Dynamic Convolutional Neural Networks as Efficient Pre-Trained Audio Models

Florian Schmid, Khaled Koutini, and Gerhard Widmer

***Abstract*—The introduction of large-scale audio datasets, such as AudioSet, paved the way for Transformers to conquer the audio domain and replace CNNs as the state-of-the-art neural network architecture for many tasks. Audio Spectrogram Transformers are excellent at exploiting large datasets, creating powerful pre-trained models that surpass CNNs when fine-tuned on downstream tasks. However, current popular Audio Spectrogram Transformers are demanding in terms of computational complexity compared to CNNs. Recently, we have shown that, by employing Transformer-to-CNN Knowledge Distillation, efficient CNNs can catch up with and even outperform Transformers on large datasets. In this work, we extend this line of research and increase the capacity of efficient CNNs by introducing dynamic CNN blocks constructed of dynamic convolutions, a dynamic ReLU activation function, and Coordinate Attention. We show that these dynamic CNNs outperform traditional efficient CNNs, such as MobileNets, in terms of the performance–complexity trade-off at the task of audio tagging on the large-scale AudioSet. Our experiments further indicate that the proposed dynamic CNNs achieve competitive performance with Transformer-based models for end-to-end fine-tuning on downstream tasks while being much more computationally efficient.**

***Index Terms*—Dynamic convolutional neural networks, dynamic convolution, dynamic ReLU, coordinate attention, audio spectrogram transformer, audio classification, pre-trained audio models, knowledge distillation.**

I. Introduction

PRE-TRAINED deep neural networks have emerged as a pivotal paradigm in the field of machine learning over the last few years. Leveraging transfer learning techniques, pre-trained models can significantly enhance the performance on downstream tasks, in particular, when the training data is insufficient to learn an end-to-end model from scratch. Such models are typically pre-trained in a supervised or self-supervised fashion on large datasets, such as ImageNet [1] in the vision domain or AudioSet [2] in the audio domain. While convolutional neural networks (CNNs) have been the method of choice during the past years to create pre-trained models in both the audio and vision domains, Transformers [3] recently surpassed them due to their ability to scale up and exploit large datasets [4], [5]. With superior performance on the large pre-training datasets, Transformers then overtook CNNs also on many downstream tasks with smaller datasets. However, Transformers are computationally costly in training and inference, as the attention operation scales quadratically with respect to the processed sequence length. For this reason, CNNs maintain their dominance on resource-constrained platforms, such as mobile devices.

Recently, it has been shown in the audio domain that efficient CNNs can achieve competitive performance with Transformer-based models on large-scale datasets when trained using Knowledge Distillation (KD) [6], [7], [8] from a Transformer ensemble. Concretely, Schmid et al. [9] train MobileNetV3s (MNs) [10] on AudioSet using offline KD from an ensemble consisting of 9 different Patchout FaSt Spectrogram Transformer (PaSST) [11] models. The resulting efficient pre-trained MNs have been shown to extract high-quality general-purpose audio embeddings that can generalize to downstream tasks in various audio domains such as music, speech, and environmental sounds [12]. Compared to Transformers, the quality of extracted audio embeddings is comparable, while the computational cost of inference is much lower. Although the MNs achieve excellent performance on AudioSet [9] and serve as high-quality audio embedding extractors [12], they remain to be tested in end-to-end fine-tuning, in which a pre-trained model is directly fine-tuned on a downstream task.

In this work, we extend this line of research and construct computationally efficient pre-trained audio models using dynamic CNN components. The term *dynamic component* refers to a function, e.g., a layer in a neural network, that is input-adaptive. That is, the function's parameters change based on the input that is processed. Specifically, we propose dynamic MobileNets (DyMNs), obtained by integrating dynamic convolutions [13], dynamic ReLU [14], and Coordinate Attention [15] into residual inverted bottleneck blocks [16].

The motivation for integrating dynamic components into MNs is driven by (1) the desire to increase the performance of efficient models without substantially increasing their computational complexity, (2) the success of the highly dynamic self-attention operation in Transformer-based models [3], and (3) the fact that

Manuscript received 24 October 2023; revised 9 February 2024; accepted 3 March 2024. Date of publication 18 March 2024; date of current version 10 April 2024. The work of Gerhard Widmer was supported by the European Research Council (ERC) through the European Union's Horizon 2020 Research and Innovation Programme, under Grant 101019375 (Whither Music?). The computational results presented were achieved using the Linz Institute of Technology (LIT) AI Lab Cluster. The LIT AI Lab was supported by the Federal State of Upper Austria. The associate editor coordinating the review of this manuscript and approving it for publication was Zafar Rafii. *(Corresponding author: Florian Schmid.)*

Florian Schmid is with the Institute of Computational Perception (CP-JKU), Johannes Kepler University Linz, 4040 Linz, Austria (e-mail: florian.schmid@jku.at).

Khaled Koutini and Gerhard Widmer are with the LIT Artificial Intelligence Lab, Johannes Kepler University Linz, 4040 Linz, Austria, and also with the Institute of Computational Perception (CP-JKU), Johannes Kepler University Linz, 4040 Linz, Austria (e-mail: khaled.koutini@jku.at; gerhard.widmer@jku.at).

Code available: https://github.com/fschmid56/EfficientAT

Digital Object Identifier 10.1109/TASLP.2024.3376984

Squeeze-and-Excitation (SE) [17], the only dynamic component in MNs, has been shown to be an important building block for MNs [9]. Regarding (1), instead of scaling CNNs by network width and depth, which increases the computational complexity substantially, a variety of lightweight dynamic components such as convolutions [13], [18], dynamic non-linearities [14], [19], and attention mechanisms [15], [17], [20], [21], [22], [23], [24] have been proposed. These dynamic components have been shown to improve the performance while only marginally adding to the computational complexity in terms of consumed multiply-accumulate operations (MACs) at inference time. Concerning (2), self-attention, a highly dynamic operation that adapts its weights based on input data, is an integral part of the successful Transformer architecture [3]. With the integration of dynamic components, we also allow CNNs to perform input-adaptive operations, such as focusing on the relevant information in spectrograms or dynamically constructing convolutional kernels best suited for a particular ambient noise level. Regarding (3), an ablation study conducted in [9] revealed that SE [17], a component that dynamically computes channel attention weights based on input data, is an integral part of MNs to achieve high performance on AudioSet [2].

We use the KD pre-training setup of [9] and show that the proposed DyMNs achieve substantially higher pre-training performance on AudioSet compared to conventional MNs. We train MNs and DyMNs of different complexity levels on the downstream tasks of polyphonic musical instrument recognition *(OpenMic dataset* [25]), environmental sound classification *(ESC50 dataset* [26]), acoustic scene classification *(TAU Urban Acoustic Scenes 2020 challenge* [27]), and sound event tagging *(FSD50K* [28]). The results show that MNs and DyMNs can attain or even surpass the performance of a single teacher model PaSST [11] on many downstream tasks while being much more computationally efficient. The proposed DyMNs outperform MNs for most of the downstream tasks and complexity levels.

To summarize our contribution, we

- propose a dynamic CNN block by integrating dynamic convolutions [13], dynamic ReLU [14], and Coordinate Attention [15] into residual inverted bottleneck blocks [16];
- introduce a component that efficiently extracts a shared context from a block input feature map to parametrize all dynamic components in the block;
- show that the proposed DyMNs outperform traditional MNs and other CNNs from the related work on AudioSet and downstream tasks while being more computationally efficient;
- show that DyMNs, while more computationally and parameter efficient, achieve competitive performance with Audio Spectrogram Transformers on AudioSet and the downstream tasks.

The remainder of this paper is structured as follows: The related work is reviewed in Section II, followed by the introduction of the proposed dynamic model in Section III. The pre-training setup and results are presented in Section IV-A; Section V then shows the experiments and results on the downstream tasks. A detailed systematic configuration study is performed in Section VI, and the paper is concluded in Section VII.

II. Related Work

In this section, the related work is covered. We start with a general recap of efficient CNN architectures and popular dynamic CNN components that were introduced to increase a CNN's computational efficiency. We then cover the literature on pre-trained audio models to set the stage for introducing our dynamic CNN, which serves as a computationally efficient, pre-trained audio model.

A. Efficient CNN Architectures

Much effort has been invested in research on designing efficient CNNs, such as the series of MobileNets [10], [16], [29], EfficientNets [30], [31], or ShuffleNets [32], [33]. MobileNetV1 [29] substantially reduces the computational complexity of conventional convolution layers by factorizing them into a depthwise and a 1x1 pointwise convolution. On top of this, MobileNetV2 [16] introduces inverted residual blocks with linear bottlenecks, leading to better accuracy and computational efficiency. MobileNetV3 [10] adds Squeeze-and-Excitation (SE) [17] layers after the depthwise filters, upgrades activation functions using swish non-linearity [34], and optimizes the global network structure using platform-aware network architecture search [35]. EfficientNet [30] builds on the MobileNetV2 inverted residual block and introduces compound scaling laws of depth, width, and input resolution. Similar to MobileNetV3, EfficientNetV2 [31] performs a neural architecture search to optimize parameter efficiency and training speed. Originally introduced in the vision domain, MobileNets and EfficientNets have been shown to provide a good performance–complexity trade-off also in the audio domain [9], [36], [37], [38].

B. Dynamic CNN Components

While scaling CNNs by width and depth typically improves performance, it significantly increases the model's complexity. In particular, the computational demand of a CNN scales with the square of the model's width. As an alternative strategy, a lot of research has focused on using a fixed number of channels more efficiently by introducing dynamic components to CNNs. While some research on dynamic convolutions [13], [18], [39], [40], [41] and dynamic non-linearities [14], [19] has been conducted, the majority of work focuses on attention mechanisms [15], [17], [20], [21], [22], [23], [24].

