Multi-Exit Vision Transformer for Dynamic Inference

Arian Bakhtiarnia, Qi Zhang and Alexandros Iosifidis

DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark

{arianbakh,qz,ai}@ece.au.dk

Abstract—**Deep neural networks can be converted to multi-exit architectures by inserting early exit branches after some of their intermediate layers. This allows their inference process to become dynamic, which is useful for time critical IoT applications with stringent latency requirements, but with time-variant communication and computation resources. In particular, in edge computing systems and IoT networks where the exact computation time budget is variable and not known beforehand. Vision Transformer is a recently proposed architecture which has since found many applications across various domains of computer vision. In this work, we propose seven different architectures for early exit branches that can be used for dynamic inference in Vision Transformer backbones. Through extensive experiments involving both classification and regression problems, we show that each one of our proposed architectures could prove useful in the trade-off between accuracy and speed.**

I. INTRODUCTION

Deep neural networks have achieved immense success in recent years [1], however, they commonly consist of many interconnected layers containing millions of parameters which require high computational resources and cause slow inference speed. Dynamic inference methods [2] allow deep models to modify their computation graph during inference in order to alleviate this problem. One such method is *early exiting* [3], [4], leading to *multi-exit architectures*, where early exit *branches* are inserted after intermediate hidden layers of the *backbone* network and provide early results, albeit with less accuracy compared to the final result of the backbone network.

Early exits are useful in computationally restricted settings such as mobile and edge computing, where early results can be used for "easy" inputs to save resources. Additionally, multi-exit architectures can be helpful in *anytime prediction* settings where the inference process may be interrupted at any time and the network is expected to provide a response even if it was interrupted before completion. Examples of anytime prediction settings are distributed environments such as edge computing systems and IoT networks, where the latency depends on the communication channels, which means the exact computation time budget is not known beforehand and varies over time. Here, the latest result provided by a multi-exit architecture can be given as output whenever the network is interrupted.

Vision Transformer [5] is a recently proposed architecture for computer vision which has since been applied to various problems, such as image classification, object detection, depth estimation, and many more [6]. To the best of our knowledge, multi-exit Vision Transformer architectures have not yet been studied in the literature, which limits the application of Vision Transformers in mobile and edge computing. In this work, we propose seven different architectures for early exit branches that can be inserted into Vision Transformer backbones. Through extensive experiments on both image classification and crowd counting, the latter being a regression problem, we show that depending on the particular problem at hand, each of these architectures has the potential to be useful in the trade-off between classification accuracy and inference speed. Our code will be made publicly available at https://gitlab.au.dk/maleci/multiexitvit.

II. RELATED WORK

A. Multi-Exit Architectures

A deep neural network (DNN) can be formulated as a function $f(X) = f_L(f_{L-1}(...f_1(X)))$ where X is the input, L is the number of layers in the DNN and f_i is the differentiable operator at layer i. The output of layer i is denoted by $h_i = f_i(h_{i-1})$ and θ_i refers to the trainable parameters of $f_i(\cdot)$. The training process for this DNN can be formulated as shown in Equation (1) where $\theta = \bigcup_{i=1}^{L} \theta_i$ is the set of all trainable parameters of the DNN, $\{X_n, y_n\}_{n=1}^{N}$ is the set of training samples and $l(\cdot)$ is a loss function.

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

In order to convert a DNN to a multi-exit architecture, an early exit branch $c_b(h_b) = y_b$ is placed at every selected branch location $b \in B \subseteq \{1, .., L\}$, where c_b is the classifier or regressor producing the early result y_b. The schematic illustration of a multi-exit architecture is shown in Figure 1 (a). Since there are multiple outputs in a multi-exit architecture, its training procedure is not as straightforward as Equation (1). Three major strategies for training multi-exit architectures exist in the literature [3]. The *classifier-wise* strategy freezes the backbone, meaning the parameters θ will not be modified, and trains the branches separately and independent of each other or the backbone. In the *end-to-end* strategy, the loss function $l_t = l + \sum_{b \in B} \lambda_b l_b$ combines the losses l_b of the early exit branches with the backbone's loss and trains the entire multi-exit architecture simultaneously. In this strategy, the contribution of the loss of the branch at location b is captured by weight score λ_b. Finally, the *layer-wise* strategy first trains the layers up to and including the first early exit

branch. Subsequently, the previous layers are frozen and the rest of the layers up to and including the second branch are trained, and this operation is repeated until the entire backbone has been trained.

