNNs) revolutionised the research field of image classification and achieved state of the art on numerous audio classification & tagging benchmarks as well [1].

In recent times, Transformer-based models [2], [3] have demonstrated superior performance over CNNs in audio tagging and classification tasks. This can be attributed to their unique capability to capture global context by employing self-attention mechanisms, treating spectrograms as sequences of patches. Despite the appealing advantages of Transformers their usage comes with a trade-off as they tend to incur substantial computational costs during both training and inference stages [4].

SS and CS are supported jointly by UK Research and Innovation and Queen Mary University of London under grant EP/S022694/1. This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/V062107/1]. EB is supported by RAEng/Leverhulme Trust Research Fellowship LTRF2223-19-106.

SS, CJ and EB are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, UK (e-mail: {s.shubhr, c.j.steinmetz, emmanouil.benetos}@qmul.ac.uk). HP is with Amazon, Cambridge, MA 02142, USA (e-mail: huypq@amazon.co.uk). DS is with Tilburg University, Bijsterveldenlaan, 5037 AB Tilburg, Netherlands (e-mail: d.stowell@tilburguniversity.edu).

The work was done when HP was at the School of Electronic Engineering and Computer Science, Queen Mary University of London, UK and prior to joining Amazon.

Sound events can be conceptualized as compositions of multiple regions in the spectrogram that are interlinked to form a coherent representation of the event. This underlying structure naturally lends itself to a graph-based representation, where each region in the spectrogram becomes a node, and the relationships between these regions are captured through edges. Graphs offer a more adaptable structure compared to grids or sequences of patches, enabling the utilization of label correlation information. This significantly enhances the overall predictive capabilities. Label co-occurrence graphs (LG) [5] represent the relationships and co-occurrence patterns among different labels in a dataset. Label Graphs have enhanced model performance in both multi-label image recognition, [5] and audio tagging [6]. Considering the advantages of graph structures and the recent successes of graph-based models in image classification [7], [8], we introduce the *Audio Tagging graph neural network* (ATGNN). This is an end-to-end graph convolution network specifically designed for audio tagging applications.

In our method, we first extract features from an input spectrogram using a backbone CNN. Each element of the resultant feature map is treated as a distinct node. To ensure efficient interaction between distant patches in the original spectrogram, we dynamically create edges between nodes based on their similarity in the feature space. This process enables information exchange across various regions of the spectrogram, allowing our model to identify and utilize intricate dependencies.

Besides using graph processing in the feature space, our approach integrates learnable label embeddings, which fulfill two key roles by capturing two unique relationships. First, these embeddings assist in modeling the semantic relationships among class labels, helping the model understand the connections between various sound event categories. Second, the label embeddings form cross-domain links between spectrogram regions and themselves, creating associations between the labels and specific areas of interest in the spectrogram. This approach enhances the overall discriminatory power of our model.

There has been a steady increase in the adoption of GNNs for audio tasks such as speech emotion recognition (SER) [9], speaker diarisation [10], [11], audio tagging [12], [6]. Our research is different from prior works as it leverages spectrogram graph structures to effectively utilize inter-region relationships, and applies an end-to-end approach to concurrently learn label correlations and label-spectrogram interactions.

The main contribution of this paper is therefore summarised as follows: (i) We model the spectrogram as a graph structure and propose an end-to-end graph based model for audio

tagging. (ii) To the best of our knowledge, this is the first attempt to simultaneously model correlations between labels and spectrograms, as well as between labels themselves, in an end-to-end fashion without relying on previously established correlations and demonstrates the effectiveness of the methodology on two widely used audio tagging datasets.

II. Model architecture

ATGNN comprises of three components: Patch GNN (PGN), Patch-Label GNN (PLG), and Label-Label GNN (LLG). PGN focuses on modeling the correlations among different patches of the input mel-spectrogram. PLG captures the relationships between these input patches and label embeddings. Finally, LLG is dedicated to modeling the correlations among various label embeddings. Following the convention of [8], we refer to the stack of single PLG and LLG block as a Multi-Label GNN (MLG) block.

