Exploring Regression-based Approach for Sound Event Detection in Noisy Environments

Soham Dinesh Tiwari
Department of Computer Science & Engineering
Manipal Institute of Technology
Manipal, India 576104

Karanth Shyam Subraya
Department of Computer Science & Engineering
Manipal Institute of Technology
Manipal, India 576104

***Abstract*—Sound-event detection enables machines to detect when a particular sound event has occurred in addition to classifying the type of event. Successful detection of various sound events is paramount in building secure surveillance systems and other smart home appliances. However, noisy events and environments exacerbate the performance of many sound event detection models, rendering them ineffective in real-world scenarios. Hence, the need for robust sound event detection algorithms in noisy environments with low inference times arises. You Only Hear Once (YOHO) is a purely convolutional architecture that uses a regression-based approach for sound-event-detection instead of the more common, frame-wise classification-based approach. The YOHO architecture proved robust in noisy environments, outperforming convolutional recurrent neural networks popular in sound event detection systems. Additionally, different ways to enhance the performance of the YOHO architecture are explored, experimenting with different computer vision architectures, dynamic convolutional layers, pretrained audio neural networks and data augmentation methods to help improve the performance of the models on noisy data. Amongst several modifications to the YOHO architecture, the Frequency Dynamic Convolution Layers helped improve the internal model data representations by enforcing frequency-dependent convolution operations, which helped improve YOHO performance on noisy audios in outdoor and vehicular environments. Similarly, the FilterAugment data augmentation method and Convolutional Block Attention Module helped improve YOHO's performance on the VOICe dataset containing noisy audios by augmenting the data and improving internal model representations of the input audio data using attention, respectively.**

Keywords*—Sound Event Detection (SED); sound event classification; frequency dynamic convolution; audio processing; FilterAugment; data augmentation; vision transformers; Pretrained Audio Neural Networks (PANN); Convolutional Block Attention Module (CBAM)

I. Introduction

In machine learning, sound event detection (SED) identifies the different sounds in an audio file and identifies the start and end time of a particular sound event in the audio. Various applications use SED, such as speech recognition, audio surveillance [1], and context-based indexing and data retrieval in a multimedia database [2].

Most of the research and development in SED today is focused on building sound event detection systems that can be trained using weakly labelled data, i.e., audios without timestamps for the occurrence of each event or unlabelled data [3]–[5]. Hence, there is an increased focus on using models that employ semi-supervised learning [4], [6], [7] to learn from weakly labelled and unlabelled data. Moreover, most of these works use sequential models or transformer architectures to leverage the sequential nature of audio data [8]–[10].

However, the consequence of using sequential and transformer-based architectures is increased model complexity, machine computation requirements, and inference times. In addition, the audios from various sources are seldom devoid of any interfering noise or disturbance in real-life. Consequently, this makes such models unsuitable for deployment in smart devices, which have constraints on the compute available and often are required to make inferences in real-time, in addition to often operating in noisy environments. Architectures with short inference times, low model parameters, and high accuracy enable smart devices to deliver accurate insights more quickly.

The You Only Hear Once (YOHO) architecture proposed by S. Venkatesh et al. draws inspiration from the famous computer vision architecture - You Only Look Once (YOLO) [11] and only makes use of different forms of convolutions, with no sequential layers. The YOHO algorithm matches the performance of the various state-of-the-art algorithms on datasets such as Music Speech Detection Dataset [12], TUT Sound Event [13], and Urban-SED datasets [14] and at lower inference times. The fast inference can be ascribed to YOHO's regression-based approach, which takes the entire audio stream as input and predicts each audio event's start and end time using regression instead of performing framewise classification.

Hence, this work tests the performance of the You Only Hear Once (YOHO) [15] algorithm on noisy audio data from different acoustic environments like indoors, outdoors, and in vehicles. The experiments show that the standard YOHO architecture is better than other popular sound event detection architectures when tackling noisy audios. Thus, we used the regression-based sound event detection approach from YOHO and combined it with other computer vision architectures, pretrained audio neural networks and methods like attention, data augmentation and dynamic convolutions to increase model performance in noisy environments. However, it was difficult to improve on the standard YOHO architecture's F1 scores. In the end, we were able to improve the performance of YOHO on the VOICe dataset by replacing the standard convolutional layers in the architecture with frequency dynamic convolution layers which helped mitigate translational invariance along the frequency axis of the log-Mel spectrograms. We also made use of the FilterAugment data augmentation method and Convolutional Block Attention Module (CBAM) to improve YOHO's

F1 scores for specific noise environments. Hence, in the end we modified the novel, purely convolutional, regression-based architecture proposed by S. Venkatesh et al. [15] to use attention, better internal data representations and data augmentation to further improve YOHO's SED performance on noisy data with a negligible increase in the model parameters.

The first section, "Introduction" explains the motivation for undertaking this work. The second chapter, "Background Theory and Literature Review", deals with the different concepts used in this work and discusses relevant research. The third chapter, "Methodology and Implementation Details", elaborates on the architecture designs and the steps undertaken in the experiments. The fourth chapter, "Results and Analysis" delineates and analyses the results. The fifth and ultimate chapter, "Conclusion and Future Work" provides a gist of the future scope of the research and possible technical improvements.

II. Background Theory and Literature Review

This section discusses the current state of research in sound event detection. It also elaborates on the different architectures and techniques in addition to the dataset and various attention and activation functions used in this work.

A. State of Research in SED

Early approaches for SED had adapted techniques from music information retrieval and speech recognition, such as Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) [16]. However, HMMs could not deal with polyphonic audio containing co-occurring sound events.

The advent of deep learning demonstrated the adeptness of deep neural networks in multi-label sound event classification and dealing with polyphonic audio. While a simple feed-forward neural network (FFN) could perform multi-label classification, it was limited in its ability to model the temporal information in signals. Consequently, plain FNNs did not gain popularity for use in SED applications.