Dynamic Convolution: In contrast to standard convolution layers, dynamic convolutions adapt the kernel weights based on global context information extracted from an input. Early approaches in the vision domain have explored generating the kernels directly [42], [43], resulting in a substantial increase in complexity as the number of parameters of convolution kernels is large. A more lightweight approach is to predict coefficients to linearly combine a fixed set of kernels. In this spirit, Conditionally Parameterized Convolutions for Efficient Inference (CondConv) [18] computes a linear mixture of K distinct trainable kernel weights. As shown in (1), the input x is convolved with K distinct kernels W_i, multiplied by weights α_i, and then summed up. Since convolution is a linear operation and distributes over addition, it suffices to perform a single convolution with the dynamically aggregated kernel. The weights α_i per kernel are computed dynamically from the input using Global Average Pooling (GAP), a trainable linear transformation, and a sigmoid activation function.

$$\begin{aligned}&\alpha_1(W_1 * x) + \alpha_2(W_2 * x) + \cdots + \alpha_K(W_K * x)\\&\quad = (\alpha_1 W_1 + \alpha_2 W_2 + \cdots + \alpha_K W_K) * x \qquad (1)\end{aligned}$$

Dynamic Convolution for Accelerating Convolutional Neural Networks (DyNet) [41] proposes a similar dynamic convolution mechanism, motivated from the perspective of extracting noise-irrelevant features, showing that dynamic filters generate more diverse feature maps with lower cross-correlation. The main difference to CondConv [18] is that DyNet [41] proposes to extract a shared context from the CNN block input to parametrize all dynamic convolutions in the block. Similar to CondConv, the context is extracted by GAP and a trainable linear transformation. Sharing the computation of the context across all dynamic components in a CNN block is an idea that we will also make use of when designing our proposed DyMNs.

Dynamic convolution is further developed in [13] by compressing the kernel space using the constraint $\sum_k \alpha_k = 1$ on the computed kernel attention weights. As a result, a smaller number of kernels K can be used, saving computations and parameters. As shown in (2), the constraint is enforced by applying a softmax instead of a sigmoid function. The dynamically predicted coefficients $\tilde{\alpha}_k$ are scaled by a temperature τ before the softmax to ensure near-uniform attention in early epochs.

$$\alpha_k = \frac{\exp(\tilde{\alpha}_k/\tau)}{\sum_k^K \exp(\tilde{\alpha}_k/\tau)} \qquad (2)$$

In the audio domain, recent developments of dynamic convolutions involve temporal dynamic convolutions (TDY) [44] and frequency dynamic convolutions (FDY) [45]. TDY dynamically adapts the filters along the time axis to consider time-varying characteristics of speech; FDY has been shown to improve sound event detection by dynamically adapting the filters along the frequency axis, addressing the fact that the frequency dimension is not shift-invariant. However, both TDY and FDY execute K parallel convolutions before combining the results, leading to considerable computational overhead during inference. Therefore, the dynamic convolution proposed in [13] will be used in this work rather than the more computationally costly versions TDY and FDY.

For completeness, we point out that dynamic convolution is not exclusively studied for audio and vision but has also been explored in NLP for generating input-aware convolutional filters based on the input sentences [39], [46], [47].

Dynamic Activation Function: Less explored in comparison to dynamic convolutions and CNN attention mechanisms are dynamic activation functions. Si et al. [19] dynamically adapt the threshold value of ReLU activations in an MLP network. Chen et al. extend this line of research by introducing dynamic ReLU (Dy-ReLU) [14], which works as a dynamic and efficient version of Maxout [48]. A hyperfunction similar to SE [17] dynamically predicts coefficients (slopes and intercepts) for M linear mappings. In the most common version of Dy-ReLU, the coefficients are spatially shared but differ across channels. This variant, known as Dy-ReLU-B, will be used in this work. After normalizing the coefficients to specific ranges, the element-wise maximum is applied across the M linear mappings. Specifically, given x_c, the input plane of the channel with index c, and the predicted coefficients α_c^m (slope) and β_c^m (intercept) for that channel, the output plane y_c is calculated as follows:

$$y_c = \max_{1 \leq m \leq M} \{\alpha_c^m x_c + \beta_c^m\} \qquad (3)$$

Dy-ReLU-B requires $2 * M * C$ dynamic coefficients, resulting from slope and intercept for M linear mappings for each of the C channels.

CNN Attention Mechanisms: While self-attention in Transformers [3] captures global dependencies in sequential data, CNN attention aims to emphasize relevant information in a feature map. Specifically, feature maps in 2D-CNNs consist of three dimensions corresponding to the number of channels C, the height H, and the width W of an image. CNN attention mechanisms typically compute recalibration weights for channels or spatial locations.

The most prominent instance of a CNN attention mechanism is the channel recalibration method SE [17] integrated into MobileNetV3 [10] and EfficientNets [30], [31]. As shown in (4), SE applies a squeeze (F_{sq}) and an excitation (F_{ex}) operation on an input $\mathbf{x}$ to obtain channel recalibration weights $\mathbf{s}$ of size C. F_{sq} applies GAP to collect contextual information, and F_{ex} captures channel-wise dependencies via a non-linear transformation with trainable parameters $\mathbf{W}$ followed by a sigmoid activation to compute the channel weights.

$$\mathbf{s} = F_{ex}(F_{sq}(\mathbf{x}), \mathbf{W}) \qquad (4)$$

Other channel attention mechanisms differ mainly in how they realize F_{sq} and F_{ex}. The Style-based Recalibration Module (SRM) [20] extends SE by combing GAP and global standard pooling to realize F_{sq}, followed by a channel-wise fully connected layer, batch norm [49], and a sigmoid activation function for F_{ex}. Global Context (GC) blocks [24] differ from the aforementioned attention mechanisms by performing additive instead of multiplicative recalibration. The recently introduced Global Response Normalization (GRN) [50] serves as a particular lightweight attention module by recalibrating the channels based on their L2 norms and adding only two learnable parameters, a scale and a shift.

Besides methods focusing only on recalibrating channels, other methods additionally compute attention weights for the two spatial dimensions. In this regard, the Convolutional Block Attention Module (CBAM) [21] computes attention weights for spatial locations of size HxW using a channel pooling operation to squeeze contextual information, followed by a sigmoid activation function. While channel and spatial recalibrations are applied sequentially in CBAM [21], Coordinate Attention (CA) [15] factorizes the attention mechanism into two parallel context encoding processes. Feature maps are aggregated by GAP along the spatial dimensions to obtain two slices of shapes CxH and CxW. These slices are processed separately and then used as spatially aware recalibration weights. Similarly, Triplet Attention (TA) [23] reduces the three dimensions of a feature map by average and max pooling, processes the three 2-dimensional slices separately, and recalibrates the feature map with the resulting three sets of recalibration weights.

To choose an appropriate CNN attention mechanism, we integrate different methods into MobileNetV3 [10] and evaluate the model's performance on AudioSet [2]. The results, listed in Table VIII and presented in Section VI-C1 in detail, reveal that the CA method performs best for audio tagging.

C. Pre-Trained Audio Models

Models in the audio domain are typically pre-trained in a supervised or self-supervised way on large-scale datasets, such as AudioSet [2]. AudioSet consists of around 2 million weakly labeled 10-second audio snippets downloaded from YouTube. The audio clips are manually annotated with 527 different event classes hierarchically sorted in an ontology.

Large-Scale Pretrained Audio Neural Networks for Audio Pattern Recognition (PANNs) [36] introduces a series of AudioSet-pre-trained CNNs of varying complexities and architectures widely used for downstream applications in audio-related domains such as sound event detection [51], automated audio captioning [52], language-based audio retrieval [53], emotion recognition [54], or even optical fiber sensing [55]. The prevalence of PANNs across many different downstream application areas underlines the community's interest in pre-trained audio models for end-to-end fine-tuning. The study presented in [36] includes MobileNetV2 [16], which shows a good performance-complexity trade-off but falls behind CNN14 [36] in terms of performance. Gong et al. [37] improve over PANNs in terms of performance and complexity by using an EfficientNet-B2 [30] pre-trained on ImageNet [1], balanced sampling, and label enhancement. Efficient Residual Audio Neural Networks for Audio Pattern Recognition (ERANNs) [56] improve the performance on AudioSet further while controlling efficiency through temporal downsampling via strided convolutions. However, despite all these improvements, CNNs have been substantially outperformed by supervised [5], [11], [38], [57] and self-supervised [58], [59], [60] Transformer models.

Recently, it has been shown that the sharp increase in audio tagging performance on AudioSet achieved by Transformers can be exploited and transferred to efficient CNNs using KD [6], [8]. In this context, Gong et al. [38] report that Transformers and CNNs are good teachers for each other, improving the performance of both models, and Schmid et al. [9] use Transformer-to-CNN KD to match the performance of a PaSST [11] Transformer model with a MobileNetV3 having only 6% of the parameters and requiring 100 times fewer MACs. These efficient, pre-trained CNNs have been shown to extract high-quality audio representations [12] and have a high potential to be fine-tuned for low-complexity on-device applications. In parallel to our work, which focuses on architectural improvements of efficient CNNs with dynamic components, Dinkel et al. [61] recently improved the KD setup introduced in [9] by using consistent ensemble distillation and an improved teacher model.

TABLE I
SPECIFICATION OF THE MOBILENETV3 [10] NETWORK ARCHITECTURE USED FOR THE PROPOSED DYMNS

Input	Operator	# out	# exp	SE	s
128 x 1000 x 1	conv2d 3 x 3, BN, HS	16	-	✗	2
64 x 500 x 16	IR block 3 x 3	16	16	✗	1
64 x 500 x 16	IR block 3 x 3	24	64	✗	2
32 x 250 x 24	IR block 3 x 3	24	72	✗	1
32 x 250 x 24	IR block 5 x 5	40	72	✓	2
16 x 125 x 40	IR block 5 x 5	40	120	✓	1
16 x 125 x 40	IR block 5 x 5	40	120	✓	1
16 x 125 x 40	IR block 3 x 3	80	240	✗	2
8 x 63 x 80	IR block 3 x 3	80	200	✗	1
8 x 63 x 80	IR block 3 x 3	80	184	✗	1
8 x 63 x 80	IR block 3 x 3	80	184	✗	1
8 x 63 x 80	IR block 3 x 3	112	480	✓	1
8 x 63 x 112	IR block 3 x 3	112	672	✓	1
8 x 63 x 112	IR block 5 x 5	160	672	✓	2
4 x 32 x 160	IR block 5 x 5	160	960	✓	1
4 x 32 x 160	IR block 5 x 5	160	960	✓	1
4 x 32 x 160	conv2d 1 x 1, BN, HS	960	-	✗	1
4 x 32 x 960	pool 4 x 32	960	-	✗	1
1^2 x 960	conv2d 1 x 1, HS, DROP	1280	-	✗	1
1^2 x 1280	conv2d, 1 x 1	K	-	✗	1

# out denotes the number of output channels of a specific operator, **# exp** the size of the expanded channel representation in inverted residual blocks, **SE** the optional use of Squeeze-and-Excitation [17], and **s** the stride. Input describes the input shape of an operator in terms of frequency bands × time frames × number of channels. The terms BN, HS, and DROP in the operator column abbreviate batch norm, hard swish, and dropout, respectively. The depicted numbers in the columns input, **# out**, and **# exp** are specific to a width multiplier of α = 1 and are scaled accordingly if α is adapted.