A. PGN: Patch GNN

The input spectrogram with size $F \times T \times 1$ is first divided into N patches, where F denotes the frequency axis dimension and T denotes the time axis dimension. Instead of directly splitting and flattening images into tokens based on a linear patch embedding layer as used in Audio Spectrogram Transformer (AST) [2], a CNN based backbone is adopted for extraction of patches. Input $X \in \mathbb{R}^{F\times T\times 1}$ is transformed to a feature map $X_t \in \mathbb{R}^M$, where $M = (F/p \times T/p) \times D$, with p as the reduction ratio and D as the feature dimension. The resulting feature map is flattened into $\mathbb{R}^{FT/p^2\times D}$ and then combined with a learnable positional encoding. The flattened feature map can be conceptualized as a set of unordered nodes.From these nodes, a k-nearest neighbor graph is constructed, using the Euclidean distance between nodes as the basis. This graph is then refined through a graph convolution layer, which facilitates the exchange of information between neighboring nodes via a message passing operation. Specifically, node x_i is updated using max-relative graph convolution [13]:

$$g(\cdot) = x^{''} = [x_i, \max(x_j - x_i) \mid j \in N(x_i)] \tag{1}$$

$$h(\cdot) = x^{'} = x^{''} W_{update}, \tag{2}$$

where $N(x_i)$ are the neighbours of the node x_i and $x^{'}$ and $x^{''}$ denote updated node embeddings through different update operations. Combination of (1) and (2) is termed as *GraphConv*. A linear layer is applied to each node before and after GraphConv to increase feature diversity and avoid oversmoothing, followed by nonlinear activation. The updated graph convolution operation can be denoted by

$$y_i = \sigma(GraphConv(W_{in}x_i))W_{out} + x_i, \tag{3}$$

where y_i denotes the updated node embedding, σ denotes non linearity, W_{in} and W_{out} are weights of fully connected layers applied before and after *GraphConv*. In a manner akin to Transformer models [14], a Feed-Forward Network (FFN) is applied to each node embedding. The FFN primarily consists of two linear layers, separated by a non-linear activation function. A single Patch GNN (PGN) block is formed by the sequence of a Graph Convolution (GraphConv) layer followed by an FFN layer.

To effectively layer multiple PGNs without causing over-smoothing from increased depth, we employ dilated aggregation in the *GraphConv* operation. Dilated convolutions were proposed in [15] as an alternative to max pool operations. Specifically, for an input graph $G = (V, E)$ with dilated k-NN and d as the dilation rate, the dilated k-NN returns the k nearest neighbors within the $k \times d$ neighborhood region by skipping every d neighbors. Dilated aggregation helps in maintaining feature diversity and reduces the over-smoothing across graph layers.

The PGN block architecture comes in two variants: isotropic and pyramid. [8]. The isotropic architecture, commonly used in Transformer-style models, maintains a consistent feature dimension across all layers. In contrast, the pyramid architecture, typically found in CNN-based structures like ResNet [16], progressively downsamples feature maps in subsequent layers.

Isotropic architecture — The number of nodes for this architecture is set to $N = FT/256$, depending on the value of F and T. The number of k in k-NN is linearly increased from k to $2 \cdot k$ across the layers.

Pyramid architecture — The pyramid architecture is used to generate multi-scale feature maps to exploit the compositional hierarchies of the input, where higher level features are obtained by composition of local level features captured in the initial layers. In addition to the learnable positional embedding, a relative positional encoding similar to [17] is used in the pyramid model. For node i and j, if the positional encoding is e_i and e_j, then their relative positional distance between them is $e_i^{\mathsf{T}} e_j$. This distance is added to the feature embedding distance to construct the k-NN graph [7].

B. PLG: Patch-Label GNN

The PLG block maps the correlation between patch and label embeddings, where patch embeddings $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N\times C}$ are the output of M PGN blocks and $L = \{l_1, l_2, \ldots, l_S\} \in \mathbb{R}^{S\times C}$ are the learnable label embeddings with dimensionality C and S number of classes. Each label node connects to its k_{plg} patch node neighbours measured by Euclidean distance, post which max-relative graph convolution is used to update the label nodes as follows:

$$l_i^{''} = [l_i, \max(l_i - xj) \mid j \in N(l_i)] \tag{4}$$

$$l_i^{'} = l_i + l_i^{''} W_{l-update} \tag{5}$$

where l_i is the i-th label node and $W_{l-update} \in \mathbb{R}^{S\times S}$ is the learnable update matrix for label nodes. The label embeddings are updated by message passing from the corresponding label nodes, which builds the correlation between the spectrogram regions and the label embeddings.