In 2017, Cakir et al. showed that the Convolutional Recurrent neural networks (CRNNs) [8] architectures were well suited for the SED task. The recurrent neural networks (RNNs) combined with convolutional neural networks (CNNs) could capture the audio's local and global context.

The following aspects helped the CRNN architecture perform well in the SED task:

- Using successive blocks of convolutional layers and non-linear transformations, the architecture learned distinguishing characteristics in the input log-Mel spectrograms and generated feature embeddings.
- The recurrent layers would then model the temporal dependencies in the feature embeddings from the convolutional layers.
- The framewise classification approach was used to predict the existence of audio events in each audio frame.

However, a drawback of the CRNN architecture was that processing times were higher, resulting from the sequential layers preventing the architecture from fully utilising GPUs' parallel computation.

To shorten inference durations without compromising high precision and recall, S. Venkatesh et al. introduced the You Only Hear Once (YOHO) architecture for SED [15]. The You Only Look Once (YOLO) algorithm, which is popular in Computer Vision, served as an inspiration for the YOHO algorithm. YOLO significantly reduced the processing durations for the input images and enabled real-time object recognition by changing the prediction of object bounding boxes from a classification problem to a regression problem. Similarly, YOHO transforms the frame-based, multi-label classification problem of acoustic boundary detection into a regression problem. To achieve this, multiple sets of three different output neurons were used, one for each class. The presence of an event class is detected by one neuron, and its start and endpoints are predicted by the other two in each set for the particular class. As a result, on numerous datasets, the YOHO article showed a higher F-measure and lower error rate than CRNNs [15].

The current focus of the majority of SED research and development is building sound event detection systems that can be trained using weakly labelled data (audios without timestamps for the occurrence of each event) or unlabelled data [3]–[5] As a result, semi-supervised learning models are being used more frequently [4], [6], [7] to learn from weakly labelled and unlabelled data. Additionally, the majority of these works take advantage of the sequential aspect of audio data by using sequential models or transformer architectures [7]–[9]. However, this work aims to explore the regression-based, supervised learning approach for SED; hence, semi-supervised learning approaches for SED are not a focus of this work.

B. VOICe Dataset

VOICe is a new dataset for developing and evaluating generalizable sound event detection and domain adaptation methods. VOICe is offered for sound event detection domain adaptation from one acoustic scene to another or between sound events with background and without background noise [17]. The VOICe dataset consists of 207 different audio mixtures containing audios with three different sound event labels:

1) babycry
2) gunshot
3) glassbreak

These audio events are then superimposed with audios from the following 3 acoustic scenes:

1) outdoor
2) indoor
3) vehicle

The noisy audios are mixed at 2 different sound-to-noise-ratio (SNR) levels:

1) -3 dB
2) -9 dB

The dataset also consists of 207 audio mixtures devoid of background noise. Consequently, there are a total of 1449 audio mixtures - 207 mixtures x 3 acoustic scenes x 2 SNRs

= 1242 noisy audios + 207 clean audios. These audio files are divided into "training", "testing", and "validation" sets. The testing and validation audio files are 60 seconds long, whereas the training audio files last about 180 seconds.

We used the noisier SNR -9 dB audios from the VOICe dataset [17] to assess the performance of YOHO and other architectures in this work on noisy audio. It is a dataset that has been artificially constructed utilising noise from various acoustic scenes and sound events from the TAU Urban Acoustic Scenes 2019 development set and TUT Rare Sound Events 2017 development set, respectively. We would have an algorithm that is resilient to noise and performs sound event recognition considerably faster than other contemporary algorithms if it can perform well on noisy data and provide scores at least matching the CRNN algorithm. Consequently, it would become a candidate model for portable sound event detection devices used in natural, noisy environments.

C. YOHO Algorithm

Instead of using the conventional, bigger YOLO architecture, S. Venkatesh et al. modified the final layers of the MobileNet [15] architecture to implement the YOHO architecture. As can be seen in Table I, YOHO is purely a convolutional neural network. The initial MobileNet architectural layers are retained in the initial half of the table. Layers adjusted for the target dataset are in the latter half.

Consequently, YOHO has faster inference times than designs with sequential layers like CRNNs because it is built as a solely convolutional neural network with no recurrent layers. The probability and the beginning and ending times of each event label are output from the neural network, which takes as input log-Mel spectrograms of the audios. In addition, because this end-to-end method directly predicts acoustic boundaries, it takes lesser time for post-processing and smoothing [18].

Log-Mel spectrograms are used as the YOHO model's input. Next, a 3x3 2D convolution with a stride of 2 is performed after reshaping the input, halving the time and frequency dimensions. The MobileNet design uses depth-wise separable convolutions [19] with 3x3 filters, followed by pointwise convolutions [20] with 1x1 filters. All convolutions except the final layer were followed by batch normalisation [21] and ReLU activations [22]. Every time a stride of two is used, the time and frequency dimensions are halved.

D. PANNs: Large-scale Pretrained Audio Neural Networks for Audio Pattern Recognition

It is common practice in computer vision and natural language processing for systems to be pretrained on large-scale datasets. The pretrained systems then generalise well to several downstream tasks. However, the research on building pretrained neural networks trained on large-scale audio datasets is limited. Pretrained audio neural networks (PANNs) [23] are trained on the large-scale AudioSet dataset [24]. These PANNs are transferred to other downstream audio-related tasks. The best PANN system achieved mean average precision (mAP) of 0.439 on AudioSet tagging, outperforming the previous state-of-the-art result of 0.392 [23].

E. ViT: Vision Transformer

The Vision Transformer, or ViT [25], is a transformer-based model architecture for image classification. Each image is split into fixed-size patches. The patches are then converted to linear embeddings using a linear transformation. The linear embeddings are then added to the positional embedding, and the resulting sequence of vectors is fed to a standard Transformer encoder. Finally, an extra learnable "classification token" in the output sequence is used to perform image classification [25].

