Single-Layer Vision Transformers for More Accurate Early Exits with Less Overhead

Arian Bakhtiarnia[a,*], Qi Zhang[a], Alexandros Iosifidis[a]

[a]*DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Finlandsgade 22, Aarhus, 8200, Midtjylland, Denmark*

Abstract

Deploying deep learning models in time-critical applications with limited computational resources, for instance in edge computing systems and IoT networks, is a challenging task that often relies on dynamic inference methods such as early exiting. In this paper, we introduce a novel architecture for early exiting based on the vision transformer architecture, as well as a fine-tuning strategy that significantly increase the accuracy of early exit branches compared to conventional approaches while introducing less overhead. Through extensive experiments on image and audio classification as well as audiovisual crowd counting, we show that our method works for both classification and regression problems, and in both single- and multi-modal settings. Additionally, we introduce a novel method for integrating audio and visual modalities within early exits in audiovisual data analysis, that can lead to a more fine-grained dynamic inference.

Keywords: dynamic inference, early exiting, multi-exit architecture, vision transformer, multi-modal, deep learning

*Corresponding author

Email addresses: arianbakh@ece.au.dk (Arian Bakhtiarnia), qz@ece.au.dk (Qi Zhang), ai@ece.au.dk (Alexandros Iosifidis)

1. Introduction

Over the past decade, deep learning has shown tremendous success across various fields, such as computer vision and natural language processing [1]. However, deep learning models are by definition composed of many layers of interconnected neurons, even reaching billions of parameters, which makes them computationally expensive. This has sparked a great deal of research in order to make deep learning models more lightweight, for which many approaches have been proposed, for instance, *model compression* methods [2] such as *quantization* [3], *pruning* [4], *low-rank approximation* [5] and *knowledge distillation* [6].

More and more emerging internet of things (IoT) applications are integrating deep learning models, such as video surveillance, voice assistants, augmented reality and cooperative autonomous driving, which are often time-sensitive and require inputs to be processed within specific deadlines [7, 8]. The heavy computational burden of deep learning becomes problematic for these time-critical IoT applications, due to resource-constrained IoT devices. *Edge computing* is a promising computing paradigm for addressing this issue, in which the deep learning task is offloaded to edge servers in the proximity of IoT devices.

Since edge computing systems introduce computation offloading over a communication network and involve multiple nodes working collaboratively in order to complete the task in a timely manner, transmission time has to be taken into account in addition to the deep learning computation time. However, transmission time may vary greatly over time and across different

channels. Consequently, deep learning models running on edge computing systems and IoT networks should be capable of *anytime prediction*, meaning they should be able to provide a valid response even if they are interrupted before traversing the entire neural network, although the model is expected to provide a better answer if it is allowed to run for longer time.

Dynamic inference approaches [9] modify the computation graph based on each input during the inference phase in order to fit the time constraints. A dynamic inference approach that particularly suits anytime prediction is *early exiting* [10], also referred to as *multi-exit architectures* or *auxiliary classifiers* in the literature. In multi-exit architectures, one or more early exit *branches* are placed after some of the intermediate hidden layers of the *backbone* network. The goal of each of these branches is to provide an early result similar to the final result of the neural network using only the features extracted up to that particular branch location. These early results are inevitably less accurate than the final result of the network. In order to achieve anytime prediction using early exiting, the latest early result can be used whenever the execution is interrupted, for instance, whenever a hard deadline is reached. Computation time can be further decreased by applying model compression techniques on the backbone of multi-exit architectures. Besides anytime prediction, early exiting can also be used in *budgeted batch classification* where a fixed amount of time is available in order to classify a set of input samples. In such a setting, the result of earlier branches can be used for "easier" samples whereas the result of later branches or the final result can be used for "harder" ones. The difficulty of each sample can be determined based on the confidence of the network about its output [11],

although other approaches exist in the literature [10].

Early exit branches are expected to have a low overhead in terms of the extra computation they introduce, since a high overhead would defeat the purpose. Therefore, they often contain only a handful of layers. Ideally, we want the accuracy of the early results to be close to that of the final result, since a higher accuracy for early exit branches means that the overall reliability of the system increases. However, the low-overhead constraint makes it quite challenging to achieve a high accuracy since the early exit branches have significantly less trainable parameters compared to the rest of the network. Several approaches for increasing the accuracy of early exits such as knowledge distillation [12], curriculum learning [13] and architectures designed specifically for early exit branches [14] have been suggested. In this paper, we propose a novel architecture in order to obtain more accurate early exits for convolutional neural network (CNN) backbones.

A neural architecture called *vision transformer* (ViT) [15] has been recently introduced for image classification which is radically different from convolutional neural networks. The building blocks of Vision Transformer have been used for early exits placed on Vision Transformer backbones [14], however, using Transformer-based early exit branches on CNN backbones is not intuitive and requires additional steps and architectural modifications. We use a modified version of this architecture instead of the usual convolution and pooling layers in early exit branches and show that our method can significantly increase the accuracy of early exits compared to conventional

architectures by fusing local and global receptive fields[1]. The contributions of this paper can be summarized as follows:

- We propose a novel architecture for early exit branches in multi-exit architectures based on vision transformers, called *single-layer vision transformer* (*SL-ViT*). We compare our method with conventional CNN-based early exit architectures across 27 scenarios involving different datasets, branch locations and backbone networks and show that our method is significantly more accurate in 26 of these scenarios, while having less overhead in terms of number of parameters and floating point operators (FLOPS). To the best of our knowledge the fusion of global and local scope in early exits has never been used in multi-exit architectures before.

- We show that our method is a general purpose approach that works across different modalities as well as multi-modal settings by investigating image classification, audio classification and audiovisual crowd counting scenarios. We also show that our method works for both classification and regression problems.

- We introduce a novel way of integrating audio and visual features in early exits using vision transformers. To the best of our knowledge, this is the first time early exits have been studied in multi-modal settings.

- We provide insight into why our method achieves better results compared to conventional CNN-based architectures by investigating the

[1]Our code will be available at `https://gitlab.au.dk/maleci/sl_vit`.

role of attention and receptive field.

- We introduce a fine-tuning strategy for SL-ViT called *copycat single-layer vision transformer* (*CC-SL-ViT*) which is based on the copycat strategy developed for CNNs [16] and show that this method can further increase the accuracy of SL-ViT early exits. To the best of our knowledge this is the first time the copycat strategy is used for vision transformers or early exits.

The rest of this paper is organized as follows: Section 2 provides an overview of the relevant literature; Section 3 describes our proposed method in detail; Section 4 explains the details of our experiments; Section 5 showcases the experiment results; and, finally, Section 6 briefly discusses the results and concludes the paper.

2. Related Work

This section provides the necessary prerequisites for understanding our method and experiments. We start by describing the particulars of multi-exit architectures. Subsequently, we provide the details of the vision transformer architecture, which is the foundation of the proposed method. Then, we briefly touch on how audio classification is normally carried out, which is included in several scenarios in our experiments. Finally, we explain another scenario investigated in our experiments, i.e. crowd counting, and how it can be approached in a multi-modal manner.

2.1. Multi-Exit Architectures

In order to describe multi-exit architectures, we use the same notation as Scardapane et al. [10] where a neural network is formulated as a function $f(X) = f_L(f_{L-1}(...f_1(X)))$. In this formulation L signifies the total number of layers in the network and f_i is the operator corresponding to layer i, which can be a convolutional layer, a fully-connected layer, a normalization layer, or any other differentiable operator. $h_i = f_i(h_{i-1})$ denotes the output of layer i, where h_0 is the input X. Finally, θ_i symbolizes the trainable parameters of layer i.

Equation (1) formulates the training process for the neural network which is achieved by tuning its parameters using an optimization algorithm on the landscape defined by a loss function. In this equation, the parameters of the neural network are denoted by $\theta = \bigcup_{i=1}^{L} \theta_i$, the training samples are signified by $\{(X_n, y_n)\}_{n=1}^{N}$, and $l(\cdot, \cdot)$ is the loss function.

$$f^* = \arg\min_{\theta} \sum_{n=1}^{N} l(y_n, f(X_n)) \tag{1}$$

Extending this notation to multi-exit architectures, $B \subseteq \{1, .., L\}$ signifies the set of selected branch locations after which early exit branches will be placed. $c_b(h_b) = y_b$ is the classifier or regressor representing the early exit branch at each branch location b, where y_b denotes the early result at that location. The schematic illustration of a multi-exit architecture is presented in Figure 1. However, since there are multiple outputs, and thus multiple loss signals in a multi-exit architecture, its training is not as straightforward.