In the end-to-end and layer-wise strategies, the number of branches and their placement create trade-offs between the accuracy of different exits. In addition, with the end-to-end strategy, the weight scores introduce new hyper-parameters. In contrast, no trade-offs or new hyper-parameters need to be considered with the classifier-wise strategy. However, since in this case the parameters of the backbone remain unchanged, fewer parameters are affected during the training of the branches. In this work we investigate all three training strategies.

It is important to note that branches placed later in the networks do not necessarily result in a higher accuracy compared to previous branches. We use the term *impractical* in order to refer to such branches, and the term *practical* for branches with a higher accuracy than all previous branches. The usage of impractical branches would not be sensible since earlier branches with a higher accuracy exist.

B. Vision Transformer

Vision Transformer (ViT) [5] is an adaptation of the Transformer architecture [7] for computer vision problems. At the core of the Transformer is the *self-attention* layer, which takes a sequence $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ as input and outputs the sequence $Z = (z_1, \ldots, z_n) \in \mathbb{R}^{n \times d_v}$, which can be formulated as Equation (2), where $Q = XW^Q$, $K = XW^K$ and $V = XW^V$ are query, key and value matrices, respectively, in which W^Q, W^K and W^V are learnable weight matrices [5]. $d_k = d_q$ are the size of the vectors in query and key matrices.

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

In order 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 several self-attention blocks, each with its own set of learnable parameters. Figure 1 (b) depicts the Vision Transformer architecture, where initially an input image is cut into several image patches. A sequence of patch embeddings is then formed by projecting each patch and concatenating a positional embedding to the resulting vector. An extra learnable *classification token* is also appended to the sequence. The sequence passes through L Transformer encoder layers, each containing multi-head attention layers among other operations. Finally, the output vector corresponding to the classification token is passed on to an MLP dubbed *classification head* to obtain the final result.

C. Attention-Free, MLP-Based Architectures

Several MLP-based architectures for computer vision that also operate on sequences of image patches have been recently proposed [8]. The aim of these architectures is to reduce the computational cost of ViT by removing the attention mechanism, while achieving a comparable performance by
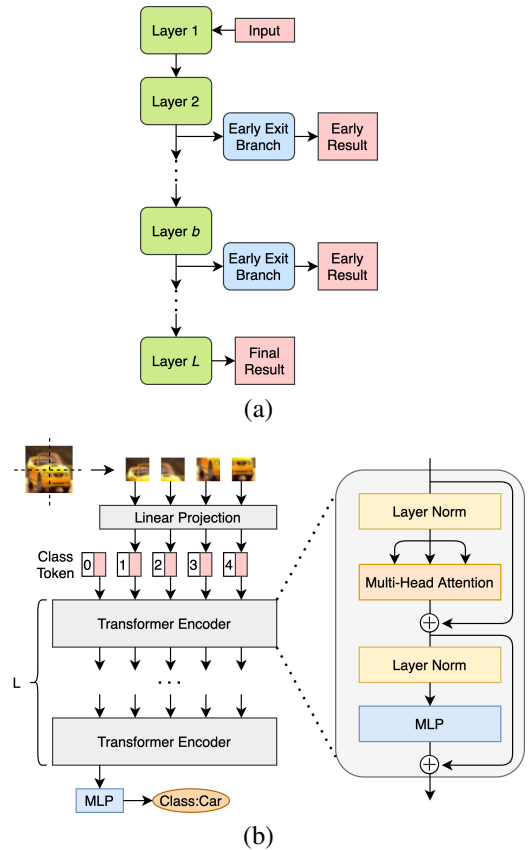

Fig. 1. (a) Schematic illustration of a multi-exit; and (b) Vision Transformer architecture.

preserving a global receptive field similar to that of ViT. Since the intermediate representations in the hidden layers of ViT is in the form of a sequence of patches, it is simple to use the building blocks of these MLP-based architectures as early exit branches placed on ViT backbones. These building blocks create more lightweight branches compared to the Transformer encoders in ViT.