F. CoAtNet: Marrying convolution and Attention for All Data Sizes

Vanilla Vision Transformers lack the inductive biases that traditional convolutional networks possess. However, Vision transformers have the advantage of utilising attention over their input [26]. Hence, a model which uses convolution and attention in machine learning benefits from two fundamental aspects – higher generalisation and higher model capacity. Convolutional layers have better generalisation, while attention in the transformer layer yields higher model capacity. Hence CoAtNet [26] is a hybrid model based on two key insights:

1) By using simple relative attention, depth-wise convolution and self-attention can be naturally fused.
2) By stacking convolution layers and attention layers in a principled manner, generalisation, capacity, and efficiency are dramatically improved.

CoAtNet has the best of both convolution networks and Transformers. CoAtNet not only has the generalisation ability of convolution networks because of favourable inductive biases but also has the advantage of superior scalability from transformers resulting in faster convergence and improved efficiency [26].

G. KNNs: Kervolutional Neural Networks

KNNs [27] are an effort to establish convolution in non-linear space. Existing CNN architectures primarily leverage activation layers which only provide point-wise non-linearity. Kervolution (kernel convolution) was introduced to approximate complex behaviours of human perception systems by leveraging the kernel trick. It claims that using the kernel trick helps it generalise convolution, capture higher-order interactions of features via patch-wise kernel functions, and enhance the model capacity without introducing additional parameters. Extensive experiments showed that kervolutional neural networks (KNN) achieve higher accuracy and faster convergence than baseline CNN [27].

H. CBAM: Convolutional Block Attention Module

Convolutional Block Attention Module (CBAM) [28] is an attention module designed for convolutional neural networks. The module sequentially computes attention maps along the channel and spatial dimensions given an intermediate feature map. The input feature maps are then multiplied by the attention maps to refine the adaptive features [28].

I. Frequency Dynamic Convolutions

When a pattern moves along the time axis in the time-frequency domain, it sounds the same because the frequency components are the same, but the timing is different. On the other hand, because the frequency component that makes up the acoustic properties of the sound event changes as it moves along the frequency axis, it would sound different [29]. The Frequency Dynamic Convolutions paper [29] highlights that due to its nature, the standard 2D convolution operation wrongly enforces translational invariance along both the time and frequency dimensions of an input log-Mel spectrogram. In contrast, log-Mel spectrograms exhibit invariance only along the time axis but not the frequency axis.

Frequency dynamic convolution [29] uses a frequency-adaptive kernel to enforce frequency-dependency on 2D convolution and improve the physical consistency of the model with sound events' time-frequency patterns. The inputs have three axes - the frequency axes, the time axes and the parallel channels. From the input, it first derives frequency-adaptive attention weights. Average pooling along the time axis is followed by two 1D convolution layers which are applied along the channel axis the input. 1D convolution was used rather than fully-connected (FC) layers to take into account neighbouring frequency components. Batch normalisation and ReLU are applied between two 1D convolution layers. The channel dimension is compressed into the number of basis kernels using 1D convolution layers. The softmax activation function is then used to normalise the values of the frequency-adaptive attention weights between zero and one.

Additionally, Softmax sets each frequency bin's weight sum to one. A temperature of 31 was applied to the softmax to achieve stable training and homogeneous learning of basis kernels. Then, using frequency-adaptive attention weights, a weighted sum of basis kernels is used to produce frequency-adaptive convolution kernels. The obtained kernel is used for frequency dynamic convolution operation just like a regular 2D convolution.

J. FilterAugment Data Augmentation

FilterAugment [30] is an improved version of frequency masking compared to SpecAugment [31]. SpecAugment involved simply masking a small time and frequency range of Mel spectrograms. While the simplicity of the time and frequency masking makes it easily adoptable during model training, they are unforgiving as they completely remove portions of information from the spectrogram.

FilterAugment was proposed to regularize acoustic models over various acoustic environments by mimicking acoustic filters. It approximates acoustic filters by applying random weights on randomly determined frequency bands, i.e., randomly increasing or decreasing the energy of these random frequency ranges of the log Mel spectrograms.

Hence, FilterAugment is an effective regularization approach for acoustic models as it extracts sound information from a wider range of frequencies and it proved that generalising the SED model over a broader frequency range enhances SED performance by a significant margin.

K. Mish Activation Function

$$f(x) = x \cdot tanh(softplus(x)) \tag{1}$$

$$softplus(x) = ln(1 + e^x) \tag{2}$$

The Mish activation function outperforms ReLU [22]. The advantages of the Mish activation function are: It has unbounded upper limit and bounded lower limit. It is a non-monotonic, self-regularized function and a self-gated function. It is continuously differentiable with infinite order. It falls under the class of $C\infty$. In contrast, ReLU falls under the class of $C0$. However, a disadvantage of Mish is that it is computationally expensive [32].

L. SERF Activation Function

Serf, or Log-Softplus ERror activation Function, is a type of activation function which is self-regularized and nonmonotonic in nature.

$$f(x) = x \cdot erf(softplus(x)) \tag{3}$$

erf is the error function. SERF outperforms Mish and ReLU on a variety of tasks like image classification, object detection, machine translation and sentiment classification on multiple datasets [33].

III. Methodology and Implementation Details

The following section describes the different modification made to each architecture for the downstream tasks as well as the different hyperparameters set for different aspects of model training. The codes have been made available on Github[1].

A. Audio Processing

The following steps are performed to process the audio files and corresponding annotations:

1) Convert stereo audio with SNR -9 dB into mono audio.
2) Read mono audio at a sample rate of 44,100 Hz, then segment it into windows, where the window length is set to 2.56s and hop length is set to 1.96s.
3) Generate model compatible output format for each window's annotations.
4) Generate log-Mel spectrograms for each window using the following parameters:
    a) Number of Mel bins = 40
    b) Number of Fast Fourier Transform components = 2048
    c) Window length for generating spectrograms = 1764 sample points
    d) Hop length for generating spectrograms = 441 sample points
    e) Minimum frequency $fmin$ of 0 Hz
    f) Maximum frequency $fmax$ of 22,050 Hz
5) Save the log-Mel spectrograms and the model compatible outputs using the $.npy$ file format for use during model training and validation.