Three different approaches for training multi-exit architectures exist in the literature [10, 17, 13]. In the first approach, called *end-to-end* training,

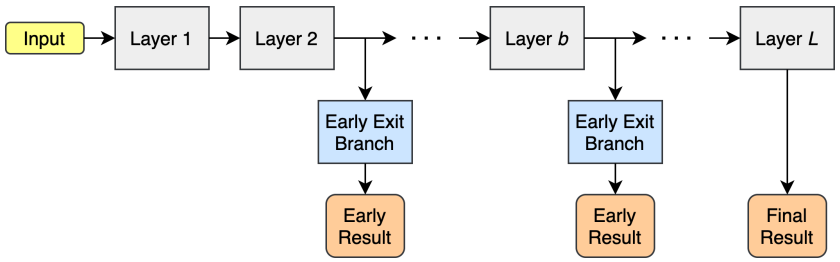

Figure 1: Schematic illustration of a multi-exit architecture with two early exits.

the loss signals of all exits are combined and backpropagated through the network at the same time. With end-to-end training, the contribution of each loss signal to the total loss is expressed with weight values, which are therefore hyper-parameters of the model.

The second approach, called *layer-wise* training, first trains the network up to and including the first exit branch. Subsequently, the part of the network that has been trained so far is frozen, meaning its parameters are not modified any further, and the remainder of the network up to and including the second exit branch is trained. This process continues until the entire network is trained. Note that with this approach, there is no guarantee that the accuracy of the final exit remains unchanged.

In the final approach, called *classifier-wise* training, the backbone network is completely frozen and each branch is trained independent of the rest of the network and other branches, meaning the parameters θ are not modified and only the parameters of the classifers/regressors $\{c_b\}, b \in B$ are trained. With this approach, no new hyper-parameters are introduced and the backbone remains unchanged. However, the early exit branches affect a lower number of trainable parameters compared to the other approaches.

In this paper, we choose to follow the classifier-wise training approach

due to its practical importance. This is because with classifier-wise training, early exit branches can be easily added on top of existing backbone networks without the need for re-training and hyper-parameter optimization, which can be computationally expensive and time consuming. Furthermore, with end-to-end and layer-wise training strategies, the number of branches and their placement can lead to further trade-offs and affect the overall performance of the model. Since branches are independently trained in the classifier-wise strategy, any number of branches can exist and a branch can be placed at any location without affecting the performance of other branches or the backbone.

It is important to mention that branches placed later in the backbone network do not necessarily result in a higher accuracy compared to branches placed earlier. The usage of such branches would therefore not be sensible since earlier branches exist that require less computation and provide more accurate results. We hereby use the term *impractical* to refer to such branches.

As previously mentioned, there are several methods that try to improve the accuracy of early exits. The method in [12] uses the combination of the distillation loss from the final exit and the loss signal from ground truth labels to train more accurate early exits using in the end-to-end training setting. The method in [18] expands on this idea by adding a third loss signal based on the difference between features of the latest early exit with earlier exits. The method in [19] proposes a technique called *gradient equilibrium* to combat the problem of gradient imbalance that surfaces when using the end-to-end strategy, which is when the variance of the gradients becomes very large when

loss signals from multiple exits are combined, leading to unstable training. Moreover, this paper introduces forward and backward knowledge transfer that aims to encourage collaboration among different exits. The method in [20] improves the accuracy of later exits by reusing predictions from earlier exits. The method in [21] circumvents the problem of impractical branches by adaptively selecting the exit location based on time budget and the specific input. The method in [22] simplifies the design of multi-exit architectures by removing the hyper-parameters of the end-to-end training strategy that specify the contribution of each loss signal.

Besides efficient inference, early exits can prove useful in several other applications, for instance, the method in [23] allows for parallel training of the segments of the DNN that exist between early exits, by training each segment based on the loss signal of the next segment obtained in the previous training stage. Moreover, early exits can be added to the network during the training in order to increase the accuracy of the backbone network and discarded after the training phase, for instance, the widely used Inception model [24] was trained in this way.

Besides early exiting, several other approaches exist for dynamic inference, for instance, layer skipping [25, 26, 27, 28] where the execution of some of the layers of the DNN are skipped, and channel skipping [29] where less impactful channels of convolutional neural networks are ignored and their computation is skipped during the inference phase. However, unlike early exits, these approaches cannot provide an output if the execution is interrupted due to a strict deadline, as these methods need to perform the computations until the very last layer.

2.2. Vision Transformer

The transformer architecture was first introduced by Vaswani et al. [30] for natural language processing, and it has recently been adapted for solving computer vision problems by Dosovitskiy et al. [15]. Vision transformer was originally developed for the problem of image classification, however, variations of vision transformer have since been applied to many computer vision problems, such as object detection, depth estimation, semantic segmentation, image generation and action recognition, as well as multi-modal data analysis tasks such as text-to-image synthesis and visual question answering [31, 32, 33].

In order to describe the vision transformer architecture, we first explain the *self-attention* layer. The input of this layer is in the form of a sequence $X = (x_1, \ldots, x_n)$ where $X \in \mathbb{R}^{n \times d}$ and d is the embedding dimension to represent each entity. Its output is in the form of $Z = (z_1, \ldots, z_n)$ where $Z \in \mathbb{R}^{n \times d_v}$. The goal of self-attention is to capture the interaction between the entities in the sequence. For this purpose, each vector x_i in the sequence is transformed into three separate vectors: the *query* vector $q_i \in \mathbb{R}^{d_q}$, the *key* vector $k_i \in \mathbb{R}^{d_k}$ and the *value* vector $v_i \in \mathbb{R}^{d_v}$, where $d_q = d_k$. To construct the output vector z_i that corresponds to the input x_i, for each vector x_j in X (including x_i itself), the scalar a_{ij} is calculated by the inner product of q_i and k_j. Output vector z_i is then calculated by summing the value vectors $v_1, \ldots, v_n$ weighted by their corresponding scalars, that is, $z_i = \sum_{j=1}^{n} a_{ij} v_j$. The scalar a_{ij} basically specifies how much attention the i-th entity should pay to the j-th entity, since a_{ij} determines the contribution of v_j to the combined output z_i. In practice, the scalars are normalized by $\sqrt{d_k}$ and

converted into probabilities using the softmax function.

If the key, query and value vectors are packed into matrices $Q = XW^Q$, $K = XW^K$ and $V = XW^V$, where W^Q, W^K and W^V are learnable weight matrices, the above operation can be rephrased as follows:

$$Z = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

In order to enable the model to capture more than one type of relationship between the entities in the sequence, self-attention is extended to *multi-head attention* by concatenating the output of h different self-attention blocks $Z_1, \ldots, Z_h$ each with its own set of learnable weight matrices, into a single matrix $Z' = [Z_0, \ldots, Z_h] \in \mathbb{R}^{n \times h.d_v}$, which is then projected using a weight matrix $W' \in \mathbb{R}^{h.d_v \times d}$.

A *transformer encoder* is constructed by passing the input sequence into a normalization layer, a multi-head attention layer, a second normalization layer and a multi-layer perceptron (MLP), respectively. Two residual connections are added, one by adding the input sequence to the output of the multi-head attention, and the other by adding the output of the multi-head attention to the output of the MLP.

Putting it all together, a vision transformer is created by first splitting the input image into patches. Subsequently, the sequence of patches is projected into a sequence of vectors and a positional embedding is added to the corresponding vector of each patch. An additional learnable embedding called *classification token* is added to the beginning of the sequence. The sequence then passes through L transformer encoders. Finally, the first vector in the output of the last transformer encoder, which corresponds to the classification token, is passed to a MLP which outputs the final classification

result. The architecture of vision transformer is depicted in Figure 2.

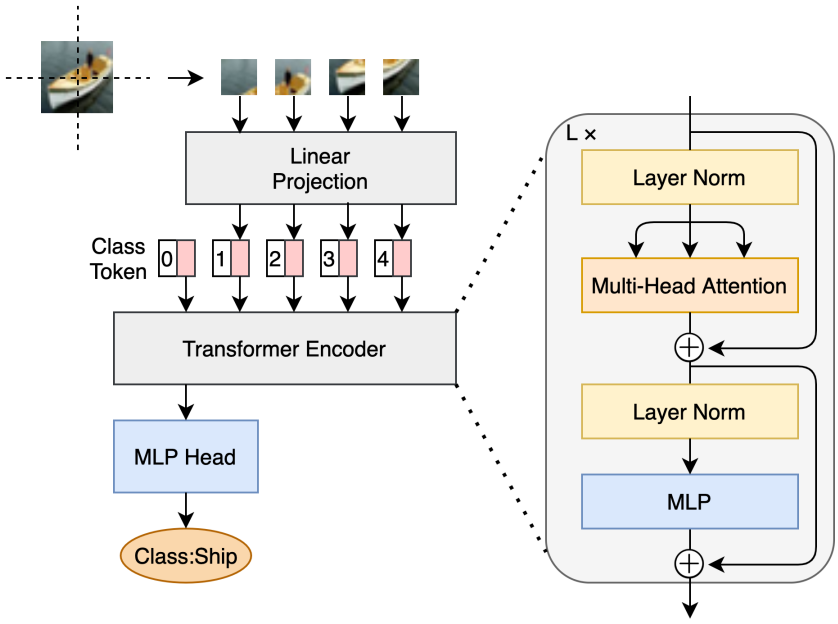

Figure 2: The vision transformer (ViT) architecture for image classification.