III. PROPOSED DYNAMIC MODEL

This section introduces the proposed dynamic model consisting of dynamic convolutions (Dy-Conv) [13], dynamic ReLU (Dy-ReLU) [14], and Coordinate Attention (CA) [15]. This design decision is based on our belief that the three dynamic methods are complementary: Dy-Conv can extract noise-invariant features [41]; Dy-ReLU increases the model's expressiveness by applying a dynamic non-linear function; and CA detects important channels, time frames, and frequency bins. These dynamic components are integrated into efficient inverted residual blocks (IR blocks) [16], as our focus is on creating efficient pre-trained audio models. In particular, we use the global network design of MobileNetV3-Large (MN) [10], as shown in Table I. MN is constructed of an input convolution, 15 IR blocks, GAP, and two fully-connected layers to make predictions for K classes. The spatial dimensions are downsampled using strided convolution, and the overall downsampling factor before the global pooling operation is 32 on both frequency bands and time frames. The initial input size of 128 frequency bins and 1000 time frames is modeled after the pre-processing setup for a 10-second audio recording, as described in Section IV-A. To adapt the model's complexity, MN is scaled by network width using a width multiplier α to modify the number of channels. MN is optimized for latency and has shown to provide an excellent performance–complexity trade-off on AudioSet [9].

In the following, we describe our proposed dynamic IR block in a top-down manner. We start by reviewing the conventional IR block in Section III-A and introduce the modifications that lead to the dynamic IR block in Section III-B. We then zoom into the four central components of the dynamic IR block: the Context Generation Module (Section III-C), Dy-Conv (Section III-D), Dy-ReLU (Section III-E), and CA (Section III-F). For all these additional components, we will identify the computationally most costly part in terms of MACs and compare it to the cost of convolutions in the conventional IR block.

A. Inverted Residual Block

IR blocks [16] are constructed of (1) a pointwise channel expansion convolution projecting the number of channels from C_{IN} to C_{EXP} using 1x1 kernels, (2) a depthwise convolution operating on each of the C_{EXP} channels independently, and (3) a pointwise projection convolution projecting the channels from C_{EXP} to C_{OUT} with 1x1 kernels. For most blocks, it holds that $C_{IN} = C_{OUT}$. However, transition blocks increase the number of channels, leading to $C_{OUT} > C_{IN}$. Each convolution is followed by a batch norm [49] and a non-linearity, except for the linear bottleneck after the last convolution. The depthwise convolution can be strided to downsample the spatial dimensions. If the spatial dimensions and the number of channels match, a residual connection from block input to block output is used.

The pointwise convolutions are the computationally most expensive operations in an IR block. Specifically, given a block with C_{OUT} output channels, C_{EXP} channels in the expanded channel representation, and spatial output dimensions of sizes T_{OUT} and F_{OUT}, the final pointwise convolution performs $C_{EXP} * C_{OUT} * T_{OUT} * F_{OUT}$ MACs.

B. Dynamic Inverted Residual Block

Starting from the conventional IR block, we apply the modifications listed below to integrate the dynamic components.

1) *Conv $\rightarrow$ Dy-Conv:* We replace all three convolutions in the IR block with Dy-Conv [13].
2) *ReLU $\rightarrow$ Dy-ReLU:* The ReLU activation function after the depthwise convolution is replaced by Dy-ReLU [14]. As shown in Section VI, using additional Dy-ReLUs after the two pointwise convolutions does not yield further improvements.
3) *SE $\rightarrow$ CA:* Instead of SE [17], CA [15] is used as the attention mechanism. As will be shown in Section VI, this replacement yields substantial performance gains.

Fig. 1 shows the overall structure of the dynamic IR block. All three types of dynamic components have in common the fact that they require statistics extracted from the input sample for dynamic parameterization and reweighting. We share the computation of these statistics across all dynamic components in a common Context Generation Module (CGM), which will be explained in detail in Section III-C. The CGM operates on the input of the IR block and outputs an embedded time and frequency sequence, i.e., two separate lists of embeddings for time frames and frequency bins (depicted as green blocks in Fig. 1). The following sections on Dy-Conv (Section III-D), Dy-ReLU (Section III-E), and CA (Section III-F) will explain in detail how these sequences are processed for dynamic parameterization and reweighting.

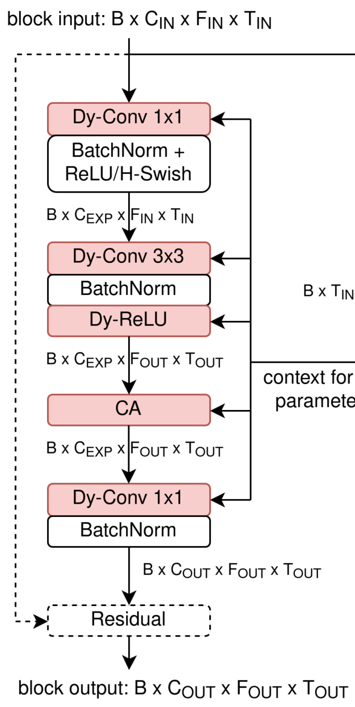
block input: B x C_IN x F_IN x T_IN
Dy-Conv 1x1
BatchNorm + ReLU/H-Swish
B x C_EXP x F_IN x T_IN
Dy-Conv 3x3
BatchNorm
Dy-ReLU
B x C_EXP x F_OUT x T_OUT
CA
B x C_EXP x F_OUT x T_OUT
Dy-Conv 1x1
BatchNorm
B x C_OUT x F_OUT x T_OUT
Residual
block output: B x C_OUT x F_OUT x T_OUT
Context Generation
S_T
S_F
B x T_IN x H
B x F_IN x H
context for dynamic parameterization

Fig. 1. Dynamic IR block: Starting from the conventional IR block, all convolutions are replaced by Dy-Conv [13]; Dy-ReLU [14] replaces the non-linear activation function after the depthwise convolution; and CA [15] is used instead of SE [17]. The Context Generation Module (CGM) operates on the block input and extracts embedded time and frequency sequences used to parametrize Dy-Convs, Dy-ReLU, and CA. Dynamic components are depicted in red, and context generation is shown in green. The shape of the input feature map size is denoted in terms of *batch size* $\times$ *channels* $\times$ *frequency bands* $\times$ *time frames*.

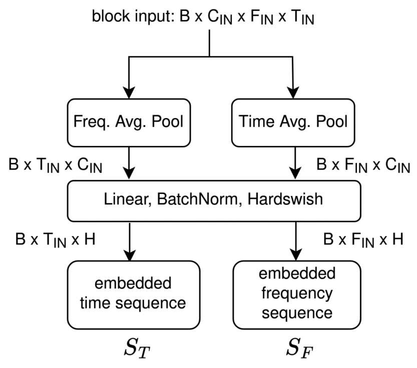

Fig. 2. Context Generation Module (CGM): a zoom into the green parts of Fig. 1. The CGM operates on the input of an IR block and outputs a time and frequency sequence embedded in a reduced channel dimension of size H. These sequences are used to parametrize Dy-Convs [13] and Dy-ReLU [14] and to compute the channel-time and channel-frequency recalibration weights for CA. The shape of the input feature map size is denoted in terms of *batch size* $\times$ *channels* $\times$ *frequency bands* $\times$ *time frames*.

C. Context Generation Module

The goal of the CGM is to collect informative statistics from an input sample that can be used to parametrize the dynamic components without creating substantial computational overhead. Fig. 2 depicts the details of this process. The CGM transforms the block input feature map into embedded time

(S_T) and frequency (S_F) sequences. Inspired by the original CA module [15], introduced in the vision domain, the input feature map is pooled separately across the two spatial dimensions to retain positional information. The resulting time and frequency sequences are then processed by a shared transformation consisting of a linear layer, a batch norm [49], and hard swish activation [10]. The linear layer embeds the channel dimension of size C_{IN} into a reduced space with H dimensions. We set H to a fraction of the expanded channel representation, such that $H = C_{EXP}/r$, where $r = 4$ in our experiments.

The computationally most expensive operation in the CGM is the linear layer, which performs $C_{IN} * H * (T_{IN} + F_{IN})$ MACs. Compared to the first pointwise convolution in the IR block, which requires $C_{IN} * C_{EXP} * T_{IN} * F_{IN}$ MACs, the computational demand of the CGM is insignificant.

D. Dy-Conv

Our implementation of Dy-Conv is based on the dynamic convolution introduced in Chen et al. [13], discussed in Section II. K different kernels are kept in parallel and are aggregated based on normalized kernel attention weights α_k. The aggregated dynamic kernel W is then constructed as a weighted sum of the K individual kernels: $W = \sum_k^K \alpha_k W_k$. Per default, we stick with the recommended settings in [13] and use $K = 4$, and linearly anneal the temperature τ for the softmax normalization, given in (2), from 30 to 1 over the first epochs of training.

We apply the transformation shown in (5) to predict the dynamic kernel attention weights from the CGM output sequences S_T and S_F. Firstly, the two sequences are concatenated and pooled over the sequence length dimension, resulting in a vector of size H. Secondly, a trainable linear transformation with parameters $W \in \mathbb{R}^{K\mathrm{x}H}$ and $b \in \mathbb{R}^K$ is used to predict the weights collected in the vector C_{dyconv}.

$$C_{\text{dyconv}} = \mathrm{Pool}(\mathrm{Concat}[S_T, S_F])W^T + b \tag{5}$$

The computationally most expensive operation is the aggregation of the K kernels, requiring at most $K * C_{EXP} * C_{OUT}$ MACs for the final pointwise convolution. Since $K = 4$ is much smaller than $T_{OUT} * F_{OUT}$, constructing the dynamic kernel is insignificant compared to the convolution itself.