[1]https://github.com/sohamtiwari3120/YOHO-on-VOICe

6) At the time of model training, FilterAugment [30] is optionally used to augment the existing dataset. It randomly increases or decreases the energy of random frequency regions of the log-Mel spectrogram.

B. Model Training Settings

The random seed was fixed at 0. The batch size for training the model was set at 32. We used the -9 dB audios from the VOICe dataset for training.

While training, the Adam [34] optimiser with the default learning rate set to 10^{-3}, epsilon value set to 10^{-7} and default weight decay set to 0. Moreover, a call-back was used to reduce the learning rate of the model by half during training whenever the validation loss during training did not decrease for 5 epochs. Gradient clipping was used to clip/keep the global norm values of the gradients less than or equal to 0.5.

The number of epochs was not fixed, and Early Stopping [35] regularization was utilised to stop model training if the validation loss would not decrease for 10 or more epochs.

C. Working Changes to YOHO Architecture

Each audio file is first segmented into overlapping windows with a window length of 2.56s and a hop length of 1.96s. The model input dimensions are determined by the length of the audio example and the number of Mel bins. For the VOICe dataset, the input log-Mel spectrogram shape is (257, 40), where 257 is the number of time steps and 40 is the number of Mel bins.

As shown in Table I, the output of the last 2D convolution layer is reshaped by flattening the last two dimensions. The final 1D convolution layer consisting of nine filters of unit length yields an output matrix of shape 9x9. The first dimension of the output corresponds to the time axis, and the second dimension corresponds to three neurons for each of the three classes in the VOICe dataset. The input spectrogram is divided into nine bins along the time axis. Every row in the output matrix represents the audio events' start and end times in the input spectrogram's corresponding bin.

As seen in Fig. 1, the first, fourth and seventh neurons perform binary classification at each time step to detect the presence of the respective audio events. The second and third neurons use regression to predict the start and end times of the first audio-event class. Similarly, for the fifth, sixth, eighth and ninth neurons and their respective second and third audio-event classes.

Table I describes the YOHO architecture for the VOICe dataset. Each convolution layer in the architecture makes use of 'same' padding to keep the shape of the output of the convolution layer same as that of its input.

D. Working Changes to CoAtNet + CBAM Architecture

The CoAtNet architecture expects input images which have equal height and width and three channels. Since the size of the log-Mel spectrograms of the VOICe dataset was (257, 40) , i.e., (number of time frames, number of mel bins), a learnable transposed convolution layer [36] is included before the CoAtNet architecture. The transposed convolution

TABLE I. YOHO'S ARCHITECTURE MODIFIED FOR VOICE DATASET

Layer type	Filters	Kernel Shape; Stride	Output shape
Reshape	-	-	(257, 40, 1)
Conv2D	32	(3, 3); 2	(129, 20, 32)
Conv2D-dw	-	(3, 3); 1	(129, 20, 32)
Conv2D	64	(1, 1); 1	(129, 20, 64)
Conv2D-dw	-	(3, 3); 2	(65, 10, 64)
Conv2D	128	(1, 1); 1	(65, 10, 128)
Conv2D-dw	-	(3, 3); 1	(65, 10, 128)
Conv2D	128	(1, 1); 1	(65, 10, 128)
Conv2D-dw	-	(3, 3); 2	(33, 5, 128)
Conv2D 256	256	(1, 1); 1	(33, 5, 256)
Conv2D-dw	-	(3, 3); 1	(33, 5, 256)
Conv2D 256	256	(1, 1); 1	(33, 5, 256)
Conv2D-dw	-	(3, 3); 2	(17, 3, 256)
Conv2D	512	(1, 1); 1	(17, 3, 512)
5x (Conv2D-dw; Conv2D)	- 512	(3, 3); (1, 1);	(17, 3, 512) (17, 3, 512)
Conv2D-dw	-	(3, 3); 2	(9, 2, 512)
Conv2D	1024	(1, 1); 1	(9, 2, 1024)
Conv2D-dw	-	(3, 3); 1	(9, 2, 1024)
Conv2D	1024	(1, 1); 1	(9, 2, 1024)
Conv2D-dw	-	(3, 3); 1	(9, 2, 1024)
Conv2D	512	(1, 1); 1	(9, 2, 512)
Conv2D-dw	-	(3, 3); 2	(9, 2, 512)
Conv2D	256	(1, 1); 1	(9, 2, 256)
Conv2D-dw	-	(3, 3); 1	(9, 2, 256)
Conv2D	128	(1, 1); 1	(9, 2, 128)
Reshape	-	-	(9, 256)
Conv1D	9	(1); 1	(9, 9)

layer would then reshape the input spectrogram to (257, 257) followed by a 2D convolutional layer which would increase the number of channels to 3. To this transformed input CBAM attention (with reduction factor of 2 and kernel size of 3) is applied before passing it onto the CoAtNet architecture. The "CCTT" variant [26] of CoAtNet architecture is used, i.e., two MobileNet Convolution Layers along with two Transformer layers. Finally, the output of the CoAtNet architecture is passed through a 1D Convolution layer to increase the number of channels to 9. The changes are described in Table II.

TABLE II. WORKING CHANGES TO COATNET + CBAM ARCHITECTURE

Name	Type	Output Shape
input	log-Mel spectrogram	(1, 257, 40)
make_input_square	ConvTranspose2d(...)	(1, 257, 257)
increase_channels_to_3	Conv2d(...)	(3, 256, 256)
cbam	CBAMBlock(...)	(3, 256, 256)
cn	CoAtNet(...)	(1, 9)
increase_1d_channels	Conv1d(...)	(9, 9)

E. Working Changes to ViT Architecture

The Vision Transformer (ViT) expects input images with three channels and height and width of even values since it splits each image into fixed-size patches. In our code, the patch size for ViT was set to 8. The depth of the transformer layer was set to 6, the number of heads to 16, the dimension of transformer tensors to 1024 and that of the feed-forward neural network layer to 2048.