ViT-EE is a method which uses transformer encoders for early exits placed on ViT backbones [14]. ViT-EE uses the exact same layer as the ViT backbone. Using the building blocks of the backbone network for early exit branches is simple and intuitive, and it is the reason why so far, mostly convolutional layers have been used for early exiting CNN backbones. However, as we show in this work, carefully designing the architecture of early exit branches can lead to significant improvements. Using Transformer-based early exit branches on CNN backbones is not intuitive, and requires additional steps such as converting tensors to patches, dealing with the classification token and fine-tuning the architecture parameters including patch size, attention heads, embedding representation, the size and number of layers

for MLP, and dropout. Moreover, we show that removing the last residual connection in the transformer encoder can improve the performance in some cases.

Furthermore, ViT backbones have a global receptive field in every layer, this means that ViT-EE is not necessarily ideal for early exits at all layers, as it adds too much overhead without providing improvements in terms of receptive field. On the other hand, CNN backbones have a limited receptive field particularly in earlier layers, therefore fusing this receptive field with a global one leads to improvements.

2.3. Audio Classification

Similar to image classification, audio classification is the problem of categorizing a given audio waveform into one of several predetermined classes. For instance, the given audio waveform could be a musical recording, and the goal could be to specify which genre of music it belongs to. To represent the input features, *spectrograms* obtained by applying short-time Fourier transform (STFT) and *Mel spectrograms* are commonly used [34], although raw audio waveforms can been used as well [35]. Mel spectrograms are spectrograms that are constructed using the *Mel scale* which is a nonlinear transformation of the frequency scale designed based on domain knowledge about the human auditory system. Various deep learning models for audio classification exist in the literature, including models that are commonly used for image classification, namely ResNet [36], DenseNet [37] and Inception [38], which have been shown to be quite effective for audio classification as well [39]. Conveniently, the same three networks have previously been used as backbone networks when investigating early exiting for image classification

[13]. Therefore we use these backbone networks for both image and audio classification in our experiments.

2.4. Audiovisual Crowd Counting

Crowd counting refers to the problem of identifying the total number of people present in a given image. Crowd counting has many applications such as safety monitoring, disaster management, design of public spaces, intelligence gathering and analysis, creation of virtual environments and forensic search [40]. With many of these applications, it is vital for the model to perform in near real-time. However, the input images in these scenarios often have high resolutions, such as HD or Full HD. Moreover, many of the available methods contain an immense number of parameters [41]. This means that crowd counting models are often very computationally expensive, therefore, dynamic inference methods such as early exiting and other lightweight deep learning methods become essential in real world applications.

Although the main objective of this task is to obtain a single count from an image, many methods treat this problem as dense prediction where the output is a *density map* depicting the density of the crowd across the input image, and the total count is calculated by the sum of all values in the density map. Therefore, in most crowd counting datasets, such as Shanghai Tech [42] and World Expo '10 [43], the locations of the heads of individuals in the image are annotated and provided as targets. A ground truth density map can then be obtained from these *head annotations* using Gaussian kernels or more complicated and specialized methods [41]. Figure 3 shows an image from the Shanghai Tech dataset and the ground truth density map that was generated from the provided head annotations using the method presented in

Figure 3: An example image from the Shanghai Tech dataset and its corresponding ground truth density map.

Zhang et al [42]. In crowd counting, *Mean Absolute Error* (*MAE*) is usually used as a measure of accuracy whereas *Mean Squared Error* (*MSE*) is used as a measure of robustness [44].

Many crowd counting methods exist in the literature [41], however, most of these methods are applied in a single-modal fashion where the input is an image or a video frame. In contrast, AudioCSRNet [45], a multi-modal extension of the widely-used CSRNet model for crowd counting [46], takes as input the ambient audio of a scene in addition to its image. The authors show that the ambient audio improves the result in situations where the image quality is not ideal, for instance, low image resolution, presence of noise, occlusion and low illumination.

In CSRNet, the features extracted from the input image by the first 10 layers of a VGG-16 [47] network pre-trained on the ImageNet dataset [48] are passed through 6 dilated convolution layers and a 1×1 convolution layer in order to obtain the density map. AudioCSRNet extends this architecture by converting each of the dilated convolution layers into a fusion block. The architecture of AudioCSRNet is depicted in Figure 4. First, a Mel spectro-

gram is obtained from the raw audio waveform. Subsequently, in each fusion block, the features extracted from the input Mel spectrogram by the first 6 layers of a VGGish [49] network pre-trained on the AudioSet dataset [49] are projected to two vectors called γ and β which represent the multiplicative and additive aspects of the audio features. The γ and β vectors are then tiled in order to match the size of the visual features. Finally, the output of the dilated convolution is element-wise multiplied by γ and added to β.

The fusion operation can be summarized as

$$v_{l+1} = \mathcal{F}_l(\gamma_l \odot D_l(v_l) + \beta_l), \tag{3}$$

where $v_l \in \mathbb{R}^{C_l \times W_l \times H_l}$ is the output of the l-th fusion block, F_l denotes an activation function, γ_l and β_l are the tiled vectors and D_l represents the l-th dilated convolution.

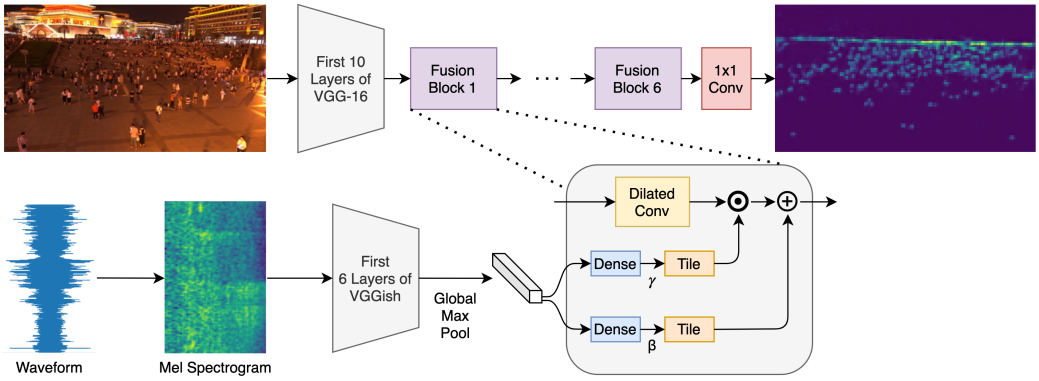

Figure 4: Architecture of AudioCSRNet.

In practice, a batch normalization layer [50] is added immediately after each dilated convolution. Furthermore, the height and width of the intermediate features remain unchanged by using padding in the convolution operations, meaning $H_l = H_{l+1}$ and $W_l = W_{l+1}$. Additionally, since the first 10

layers of VGG-16 decrease both height and width by a factor of 8 via several pooling operations, the final result of the network needs to be upsampled by a factor of 8 in order to match the resolution of the input image. It is important to preserve the total sum of the density map during this upsampling operation, since it represents the total count.

3. Single-Layer Vision Transformers for Early Exits

We assume a pre-trained and high performing backbone network is already available. Due to time constraints arising from the particular application, it is desirable that the network provides a result within the specific deadline rather than not providing a result at all, even though this result may be less accurate than it would be if time constraints did not exist. Therefore, the backbone needs to be augmented with early exit branches to allow for dynamic inference and anytime prediction. As previously mentioned, we use the classifier-wise approach for training the early exit branches since it results in "plug-and-play" branches that can easily be added to the backbone network without any re-training or hyper-parameter tuning.

3.1. SL-ViT

Typically, the architecture of early exit branches starts with one or more convolution layers, although some may have no convolutions at all. Afterwards, they may have a pooling layer, which may be global pooling, and one MLP [51, 11]. Here, as a baseline, we choose to utilize the architecture depicted in Figure 5 with one 3×3 convolution, followed by a 2×2 max pooling layer and finally a MLP. The size of the max pooling layer is increased to 4×4 for crowd counting since the input images have a very high resolution.

Additionally, we use dropout [52] inside the MLP to avoid overfitting. We use a single convolution since early exits with two or more convolution layers have a high overhead and may even lead to lower accuracy [11]. Early exits without convolutions are sometimes used very late in the network, however, since they are straightforward and leave no room for modifications, we do not apply our method for such cases. The resulting architecture is a common setup within the literature, and is effectively the same architecture used for earlier exits by Hu et al. [51].