E. Dy-ReLU

Our specific implementation of dynamic ReLU is the spatially-shared and channel-wise Dy-ReLU-B [14] discussed in Section II-B. Recall from (3) that Dy-ReLU-B requires a total of $2 * M * C$ dynamic coefficients to determine the shape of the non-linearity for each of the C channels.

To obtain predictions from the CGM output sequences for the $2 * M * C$ dynamic coefficients, we apply the transformation shown in (6). The difference from the Dy-Conv coefficient prediction is only in the shape of the trainable linear transformation and the resulting size of the coefficient vector C_{dyrelu}. Specifically, we use $W \in \mathbb{R}^{2*M*C_{EXP}\mathrm{x}H}$ and $b \in \mathbb{R}^{2*M*C_{EXP}}$ resulting in a vector C_{dyrelu} of size $2 * M * C$. Most commonly, $M = 2$ linear mappings are used [14], which is also the default value in our setup.

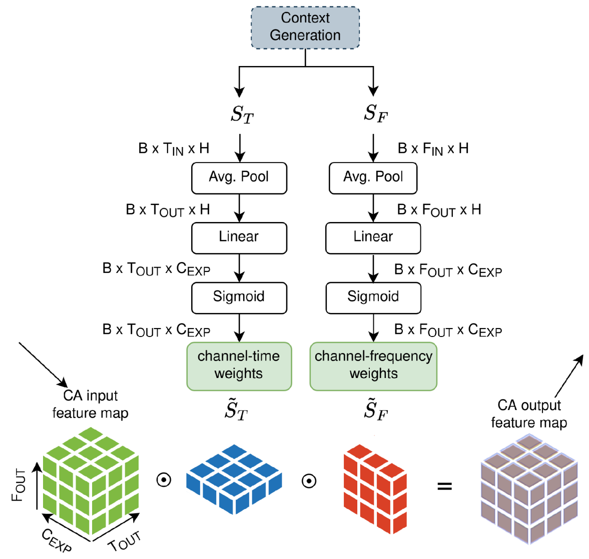

Fig. 3. Coordinate Attention (CA): CA highlights important channels, frequency bins, and time frames by recalibrating the feature map by element-wise multiplication with attention weights. It operates on the output sequences of the CGM and transforms them separately into the channel-time $\tilde{S}_T$ and channel-frequency $\tilde{S}_F$ attention weights. Average Pooling with a kernel size of 3 and a stride of 2 is used in case of a strided IR block. The linear transformation upsamples the number of channels from H to C_{EXP} to match the dimensionality of the feature map after the depthwise convolution.

$$C_{\text{dyrelu}} = \mathrm{Pool}(\mathrm{Concat}[S_T, S_F])W^T + b \tag{6}$$

The computational cost of Dy-ReLU is dominated by the calculation of elementwise linear mappings ($\alpha_c^m x_c + \beta_c^m$) as part of (3). The total cost of computing M linear mappings is $M * C_{EXP} * T_{OUT} * F_{OUT}$ MACs. Since $M = 2$ is much smaller than the number of block output channels C_{OUT}, the cost of DyReLU is insignificant compared to the cost of the final pointwise convolution ($C_{OUT} * C_{EXP} * T_{OUT} * F_{OUT}$) as discussed in Section III-A.

F. Coordinate Attention

The purpose of CA [15] is to emphasize important spatial positions and channels by recalibrating a feature map with channel-time and channel-frequency weights. As depicted in Fig. 3, CA takes as input the embedded time and frequency sequences as produced by the CGM, performs separate transformations per sequence, and outputs the respective attention weights. (7) shows the transformation from the embedded sequences $S_{\{T,F\}}$ to the respective attention weights $\tilde{S}_{\{T,F\}}$. To match the dimensions of the feature map, an average pooling operation with a kernel size of 3 and a stride of 2 is applied in case of a strided depthwise convolution in the IR block. The result is processed by a trainable linear layer with parameters $W \in \mathbb{R}^{C_{EXP}\mathrm{x}H}$ and $b \in \mathbb{R}^{C_{EXP}}$ to match the channel dimension of the feature map. The final sigmoid function converts the resulting sequences into attention weights that are used to recalibrate the feature map via elementwise multiplications.

$$\tilde{S}_{\{T,F\}} = \text{Sigmoid}(\text{Pool}(S_{\{T,F\}})W^T + b) \tag{7}$$

The computationally most expensive operations in CA are the linear layers, which jointly perform $H * C_{EXP} * (T_{OUT} + F_{OUT})$ MACs. Compared to the final pointwise convolution in the IR block, which requires $C_{OUT} * C_{EXP} * T_{OUT} * F_{OUT}$ MACs, the computational demand of CA is negligible since $C_{OUT} \approx H$ and the product of the spatial dimensions is typically much larger than their sum.

IV. Pre-Training on Large-Scale Audio Tagging

In this section, we report the pre-training results of the introduced dynamic CNNs on the task of large-scale audio tagging on AudioSet [2]. That is, models need to assign one or multiple labels out of 527 classes to 10-second audio clips. Since AudioSet must be downloaded from YouTube, different proportions of the dataset can be successfully downloaded. In this regard, our setup is strictly comparable to the dataset used in [11] and [9]. The proposed DyMN is scaled to three different complexities using width multipliers $\alpha \in \{0.4, 1, 2\}$. Adapting the width multiplier changes the number of channels in the IR blocks while keeping the total number of blocks constant. We denote the resulting models as DyMN small (DyMN-S), DyMN medium (DyMN-M), and DyMN large (DyMN-L). By default, we replace all 15 IR blocks with their dynamic counterparts; however, we will also discuss the effect of applying the dynamic IR blocks selectively in Section VI-B2. The results will be compared in terms of parameter and computational efficiency to other models pre-trained on AudioSet from the related work. In particular, we are interested in a comparison to the non-dynamic counterpart of our proposed DyMNs, the efficient MNs used in [9].

A. Pre-Training Setup

1) Preprocessing and Augmentation: To pre-train our models on AudioSet, we match the preprocessing used in [11] and [9]. We use mono audio with a sampling rate of 32 kHz and apply STFT with a window length of 25 ms and a hop size of 10 ms. Mel spectrograms are computed using a Mel filterbank with 128 frequency bins, and the minimum and maximum frequencies are randomly perturbed within a range of 10 Hz and 2 kHz, respectively. Aligned with [9], Mixup [62] with a mixing coefficient of 0.3 is the only spectrogram-level data augmentation used since we are using offline KD as described in Section IV-B and it has been shown that consistent KD is beneficial [9], [63].

2) Training: Models are trained for a total of 200 epochs, and 100,000 samples are drawn at random without replacement from the full AudioSet in each epoch. We use a learning rate scheduler consisting of an exponential warmup phase until epoch 8, followed by a constant peak learning rate phase, 95 epochs of linear rampdown, and 25 epochs of fine-tuning with 1% of the peak learning rate. The peak learning rates are set to 2×10^{-3}, 1×10^{-3}, and 5×10^{-4} for DyMN-S/M/L, respectively. We use the Adam optimizer [64] with a batch size of 120. We adopt the importance sampling strategy based on label frequency from [11] to counter the long tail of infrequent classes. The results presented in this section are achieved by DyMNs pre-trained on ImageNet [1], which has been shown to improve performance substantially [37].

3) Dynamic Component Settings: The temperature τ used in the context of Dy-Convs (see (1)) is linearly annealed from 30 to 1 in the first 30 epochs of training. The sequence embedding dimension H, as defined in the CGM (Fig. 2), is set to C_{EXP}/r with $r = 4$. However, we additionally restrict its size between 32 and 128 to ensure that the sequences can capture enough information about the feature map in the early layers and do not get unnecessarily complex in the final layers. These bounds are scaled accordingly with the model's width multiplier α.

B. Offline Knowledge Distillation

We copy the Transformer-to-CNN KD method introduced in [9] to train our DyMNs. Specifically, the DyMNs act as students in the KD setup and optimize the loss given in (8). The loss is a weighted sum of label loss L_l and distillation loss L_{kd}, traded off by a hyperparameter λ. y denotes the AudioSet labels, z_S and z_T are the student and teacher logits, respectively, δ is the sigmoid activation function, and Binary-Cross-Entropy is applied for L_l and L_{kd}.

$$Loss = \lambda L_l(\delta(z_S), y) + (1 - \lambda) L_{kd}(\delta(z_S), \delta(z_T)) \tag{8}$$

The teacher logits z_T are constructed by ensembling the logits of 9 different PaSST models [11], achieving a mean average precision (mAP) of 49.5 on the AudioSet evaluation set. Aligned with [9], we pre-compute the ensemble logits for all recordings in the training set to speed up the training of the DyMN students and use λ = 0.1 to emphasize the distillation loss.

C. Results on AudioSet

In the following, we will compare the computational efficiency and the parameter efficiency across different models trained on AudioSet. In particular, the DyMNs are trained in the same setup as the MNs in [9], which permits a fair comparison in assessing the effect of the proposed dynamic IR block. All results presented throughout this paper are averages of at least 3 independent runs and the last 10 epochs of training.

1) Computational Complexity: The number of MACs is calculated on 10-second audio recordings using the model profiler contained in Microsoft's DeepSpeed framework [66]. The results are presented in Fig. 4 and Table II. Fig. 4 plots the performance of different models against the consumed MACs and compares the proposed DyMNs (red stars) to other popular CNNs (circles) and Transformers (crosses) trained on AudioSet. In addition, a detailed comparison between DyMNs and the MNs from [9] is offered in Table II. The horizontal lines divide the table into three sections, comparing models of similar computational complexities.