Table III shows that the input log-Mel spectrogram is passed through a 2D Convolutional layer. This layer increases the number of channels to 3 and reduces the height of the image to an even value. The ViT layer outputs a 1D vector of 1024 values and 161 channels. Consequently, the output is passed through two successive 1D convolutional layers to obtain the final output in the desired shape (9, 9).

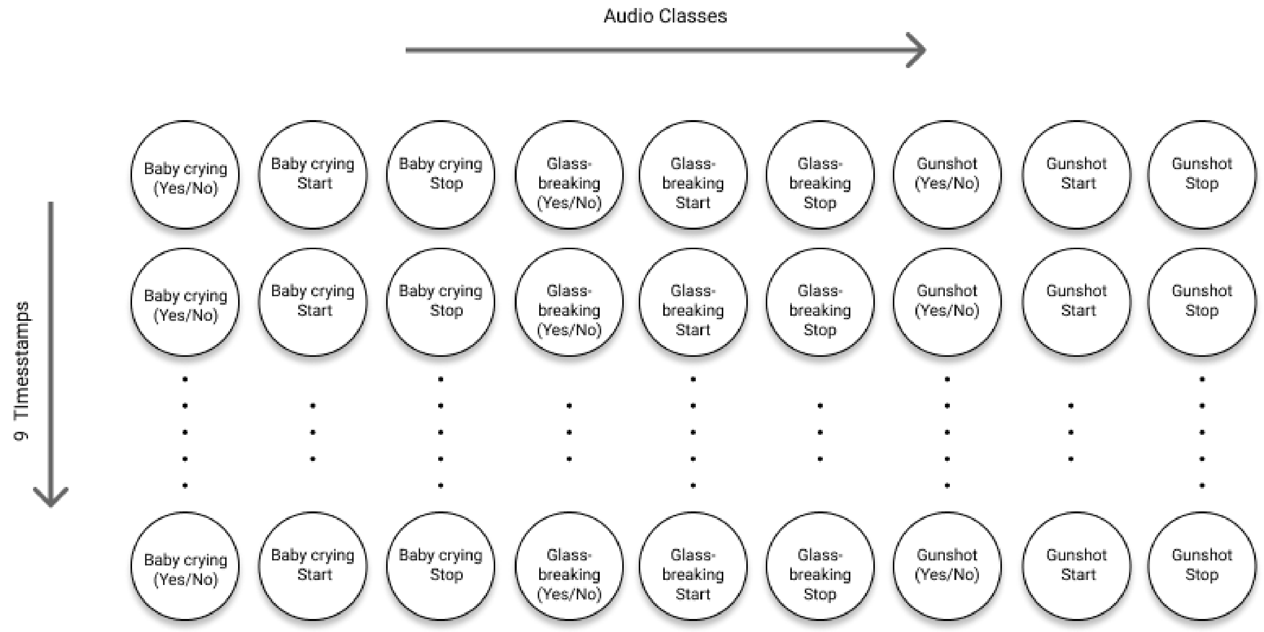

Fig. 1. Visualisation of Output Layer of YOHO for VOICe Dataset.

TABLE III. WORKING CHANGES TO VIT ARCHITECTURE

Name	Type	Output Shape
input	log-Mel spectrogram	(1, 257, 40)
increase_channels_to_3_and_reduce_height	Conv2d(...)	(3, 256, 40)
ViT	ViT(...)	(161, 1024)
head.0	Conv1d(...)	(9, 512)
head.1	Conv1d(...)	(9, 9)

F. Working Changes to YOHO + KNN Architecture

The YOHO architecture was modified by replacing every standard 2D convolution layer in the architecture by a 2D kervolution layer with a linear kernel.

G. Working Changes to YOHO + PANN Architecture

The CNN10 [23] variant of the PANN architecture expects log-Mel spectrograms with 64 Mel bins as input. Hence as described in Table IV, the input is processed and made compatible with PANN. PANN's output is then passed through a couple of transpose convolution layers followed by a couple of convolution layers to make the output compatible for YOHO. During training and inference, the weights of PANN pretrained on the large-scale AudioSet dataset were used for the CNN10 architecture.

TABLE IV. WORKING CHANGES TO YOHO + PANN ARCHITECTURE

Name	Type	Output Shape
input	log-Mel spectrogram	(1, 257, 40)
transpose	input.transpose(0, 2)	(40, 257, 1)
increase_channels_to_64	Conv2d(...)	(64, 257, 1)
transpose	input.transpose(0, 2)	(1, 257, 64)
PANN	CNN10(...)	(512, 16, 4)
transpose_conv2d_1	ConvTranspose2d(...)	(256, 16, 40)
transpose_conv2d_2	ConvTranspose2d(...)	(128, 257, 40)
reduce_channels_1	Conv2d(...)	(64, 257, 40)
reduce_channels_2	Conv2d(...)	(1, 257, 40)
YOHO	YOHO(...)	(9, 9)

H. Working Changes to YOHO + CBAM Architecture

The YOHO architecture is modified by inserting a Convolutional Block Attention Module (CBAM) before the input to every 2D depth wise convolution layer. The reduction factor was set to 2 and the kernel size set to 3 for every CBAM layer. The number of channels for each cbam layer was set to the number of channels of the output from the previous layer.