C. LLG: Label-Label GNN

The LLG block is used to map correlation between different label embeddings. The updated label embeddings from PLG

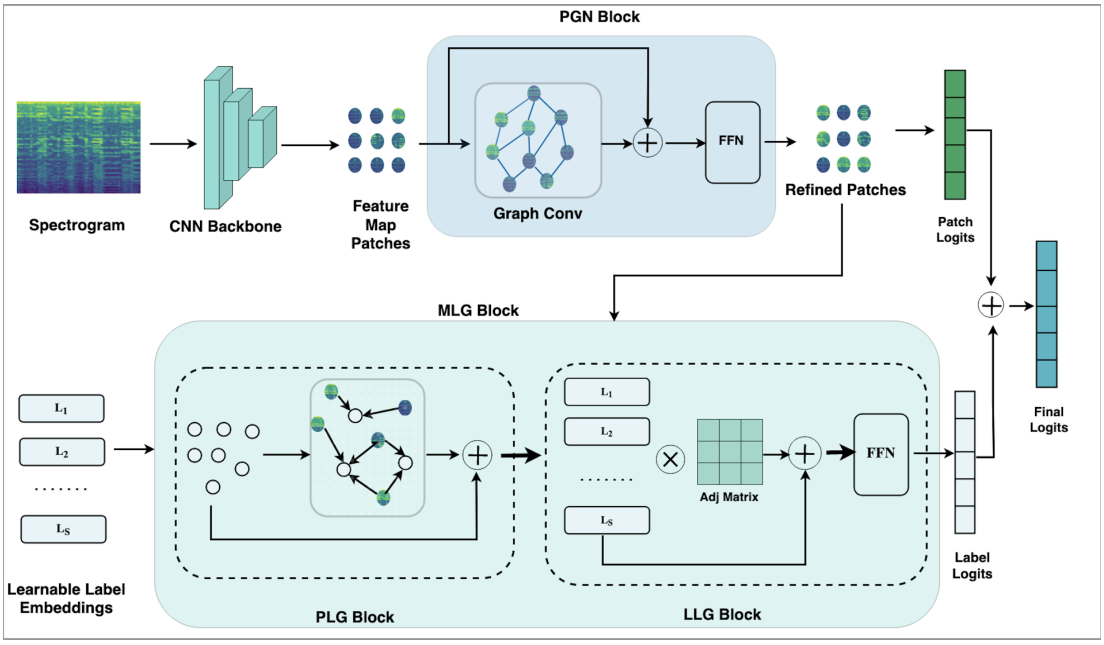

Fig. 1. **ATGNN**: The input spectrogram is passed through a CNN backbone and a k-nearest neighbour graph is constructed with the the feature map pixel as nodes. The patch nodes are updated with graph convolution across M PGN blocks and then fed into an MLG block where cross-correlation between learnable label embeddings and patch nodes is learnt. Each label node connects to its nearest patch nodes and is updated using graph convolution. The LLG block maps label-label correlation using a learned adjacency matrix. A stack of PGN and MLG blocks is used to refine the patch and label embeddings and is combined for final prediction.

blocks are used as nodes in a fully connected graph and each node is updated by aggregating information from the other nodes in the following manner:

$$\hat{L} = AL^{'} + L^{'}, \tag{6}$$

where $L^{'} = [l_1^{'}, l_2^{'}, \ldots, l_S^{'}] \in R^{S\times C}$ are updated label embeddings from the PLG block. $A \in R^{S\times S}$ is a learnable adjacency matrix with random initialisation. $\hat{L} = [\hat{l_1}, \hat{l_2}, \ldots, \hat{l_S}] \in R^{S\times C}$ is the matrix of refined label node embeddings. A is learnt during training of the model and captures the latent label correlations.

D. Prediction

The final output of the PGN block is aggregated via global average pooling and a set of 1×1 convolution layers with nonlinear activation are applied to obtain the final patch logits:

$$Y_{patch} = Conv_p(AvgPool(X_{patch})), \tag{7}$$

where $Y_{patch} \in \mathbb{R}^{1\times S}$ denotes the final patch logits. $Conv_p$ denotes the set of 1×1 convolution layer, $AvgPool$ denotes the average pooling operation applied on the final output $X$of the PGN block.