The plot shows that DyMNs (red line) and MNs [9] (green line) trained in the Transformer-to-CNN KD setup [9] outperform other CNNs in terms of both prediction performance and computational efficiency. The crosses depict the Transformer models PaSST-S [11], which is used as the teacher for MNs and

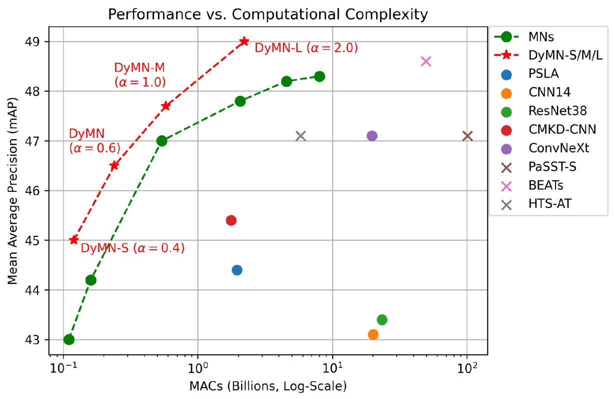

Fig. 4. Plot compares the performance – computational complexity trade-off across different single (i.e., non-ensemble) models on AudioSet. CNNs (MNs [9], PSLA [37], CNN14 [36], ResNet38 [36], CMKD-CNN [38], ConvNeXt [65]) are shown as circles; Transformer models (PaSST-S [11], BEATs [59] and HTS-AT [57]) are depicted as crosses; and our proposed DyMNs are indicated by the red stars. To aid the visual comparison, we connect the width-scaled MNs and DyMNs by lines and add a DyMN with $\alpha = 0.6$ in addition to DyMN-S/M/L. The computational complexity is measured in terms of multiply-accumulate operations (MACs) plotted in log scale on the x-axis.

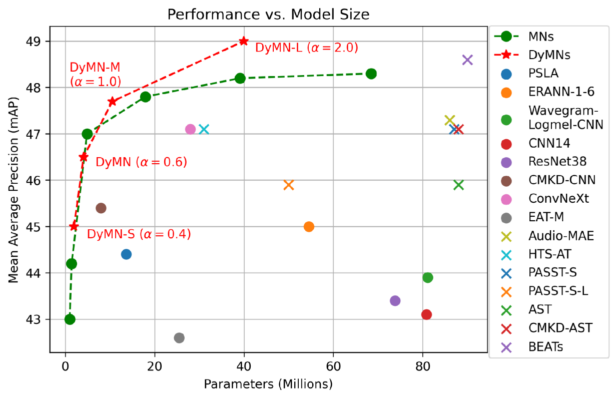

Fig. 5. Plot compares the parameter-efficiency across multiple different single models on AudioSet [2]. CNNs (MNs [9], PSLA [37], ERANN [56], Wavegram-Logmel CNN [36], CNN14 [36], ResNet38 [36], CMKD-CNN [38], ConvNeXt [65], EAT-M [68]) are denoted as circles. Transformers (Audio-MAE [58], HTS-AT [57], PaSST-S [11], PaSST-S-L [11], AST [5], CMKD-AST [38], BEATs [59]) are denoted as crosses. Our proposed DyMNs are indicated by the red stars. To aid the visual comparison, we connect the width-scaled MNs and DyMNs by lines and add a DyMN with $\alpha = 0.6$ in addition to DyMN-S/M/L.

TABLE II
This Table Compares MobileNets (MNs) From [9] Using the Default IR Block to DyMNs Using the Proposed Dynamic IR Block

Model	α	**MACs**	**Params**	**mAP**
MN	0.4	111M	0.98M	43.0
MN	0.5	160M	1.43M	44.2
DyMN-S	0.4	117M	1.96M	**45.0**
MN	1.0	541M	4.88M	47.0
DyMN-M	1.0	582M	10.55M	**47.7**
MN	2.0	2,060M	17.91M	47.8
MN	3.0	4,550M	39.09M	48.2
MN	4.0	8,030M	68.43M	48.3
DyMN-L	2.0	2,220M	39.97M	**49.0**

α denotes the width multiplier used to scale MNs and DyMNs. Multiply-accumulate operations (MACs) and Parameters are specified in millions, and mAP denotes the performance on AudioSet [2]. The presented models are grouped into three categories of similar computational complexity levels. As DyMNs and MNs are trained using the same setup, the performance difference can be attributed to the proposed dynamic IR block.
The best values are highlighted in bold.

DyMNs;[1] BEATs [59], which achieves the best performance on AudioSet; and HTS-AT [57], a particularly efficient implementation based on the Swin Transformer [67]. While both DyMNs and MNs can outperform a single-teacher model, the PaSST ensemble teacher performance of 49.5 mAP is not reached. However, DyMN-L outperforms even the best-performing Transformer model BEATs [59] while requiring less than 5% of its MACs.

Table II shows that integrating the dynamic components only marginally increases the number of consumed MACs. Using the dynamic instead of the conventional IR blocks leads to a computational overhead of around 7–10% for DyMN-S/M/L compared to MNs with the same number of channels, i.e., the same value for α. However, the results show that dynamic IR blocks can boost performance substantially, with DyMN-S/M/L improving over their static counterpart with a matching width multiplier by 2.0, 0.7, and 1.2 points in mAP, respectively. In particular, it should be pointed out that using the dynamic IR blocks is more efficient than scaling up the number of channels. DyMN-S ($\alpha = 0.4$) has 25% fewer MACs and improves performance by 0.8 points in mAP compared to MN ($\alpha = 0.5$), DyMN-M ($\alpha = 1.0$) almost matches the performance of the MN with twice the width ($\alpha = 2.0$), and DyMN-L ($\alpha = 2.0$) outperforms even the largest MN ($\alpha = 4.0$) while requiring approximately 4 times less MACs.

2) Parameter Complexity: The parameter efficiency on AudioSet is compared across different CNNs (circles) and Transformers (crosses) in Fig. 5. Both MNs and DyMNs outperform a variety of Transformers and CNNs in terms of the performance–complexity trade-off. BEATs [59] is the only model that outperforms the best MNs, but DyMN-L achieves a higher mAP with less than half the number of parameters.

Although the dynamic convolutions require more than K times as many parameters as conventional convolution layers, and the fully connected layers predicting the Dy-ReLU activation coefficients are also non-negligible, DyMNs are competitive with MNs in terms of parameter efficiency. In particular, for larger width multipliers, Fig. 5 shows that using dynamic IR blocks instead of increasing α, which increases the number of parameters approximately by α^2, is more efficient. Specifically, the results in Table II show that the performance of MNs only increases marginally when using α values > 2. However, the performance can be boosted substantially using the proposed dynamic IR blocks, with DyMN-L outperforming MN ($\alpha = 4.0$) by 0.7 points in mAP using 42% less parameters.

[1] More precisely: we show here one single Transformer model from the teacher ensemble, in order to only have comparable single models in the plot. The 9-model PaSST ensemble would figure at $(6.4 * 10^2, 49.5)$ in Fig. 4; in Fig. 5, it would completely distort the plot, with a parameter complexity of 775.3 million.

V. Experiments on Downstream Tasks

The previous section has shown that the introduction of dynamic components to MNs increases the pre-training performance on large-scale AudioSet. However, the main question is whether the performance gain during pre-training can be carried over to downstream tasks. We fine-tune AudioSet-pre-trained MNs and DyMNs on the tasks of polyphonic musical instrument recognition, environmental sound classification, sound event tagging, and acoustic scene classification and compare their performance against each other, the pre-training teacher Transformer PaSST [11], and the state of the art on the respective tasks. Furthermore, we share the same fine-tuning pipeline across all downstream tasks and only adapt the learning rate to show that no extensive hyperparameter tuning is required for high performance on downstream tasks.

TABLE III
Results on the Downstream Tasks Based on OpenMic [25], ESC50 [26], FSD50K [28], and DCASE20 [27]

Model	**Task**			
	OpenMic	ESC50	FSD50K	DCASE20
Baseline	79.5 [25]	76.9 [26]	43.4 [28]	54.1 [27]
SOTA	84.3 [11]	**98.1** [59]	**69.7** [71]	**76.6** [73]
SOTA CNN	83.1 [69]	96.7 [70]	58.6 [70]	73.7 [72]
PaSST-S	84.3	96.8	65.3	75.6
MN ($\alpha = 0.4$)	82.0	93.2	59.7	69.5
DyMN-S	83.6	96.4	61.9	72.9
MN ($\alpha = 1.0$)	83.8	96.4	65.1	72.1
DyMN-M	84.4	96.4	64.2	73.6
MN ($\alpha = 2.0$)	84.7	96.9	65.4	73.2
DyMN-L	**85.5**	97.4	65.5	75.7
MN ($\alpha = 3.0$)	84.8	96.8	65.6	73.5

OpenMic and FSD50K use mean average precision; ESC50 and DCASE20 use accuracy as the metric.
The best values are highlighted in bold.

A. Tasks

1) Polyphonic Musical Instrument Recognition: This task is to recognize all instruments present in an audio clip. It is based on the OpenMic dataset [25], which consists of 20,000 10-second audio clips. Each clip is annotated with multiple tags out of 20 different classes. The performance metric is mean average precision (mAP). The state of the art for this task is the Transformer model PaSST [11], which surpassed receptive-field-regularized CNNs [69] as the previous state-of-the-art method.

2) Environmental Sound Classification: This task is to classify 5-second audio recordings into one out of 50 different classes. It is based on the ESC50 [26] dataset, consisting of 2,000 environmental sound recordings. The performance metric for this task is accuracy, and we report the results in terms of average accuracy on the 5 official folds [26]. The state-of-the-art model on this dataset is the Transformer BEATs [59]; in terms of CNNs, the fine-tuned audio encoder of CLAP [70] has the lead.

3) Sound Event Tagging: The FSD50K dataset [28] consists of 51,197 recordings that are annotated with 200 event classes taken from the AudioSet [2] ontology. It is the second-largest publicly available general-purpose sound event tagging dataset after AudioSet, consisting of 100 hours of audio. FSD50K is separated into training, validation, and evaluation splits. We use the validation split to set up our fine-tuning pipeline that is shared across all models and downstream tasks. The performance is reported in terms of mAP on the evaluation set. The multi-modal, giant-size ONE-PEACE [71] Transformer with 4B parameters achieves state-of-the-art results on this dataset. On the CNN side, the fine-tuned audio encoder of CLAP [70] achieves the highest mAP.

4) Acoustic Scene Classification: This task is to classify 10-second audio recordings into one out of ten different acoustic scenes. The TAU Urban Acoustic Scenes 2020 Mobile dataset [27] has been used in the DCASE 2020 Challenge Task 1 and consists of 13,965 recordings in the train set and 2,979 in the test set. The performance is measured in terms of accuracy. It is particularly difficult to find models that generalize well on this dataset since it is recorded with a limited number of microphones, some of which are completely unseen during training and cause a distribution shift at test time. PaSST-S [11] is the state-of-the-art method on this dataset, outperforming the top CNN [72] from the DCASE 2020 Challenge [27].