I. Working Changes to YOHO + Custom Rectangular Kernel Architecture

In an attempt to mitigate the problem of convolutional neural networks wrongly enforcing translational invariance along the frequency axis of the log-Mel spectrograms, a custom convolutional layer with "rectangular kernels" was applied before being input to the YOHO architecture. Fig. 2 shows that the layer consisted of 4 parallel 2D convolution layers:

1) Conv2D layer with filter size (3, 3)
2) Conv2D layer with filter size (log-Mel height//4, 3)
3) Conv2D layer with filter size (log-Mel height//2, 3)
4) Conv2D layer with filter size (log-Mel height*3//4,3)

J. Working Changes to YOHO + FilterAugment Architecture

The YOHO architecture remains the same. The difference is during training, FilterAugment data augmentation is used. The decibel range was set to (-6, 6), the band number range to (3, 6), minimum bandwidth to 6 and a linear filter type was used. The same set of hyperparameters are used for all experiments using FilterAugment data augmentation.

K. Working Changes to YOHO + FDY Architecture

The YOHO architecture was modified by replacing every standard 2D convolutional layer in the architecture by a 2D frequency dynamic convolution layer with the number of basis

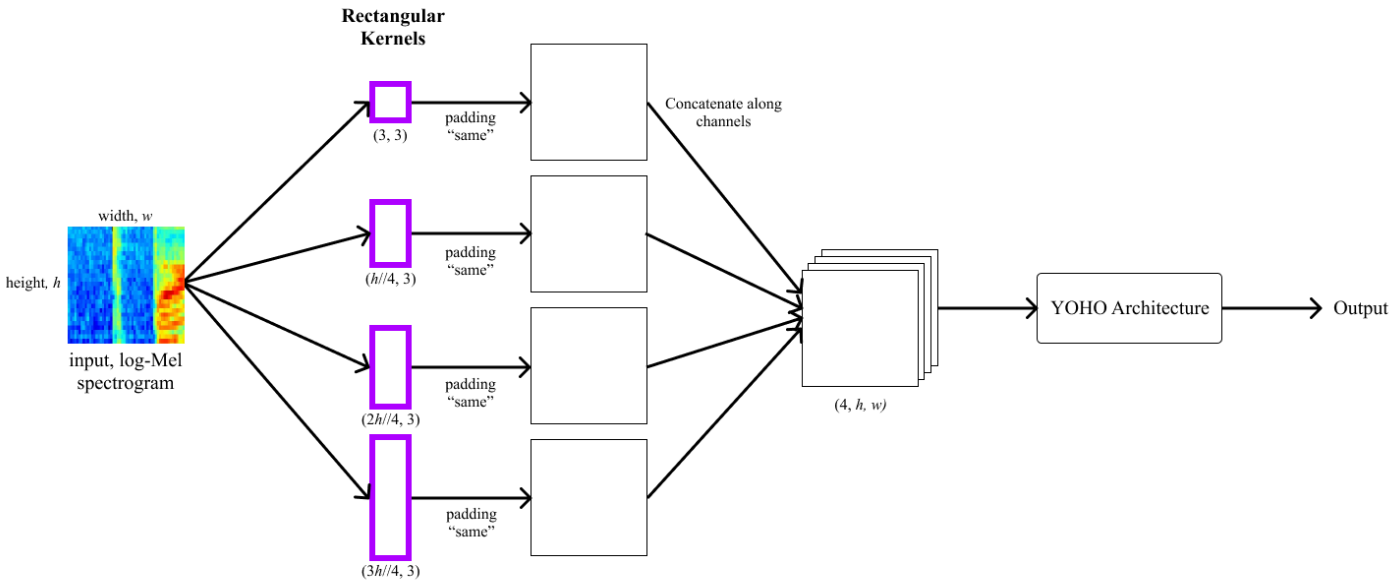

Fig. 2. Visualisation of Rectangular Kernel Layer Preceeding YOHO.

kernels set to 4 and the temperature set to 31 in every such layer. The same hyperparameters are used in all experiments using frequency dynamic convolutions.

L. Working Changes to YOHO + FDY + FilterAugment Architecture

The architecture is same as that described in the previous section, the difference being that the FilterAugment data augmentation method was being used during the training of the above architecture.

M. Working Changes to YOHO + FDY + FilterAugment + CBAM Architecture

The architecture is same as that described in the previous section, the difference being that the CBAM attention module (with reduction factor of 2 and kernel size of 3) is inserted before the input to every depth wise frequency dynamic convolutional layer.

IV. Results and Analysis

The results of all the experiments have been tabulated in Table V. More detailed results have been made available on Weights and Biases[2].

The results illustrate the following points:

1) YOHO, with an average F1 score of 0.8738, outperforms the CRNN architecture, which has an average F1 score of 0.7603, as given in [17].
2) The larger and more complex architectures, such as CoAtNet, ViT and PANN (with YOHO) with average F1 scores of 0.7589, 0.7966 and 0.8594 respectively are unable to match standard YOHO architecture's 0.8738 F1 score. This is surprising as these architectures are currently one of the best performing models in the multiple applications of Computer Vision and audio processing.
3) On average, the best performing architecture was the YOHO + CBAM architecture, with an average F1 score of 0.874. This architecture entailed adding attention to YOHO, which benefits from the inductive biases of a convolutional network. This is again surprising as the CoAtNet architecture, which combines the global attention of transformers and the inductive biases of the convolutional networks could not perform as well.
4) The proposed rectangular kernels layer (average F1 score of 0.8676), a simplistic approach to improve internal model representation of audio log-Mel spectrograms performed better than larger architectures like CoAtNet (average F1 score 0.7589), CRNN (average F1 score 0.7603), ViT (average F1 score 0.7966) and YOHO + PANN (average F1 score 0.8594) on the dataset.
5) YOHO modified with Kervolutional layers reported the lowest F1 scores in the Outdoor - 0.7883 and Vehicle - 0.8167 noise environments compared to all other variants of YOHO.
6) YOHO + FilterAugment reported the best F1 scores in the outdoor and the vehicle noise environments - 0.879 and 0.8918, respectively. In contrast, it performed the worst out of all architectures in the indoor noise environment, with an F1 score of 0.3871.
7) YOHO + FDY + FilterAugment reported very high scores in the outdoor and noise environments - 0.8746 and 0.8903, respectively. Moreover, frequency dynamic convolution layers seemed to counteract to some extent the drop in performance when using FilterAugment for indoor audios with an F1 score of 0.7681.
8) Combining YOHO + FDY + FilterAugment + CBAM did not perform as expected. Its F1 scores (0.8668, 0.8815 and 0.4534) are lesser than its individual