One such architecture called *MLP-Mixer* [9] is shown in Figure 2 (a). Each *mixer layer* in MLP-Mixer consists of *token mixing* and *channel mixing* operations, which are formulated as Equations (3a) and (3b), where $f_1(\cdot) \ldots f_4(\cdot)$ are linear layers and $\sigma(\cdot)$ is the GELU activation function. The output of the final mixer layer is passed on to a global average pooling layer and then a fully connected layer.

$$U = X + f_2(\sigma(f_1(Norm(X)^T)))^T \tag{3a}$$

$$Y = U + f_4(\sigma(f_3(Norm(U)))) \tag{3b}$$

A similar architecture called *ResMLP* [10] is shown in Figure 2 (b). Each *ResMLP layer* consists of a *cross-patch sublayer* and a *cross-channel sublayer*, which are formulated as Equations (4a) and (4b). In ResMLP, normalization is carried out using an affine transformation instead of layer normalization, as shown in Equation (4c) where α and β are learnable vectors that scale and shift the input. Similarly, the
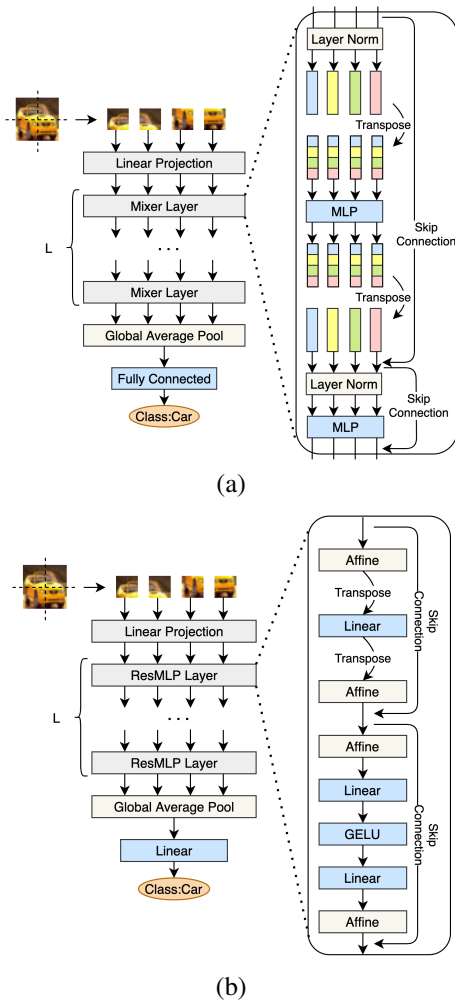
Fig. 2. (a) MLP-Mixer architecture; and (b) ResMLP architecture.

output of the final ResMLP layer is passed on to a global average pooling layer and then a fully connected layer.

$$U = X + Norm(f_1(Norm(X)^T)^T) \quad (4a)$$

$$Y = U + Norm(f_3(\sigma(f_2(Norm(U))))) \quad (4b)$$

$$Norm(X) = Aff_{\alpha,\beta}(X) = Diag(\alpha)X + \beta \quad (4c)$$

III. MULTI-EXIT VISION TRANSFORMER

We assume a high-performing ViT backbone is available for the problem at hand, and the goal is to convert this backbone to a multi-exit architecture in order to allow for dynamic inference. We propose seven different architectures for the early exit branches added after intermediate layers of a ViT backbone. The most intuitive approach, which we call *MLP-EE*, is to add an MLP to the classification token of the intermediate layer, similar to the classification head in the ViT backbone. Even though MLP-EE is very lightweight, it may not contain enough parameters and layers to extract useful features, particularly for exits placed early. Moreover, it does not process tokens other than the classification token.

Another approach is to convert the sequence of token vectors in the intermediate layers of the ViT backbone to a 2D grid and further process them using convolutional filters, leading to 3 different architectures we call *CNN-Ignore-EE*, *CNN-Add-EE* and *CNN-Project-EE*, each handling the classification token in a different way. Note that even though the intermediate layer is in the form of a sequence, each vector in the sequence corresponds to a patch of the input image, therefore putting the vectors back in a 2D grid simulates their original neighborhood which is essential when using convolutional filters that have a local receptive field. The motivation behind this approach is that convolutional filters are the current approach in the literature for early exiting [3], [11], [12] and can act as a baseline for the other proposed architectures. Furthermore, convolutional filters introduce low overhead in terms of parameters and computation. Additionally, a fusion of CNNs that can capture local structure very well but can not handle long range interactions, with ViTs which can process long range interactions, seems natural and may combine the advantages of both [8].