B. Fine-Tuning Setup

For fine-tuning models on downstream tasks, the preprocessing of audio recordings applied in the pre-training stage, as discussed in Section IV-A, is matched. Except for the learning rate, we share the fine-tuning pipeline across all tasks and models. We use the Adam optimizer and train for 80 epochs. The learning rate schedule includes an exponential warmup phase for 10 epochs, followed by a linear rampdown for 65 epochs and 5 final epochs with 1% of the peak learning rate. The temperature τ to compute the attention weights for Dy-Conv is fixed to 1. As for data augmentation, we randomly roll the waveform over time in a maximum range of ± 125 ms. We use a two-level mixup [62], both on the raw waveforms and on the spectrogram level, and the audio waveform is multiplied to change the gain by ± 7 dB. The min and max frequencies of the mel filterbank are randomly perturbed within ranges of 10 Hz and 2 kHz, respectively. Interestingly, a critical performance factor on the downstream tasks is the weight decay, which must be set to 0 to achieve high performance.

C. Results

The results for the four downstream tasks are given in Table III. For each of the tasks, we specify the baseline performance (Baseline), global state of the art (SOTA), state of the art among CNNs (SOTA CNN), and the performance of the AudioSet teacher model PaSST-S [11]. We also compare our proposed DyMNs to the MNs with matching width multipliers and add an MN with an increased width of $\alpha = 3.0$.

1) DyMNs vs. MNs: The DyMNs outperform the MNs with matching width across all tasks and model sizes, except for $\alpha = 1.0$ on FSD50K. DyMN-L even outperforms MN ($\alpha = 3.0$) on OpenMic, ESC-50, and DCASE20 while requiring less than half of its MACs, as shown in Table II. These results underline the fact that dynamic components can increase channel efficiency and generalization performance. It further shows that using the dynamic components is a more efficient solution than simply scaling up the width by increasing α.

2) DyMNs vs. PaSST: DyMN-L outperforms the pre-training teacher model PaSST-S [11] on all four downstream tasks while requiring less than half of its parameters and less than 3% of its MACs for computing the predictions for a 10-second audio recording. DyMN-M achieves comparable performance on OpenMic and ESC-50, being 8 times smaller and requiring less than 1% of the number of MACs compared to PaSST.

3) DyMNs vs. SOTA CNN: DyMN-L beats the state-of-the-art CNN performance on all four downstream tasks. On FSD50K, even the most lightweight models, DyMN-S and MN ($\alpha = 0.4$), outperform the top CNNs such as PSLA [37] (56.7), CMKD-CNN [38] (58.2) or CLAP [70] (58.6). While CMKD-CNN is the smallest of the aforementioned CNNs with 8M parameters, MN ($\alpha = 0.4$) and DyMN-S are below 1M and 2M parameters, respectively.

4) DyMNs vs. SOTA: On OpenMic, DyMN-M and DyMN-L outperform the state-of-the-art performance held by the Transformer model PaSST [11]. On ESC-50, DyMN-L lags slightly behind the top method BEATs [59] but outperforms other recent Transformers such as AST [5] (95.7), PaSST [11] (96.8) or HTS-AT [57] (97.0). However, DyMN-L is much more lightweight compared to BEATs, having less than half of its parameters and less than 5% of its MACs. On FSD50K, besides the giant-size model ONE-PEACE [71] with 4B parameters, DyMN-L outperforms other recent Transformers such as PaSST [11] (65.3) and CMKD-AST [38] (61.7) and on DCASE20, DyMN-L lags only behind a specific version of PaSST (PaSST-B) that uses no patchout (76.6).

VI. Systematic Configuration Study

The purpose of this section is to gain insights into the training setup, justify the design decisions that led to the final dynamic IR block presented in Section III, and show that the proposed variant has turned out to be beneficial across a variety of other configurations.

In the following, we assess the impact of KD and ImageNet pre-training in Section VI-A. We then present configuration studies for the proposed dynamic IR block in Section VI-B and delve into the details of the individual dynamic components in Sections VI-C, VI-D, and VI-E, followed by a study on different context generation variants in Section VI-F. All experiments are conducted on AudioSet [2] using DyMN-M and MN with the default width of $\alpha = 1.0$. Except for the results presented in Section VI-A, the results are obtained without ImageNet pre-training. In the following tables, the default values in our setup are indicated in bold. The tables also list the number of MACs and parameters in millions for each experiment. In this regard, we want to remind the reader that special attention should be paid to the number of MACs, as computational efficiency is the main objective of this work.

A. *Impact of Knowledge Distillation and ImageNet Pre-Training*

The purpose of this section is to study the impact of KD and ImageNet pre-training on MNs and DyMNs. For experiments outside of KD, the training setup needs to be adapted in order to account for the changed training objective. In this regard, we use a learning rate of 5×10^{-4} and include the data augmentation techniques described in Section V-B. Additionally, we apply SpecAugment [74], similar to [11]. Based on the results presented in Table IV, we can make the following three observations:

TABLE IV
Comparing the Performance Impact of ImageNet Pre-Training (PT) and Knowledge Distillation (KD) on MNs and DyMNs

Architecture	*KD*	*PT*	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	✓	✓	582M	10.55M	**47.7**	*Ref. Val.*
MN [9]	✓	✓	541M	4.88M	47.0	–0.7
DyMN	✓	✗	582M	10.55M	47.5	–0.2
MN [9]	✓	✗	541M	4.88M	45.8	–1.9
DyMN	✗	✓	582M	10.55M	43.2	–4.5
MN	✗	✓	541M	4.88M	44.1	–3.6
DyMN	✗	✗	582M	10.55M	43.0	–4.7
MN	✗	✗	541M	4.88M	42.4	–5.3

The default values in our setup are indicated in bold.

1) In line with [9], KD is a very important ingredient in the training setup and accounts for a performance improvement of 2.9 and 4.5 points mAP for MN and DyMN, respectively. The difference between the improvements underlines that the increased learning capacity of DyMNs allows them to better exploit the rich training signal when distilling knowledge from a large transformer ensemble. However, we also observed that DyMNs are much more prone to overfitting than MNs when trained outside of KD.
2) While the improvements of ImageNet pre-training are marginal for DyMN with and without KD (0.2 points mAP), the benefits are much larger for MN (1.2 and 1.7 points mAP, respectively). We therefore conclude that the dynamic components learn domain-specific properties and are less robust to the modality shift from the vision to the audio domain.
3) KD, pre-training, and dynamic components are complementary. For the proposed DyMNs, KD contributes the largest performance improvement (4.5 points mAP), followed by the dynamic components (0.7 points mAP) and pre-training (0.2 points mAP).

B. *Dynamic IR Block*

In this section, we perform a configuration study based on the proposed dynamic IR block. We investigate the effect of the individual dynamic components, the impact of applying the dynamic IR blocks selectively, and the effect of applying Dy-Conv and Dy-ReLU at different positions in the block.

1) Importance of Dynamic Components: Table V presents the results for the proposed DyMN, the MobileNetV3 [10] Baseline (MN Baseline) from [9], a fully static MN with no SE [17] (MN Static), and all other combinations of the three dynamic components in DyMNs.

The results show that dynamic, input-dependent processing is important. The proposed DyMN improves the performance by 3.1 points mAP over the static MN. While Dy-ReLU and

TABLE V
IMPORTANCE OF DY-CONV, DY-RELU, AND CA IN THE PROPOSED DYNAMIC IR BLOCK

Dynamic Components	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	582M	10.55M	**47.5**	*Ref. Val.*
MN Baseline [9]	541M	4.88M	45.8	–1.7
MN Static (no SE)	540M	3.36M	44.4	–3.1
– Dy-Conv	577M	6.69M	47.2	–0.3
– Dy-ReLU	580M	8.40M	46.9	–0.6
– CA	554M	9.47M	46.9	–0.6
– Dy-Conv, Dy-ReLU	574M	4.54M	46.7	–0.8
– Dy-Conv, CA	548M	5.62M	46.6	–0.9
– Dy-ReLU, CA	551M	7.32M	46.1	–1.4

– denotes that the respective dynamic components are removed from proposed DyMN.
The default values in our setup are indicated in bold.

TABLE VI
EFFECT OF SELECTIVE APPLICATION OF DYNAMIC IR BLOCKS

Dynamic Blocks	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	582M	10.55M	**47.5**	*Ref. Val.*
MN Baseline [9]	541M	4.88M	45.8	–1.7
First 5 blocks dynamic	543M	3.52M	46.2	–1.3
Mid 5 blocks dynamic	545M	4.09M	46.3	–1.2
Last 5 blocks dynamic	572M	9.67M	46.9	–0.6
Replace SE	574M	9.83M	47.1	–0.4

The default values in our setup are indicated in bold.

CA are of equal importance, Dy-Conv leads to the smallest improvements. However, all three dynamic components are beneficial for the overall performance and improve over MN Baseline.

2) Selectively Applying the Dynamic IR Block: The purpose of this experiment, with results summarized in Table VI, is to determine at which positions in the model the dynamic blocks have the highest impact. MN has a total of 15 IR blocks, all of which are replaced by dynamic IR blocks in the proposed DyMN. In this study, we replace only the first, middle, and last 5 blocks in the MN with dynamic IR blocks and keep conventional IR blocks at the remaining positions. Additionally, the setting *Replace SE* uses the dynamic IR block only at the positions at which the original MN uses SE, resulting in 8 out of the 15 IR blocks being dynamic.

Table VI shows that dynamic blocks are beneficial at different positions in the model. Each selective variant outperforms the MN Baseline from [9]. Applying the dynamic IR block to the first 5 layers causes the least overhead in terms of MACs and parameters, but the performance benefits the most when the last 5 layers in the network use the dynamic IR block. If, besides computational efficiency, parameter efficiency is the main concern for an application, applying the dynamic IR block only to the first layers offers better performance with fewer parameters when compared to the conventional MN architecture.

3) Effects of Dy-Conv and Dy-ReLU Positions: The proposed dynamic IR block replaces all convolution layers with Dy-Conv and uses Dy-ReLU only after the depthwise convolution. Table VII shows the results for applying Dy-Conv and Dy-ReLU at alternative positions. Pos. 1, 2, and 3 describe the first, second, and third convolutions in the dynamic IR block (shown in Fig. 1) and the activation functions that follow them. In case of Dy-ReLU at Pos. 3, we add an additional Dy-ReLU after the final pointwise convolution.