[2]https://wandb.ai/sohamtiwari3120/YOHO-on-VOICe?workspace=user-sohamtiwari3120

TABLE V. Results of All Experiments

Architecture	Outdoor F1	Vehicle F1	Indoor F1	Average F1
CoAtNet	0.7566	0.7719	0.7483	0.7589
CRNN	0.7500	0.8004	0.7305	0.7603
ViT	0.7902	0.8223	0.7774	0.7966
YOHO	0.8714	0.8884	**0.8615**	0.8738
YOHO + KNN	0.7883	0.8167	0.7732	0.7927
YOHO + PANN (cnn10)	0.8598	0.8712	0.8472	0.8594
YOHO + Rectangular Kernels	0.8670	0.8792	0.8567	0.8676
YOHO + CBAM	0.8724	0.8905	0.8591	**0.8740**
YOHO + FDY	0.8674	0.8864	0.8607	0.8715
YOHO + Filt_Aug	**0.8790**	**0.8918**	*0.3871*	0.7193
YOHO + FDY + FILT_AUG	0.8746	0.8903	0.7681	0.8443
YOHO + FDY + FILT_AUG + CBAM	0.8668	0.8815	*0.4534*	0.7339

components - YOHO, YOHO + CBAM, YOHO + FilterAugment and YOHO + FDY + FilterAugment.

Table V shows that the standard YOHO architecture is adept at handling noisy audios, and few architectures and other variants of YOHO could improve upon the standard architecture's scores. It seems that complex architectures with a large number of parameters performed worse than standard YOHO on the VOICe dataset. One reason could be less training data, which large architectures usually require. The VOICe dataset could be sufficient for YOHO but insufficient for the more complex architectures.

Furthermore, architectures with sequential layers and transformers like CRNN and ViT performed worse than the purely convolutional YOHO architecture and its variants. Purely convolutional architectures can better utilise GPUs and hence train faster and longer. In addition, convolutional neural networks possess inductive biases, which help them generalise to unseen datasets. Hence, another reason sequential layer-based architectures did not perform well on noisy audios could be the lack of inductive biases and poor generalizability compared to YOHO. However, the reason why CoAtNet, which combines the inductive biases of convolutional layers and the attention of transformers, performed poorly on the dataset is unknown. The poor performance of CoAtNet is confounding, especially since YOHO combined with attention in the form of CBAM performed very well on audios from all three noise environments.

On the other hand, the experiments using convolutional block attention module (CBAM), FilterAugment and Frequency Dynamic Convolution (FDY) layers hint that improving internal model representations of the input audio data and data augmentation hold the key to achieving robust sound event detection on noisy audios.

V. Conclusion and Future Work

The standard YOHO architecture is resilient to noise in audios, and its lightweight architecture with low inference times makes it a candidate model for use in portable sound event detection applications. Furthermore, this work found that combining the YOHO architecture with Frequency Dynamic Convolutions, Convolutional Block Attention Module, and FilterAugment data augmentation, helped obtain high F1 scores in specific noisy environments. This increase in performance can be attributed to improved internal model representations of the input audios with the help of data augmentation, which entailed a negligible increase in the number of parameters and model inference times.

The results suggest that improving audio representations in the model and data augmentation could help obtain better performance of SED models in noisy environments. Hence, future research can explore better representations for audios and research ways to mitigate the translational invariance along the frequency axis enforced by standard convolutional neural networks. This could involve improving the proposed rectangular kernels layer. Additionally, the models could be trained with representations from Google LEAF [37], which was one of the experiments we wanted to conduct. However, at the time of writing this paper, we could not find links to pretrained weights for the Google LEAF architecture and our compute and data were inadequate to train the architecture from scratch.

Another possible avenue of research could be to determine why the FilterAugment data augmentation method adversely affects only the audios containing noise from indoor environments while showing the opposite, helpful effects in audios containing vehicular and outdoor noise. This study could help unlock insights about noise in indoor environments and could help in the development of intelligent home audio devices.

Another direction of research would be to first train the larger and more complex architectures with more data to try and determine the ceiling for their performance in noisy environments. Furthermore, these experiments could help assess whether their performance can be improved using additional data or other factors that govern the model performance.

Finally, to determine why YOHO with convolutional block attention module performs so well and why CoAtNet does not. This could help us understand why CoAtNet's architecture, which benefits from inductive biases of convolutional networks and the global attention mechanism of transformer architectures, did not perform well in audios from noisy environments.

This work showed that the promising new regression-based, purely convolutional sound event detection architecture called You Only Hear Once can be effectively used in noisy environments. Furthermore, the regression-based approach was experimented with and used with other architectures and methods to find ways to better improve robustness to noise. The results indicate that better audio representations for the model and data augmentation techniques can help boost the performance of SED systems. Hopefully, this work will encourage more research in developing better audio representations and making audio-related models more robust to noise.

ACKNOWLEDGMENT

We want to thank the faculty of the Computer Science Department at Manipal Institute of Technology, Manipal, India, for their continued support and guidance. Special appreciation for Dr. Manjunath Mulimani (Email manjunath.mulimani@tuni.fi), who, during his tenure at Manipal Institute of Technology, guided us and helped us navigate various technical difficulties and advised possible research directions throughout the lifetime of this work. The authors would also like to thank Dr. Chng Eng Siong, Andrew Koh and Tanmay Khandelwal, part of the Speech Lab at Nanyang Technological University, Singapore.