For label nodes, a readout function R projects each of the S label embeddings $l_i^{'}$ using a projection matrix $W^p \in \mathbb{R}^{S\times d}$ where W_i^p is the learned output vector for $l_i^{'}$. This is akin to applying a separate linear layer to map the presence of each label in the given input:

$$\hat{y}_i = W_i^p l_i^{'\mathsf{T}}. \tag{8}$$

The classification score of each label embedding are concatenated as $\hat{Y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_S] \in \mathbb{R}^{1\times S}$. The final score is obtained as:

$$Y = sigmoid(Y_{patch} + \hat{Y}). \tag{9}$$

III. Experiments

A. Datasets

We evaluated the proposed models on two commonly used datasets: AudioSet [18] and FSD50K [19]. AudioSet is a large scale weakly labelled audio event dataset of over 2 million 10-second audio clips extracted from YouTube videos. The audio events in the datasets are categorised into 527 predefined labels with each audio clip containing one or more audio events. We use the balanced subset of this dataset comprising 20550 training samples and 1887 evaluation samples. FSD50K [19] is a publicly available weakly labelled dataset comprising sound event audio clips categorized across 200 classes drawn from the AudioSet ontology. The dataset comprises 3 subsets: training, validation and evaluation subset consisting of 37134, 4170 and 10231 samples respectively. The lengths of these audio clips vary, ranging from 0.3 to 30 seconds.

B. Training pipeline

To ensure an impartial evaluation, we adopted the training pipeline outlined in [20]. The specifics of this approach are as follows::

Data Preprocessing — The audio files were first resampled to 16 kHz and a short-time Fourier transform (STFT) with a window size of 25ms and hop length of 10 ms, was applied to each audio clip to generate spectrograms. Following this,

TABLE I
MEAN AVERAGE PRECISION VS k-NN VALUES FOR PGN BLOCK

k	9	12	15	18	20
mAP	57.9	57.7	57.7	57.9	57.6

we used a 128-dimensional mel filter bank and performed a logarithmic operation to extract the log-mel spectrograms. To accommodate the varying durations of audio clips, we employed zero-padding to standardize the length to 1024 frames for the FSD50K dataset and 1056 frames for the AudioSet experiments.

Model details — The PGN block of the proposed models was initialised with ImageNet pre-trained weights from [7] for all the experiments. The MLG blocks were trained from scratch. The terms "iso" and "pyr" respectively denote isometric and pyramid versions of the PGN block. The pyramid model has two different sizes - small (s) and medium (med). In case of pyramid model, a stage-wise approach was followed where each stage consists of M PGN blocks followed by P MLG blocks. In case of pyr-s, we found the best results with M = $[2, 2, 6, 2]$ and P = $[1, 1, 3, 1]$. For pyr-med, best results were achieved with M = $[2, 2, 16, 2]$ and P = $[1, 1, 6, 1]$. In our experiments, we varied k from 9 to 20 as shown in Table I for the PGN block and observed that the k values did not change the results significantly, hence for all experiments we used k and k_{plg} = 9.

Balanced Sampling — We adopted a random balanced sampling approach for FSD50K, where each audio clip is assigned a sampling weight such that higher weight is assigned to clips containing rare events. For more details, refer to [20].

Data Augmentation — We used mixup [21] data augmentation (mixup ratio = 0.5) and time-frequency masking [22] with maximum time mask of 192 frames and maximum frequency mask of 48 bins for all the experiments.

Label Enhancement — Label noise is prevalent in both FSD50K and AudioSet, hence we adopted the enhanced labels proposed in [20].

Training details — We used an initial learning rate of 5e-4 and linear learning rate warm-up strategy for the first 1,000 iterations. The learning rate was halved every 5 epochs after the 10th epoch for FSD50K experiments and after every 5 epochs after the 35th and 10th epoch for the AudioSet experiments. All models were trained for 50 epochs on FSD50K and 60 epochs on AudioSet with batch size of 24 along with the Adam optimizer [23] and binary cross-entropy.

IV. RESULTS AND DISCUSSION

In this section we present the results of our experiments. Tables II and III showcase the results obtained on the evaluation set of AudioSet balanced and FSD50k. Our primary comparison for ATGNN focuses on the AST and PSLA models, both of which have achieved state-of-the-art results on the AudioSet and FSD50K datasets. Table II illustrates that the isotropic version of ATGNN achieves performance comparable to the AST model, while also surpassing the PSLA model. Moreover, both the pyramid versions of ATGNN outperform the AST model. Our results for AST differs from the results reported in the original paper [2] due to mismatch in our training set size with [20]. hence it was not possible to use the label enhancement files provided by [20].

TABLE II
RESULTS ON AUDIOSET-BALANCED. $*$ INDICATES OUR RUN.