TABLE VII
EFFECT OF VARYING THE POSITION OF DY-CONVS AND DY-RELU IN THE DYNAMIC IR BLOCK

Dy-Conv/Dy-ReLU Pos.	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	582M	10.55M	**47.5**	*Ref. Val.*
Dy-Conv Pos. 2	577M	6.97M	47.1	–0.4
Dy-ReLU Pos. 1	582M	10.55M	47.2	–0.3
Dy-ReLU Pos. 2 + 1	584M	12.70M	47.4	–0.1
Dy-ReLU Pos. 2 + 3	582M	11M	47.2	–0.3

The default values in our setup are indicated in bold.

TABLE VIII
MOBILENETV3 WITH DIFFERENT ATTENTION MECHANISMS INTEGRATED INTO ALL IR BLOCKS BEFORE THE FINAL POINTWISE CONVOLUTION

Attention Method	**MACs**	**Params**	**mAP**	**Diff.**
MN Static	540M	3.36M	44.4	*Ref. Val.*
MN w. CA	615M	5.78M	**46.7**	+2.3
MN w. TA	577M	3.37M	46.1	+1.7
MN w. SRM	540M	3.38M	45.9	+1.5
MN w. SE	541M	4.97M	45.8	+1.4
MN w. GRN	540M	3.36M	45.3	+0.9
MN w. CBAM	548M	4.96M	44.6	+0.2
MN w. GC	548M	4.98M	44.2	-0.2

The default values in our setup are indicated in bold.

The results show that replacing all convolution layers with Dy-Conv is beneficial, and the proposed Dy-ReLU variant, where we have a single Dy-ReLU at Pos. 2, achieves the best performance. In particular, adding additional Dy-ReLUs does not improve results further.

C. Attention Mechanism

This section presents a study on the choice of attention mechanism and the impact of channel-frequency and channel-time recalibration in CA.

1) Choice of Attention Mechanism: Table VIII shows the results for integrating different popular attention mechanisms (CA [15], TA [23], SRM [20], GRN [50], SE [17], CBAM [21], and GC [24]) into MN. All attention mechanisms are integrated before the final pointwise convolution into all 15 IR blocks.

While several different attention methods are capable of achieving substantial improvements over the static MN with no attention mechanism, CA leads to the largest improvement and is therefore the attention mechanism of choice for our proposed dynamic IR block. Additionally, the overhead in terms of MACs caused by using CA is much smaller in the dynamic IR block since the proposed CGM extracts context from the block input (C_{IN} channels) instead of the expanded representation (C_{EXP} channels) and shares the computation of the context with Dy-ReLU and Dy-Convs.

2) Channel-Time and Channel-Frequency Recalibration in CA: CA performs recalibration of the feature map with channel-time and channel-frequency attention weights. The results given

TABLE IX
EFFECT OF APPLYING CA SELECTIVELY FOR CHANNEL-TIME AND CHANNEL-FREQUENCY RECALIBRATION

CA Recalibration	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	582M	10.55M	**47.5**	*Ref. Val.*
Only channel-frequency	557M	10.01M	47.3	–0.2
Only channel-time	579M	10.01M	47.2	–0.3

The default values in our setup are indicated in bold.

TABLE X
VARYING THE NUMBER OF DYNAMIC KERNELS K AND THE TEMPERATURE τ FOR COMPUTING THE ATTENTION WEIGHTS IN DY-CONV

DyConv	**MACs**	**Params**	**mAP**	**Diff.**
K=2	579M	7.98M	47.3	–0.2
K=**4**	582M	10.55M	**47.5**	*Ref. Val.*
K=6	584M	13.12M	47.5	0.0
τ **annealing**	582M	10.55M	**47.5**	*Ref. Val.*
$\tau = 1$	582M	10.55M	47.4	–0.1
$\tau = 10$	582M	10.55M	47.4	–0.1
$\tau = 30$	582M	10.55M	47.4	–0.1

The default values in our setup are indicated in bold.

in Table IX assess the importance of these two recalibration steps. While the channel-frequency weights are slightly more important than the channel-time weights, using both of them leads to the best results. Since the computation of channel-time weights is costly in terms of MACs compared to the other dynamic components, using only channel-frequency weights can be considered a lightweight alternative with minimal computational overhead.

D. Dy-Conv

In this section, the impact of the two hyperparameters of Dy-Conv, the number of kernels K and the temperature τ, is assessed.

1) Number of Dynamic Kernels K: The number of kernels K specifies how many different kernels are aggregated in each Dy-Conv layer. Table X shows the results for $K \in \{2, 4, 6\}$. The performance improves only marginally from $K = 2$ to $K = 4$ kernels and plateaus for larger values of K.

2) Temperate τ: The temperature τ affects the computation of kernel attention weights, as shown in (2). Aligned with [13], by default, we use a temperature schedule for τ and anneal it from 30 to 1 over the first 30 epochs of training. This ensures near-uniform attention weights in the first epochs to properly update all kernels. Table X compares the temperature schedule to the results of using a constant temperature ($\tau \in \{1, 10, 30\}$). The results show that the performance is stable across different constant temperature values in our setup. However, keeping the temperature constant leads to a slight performance decrease, underlining the advantage of using a temperature schedule.

E. Dy-ReLU

An important hyperparameter of Dy-ReLU is the number of linear mappings M that the max operation, shown in (3), acts on. The results for $M \in \{1, 2, 3\}$ are shown in Table XI. $M = 1$ results in a dynamic linear function, while $M = 2$ and $M = 3$ are dynamic non-linear functions. The non-linear functions outperform the linear function, and aligned with the findings in [14], $M = 2$ and $M = 3$, they achieve similar performance.

TABLE XI
NUMBER OF LINEAR MAPPINGS M IN DY-RELU

M	**MACs**	**Params**	**mAP**	**Diff.**
1	581M	9.47M	47.2	–0.3
2	582M	10.55M	47.5	*Ref. Val.*
3	583M	11.62M	47.4	–0.1

The default values in our setup are indicated in bold.

TABLE XII
DIFFERENT SETTINGS FOR CONTEXT GENERATION

Context Method	**MACs**	**Params**	**mAP**	**Diff.**
Proposed DyMN	582M	10.55M	**47.5**	*Ref. Val.*
no shared context	583M	12.09M	47.3	–0.2
no shared seq. parameters	582M	10.65M	47.4	–0.1
concat pooled seq.	584M	12.69M	47.4	–0.1

The default values in our setup are indicated in bold.

F. Context Generation

In this section, different variants of context generation are studied. In particular, architectural variants are discussed in Section VI-F1, and modifications of the context size H are investigated in Section VI-F2.

1) Different Architectural Variants for Context Generation: Table XII contains results for the following modifications to the context generation process:

- *no shared context:* Refers to a setting in which Dy-Conv and Dy-ReLU extract their own context by GAP and a learnable non-linear transformation, as originally proposed in [13] and [14], respectively. In this case, Dy-Conv and Dy-ReLU do not make use of the CGM output sequences. This experiment tests whether the shared CGM is capable of extracting a sufficiently rich global context that can be used to parametrize all dynamic components in a block.
- *no shared seq. parameters:* Indicates that, in contrast to the CGM setup in Fig. 2, the linear layer and batch norm parameters are not shared across the sequences. Instead, two sets of parameters are learned to transform the time and frequency sequences. The motivation for this experiment is to decouple transformations involving time and frequency information, which, in contrast to the height and width of an image, encode different physical properties.
- *concat pooled seq.:* Describes a setting for which (5) and (6) are modified. Specifically, the sequences S_T and S_F are pooled separately, and the vectors of size H are concatenated, resulting in a context vector of size $2 * H$. Aligned with the motivation of the last experiment, with this experiment, we try to avoid mixing time and frequency information.

The results presented in Table XII show that all of these modifications lead to a slight decrease in performance, despite all of them increasing the number of parameters. The proposed design of the context generation is based on the findings of these

TABLE XIII
VARYING THE SEQUENCE EMBEDDING SIZE H

Context Size	**MACs**	**Params**	**mAP**	**Diff.**
$r = 4$	582M	10.55M	**47.5**	*Ref. Val.*
$r = 8$	573M	9.85M	47.4	–0.1
$r = 16$	562M	8.69M	47.3	–0.2
$H_{MIN} = 16$	581M	10.53M	47.4	–0.1
$H_{MIN} = 32$	582M	10.55M	**47.5**	*Ref. Val.*
$H_{MIN} = 64$	587M	10.7M	47.4	–0.1
$H_{MAX} = 64$	568M	9.09M	47.2	–0.3
$H_{MAX} = 128$	582M	10.55M	**47.5**	*Ref. Val.*
$H_{MAX} = 256$	595M	12.21M	47.5	0.0

The default values in our setup are indicated in bold.

experiments; the CGM follows the feature encoding process used in CA [15], and the dynamic coefficients of Dy-Conv and Dy-ReLU are derived as defined in (5) and (6), respectively.

2) Varying the Sequence Embedding Size H: As stated in Section III-C, the embedding dimension for the time and frequency sequences, computed by the CGM, is defined as $H = C_{EXP}/r$. Additionally, H is clipped between a lower bound ($H_{MIN} = 32$) and an upper bound ($H_{MAX} = 128$) that are scaled accordingly with the model's width α. Table XIII shows the results for different values of r, H_{MIN}, and H_{MAX}. The results indicate that reducing the sequence embedding dimension H by either increasing r or decreasing H_{MIN} and H_{MAX} leads to a decrease in mAP. This shows that assigning a certain capacity to the global context is important for the dynamic components to exploit their full potential. However, increasing H by increasing the lower and upper bounds ($H_{MIN} = 64$ and $H_{MAX} = 256$) does not yield further performance improvements and shows that the performance saturates after a certain context size is reached.

VII. Conclusion

In this work, we proposed dynamic convolutional neural networks as efficient pre-trained audio models. We integrated dynamic convolutions, dynamic ReLU, and Coordinate Attention into efficient inverted residual blocks and share the computation of a global context for dynamic parameterization across all dynamic modules in a block. The resulting models, named DyMNs, are pre-trained on AudioSet at three different complexity levels using Transformer-to-CNN Knowledge Distillation. DyMNs show a beneficial performance–complexity trade-off compared to their non-dynamic counterparts and other Transformers and CNNs. Specifically, Dy-MN-L achieves a pre-training performance of 49.0 mAP on AudioSet, outperforming current popular Audio Spectrogram Transformers. Experiments on downstream tasks indicate that the proposed DyMNs outperform other CNNs by a large margin and are highly competitive with Audio Spectrogram Transformers while being much more computationally efficient. Furthermore, we show that DyMNs are suitable for simple task-specific fine-tuning by sharing the same fine-tuning pipeline across all downstream tasks. In short, DyMNs are efficient, high-performing, easy-to-fine-tune audio models that can have a large impact on the audio community, especially in the context of resource-critical applications.