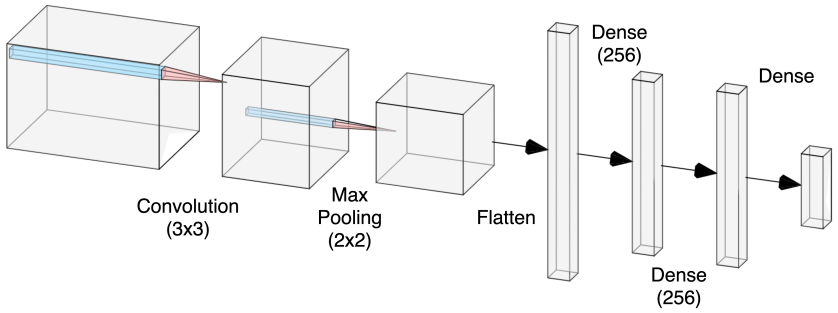

Figure 5: Architecture of CNN early exit branches. Size of the flattened feature vector depends on the dimensions of the features at the specific branch location. For branches placed on the AudioCSRNet backbone, max pooling size is increased to 4x4 since the input images have a high resolution. Figure created using the NN-SVG tool [53].

Our method called *single-layer vision transformer* or *SL-ViT* for short, is an alternative architecture for early exit branches that can achieve a higher accuracy compared to the aforementioned baseline, while having less overhead in terms of the number of parameters as well as floating point operations per second (FLOPS). Our proposed architecture is based on the vision transformer architecture introduced in section 2.2, where instead of the input image, we split the intermediate features at the branch location into patches

(sub-tensors) and pass them to a vision transformer.

The choice of vision transformer architecture is primarily due to its global receptive field. Receptive field is crucial in many deep learning problems, including ones studied in this work. The receptive field of state-of-the-art CNNs developed for image classification has steadily increased over time and is correlated with increased classification accuracy [54]. Additionally, in audio classification using spectrograms, each location relates to a different frequency band in a different window of time. It is reasonable to assume that processing combinations of frequencies and windows that are not necessarily adjacent could be of importance. Moreover, many crowd counting methods have made use of global information through visual attention mechanisms and dilated convolutions [41]. Since the receptive field is particularly limited in early layers of CNN backbones, choosing an architecture for early exit branches with a global receptive field could be beneficial.

Many other designs strive to increase the receptive field in their building blocks, for instance, the *pyramid pooling module (PPM)* in PSPNet [55] or *atrous spatial pyramid pooling (ASPP)* in DeepLab [56]. However, they all fall short in comparison with the global receptive field of transformers. PPM increases the receptive field through aggregating different levels of pooling, which means far locations have only access to coarse representations of each other, and ASPP has holes in its receptive field.

It is important to mention that the local receptive field of convolutional layers is not fundamentally bad. On the contrary, it plays a key role in representation learning and extracting local information, especially in the early layers of the network where the receptive field of the convolutional filters

is small. Filters in successive convolutional layers have increasingly larger receptive fields, therefore, final layers in a CNN architecture have filters of large enough receptive fields that can effectively aggregate information from the entire input image to provide a proper response. However, this process of cascading local receptive fields of increasing size requires the number of layers in the CNN to be large, or at least all the layers in the network to be traversed in order to provide the network's response. When an early exit is added at an early layer, this chain of increasingly larger receptive fields is broken, and an early exit that has a local receptive field may not be able to effectively aggregate all required information in the image to provide a suitable response. This situation is the motivation behind the proposed branch architecture, which fuses the local receptive field of the layer in the network where the early exit branch is attached, with the global receptive field of the early exit, in order to effectively aggregate information from the entire input and provide a more accurate response. Indeed, the original Vision Transformer paper [15] attributes the success of their model to the combination of local and global receptive fields and shows that even in very early layers, this ability to integrate information globally is indeed used by the model.

There are some crucial differences between the original vision transformer and the architecture in our method. First, in order to introduce a low overhead for early exit branches, we only use a single transformer encoder layer instead of the original 12 to 36 layers, meaning that $L = 1$ in our case. Secondly, we do not utilize a separate classification token and instead pass the entire output of the transformer encoder layer to the MLP head. This

is possible because the width and height of tensors are generally reduced throughout CNN backbones by pooling operations, and thus the number of patches in our architecture is lower than that of the original vision transformer. In addition to the number of patches, the size of the embedding dimension (d) is also reduced in our proposed architecture, introducing far less parameters when passing the entire output of the last transformer encoder layer to the MLP head, even with high-resolution inputs such as in our crowd counting experiments. Variations of our architecture have 5×5, 7×7 or 16×9 patches and embedding dimensions of 32 or 36, whereas different versions of the original vision transformer have 14×14 or 16×16 patches and embedding dimensions of 768, 1024 or 1280. We empirically found that using the entire transformer encoder output instead of just one classification token can increase the accuracy, perhaps because in a single-layer version, there are not enough layers for the classification token to learn to properly summarize other patches. Our proposed architecture is shown in Figure 6. It is also important to note that the MLP head used in our architecture is exactly the same as the MLP in the CNN early exit architecture.

Our model has several hyper-parameters, namely the size of each patch, the embedding dimension d and the number of attention heads h in multi-head attention. The patch size creates a trade-off where smaller patches result in a more fine-grained attention mechanism while increasing the total number of parameters in a bi-quadratic fashion. Therefore, similar to the original vision transformer, we choose the size of the patch to be close to the square root of the height and width of the input features. We also make sure that the size of the patch can divide the size of the input features to avoid

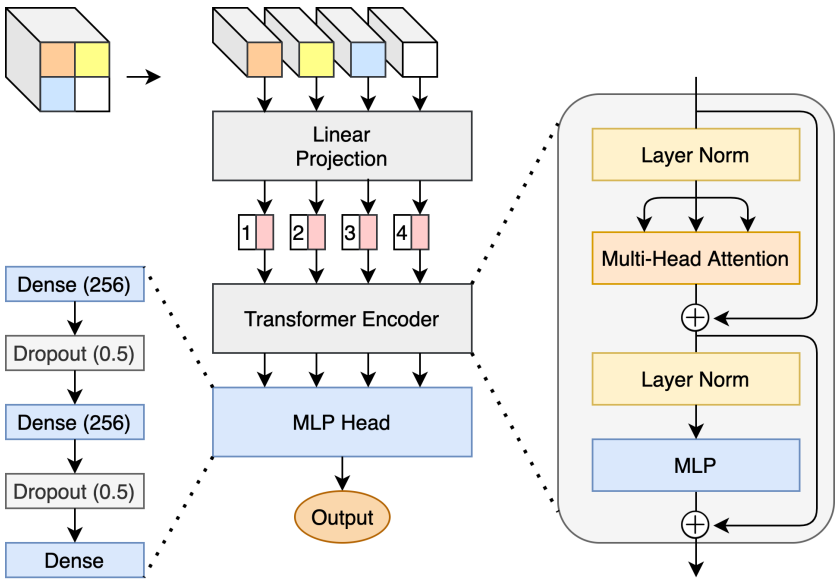

Figure 6: Architecture of SL-ViT early exit branches. Unlike typical vision transformers, only a single transformer encoder layer is used, extra learnable classification token is not added to the sequence and the entire output of the transformer encoder is passed on to the MLP head. The MLP head is the same as CNN early exit branches.

padding, for instance, a patch size of 4×4 for input features of size 28×28. We perform a grid search to find the values of d and h that result in the highest accuracy, while keeping the total number of parameters less than or equal to that of the CNN early exit counterpart.

At a first glance, it might seem like the SL-ViT architecture introduces more hyper-parameters than the conventional CNN architecture, however, the CNN architecture includes many design choices as well, such as the number of filters, filter size, padding, dilation, stride, pooling type and pooling size. The design choices for CNN architectures might seem simpler since they have been studied more extensively compared to vision transformers which were introduced more recently.

3.2. Audiovisual SL-ViT

With audiovisual backbones such as the AudioCSRNet model for audio-visual crowd counting, described in section 2.4, it is desirable to have audio-visual early exits that use both visual and audio features in order to achieve a higher accuracy. The simplest way to have such branches is to add the branches after the blocks where the fusion of visual and audio features take place. However, with our proposed SL-ViT architecture, it is also possible to include audio features as one or more patches alongside other patches, and directly fuse the features in the early exit. The advantage of this approach is that since in vision transformers, any of the patches can pay attention to any other patch, the visual features can be fused with the audio features without being directly impacted and modified. In contrast, since convolutional filters only take the immediate vicinity into account, the audio features must be present in every location. One option is to concatenate the visual features

and the tiled audio features along the depth. However, that would greatly increase the amount of computation for each fusion operation, therefore intrusive operations such as element-wise multiplication and addition are used instead.