On the other hand, the local receptive field of CNN-based early exits may prove to be a drawback. An alternative that can overcome this limitation is using the Transformer encoder layer instead of the convolutional filters, which we call *ViT-EE*. Indeed, it has been shown that Transformer encoder layers can create superior early exits for CNN backbones by introducing a global receptive field [13]. However, since the layers of ViT backbones already have a global receptive field, it is not clear whether ViT-EE will have the same advantage over CNN-based early exits in ViT backbones as well. Another advantage of using Transformer encoder is the simplicity of its structure, which means it can handle various other data types such as point-clouds and even cross-modal data [8], [13]. The main drawback of ViT-EE is its high overhead, however, the building blocks of the recently proposed attention-free MLP-based architectures can serve as more lightweight alternatives that still maintain a global receptive field and structure simplicity, leading to *ResMLP-EE* and *MLP-Mixer-EE*.

Formally, the output of Transformer encoder b, denoted by P^b, consists of patch embeddings $p_1^b, \ldots, p_N^b$ corresponding to the input image patches, as well vector p_0^b corresponding to the classification token. Since the shape of the intermediate representations is the same for all of the hidden layers, without loss of generality, we assume that the early exit branch is to be placed after Transformer encoder b. In MLP-EE, shown in Figure 3 (a), P^b is normalized to obtain $\bar{P}^b = Norm(P^b)$. Subsequently, an MLP consisting of three dense layers with two dropout layers in between takes $\bar{p}_0^b$ as input, where $\bar{P}^b = (\bar{p}_0^b, \ldots, \bar{p}_N^b)$, and outputs the early result. The MLP layers in all our proposed architectures have the same three layers. In ViT-EE, shown in Figure 3 (b), P^b is given as input to a Transformer encoder layer [13]. The output of the Transformer encoder is then normalized and passed on to an MLP, similar to the previous architecture.

In CNN-based early exits, the N patch embeddings $p_1^b, \ldots, p_N^b$ can be reshaped into a tensor $C^b \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times d_v}$,
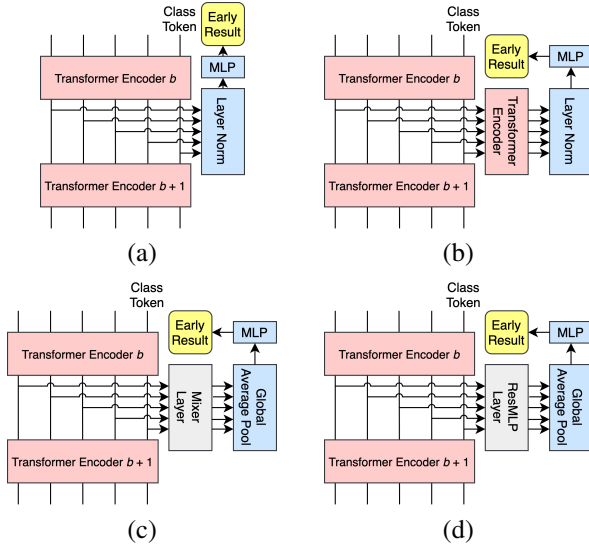
Fig. 3. (a) MLP-EE; (b) ViT-EE; (c) MLP-Mixer-EE; and (d) ResMLP-EE early exit branch architectures.

Fig. 4. (a) CNN-Add-EE; (b) CNN-Project-EE and (c) CNN-Ignore-EE early exit branch architectures.

akin to an intermediate representation in a CNN backbone, with height and width of $\sqrt{N}$ and d_v channels, and then passed on to a convolution layer, a max pooling layer and an MLP to obtain the early result. However, it is not clear what should be done with classification token $\bar{p}_0^b$. A similar situation arises in dense prediction using Vision Transformers, where three ways for dealing with the classification token are proposed [14]. In CNN-Add-EE, the classification token is added to every patch embedding, leading to $\bar{C}^b = (p_1^b + p_0^b, p_2^b + p_0^b, \ldots, p_N^b + p_0^b)$; in CNN-Project-EE, the classification token is concatenated to every patch embedding, leading to $\bar{C}^b = (concat(p_1^b, p_0^b), concat(p_2^b, p_0^b), \ldots, concat(p_N^b, p_0^b))$; and in CNN-Ignore-EE, the classification token is ignored and discarded, leading to $\bar{C}^b = C^b$. These three alternative architectures are depicted in figure 4.