REFERENCES

[1] S. Ntalampiras, "Audio surveillance." [Online]. Available: www.witpress.com,

[2] "Sound Event Detection - Toni Heittola." [Online]. Available: https://homepages.tuni.fi/toni.heittola/research-sound-event-detection

[3] "Sound Event Detection in Domestic Environments - DCASE." [Online]. Available: https://dcase.community/challenge2022/task-sound-event-detection-in-domestic-environments

[4] H. Dinkel, X. Cai, Z. Yan, Y. Wang, J. Zhang, and Y. Wang, "Detection and Classification of Acoustic Scenes and Events 2021 A LIGHTWEIGHT APPROACH FOR SEMI-SUPERVISED SOUND EVENT DETECTION WITH UNSUPERVISED DATA AUGMENTATION." [Online]. Available: https://github.com/pytorch/pytorch

[5] A. Cheung, Q. Tang, C.-C. Kao, M. Sun, and C. Wang, "Detection and Classification of Acoustic Scenes and Events 2021 IMPROVED STUDENT MODEL TRAINING FOR ACOUSTIC EVENT DETECTION MODELS."

[6] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah, "Large-Scale Weakly Labeled Semi-Supervised Sound Event Detection in Domestic Environments," 7 2018. [Online]. Available: https://arxiv.org/abs/1807.10501v1

[7] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Detection and Classification of Acoustic Scenes and Events 2020 CONFORMER-BASED SOUND EVENT DETECTION WITH SEMI-SUPERVISED LEARNING AND DATA AUGMENTATION."

[8] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 6 2017.

[9] D. De Benito-Gorron, D. Ramos, and D. T. Toledano, "A Multi-Resolution CRNN-Based Approach for Semi-Supervised Sound Event Detection in DCASE 2020 Challenge," *IEEE Access*, vol. 9, pp. 89 029–89 042, 2021.

[10] K. Miyazaki, T. Komatsu, T. Hayashi, S. Watanabe, T. Toda, and K. Takeda, "Detection and Classification of Acoustic Scenes and Events 2020 CONVOLUTION-AUGMENTED TRANSFORMER FOR SEMI-SUPERVISED SOUND EVENT DETECTION Technical Report."

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 779–788, 6 2015. [Online]. Available: https://arxiv.org/abs/1506.02640v5

[12] "2018:Music and/or Speech Detection - MIREX Wiki." [Online]. Available: https://music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection

[13] "Acoustic scene classification - DCASE." [Online]. Available: http://dcase.community/challenge2017/task-acoustic-scene-classification

[14] "URBAN-SED - Home." [Online]. Available: http://urbansed.weebly.com/

[15] S. Venkatesh, D. Moffat, and E. Reck Miranda, "You Only Hear Once: A YOLO-like Algorithm for Audio Segmentation and Sound Event Detection." [Online]. Available: https://github.com/satvik-venkatesh/you-only-hear-once

[16] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound Event Detection: A Tutorial."

[17] S. Gharib, K. Drossos, E. Fagerlund, and T. Virtanen, "VOICe: A Sound Event Detection Dataset For Generalizable Domain Adaptation." [Online]. Available: https://doi.org/10.5281/zenodo.3514950

[18] S. Tiwari, K. Lakhotia, and M. Mulimani, "Evaluating robustness of You Only Hear Once(YOHO) Algorithm on noisy audios in the VOICe Dataset," 11 2021. [Online]. Available: https://arxiv.org/abs/2111.01205v1

[19] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 1800–1807, 10 2016. [Online]. Available: https://arxiv.org/abs/1610.02357v3

[20] "Pointwise Convolution Explained — Papers With Code." [Online]. Available: https://paperswithcode.com/method/pointwise-convolution

[21] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2 2015. [Online]. Available: https://arxiv.org/abs/1502.03167v3

[22] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," 3 2018. [Online]. Available: https://arxiv.org/abs/1803.08375v2

[23] 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, 12 2019. [Online]. Available: https://arxiv.org/abs/1912.10211v5

[24] 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," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 776–780, 6 2017.

[25] 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," 10 2020. [Online]. Available: https://arxiv.org/abs/2010.11929v2

[26] Z. Dai, H. Liu, Q. V. Le, and M. Tan, "CoAtNet: Marrying Convolution and Attention for All Data Sizes," 6 2021. [Online]. Available: https://arxiv.org/abs/2106.04803v2

[27] C. Wang, J. Yang, L. Xie, and J. Yuan, "Kervolutional Neural Networks."

[28] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11211 LNCS, pp. 3–19, 7 2018. [Online]. Available: https://arxiv.org/abs/1807.06521v2

[29] H. Nam, S.-H. Kim, B.-Y. Ko, and Y.-H. Park, "Frequency Dynamic Convolution: Frequency-Adaptive Pattern Recognition for Sound Event Detection." [Online]. Available: https://github.com/frednam93/FDY-SED

[30] H. Nam, S.-H. Kim, and Y.-H. Park, "FilterAugment: An Acoustic Environmental Data Augmentation Method," pp. 4308–4312, 10 2021. [Online]. Available: https://arxiv.org/abs/2110.03282v4

[31] 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," *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, vol. 2019-September, pp. 2613–2617, 4 2019. [Online]. Available: http://arxiv.org/abs/1904.08779 http://dx.doi.org/10.21437/Interspeech.2019-2680

[32] D. Misra, "Mish: A Self Regularized Non-Monotonic Activation Function," 8 2019. [Online]. Available: https://arxiv.org/abs/1908.08681v3

[33] S. Nag and M. Bhattacharyya, "SERF: Towards better training of deep neural networks using log-Softplus ERror activation Function," 8 2021. [Online]. Available: https://arxiv.org/abs/2108.09598v3

[34] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[35] "Early Stopping Explained — Papers With Code." [Online]. Available: https://paperswithcode.com/method/early-stopping

[36] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," 5 2016. [Online]. Available: https://arxiv.org/abs/1605.06211v1

[37] N. Zeghidour, O. Teboul, F. de Chaumont Quitry, and M. Tagliasacchi, "{LEAF}: A learnable frontend for audio classification," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=jM76BCb6F9m