3.3. Copycat SL-ViT

Finally, we introduce a fine-tuning strategy for SL-ViT branches that can further increase their accuracy. Correia-Silva et al. [16] developed a method called *copycat CNN* where they create a "fake" dataset by taking images from another domain, giving them as input to a network trained on the target domain, and recording the output of the network as labels for these images. For instance, images from the ImageNet dataset [48] can be given to a network trained on the CIFAR-10 dataset [57], where the image of a camel may be labelled as a "dog" since there are no labels for "camel" in CIFAR-10. This fake dataset is then combined with a dataset for the target domain and used to train a new network. We use this strategy to fine-tune an already trained SL-ViT branch and obtain a *copycat single-layer vision transformer* (*CC-SL-ViT*). Note that the ratio of the fake data mixed with the available dataset is a hyper-parameter of this fine-tuning strategy.

4. Experimental Setup

In this section, we provide the details of our experiments. We begin by giving a short summary of the datasets as well as the training details for the backbone networks. We then lay out the details of the branch architectures, their training procedure and their placement on the backbone networks, and finally explain how the copycat strategy was used to fine-tune the branches.

A total of 27 different scenarios were tested in our experiments. For both image and audio classification, two datasets, three backbone networks and two different branch locations on each backbone were tested. In addition, three different branch locations for the audiovisual crowd counting backbone network were covered. All experiments were repeated 5 times and the average accuracy as well as the standard deviation were recorded. 4 $\times$ Nvidia 2080Ti GPUs were used for the training of our models.

4.1. Datasets

4.1.1. CIFAR-10 and CIFAR-100

These are widely-used datasets for image classification [57]. Both datasets consist of 60,000 color images of size 32 $\times$ 32 pixels and their corresponding class labels. The images in CIFAR-10 and CIFAR-100 are categorized into 10 and 100 different classes, respectively. We use 40,000 examples for training, 10,000 for validation and another 10,000 for testing. Since our backbone networks are pre-trained on ImageNet which consists of 224 $\times$ 224 pixel images, we resize each image to these dimensions before passing them into the network.

4.1.2. Speech Commands (SC)

A well-known audio dataset of spoken words [58]. It consists of 100,503 1-second audio clips with a sampling rate of 16kHz, each labelled as one of 12 classes: 10 different spoken words such as "Yes", "No", "Down" and "Stop" as well as one class for background noise and another for unknown words. We use 85,511 examples for training, 10,102 for validation and 4,890 for testing. We convert the raw audio waveforms into spectrograms using

short-time Fourier transform (STFT) with a window size of 255 samples and step size of 128 samples, and resize the resulting spectrograms to 224×224 before passing them into the network.

4.1.3. GTZAN

It is the most widely-used dataset for music genre recognition [59]. The original dataset consists of 10 genres such as "Pop" and "Rock" and 100 30-second audio clips per genre with a sampling rate of 22,050Hz. We follow the common approach to split each audio clip into 10 separate 3-second clips in order to increase the size of the dataset to 10,000. We use 8,000 examples for training, 1,000 for validation and another 1,000 for testing. Following the approach of Palanisamy et al. [39] where different spectrograms with different parameters are placed in each channel of the input image, we use one spectrogram obtained from STFT with window size of 512 samples and step size of 256 samples as well as two Mel spectrograms with 128 bins and window sizes of 100 and 50 milliseconds, and step sizes of 50 and 25 milliseconds, respectively.

4.1.4. DISCO

An audiovisual dataset for crowd counting which contains 1,935 images of Full HD resolution (1920×1080) [45]. For each image, a corresponding 1-second audio clip of ambient sounds with a sampling rate of 48kHz, starting 0.5 seconds before the image was taken and ending 0.5 seconds afterwards, exists as well. The labels are provided in the form of head annotations in the image. At the time of this writing, DISCO is the only publicly available dataset for audiovisual crowd counting. We use 1435 examples for train-

ing, 200 for validation and 300 for testing. The input image is resized to 1024×576 pixels to reduce memory and computation requirements. Similar to Hershey et al. [49], the input audio waveform is transformed into a Mel spectrogram with 64 bins, window size of 25 milliseconds and step size of 10 milliseconds. Following Hu et al. [45] the ground truth density maps are obtained by convolving the head annotations with a 15×15 Gaussian kernel $\mathcal{K} \sim \mathcal{N}(0, 4.0)$.

4.2. Backbone networks

Transfer learning is used to train the ResNet152, DenseNet201 and InceptionV3 backbone networks for both image and audio classification. The backbone networks are all pre-trained on the ImageNet dataset and the top layer is replaced. We found that instead of adding just one dense layer at the top, as is common in transfer learning, using two dense layers and a dropout layer in between leads to a higher accuracy in our case. The resulting network is then trained using the Adam optimizer [60] with a learning rate of 10^{-4} and categorical cross-entropy loss function. The learning rate is reduced by a factor of 0.6 on plateau with a tolerance of 2 epochs, and an early stopping mechanism with a tolerance of 5 epochs is used.

The audiovisual crowd counting backbone is trained in two stages. We first train a network with the AudioCSRNet architecture described in Section 2.4 for 100 epochs. L_2 norm is used as loss function and AdamW [61] with a learning rate of 10^{-5} and weight decay of 10^{-4} is used as optimizer, where the learning rate is multiplied by a factor of 0.99 each epoch. This is the same training procedure used in the original paper [45]. Subsequently, in order to convert the problem from dense prediction to regression, a dense

layer with an output size of one is added after the last layer of the trained AudioCSRNet. This layer is initialized as a sum, meaning the initial weights are all equal to one and no bias is used. Then the entire network is re-trained for another 100 epochs using MAE as loss function instead of the previous L_2 loss, a learning rate of 10^{-6} and weight decay of 10^{-5}. The learning rate is similarly multiplied by a factor of 0.99 every epoch. The resulting model achieves a MAE of 13.63 which is even lower than the MAE of 14.27 reported in the original paper. However, the output of the network is just a single number representing the total count instead of a density map. The final accuracy of all trained backbones can be seen in Table 1.

When training the backbone networks, in order to fit the limitations of our available computational resources, the batch sizes are adjusted and some layers of the backbone networks are frozen. All backbone networks were trained with a batch size of 32 except AudioCSRNet which has a batch size of 4 as well as InceptionV3 when trained on CIFAR-10 and CIFAR-100 which has a batch size of 64. All layers of the backbone networks were trained, except in the case of ResNet152 and DenseNet201 when trained on CIFAR-10 and CIFAR-100 where only the batch normalization layers were trained. We found that training only the batch normalization layers is sufficient to achieve a high-performing backbone network in these cases [62].

4.3. Branches

All branches were trained from scratch using the He initialization method [63] and the Adam optimizer with a learning rate of 10^{-4} where the learning rate is reduced by a factor of 0.6 on plateau with a tolerance of 2 epochs, and an early stopping mechanism with a tolerance of 5 epochs is utilized.

Table 1: Performance of backbone networks on each dataset

Backbone	CIFAR-10 Acc.	CIFAR-100 Acc.	SC Acc.	GTZAN Acc.	DISCO MAE
ResNet152	95.36%	82.25%	95.85%	91.29%	-
DenseNet201	96.48%	82.53%	96.36%	92.09%	-
InceptionV3	96.56%	83.80%	94.93%	87.79%	-
AudioCSRNet	-	-	-	-	13.63

The branches on classification backbones use a categorical cross-entropy loss function whereas the branches on the audiovisual crowd counting backbone use mean absolute error loss. The training batch size for branches were 64 in scenarios involving CIFAR-10, CIFAR-100 and Speech Commands, 32 in scenarios involving GTZAN and 4 in scenarios involving DISCO.

Table 2 shows the location of the branches placed on each backbone network. For the AudioCSRNet backbone network, branch V1 uses only the output of the VGG-16 layers, therefore, it only has access to the visual features. Branch AV1 uses the outputs of both VGG-16 and VGGish, therefore it has access to both audio and visual features. In this branch location, the fusion of audio and visual features is performed as described in Section 3 for the SL-ViT architecture, and similar to the fusion blocks in AudioCSRNet for the CNN architecture, however, without dilation. Finally, branch AV2 is placed after the first fusion block in AudioCSRNet, therefore audio and visual features have already been fused and thus fusion operation is not required within the branches. Adding branches after the second fusion block or later would not be reasonable since more than 85% of the computation of the backbone is carried out before that point, and thus the acceleration

resulting from early exits would be negligible.

Table 2: Placement of branches for each backbone betwork

Backbone	BN*	Branch Placed After
DenseNet201	1	Transition Layer 1
	2	Transition Layer 2
ResNet152	1	12th Convolution
	2	36th Convolution
InceptionV3	1	First Filter Concat
	2	Second Filter Concat
AudioCSRNet	V1	Last Layer of VGG
	AV1	Last Layers of VGG and VGGish
	AV2	First Fusion Block

*Branch Number

4.4. SL-ViT and CC-SL-ViT Parameters

Table 3 summarizes the hyper-parameters used for the SL-ViT branches in each scenario. "Patch Size" shows the width and height of each image patch, "Patches" denotes the resulting number of patches across width and height of the input image, d is the size of embedding dimension and h is the number of heads in multi-head attention.