As previously mentioned, the building blocks of attention-free MLP-based architectures can be low-overhead alternatives for ViT-EE which uses a Transformer encoder layer. Figure 3 (c) and (d) shows the MLP-Mixer-EE and ResMLP-EE early exit branch architectures, respectively. Note that similar to the original MLP-Mixer and ResMLP architectures, the output of the mixer layer and the ResMLP layer are passed on to a global average pooling layer.

IV. RESULTS

For the image classification experiments, we use CIFAR-10, CIFAR-100 and Fashion MNIST datasets [15], [16]. We use ViT-B/16 architectures with the original pre-trained weights provided by the authors [5] as backbones, and we train them on our target datasets using a cross-entropy loss function. For the regression experiments, we investigate crowd counting, which is the problem of counting the total number of people present in a given image [17]. We use DISCO [18] as the dataset and Tra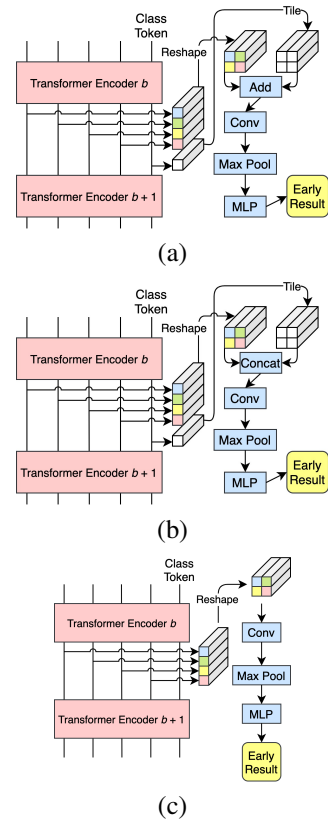nsCrowd [19] which is a ViT-based architecture as the backbone. Mean absolute error (MAE) is commonly used to evaluate the accuracy of crowd counting models [20]. All three backbones have 12 Transformer encoder layers.

All our models were trained using the Adam optimizer [21] with learning rates of $\{10^{-3}, 10^{-4}, 10^{-5}\}$ and the best result is selected. 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.

Note that while early exits have been recently attached to high-performing CNN backbones [13], [22], [23], there is no prior work for early exits on Vision Transformer backbones. Since the performance obtained by Vision Transformer backbones is improved by a large margin, we omit listing the comparison with early exits on CNN backbones in the following results.

A. Classifier-Wise

With the classifier-wise training strategy, since the branches do not affect each other or the backbone, we place and train early exit branches with each of our proposed architectures at every single layer in the backbone. For each early exit branch, we record the accuracy of its early results as well as the total FLOPs up to and including the branch. The results are depicted in Figure 5, where all practical early exits are circled. In addition, the accuracy of the final exit of the backbone is shown in these figures.

These results can be used to select a collection of lightweight high-performing branches. With dynamic inference, it is desirable for the model to be as fine-grained as possible, therefore, with the classifier-wise strategy where placing more branches does not affect other branches or the backbone, it is desirable to place as many branches as possible on the backbone. To make this more clear, imagine a scenario where only two exit options are available: option A with 80% accuracy and 3B FLOPS, and option B with 90% accuracy and 6B FLOPS. If the computation budget at hand is 5B FLOPS, the only possible option to choose is A, resulting in 80% accuracy. However, with a finer-grained model that also includes option C with 85% accuracy and 4B FLOPS, choosing option C leads to 85% accuracy. Hence we examine all possible branch locations: if there exists a single practical branch at a location, that branch should be added at that location; if there are no practical branches at a location, then no branches should be added there, since more accurate and more lightweight exits are available; and if there are more than one practical branches at a location (for instance, with the DISCO dataset in Figure 5 (a), both CNN-Ignore-EE and CNN-Project-EE make practical branches at layer 2) it means that there is a trade-off between accuracy and computation at that location, and the proper branch should be selected based on the particular application. Alternatively, it is possible to deploy multiple branches at the same location simultaneously, and exit the one that fits the budget during inference. Note that with the classifier-wise strategy, there can be different branch types on the same backbone, for instance, there can be a CNN-Add-EE branch at location 1 and a ViT-EE branch at location 2.