References

[1] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[2] J. F. Gemmeke et al., "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 776–780.

[3] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[4] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.

[5] Y. Gong, Y. Chung, and J. R. Glass, "AST: Audio spectrogram transformer," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2021, pp. 571–575.

[6] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[7] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2006, pp. 535–541.

[8] J. Ba and R. Caruana, "Do deep nets really need to be deep?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[9] F. Schmid, K. Koutini, and G. Widmer, "Efficient large-scale audio tagging via transformer-to-CNN knowledge distillation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2023, pp. 1–5.

[10] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.

[11] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2022, pp. 2753–2757.

[12] F. Schmid, K. Koutini, and G. Widmer, "Low-complexity audio embedding extractors," in *Proc. IEEE Eur. Signal Process. Conf.*, 2023, pp. 451–455.

[13] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11027–11036.

[14] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic relu," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 351–367.

[15] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13713–13722.

[16] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[17] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[18] B. Yang, G. Bender, Q. V. Le, and J. Ngiam, "CondConv: Conditionally parameterized convolutions for efficient inference," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1305–1316.

[19] J. Si, S. L. Harris, and E. Yfantis, "A dynamic reLU on neural network," in *Proc. IEEE Dallas Circuits Syst. Conf.*, 2018, pp. 1–6.

[20] H. Lee, H. Kim, and H. Nam, "SRM: A style-based recalibration module for convolutional neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1854–1862.

[21] S. Woo, J. Park, J. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

[22] J. Park, S. Woo, J. Lee, and I. S. Kweon, "BAM: Bottleneck attention module," in *Proc. Brit. Mach. Vis. Conf.*, 2018, Art. no. 147.

[23] D. Misra, T. Nalamada, A. U. Arasanipalai, and Q. Hou, "Rotate to attend: Convolutional triplet attention module," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 3138–3147.

[24] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "GCNet: Non-local networks meet squeeze-excitation networks and beyond," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1971–1980.

[25] E. Humphrey, S. Durand, and B. McFee, "OpenMIC-2018: An open data-set for multiple instrument recognition," in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2018, pp. 438–444.

[26] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proc. Annu. ACM Multimedia Conf.*, 2015, pp. 1015–1018.

[27] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in DCASE 2020 challenge: Generalization across devices and low complexity solutions," in *Proc. Workshop Detection Classification Acoustic Scenes Events*, 2020, pp. 56–60.

[28] E. Fonseca, X. Favory, J. Pons, F. Font, and X. Serra, "FSD50K: An open dataset of human-labeled sound events," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 829–852.

[29] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[30] M. Tan and Q. V. Le, "Efficient Net:Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[31] M. Tan and Q. V. Le, "EfficientNetV2: Smaller models and faster training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10096–10106.

[32] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[33] N. Ma, X. Zhang, H. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 122–138.

[34] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *Proc. Int. Conf. Learn. Representations Workshop Track*, 2018.

[35] M. Tan et al., "MnasNet:Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[36] 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.

[37] Y. Gong, Y. A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 3292–3306, 2021.

[38] Y. Gong, S. Khurana, A. Rouditchenko, and J. R. Glass, "CMKD: CNN/transformer-based cross-model knowledge distillation for audio classification," 2022, *arXiv:2203.06760*.

[39] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," in *Proc. Int. Conf. Learn. Representations*, 2019.

[40] T. Verelst and T. Tuytelaars, "Dynamic convolutions: Exploiting spatial sparsity for faster inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2317–2326.

[41] Y. Zhang, J. Zhang, Q. Wang, and Z. Zhong, "DyNet: Dynamic convolution for accelerating convolutional neural networks," 2020, *arXiv:2004.10694*.

[42] X. Jia, B. D. Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 667–675.

[43] B. Klein, L. Wolf, and Y. Afek, "A dynamic convolutional layer for short range weather prediction," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4840–4848.

[44] S. Kim, H. Nam, and Y. Park, "Temporal dynamic convolutional neural network for text-independent speaker verification and phonemic analysis," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 6742–6746.

[45] H. Nam, S. Kim, B. Ko, and Y. Park, "Frequency dynamic convolution: Frequency-adaptive pattern recognition for sound event detection," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2022, pp. 2763–2767.

[46] D. Shen, M. R. Min, Y. Li, and L. Carin, "Learning context-sensitive convolutional filters for text processing," in *Proc. Conf. Empirical Methods Natural Lang.*, 2018, pp. 1839–1848.

[47] J. Gong, X. Qiu, X. Chen, D. Liang, and X. Huang, "Convolutional interaction network for natural language inference," in *Proc. Conf. Empirical Methods Natural Lang. Proc.*, 2018, pp. 1576–1585.

[48] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1319–1327.

[49] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[50] S. Woo et al., "ConvNeXt V2: Co-designing and scaling convnets with masked autoencoders," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16133–16142.

[51] F. Ronchini, S. Cornell, R. Serizel, N. Turpault, E. Fonseca, and D. P. W. Ellis, "Description and analysis of novelties introduced in DCASE task 4 2022 on the baseline system," in *Proc. Workshop Detection Classification Acoust. Scenes Events*, 2022, pp. 171–175.

[52] X. Mei, X. Liu, M. D. Plumbley, and W. Wang, "Automated audio captioning: An overview of recent progress and new challenges," *EURASIP J. Audio Speech Music Process.*, vol. 2022, no. 1, 2022, Art. no. 26.

[53] H. Xie, S. Lipping, and T. Virtanen, "Language-based audio retrieval task in DCASE 2022 challenge," in *Proc. Workshop Detection Classification Acoustic Scenes Events*, 2022, pp. 216–220.

[54] C. L. Jiménez, D. Griol, Z. Callejas, R. Kleinlein, J. M. Montero, and F. F. Martínez, "Multimodal emotion recognition on RAVDESS dataset using transfer learning," *Sensors*, vol. 21, no. 22, 2021, Art. no. 7665.

[55] N. Tonami, S. Mishima, R. Kondo, K. Imoto, and T. Hino, "Event classification with class-level gated unit using large-scale pretrained model for optical fiber sensing," in *Proc. Workshop Detection Classification Acoust. Scenes Events*, 2023, pp. 196–200.

[56] S. Verbitskiy, V. B. Berikov, and V. Vyshegorodtsev, "ERANNs: Efficient residual audio neural networks for audio pattern recognition," *Pattern Recognit. Lett.*, vol. 161, pp. 38–44, 2022.

[57] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 646–650.

[58] P. Huang et al., "Masked autoencoders that listen," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 28708–28720.

[59] S. Chen et al., "BEATs: Audio pre-training with acoustic tokenizers," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 5178–5193.

[60] Y. Gong, C. Lai, Y. Chung, and J. R. Glass, "SSAST: Self-supervised audio spectrogram transformer," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 10699–10709.

[61] H. Dinkel, Y. Wang, Z. Yan, J. Zhang, and Y. Wang, "CED: Consistent ensemble distillation for audio tagging," 2023, *arXiv:2308.11957*.

[62] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018.

[63] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10915–10924.

[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[65] T. Pellegrini, I. K. Hassani, E. Labbé, and T. Masquelier, "Adapting a ConvNeXt model to audio classification on AudioSet," 2023, *arXiv:2306.00830*.

[66] J. Rasley, S. Rajbhandari, O. Ruwase, and Y. He, "DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 3505–3506.

[67] Z. Liu et al., "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 9992–10002.

[68] A. Gazneli, G. Zimerman, T. Ridnik, G. Sharir, and A. Noy, "End-to-end audio strikes back: Boosting augmentations towards an efficient audio classification network," 2022, *arXiv:2204.11479*.

[69] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Receptive field regularization techniques for audio classification and tagging with deep convolutional neural networks," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 29, pp. 1987–2000, 2021.

[70] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "CLAP: Learning audio concepts from natural language supervision," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2023, pp. 1–5.

[71] P. Wang et al., "ONE-PEACE: Exploring one general representation model toward unlimited modalities," 2023, *arXiv:2305.11172*.

[72] S. Suh, S. Park, Y. Jeong, and T. Lee, "Designing acoustic scene classification models with CNN variants," Electron. Telecommun. Res. Inst., Daejeon, South Korea, Tech. Rep., 2020.

[73] T. Morocutti, F. Schmid, K. Koutini, and G. Widmer, "Device-robust acoustic scene classification via impulse response augmentation," in *Proc. IEEE Eur. Signal Process. Conf.*, 2023, pp. 176–180.

[74] D. S. Park et al., "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Conf. Int. Speech Commun. Assoc.*, 2019, pp. 2613–2617.

Florian Schmid received the B.S. and M.S. degrees in computer science from Johannes Kepler University (JKU), Linz, Austria, in 2021 and 2022, respectively. He is currently working toward the Ph.D. degree. He is also a University Assistant with the Institute of Computational Perception, Johannes Kepler University (JKU). His research interests include efficient deep learning, acoustic scene classification, acoustic event detection, pre-trained audio models, and general-purpose audio representations.

Khaled Koutini is currently a Postdoctoral Researcher with the Linz Institute of Technology AI Lab & the Institute of Computational Perception, Johannes Kepler University, Linz, Austria. His research interests include acoustic signal processing, learning general audio representations, audio classification and tagging, machine learning, neural networks, and generalization and robustness in machine learning.

Gerhard Widmer is currently a Professor and the Head of the Institute of Computational Perception, Johannes Kepler University, Linz, Austria. His work is published in a wide range of scientific fields, from AI and machine learning to audio, multimedia, musicology, and music psychology. His research interests include AI, machine learning, intelligent music processing, and computational musicology. He is a Fellow of the European Association for Artificial Intelligence and a Member of the Austrian Academy of Sciences. He was the recipient of Austria's highest research awards (START and Wittgenstein Prize), and currently holds an ERC Advanced Grant for research on AI & music.