For copycat SL-ViT, images from the Tiny ImageNet dataset, which are the images from ImageNet down-sampled to 32×32, were given to the InceptionV3 backbone trained on CIFAR-10, and the outputs were used to create the fake dataset. Then the fake dataset was mixed with CIFAR-10 with a 2-to-1 ratio and used for re-training.

Table 3: Hyper-parameters of SL-ViT for different backbone networks and branches

Backbone	Dataset	BN*	Patch Size	Patches	d	h
DenseNet201	all	all	4x4	7x7	32	12
ResNet152	SC	2	4x4	7x7	32	24
	GTZAN	2	4x4	7x7	32	24
	Other		4x4	7x7	32	12
InceptionV3	CIFAR-100	all	5x5	5x5	36	8
	Other		5x5	5x5	32	12
AudioCSRNet	DISCO	all	8x8	16x9	32	12

*Branch Number

5. Results

The results of our experiments are presented in Tables 4 to 8. In these Tables, the final accuracy, the total FLOPS of the model up to and including the branch and the number of parameters of just the early exit branch are compared between the CNN architecture and the SL-ViT architecture. Higher accuracies, lower errors, lower number of parameters and lower total FLOPS are highlighted in these tables. Furthermore, the acceleration caused by SL-ViT early exits, defined as the total FLOPS of the backbone network divided by the total FLOPS of the model up to and including the SL-ViT branch, is also provided.

Several observations can be made about these results. First, in all scenarios except one, SL-ViT early exits achieve a significantly higher accuracy. Even in the one exceptional scenario, namely branch 2 of ResNet152 in Table 6, the accuracy of SL-ViT is very close to its CNN counterpart. Secondly, while in some cases SL-ViT branches have an equal number of parameters

compared to CNN branches, in all scenarios, the total FLOPS of SL-ViT branches is lower, therefore SL-ViT branches are always more lightweight. Thirdly, in one scenario, namely branch 2 of ResNet152 in Table 7, removing the last residual connection in the SL-ViT architecture significantly improved the accuracy of the SL-ViT branch. Finally, in the AV2 branch location in Table 8, both CNN and SL-ViT are impractical branches since earlier branches with higher accuracies exist. This is perhaps due to the intrusive fusion operation in the first fusion block which might initially make the intermediate features more obscure. Nonetheless, even in this case, SL-ViT is more accurate.

Table 4: Comparison of different early exit architectures on the CIFAR-10 dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	66.74 ± 0.57%	**70.79 ± 0.72%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2	79.31 ± 0.81%	**81.18 ± 0.52%**	0.83M	**0.79M**	5.33B	**5.26B**	4.29
DenseNet201	1	71.27 ± 0.36%	**76.38 ± 0.33%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	80.64 ± 0.29%	**83.53 ± 0.37%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	77.27 ± 0.58%	**79.99 ± 0.20%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	79.55 ± 0.24%	**81.72 ± 0.53%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

Table 9 shows the result of applying the copycat fine-tuning strategy to SL-ViT branches for the CIFAR-10 dataset. Observe than in all cases, the accuracy is significantly increased compared to SL-ViT, which itself was more accurate than CNN based on Table 4. We also tested this strategy for the CIFAR-100 dataset with 10-to-1, 2-to-1 and 1-to-1 ratios of fake and real data, however, neither improved the overall accuracy. Perhaps another

Table 5: Comparison of different early exit architectures on the CIFAR-100 dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	34.93 ± 0.52%	**38.59 ± 1.40%**	0.80M	**0.61M**	1.66B	**1.64B**	13.77
	2	47.39 ± 0.65%	**53.93 ± 0.68%**	0.86M	**0.81M**	5.33B	**5.26B**	4.29
DenseNet201	1	33.91 ± 1.00%	**42.50 ± 0.69%**	0.80M	**0.61M**	2.55B	**2.53B**	3.39
	2	47.22 ± 0.45%	**50.76 ± 1.01%**	0.82M	**0.68M**	4.21B	**4.17B**	2.06
InceptionV3	1	43.18 ± 0.69%	**46.86 ± 0.57%**	0.63M	0.63M	2.17B	**2.14B**	2.66
	2	44.87 ± 0.83%	**49.07 ± 0.55%**	0.63M	0.63M	2.53B	**2.50B**	2.28

Table 6: Comparison of different early exit architectures on the Speech Commands dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	75.80 ± 0.73%	**84.05 ± 0.31%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2	**89.78 ± 0.24%**	89.63 ± 0.52%	0.84M	0.84M	5.33B	**5.26B**	4.29
DenseNet201	1	72.78 ± 0.64%	**87.94 ± 0.85%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	86.56 ± 0.61%	**90.93 ± 0.52%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	84.64 ± 0.88%	**87.62 ± 0.65%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	87.08 ± 1.11%	**88.33 ± 0.92%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

Table 7: Comparison of different early exit architectures on the GTZAN dataset

Backbone	Branch	Accuracy		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
ResNet152	1	67.01 ± 1.11%	**73.27 ± 0.91%**	0.78M	**0.59M**	1.66B	**1.64B**	13.77
	2*	80.26 ± 2.07%	**81.56 ± 1.57%**	0.83M	0.83M	5.33B	**5.26B**	4.29
DenseNet201	1	70.65 ± 1.23%	**76.38 ± 1.94%**	0.78M	**0.59M**	2.55B	**2.53B**	3.39
	2	81.72 ± 0.62%	**84.00 ± 1.67%**	0.80M	**0.66M**	4.21B	**4.17B**	2.06
InceptionV3	1	77.86 ± 0.90%	**79.42 ± 0.99%**	0.61M	**0.56M**	2.17B	**2.14B**	2.65
	2	78.90 ± 0.90%	**79.90 ± 0.79%**	0.61M	**0.56M**	2.53B	**2.49B**	2.28

*The last residual connection in the SL-ViT architecture was removed in this case

Table 8: Comparison of Different Early Exit Architectures on the DISCO Dataset

Backbone	Branch	MAE		Branch Params		Total FLOPS		Acceleration
		CNN	SL-ViT	CNN	SL-ViT	CNN	SL-ViT	SL-ViT
AudioCSRNet	V1	16.99 ± 0.28	**15.04 ± 0.71**	2.50M	**2.35M**	329.77B	**328.72B**	1.49
	AV1	17.00 ± 0.23	**14.58 ± 0.64**	2.52M	**2.36M**	331.37B	**330.31B**	1.48
	AV2	17.90 ± 0.25	**17.03 ± 1.04**	2.50M	**2.35M**	374.86B	**373.81B**	1.31

mixing ratio, choice of dataset and network to generate the fake dataset, optimizer or hyper-parameters such as learning rate may result in improvements for CIFAR-100.

Table 9: Effect of Copycat strategy demonstrated on the CIFAR-10 dataset

Backbone	Branch	Accuracy	
		SL-ViT	CC-SL-ViT
ResNet152	1	70.79 ± 0.72%	**71.61 ± 0.45%**
	2	81.18 ± 0.52%	**83.41 ± 0.15%**
DenseNet201	1	76.38 ± 0.33%	**78.34 ± 0.31%**
	2	83.53 ± 0.37%	**84.89 ± 0.43%**
InceptionV3	1	79.99 ± 0.20%	**80.78 ± 0.23%**
	2	81.72 ± 0.53%	**82.20 ± 0.40%**

Table 10: Comparison of improvements gained by SL-ViT with gains from knowledge distillation for the CIFAR-10 dataset.

Backbone	Branch	CNN (Baseline)	CNN with KD	SL-ViT (Ours)
ResNet152	1	66.74 ± 0.57%	69.31 ± 0.28%	**70.79 ± 0.72%**
	2	79.31 ± 0.81%	78.79 ± 0.61%	**81.18 ± 0.52%**
DenseNet201	1	71.27 ± 0.36%	73.93 ± 0.15%	**76.38 ± 0.33%**
	2	80.64 ± 0.29%	81.56 ± 0.12%	**83.53 ± 0.37%**
InceptionV3	1	77.27 ± 0.58%	78.37 ± 0.34%	**79.99 ± 0.20%**
	2	79.55 ± 0.24%	80.41 ± 0.43%	**81.72 ± 0.53%**

Even though other early exit methods focus on improving the training procedure and can be used in combination with our proposed architecture, comparing the improvements gained by utilizing such methods with improve-

ments gained from our approach can still provide insights into the significance of architecture design for early exits. Table 10 contains comparisons with knowledge distillation-based training similar to the method in. [12] for the CIFAR-10 dataset. Observe that in all cases, SL-ViT obtains a significantly higher accuracy compared to knowledge distillation.

5.1. Ablation Studies

Table 11 showcases the effect of using different architectural parameters on the accuracy of both SL-ViT and CNN branches. Where not specified, the CNN early exits have a 3×3 kernel size with no dilation, and the SL-ViT early exits have 12 attention heads, which are the baselines presented in previous tables. Other parameters such as the number of convolutional filters and padding size are adjusted accordingly in order to keep the number of parameters close to the baselines.