Several observations can be made from these results. First, all of our proposed architectures create at least one practical branch. As expected, MLP-EE does not contain enough parameters and layers to extract useful features in early locations on its own, and thus performs poorly, while it catches up in the later locations where the features extracted by the intermediate layers can compensate. Furthermore, MLP-EE only processes the classification head, which contains only low-level features in very early layers. However, MLP-EE always creates the first practical branch as it is the most lightweight. Secondly, CNN-based branches outperform other types in the first few locations. This is likely because the fusion of convolutional layers that capture local interactions well, with the global attention of the backbone, combines the best of both worlds. However, this effect seems to fade in later locations, perhaps since several layers of the backbone are able to capture both local and global interactions fairly well. In addition, CNN-Ignore-EE outperforms other CNN-based early exits in most of these early cases, as the classification token in the very early layers contains only low-level features. Thirdly, aside from the very early locations where CNN-based branches dominate, ResMLP-EE performs better in classification problems, while ViT-EE performs better in crowd counting. Evident from the use of visual attention mechanisms and dilated convolutions in many high-performing models for crowd counting [17], global

information such as perspective plays an important role in crowd counting, therefore, ViT-EE which can capture multiple types of attention through the use of the multi-head attention layer in Transformer encoder, outperforms ResMLP-EE which does not include a mechanism for handling multiple types of attention. Fourthly, observe that MLP-Mixer-EE outperforms ResMLP-EE in most locations in the crowd counting cases. This is because the affine transformation in ResMLP can be used instead of normalization when the training is stable [10], however, with crowd counting, the training process is not as stable as image classification.

Note that in the last six locations in CIFAR-10 and Fashion MNIST cases, the differences between the performance of different branch types are minuscule, and therefore less informative. Moreover, observe that unlike multi-exit architectures with CNN backbones, branches placed later on a ViT backbone do not necessarily provide a higher accuracy compared to previous branches. This is because in CNN backbones, the network has a very local receptive field in the early layers, and the receptive field gradually increases throughout the network, whereas ViTs have a global receptive field from the very first layer. This means that the accuracy of later branches of CNN backbones is expected to increase since the receptive field has increased, whereas in ViT backbones, later intermediate layers do not have any advantages in terms of the receptive field, thus their branches may obtain a lower accuracy. Finally, note that in the DISCO experiments, some of the very late early exit branches achieve a lower MAE compared to the final exit. This is because the MLP in our proposed architectures consists of three layers while the MLP in the ViT-B/16 backbone has one.

B. End-to-End and Layer-Wise

Unlike the classifier-wise training strategy, it is not possible to conduct a comprehensive study of the end-to-end and layer-wise strategies, since there are $2^{L-1} - 1$ possible branch placements. In addition, the end-to-end strategy can have infinitely many weight values for each of the placements. Therefore, for these training strategies, we only investigate two cases; one where a single early exit branch is placed after the sixth layer; the other where three branches are placed after the third, sixth and ninth layers. In both cases, the contribution of the final exit to the loss is double the contribution of the early exits.