Model	#Params	#GFLOPs	mAP
PANNS [24]	81.0 M	29.67	0.278
PSLA [20] $*$	13.6 M	1.09	0.308
AST [2] $*$	88.7 M	48.67	0.330
ATGNN-iso$*$	38.7 M	14.93	0.330
ATGNN-pyr-s$*$	36.4 M	14.72	0.335
ATGNN-pyr-s (-MLG)$*$	27.3 M	12.94	0.331
ATGNN-pyr-med$*$	62.7 M	27.2	**0.336**
ATGNN-pyr-med (-MLG)$*$	51.2 M	24.93	0.332

TABLE III
RESULTS ON FSD50K. $*$ INDICATES OUR RUN.

Model	#Params	#GFLOPs	mAP
FSD50K Baseline [19]	0.27M	-	0.434
Wav2CLIP [25]	-	-	0.431
Audio Transformers [26]	2.3M	-	0.537
PSLA [20]$*$	13.6M	1.09	0.559
AST [2]$*$	88.7M	48.67	0.572
ATGNN-iso$*$	38.7M	14.93	0.570
ATGNN-pyr-s$*$	36.4M	14.72	0.583
ATGNN-pyr-s (-MLG)$*$	27.3M	12.94	0.579
ATGNN-pyr-med$*$	62.7M	27.2	**0.585**
ATGNN-pyr-med$*$(-MLG)	51.2M	24.93	0.580

The findings for the FSD50K dataset show a similar trend. Here, the isotropic architecture of ATGNN competes well with the AST model, while the pyramid architecture surpasses both PSLA and AST in performance. As shown in Tables II and III, the MLG block brings an additional benefit of ≈ 0.4 mAP to the overall score, implying that the SGN block can be used as a standalone model for obtaining comparable results to the SOTA models for audio classification tasks.

V. CONCLUSIONS

We introduced an end-to-end Graph Neural Network (GNN) model for audio tagging, adept at learning both feature and label correlations. This model combines a CNN for extracting local features with graph convolution operations to map global contexts. Crucially, it learns cross-correlations between label and feature embeddings, consistently surpassing state-of-the-art models on two major audio classification benchmarks. Future research will explore new approaches to graph construction, such as using visibility graphs for time series or viewing spectrograms as 3D-point clouds. Our work aims to position graph-based models as a potent alternative to convolution and attention-based architectures in the realm of audio classification.

REFERENCES

[1] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[2] Y. Gong, Y.-A. Chung, and J. Glass, "Ast: Audio spectrogram transformer," *arXiv preprint arXiv:2104.01778*, 2021.

[3] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[4] X. Liu, H. Peng, N. Zheng, Y. Yang, H. Hu, and Y. Yuan, "Efficientvit: Memory efficient vision transformer with cascaded group attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 420–14 430.

[5] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5177–5186.

[6] H. Shrivastava, Y. Yin, R. R. Shah, and R. Zimmermann, "Mt-gcn for multi-label audio-tagging with noisy labels," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 136–140.

[7] K. Han, Y. Wang, J. Guo, Y. Tang, and E. Wu, "Vision gnn: An image is worth graph of nodes," in *NeurIPS*, 2022.

[8] R. Yao, S. Jin, W. Liu, C. Qian, P. Luo, and J. Wu, "Ml-vig: Multi-label image recognition with vision graph convolutional network."

[9] J. Liu, Y. Song, L. Wang, J. Dang, and R. Yu, "Time-frequency representation learning with graph convolutional network for dialogue-level speech emotion recognition." in *Interspeech*, 2021, pp. 4523–4527.

[10] Y. Kwon, H.-S. Heo, J.-w. Jung, Y. J. Kim, B.-J. Lee, and J. S. Chung, "Multi-scale speaker embedding-based graph attention networks for speaker diarisation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 8367–8371.

[11] J. Wang, X. Xiao, J. Wu, R. Ramamurthy, F. Rudzicz, and M. Brudno, "Speaker diarization with session-level speaker embedding refinement using graph neural networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7109–7113.

[12] Y. Sun and S. Ghaffarzadegan, "An ontology-aware framework for audio event classification," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 321–325.

[13] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[16] 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.

[17] 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*, 2021, pp. 10 012–10 022.

[18] J. F. Gemmeke, D. P. 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 *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[19] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "Fsd50k: an open dataset of human-labeled sound events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 829–852, 2021.

[20] 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*, vol. 29, pp. 3292–3306, 2021. [Online]. Available: https://doi.org/10.1109%2Ftaslp.2021.3120633

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[22] 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.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] 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.

[25] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2clip: Learning robust audio representations from clip," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4563–4567.

[26] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.