These results support our hypothesis that the improvements of SL-ViT are due to the fusion of local and global receptive fields. First, by increasing the number of attention heads in SL-ViT, the accuracy increases significantly while the parameters only slightly increase, hinting that learning multiple types of attention plays a major role in SL-ViT. Secondly, by increasing the CNN kernel size from 3×3 to 15×15 the accuracy is improved, yet it is still lower than that of SL-ViT. This is because even a large filter size does not provide a global receptive field. On the other hand, adding dilation to CNN decreases its accuracy compared to the CNN baseline. This is because dilated convolutions create holes in the receptive field, which increases the receptive field yet loses important local information. Thirdly, using two CNN layers also improves the accuracy compared to the CNN baseline, however, a higher

gain in accuracy was achieved using a larger kernel size. Moreover, two SL-ViT layers still obtain a higher accuracy compared to two CNN layers while having a lower overhead in terms of parameters. Finally, we show that even if the backbone is not pre-trained on ImageNet and is trained completely from scratch, SL-ViT still obtains a higher accuracy compared to CNN.

Table 11: Ablation studies: the effect of the number of attention heads, number of layers, dilation, kernel size and backbone pre-training on the accuracy of early exits placed on the first branch location of a ResNet152 backbone trained on the CIFAR-10 dataset

Architecture Params	Accuracy	Branch Params	FLOPS
SL-ViT (1 head)	67.92 ± 0.86%	0.55M	1.64B
SL-ViT (2 heads)	68.65 ± 0.90%	0.55M	1.64B
SL-ViT (4 heads)	69.08 ± 1.07%	0.56M	1.64B
SL-ViT (8 heads)	69.85 ± 1.12%	0.58M	1.64B
SL-ViT (12 heads)	**70.79 ± 0.72%**	0.59M	1.64B
SL-ViT (16 heads)	70.76 ± 0.40%	0.61M	1.64B
CNN (3×3 kernel)	66.74 ± 0.57%	0.78M	1.66B
CNN (11×11 kernel)	69.71 ± 1.06 %	0.78M	1.88B
CNN (15×15 kernel)	69.90 ± 0.68%	0.79M	2.02B
CNN (dilation 2)	66.61 ± 0.47%	0.78M	1.66B
CNN (dilation 3)	65.43 ± 0.32%	0.78M	1.66B
SL-ViT (2 layers)	**71.89 ± 0.75%**	0.65M	1.64B
CNN (2 layers)	67.68 ± 1.06%	0.78M	1.66B
SL-ViT (no backbone pre-training)	**63.14 ± 0.57%**	0.59M	1.64B
CNN (no backbone pre-training)	62.86 ± 0.99%	0.78M	1.66B

Finally, we discovered that removing the second residual connection in the transformer encoder may lead to an increase in the overall accuracy of our method. This effect was moderate in most cases, yet quite significant in others. An example of this effect is shown in Table 12 for the Speech Commands dataset. We chose to keep the residual connection whenever the

effect was moderate and only remove it if it leads to a significantly higher accuracy. Such cases are highlighted in our experiments (Table 7).

Table 12: Ablation studies: the effect of removing the last residual connection in the transformer encoder for the Speech Commands dataset

Backbone	Branch Number	Accuracy of SL-ViT	Accuracy of SL-ViT without the Last Residual
ResNet152	1	**84.05 ± 0.31%**	83.67 ± 0.85%
	2	**89.63 ± 0.52%**	85.79 ± 0.58%
DenseNet201	1	87.94 ± 0.85%	**88.35 ± 0.24%**
	2	90.93 ± 0.52%	**91.08 ± 0.52%**
InceptionV3	1	**87.62 ± 0.65%**	86.10 ± 0.32%
	2	**88.33 ± 0.92%**	88.21 ± 0.45%

5.2. Early Exit Procedure

Since our method improves the accuracy in all early exit locations, it provides improvements regardless of which early exit procedure is used. For instance, suppose a confidence-based method is used where the result of an early exit branch is selected as the final answer if it is confident enough. In this setting, our method will lead to faster inference on average, since more accurate branches lead to higher confidence.

Another example would be the anytime prediction setting explained in the introduction, for instance, an edge server which receives inputs from many IoT devices and needs to provide a response for each input within a strict deadline. The transmission time from the IoT devices to the server changes over time due to network congestion. Moreover, the computational workload of the server varies over time, therefore, the time budget available for each

input is not known beforehand, and the inference can be interrupted at any moment. In this case, the output of the latest exit is used as the final answer. In such a setting, our method will lead to more accurate results and faster inference, since SL-ViT exits are more accurate and have less overhead.

To make this more clear, we have conducted experiments within the anytime prediction setting, where a random time budget is assigned to each image in the CIFAR-10 test set. We use the DenseNet backbone and the two branch locations specified in Table 2. We compare the average accuracy and FLOPS between the case where SL-ViT branches are used and the case where CNN branches are utilized. The results of these experiments are shown in Table 13. It can be observed that the multi-exit network with SL-ViT branches achieves a significantly higher average accuracy while having lower average FLOPS.

Table 13: Comparison of the average accuracy and FLOPS in the anytime prediction setting between a multi-exit DenseNet with SL-ViT early exits and one with CNN early exits.

Model	Average Accuracy	Average FLOPS
Multi-Exit DenseNet with CNN Branches	82.79 ± 0.17%	5.11B
Multi-Exit DenseNet with SL-ViT Branches	**85.65 ± 0.21%**	**5.09B**

6. Discussion and Conclusion

We showed that the proposed architecture for early exit branches, namely single-layer vision transformer (SL-ViT) can consistently obtain a signifi-

cantly higher accuracy compared to conventional methods while introducing a lower overhead in terms of FLOPS. We showed that our method works for both classification and regression problems, in both single and multi-modal scenarios, and across different backbone networks and branch locations.

As previously mentioned, one possible explanation for why SL-ViT performs better, is the fact that even a single layer of transformer encoder has a global receptive field since each patch can attend to any other patch, while a convolutional layer has a limited receptive field and can only access the immediate vicinity based on its filter size. There are several clues that point to this explanation. First, Table 11 suggests that the attention mechanism plays a major role in the accuracy improvements. Secondly, based on Tables 4 to 8, the accuracy improvements are generally higher in earlier branches, where the receptive field of the backbone network up to the branch location is lower compared to later branches. Finally, the incorporation of global scale and global information such as perspective is known to be of great importance in crowd counting, and many crowd counting methods utilize visual attention mechanisms and dilated convolution layers to this end [41], which can explain why our method performs well for this problem.

Moreover, we showed that our fine-tuning strategy, namely Copycat SL-ViT, has the potential to further increase the accuracy of SL-ViT branches. It is well-known that with deep learning, more data almost always improves the final outcome, and this is especially true for vision transformers which are known to be data-hungry [32]. The copycat strategy can at times artificially increase the size of the dataset without introducing too much noise and thus improve the final result.

Furthermore, we introduced a novel approach for fusing audio and visual features within early exits using vision transformers. The importance of fusion inside early exits is that it creates much more options for branch locations, since a combination of any layer in the visual channel of the backbone network with any layer in the audio channel of the backbone can be selected. This allows for a more fine-grained dynamic inference, meaning a more recent result is available whenever the inference is interrupted in an anytime prediction setting, which is likely to be more accurate than earlier results.

Acknowledgment

This work was partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 957337, and by the Danish Council for Independent Research under Grant No. 9131-00119B. This publication reflects the authors views only. The European Commission and the Danish Council for Independent Research are not responsible for any use that may be made of the information it contains.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444. doi:10.1038/nature14539.
URL https://doi.org/10.1038/nature14539

[2] Y. Cheng, D. Wang, P. Zhou, T. Zhang, Model compression and acceleration for deep neural networks: The principles, progress, and challenges, IEEE Signal Processing Magazine 35 (1) (2018) 126–136.

doi:10.1109/msp.2017.2765695.
URL https://doi.org/10.1109/msp.2017.2765695

[3] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-net: ImageNet classification using binary convolutional neural networks, in: Computer Vision – ECCV 2016, Springer International Publishing, 2016, pp. 525–542. doi:10.1007/978-3-319-46493-0_32.
URL https://doi.org/10.1007/978-3-319-46493-0_32

[4] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, arXiv:1608.08710 (Unpublished results).

[5] D. T. Tran, A. Iosifidis, M. Gabbouj, Improving efficiency in convolutional neural network with multilinear filters, Neural Networks 105 (2018) 328–339.

[6] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv:1503.02531 (Unpublished results).

[7] J. Chen, X. Ran, Deep learning with edge computing: A review, Proceedings of the IEEE 107 (8) (2019) 1655–1674. doi:10.1109/jproc.2019.2921977.
URL https://doi.org/10.1109/jproc.2019.2921977

[8] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Communications Surveys & Tutorials 22 (2) (2020) 869–904. doi:10.1109/comst.2020.2970550.
URL https://doi.org/10.1109/comst.2020.2970550

[9] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, Y. Wang, Dynamic neural networks: A survey, arXiv:2102.04906 (Unpublished results).