Results are summarized in Tables I and II. In all image classification cases, final accuracy is decreased compared to the backbones without early exits, which have an accuracy of 98.31% for CIFAR-10, 91.24% for CIFAR-100 and 95.00% for Fashion MNIST. However, in crowd counting, the final MAE is improved from the original 11.07 when a single MLP-EE, ViT-EE or MLP-Mixer-EE branch is used. Similar to the classifier-wise strategy, in both cases involving the DISCO dataset, MLP-Mixer-EE outperforms ResMLP-EE for the same reason explained above. Furthermore, ViT-EE outperforms other branch types in most cases, particularly when there are 3 exit branches, and performs very high in others. Since the end-to-end training strategy affects the
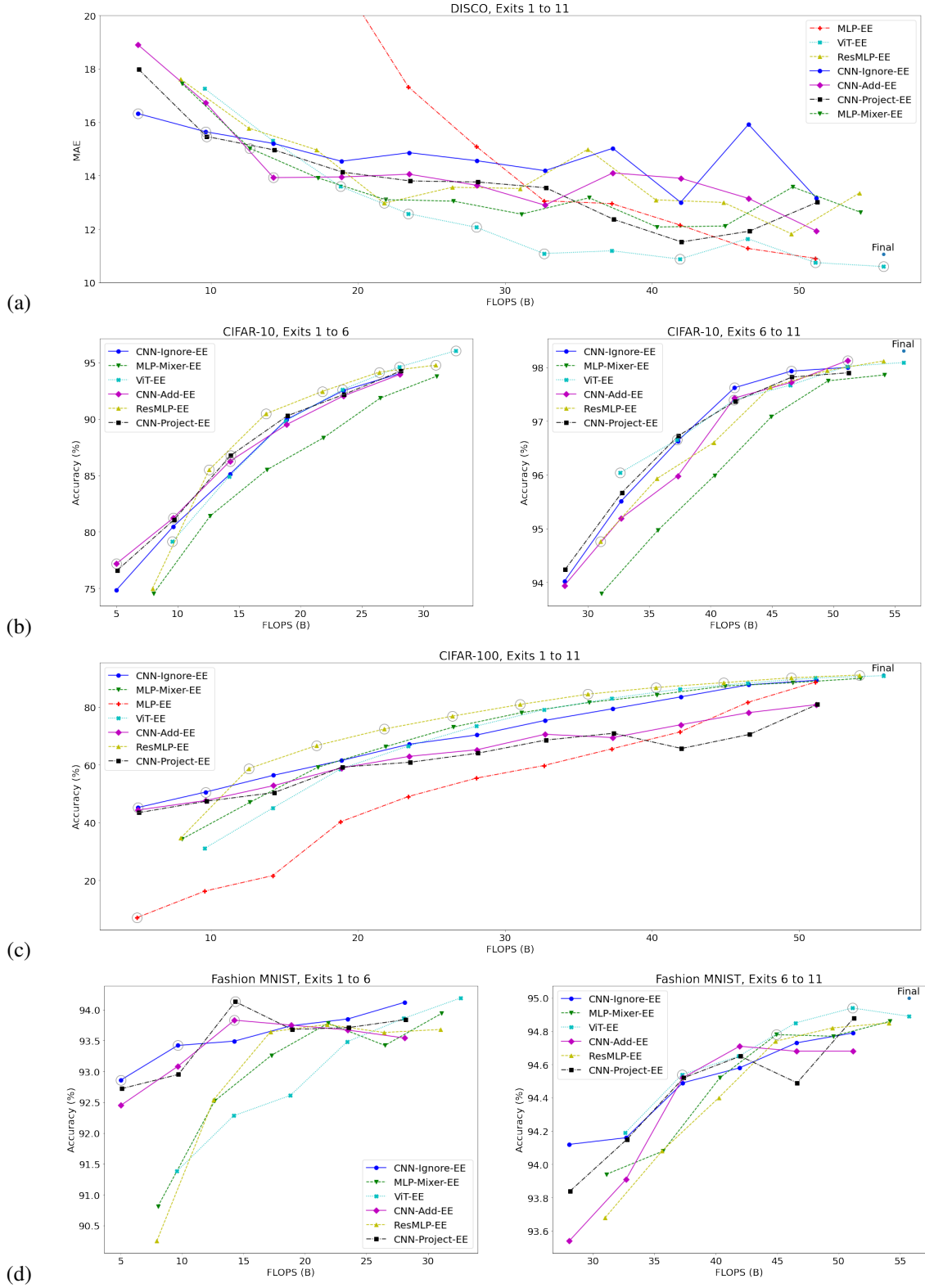
Fig. 5. Performance of different multi-exit architectures on (a) DISCO; (b) CIFAR-10; (c) CIFAR-100 and (d) Fashion MNIST datasets trained with classifier-wise strategy. Practical early exit branches are circled. In order to highlight the differences, early exits with MLP-EE architecture are removed in (b) and (d) and branches 1 to 6 and 6 to 11 are separated.