[10] S. Scardapane, M. Scarpiniti, E. Baccarelli, A. Uncini, Why should we add early exits to neural networks?, Cognitive Computation 12 (5) (2020) 954–966. doi:10.1007/s12559-020-09734-4.
URL https://doi.org/10.1007/s12559-020-09734-4

[11] S. Teerapittayanon, B. McDanel, H. Kung, BranchyNet: Fast inference via early exiting from deep neural networks, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016. doi:10.1109/icpr.2016.7900006.
URL https://doi.org/10.1109/icpr.2016.7900006

[12] M. Phuong, C. Lampert, Distillation-based training for multi-exit architectures, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, 2019. doi:10.1109/iccv.2019.00144.
URL https://doi.org/10.1109/iccv.2019.00144

[13] A. Bakhtiarnia, Q. Zhang, A. Iosifidis, Improving the accuracy of early exits in multi-exit architectures via curriculum learning, in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–8. doi:10.1109/IJCNN52387.2021.9533875.

[14] A. Bakhtiarnia, Q. Zhang, A. Iosifidis, Multi-exit vision transformer for dynamic inference, The 32nd British Machine Vision Conference (BMVC 2021) (2021).

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: International Conference on Learning Representations, 2021.
URL https://openreview.net/forum?id=YicbFdNTTy

[16] J. R. Correia-Silva, R. F. Berriel, C. Badue, A. F. de Souza, T. Oliveira-Santos, Copycat CNN: Stealing knowledge by persuading confession with random non-labeled data, in: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, 2018. doi:10.1109/ijcnn.2018.8489592.
URL https://doi.org/10.1109/ijcnn.2018.8489592

[17] E. Baccarelli, S. Scardapane, M. Scarpiniti, A. Momenzadeh, A. Uncini, Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported IoT applications, Information Sciences 521 (2020) 107–143. doi:10.1016/j.ins.2020.02.041.
URL https://doi.org/10.1016/j.ins.2020.02.041

[18] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, K. Ma, Be your own teacher: Improve the performance of convolutional neural networks via self distillation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3713–3722.

[19] H. Li, H. Zhang, X. Qi, R. Yang, G. Huang, Improved techniques for

training adaptive deep networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1891–1900.

[20] M. Wołczyk, B. Wójcik, K. Bałazy, I. Podolak, J. Tabor, M. Śmieja, T. Trzcinski, Zero time waste: Recycling predictions in early exit neural networks, Advances in Neural Information Processing Systems 34 (2021).

[21] Z. Jie, P. Sun, X. Li, J. Feng, W. Liu, Anytime recognition with routing convolutional networks, IEEE transactions on pattern analysis and machine intelligence 43 (6) (2019) 1875–1886.

[22] J. Pomponi, S. Scardapane, A. Uncini, A probabilistic re-intepretation of confidence scores in multi-exit models, Entropy 24 (1) (2021) 1.

[23] H. Lee, C.-J. Hsieh, J.-S. Lee, Local critic training for model-parallel learning of deep neural networks, IEEE Transactions on Neural Networks and Learning Systems (2021).

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[25] M. Elbayad, J. Gu, E. Grave, M. Auli, Depth-adaptive transformer, in: International Conference on Learning Representations, 2020.
URL `https://openreview.net/forum?id=SJg7KhVKPH`

[26] A. Graves, Adaptive computation time for recurrent neural networks, arXiv:1603.08983 (2016).

[27] A. Banino, J. Balaguer, C. Blundell, Pondernet: Learning to ponder, arXiv:2107.05407 (2021).

[28] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, J. E. Gonzalez, Skipnet: Learning dynamic routing in convolutional networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[29] X. Gao, Y. Zhao, L. Dudziak, R. D. Mullins, C. Xu, Dynamic channel pruning: Feature boosting and suppression, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=BJxh2j0qYm

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[31] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, D. Tao, A survey on visual transformer, arXiv:2012.12556 (Unpublished results).

[32] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, M. Shah, Transformers in vision: A survey, arXiv:2101.01169 (Unpublished results).

[33] S. F. Bhat, I. Alhashim, P. Wonka, Adabins: Depth estimation using adaptive bins, arXiv:2011.14141 (Unpublished results).

[34] K. Choi, G. Fazekas, M. Sandler, Automatic tagging using deep convolutional neural networks, arXiv:1606.00298 (Unpublished results).

[35] J. Lee, T. Kim, J. Park, J. Nam, Raw waveform-based audio classification using sample-level cnn architectures, arXiv:1712.00866 (Unpublished results).

[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.90.
URL https://doi.org/10.1109/cvpr.2016.90

[37] G. Huang, Z. Liu, L. V. D. Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017. doi:10.1109/cvpr.2017.243.
URL https://doi.org/10.1109/cvpr.2017.243

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.308.
URL https://doi.org/10.1109/cvpr.2016.308

[39] K. Palanisamy, D. Singhania, A. Yao, Rethinking cnn models for audio classification, arXiv:2007.11154 (Unpublished results).

[40] V. A. Sindagi, V. M. Patel, A survey of recent advances in CNN-based single image crowd counting and density estimation, Pattern Recognition Letters 107 (2018) 3–16. doi:10.1016/j.patrec.2017.07.007.
URL https://doi.org/10.1016/j.patrec.2017.07.007

[41] G. Gao, J. Gao, Q. Liu, Q. Wang, Y. Wang, Cnn-based density estimation and crowd counting: A survey, arXiv:2003.12783 (Unpublished results).

[42] Y. Zhang, D. Zhou, S. Chen, S. Gao, Y. Ma, Single-image crowd counting via multi-column convolutional neural network, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016. doi:10.1109/cvpr.2016.70.
URL https://doi.org/10.1109/cvpr.2016.70

[43] C. Zhang, H. Li, X. Wang, X. Yang, Cross-scene crowd counting via deep convolutional neural networks, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015. doi:10.1109/cvpr.2015.7298684.
URL https://doi.org/10.1109/cvpr.2015.7298684

[44] F. Dai, H. Liu, Y. Ma, J. Cao, Q. Zhao, Y. Zhang, Dense scale network for crowd counting, arXiv:1906.09707 (Unpublished results).

[45] D. Hu, L. Mou, Q. Wang, J. Gao, Y. Hua, D. Dou, X. X. Zhu, Ambient sound helps: Audiovisual crowd counting in extreme conditions, arXiv:2005.07097 (Unpublished results).

[46] Y. Li, X. Zhang, D. Chen, CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2018. doi:10.1109/cvpr.2018.00120.
URL https://doi.org/10.1109/cvpr.2018.00120

[47] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1409.1556

[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009. doi:10.1109/cvpr.2009.5206848.
URL https://doi.org/10.1109/cvpr.2009.5206848

[49] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, K. Wilson, CNN architectures for large-scale audio classification, in: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017. doi:10.1109/icassp.2017.7952132.
URL https://doi.org/10.1109/icassp.2017.7952132

[50] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: F. Bach, D. Blei (Eds.),

Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, pp. 448–456.
URL http://proceedings.mlr.press/v37/ioffe15.html

[51] T. Hu, T. Chen, H. Wang, Z. Wang, Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference, in: ICLR, 2020.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[53] A. LeNail, NN-SVG: Publication-ready neural network architecture schematics, Journal of Open Source Software 4 (33) (2019) 747. doi:10.21105/joss.00747.
URL https://doi.org/10.21105/joss.00747

[54] A. Araujo, W. Norris, J. Sim, Computing receptive fields of convolutional neural networks, DistillHttps://distill.pub/2019/computing-receptive-fields (2019). doi:10.23915/distill.00021.

[55] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6230–6239. doi:10.1109/CVPR.2017.660.

[56] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Transactions on Pattern Analy-

sis & Machine Intelligence 40 (04) (2018) 834–848. doi:10.1109/TPAMI.2017.2699184.

[57] A. Krizhevsky, Learning multiple layers of features from tiny images (Unpublished results).

[58] P. Warden, Speech commands: A dataset for limited-vocabulary speech recognition, arXiv:1804.03209 (Unpublished results).

[59] G. Tzanetakis, P. Cook, Musical genre classification of audio signals, IEEE Transactions on Speech and Audio Processing 10 (5) (2002) 293–302. doi:10.1109/tsa.2002.800560.
URL https://doi.org/10.1109/tsa.2002.800560

[60] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
URL http://arxiv.org/abs/1412.6980

[61] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=Bkg6RiCqY7

[62] J. Frankle, D. J. Schwab, A. S. Morcos, Training batchnorm and only batchnorm: On the expressive power of random features in cnns, arXiv:2003.00152 (Unpublished results).

[63] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.