Branch Arch.	CIFAR-10 Acc.		CIFAR-100 Acc.		Fasion MNIST Acc.		DISCO MAE		FLOPS
	Early@6	Final	Early@6	Final	Early@6	Final	Early@6	Final	
MLP-EE	94.90%	96.73%	81.12%	87.45%	**94.47%**	94.85%	**11.04**	**10.72**	28.04
CNN-Ignore-EE	**95.95%**	97.10%	77.97%	85.90%	94.43%	94.67%	21.88	11.09	28.10
CNN-Add-EE	94.94%	96.87%	75.75%	86.96%	94.22%	94.77%	18.46	11.24	28.10
CNN-Project-EE	94.66%	96.80%	77.95%	86.89%	94.33%	94.69%	18.23	11.29	28.16
ViT-EE	95.89%	96.99%	**85.23%**	**89.44%**	94.39%	94.84%	11.06	11.01	32.65
MLP-Mixer-EE	95.78%	97.07%	81.72%	87.53%	94.41%	94.88%	13.03	10.93	31.11
ResMLP-EE	95.44%	**97.35%**	82.41%	87.57%	94.38%	**94.95%**	16.99	11.36	31.02

TABLE I

PERFORMANCE OF MULTI-EXIT ARCHITECTURES WITH ONE BRANCH, TRAINED WITH END-TO-END STRATEGY. THE LAST COLUMN SHOWS THE FLOPS UP TO AND INCLUDING THE BRANCH.

Branch Arch.	CIFAR-10 Acc.				CIFAR-100 Acc.				DISCO MAE			
	Early@3	Early@6	Early@9	Final	Early@3	Early@6	Early@9	Final	Early@3	Early@6	Early@9	Final
MLP-EE	87.21%	94.48%	95.64%	96.19%	61.00%	79.83%	84.42%	86.46%	13.77	11.54	**11.55**	11.44
CNN-Ignore-EE	91.44%	95.68%	96.55%	96.56%	65.08%	79.35%	84.74%	86.32%	20.99	23.65	20.88	11.42
CNN-Add-EE	90.27%	95.63%	96.80%	96.94%	62.66%	78.86%	85.11%	87.01%	18.33	19.25	18.92	11.77
CNN-Project-EE	91.19%	95.77%	96.81%	96.99%	64.26%	78.63%	84.47%	86.19%	21.25	21.46	17.99	11.49
ViT-EE	92.35%	**96.01%**	**97.25%**	**97.33%**	**74.73%**	**84.31%**	**87.43%**	**87.88%**	12.76	**11.27**	11.59	11.18
MLP-Mixer-EE	91.16%	95.99%	96.96%	96.99%	66.24%	81.84%	86.68%	87.29%	**12.24**	13.95	15.29	11.46
ResMLP-EE	**92.53%**	95.87%	96.76%	96.86%	70.45%	82.61%	87.36%	87.85%	14.15	14.71	17.05	**11.09**

TABLE II

PERFORMANCE OF MULTI-EXIT ARCHITECTURES WITH 3 BRANCHES, TRAINED WITH END-TO-END STRATEGY.

parameters of the backbone, perhaps ViT-EE branches have the least negative impact on the backbone due to the similarity of their architecture with the layers of the backbone. This is further supported by the fact that CNN-based branches whose architectures differ the most from that of the backbone, typically perform much worse.

With the layer-wise strategy, we encounter a problem. Since Vision Transformers are data-hungry [6], they need to be pre-trained on very large datasets. For the first step of the layer-wise strategy where all layers up to and including the first early exit branch are trained, pre-trained weights exist, therefore, the training procedure achieves results with a high accuracy on par with their end-to-end counterpart. For instance, in the case of CIFAR-10, CNN-Ignore-EE achieves 95.97% accuracy at the sixth layer. However, for subsequent steps, the original pre-trained weights can not be used since the weights of earlier layers have changed. We tested the training process with no pre-trained weights, with the original pre-trained weights as well as pre-trained weights from the end-to-end strategy, however, neither achieved a high accuracy.

V. CONCLUSION AND FUTURE DIRECTION

We proposed seven architectures for early exiting Vision Transformer backbones, provided the motivations behind each of these architectures, and showed that depending on the branch locations, training strategy and the problem at hand, any of our proposed architectures can prove useful. Furthermore, we provided recommendations on selecting lightweight high-performing branches based on the results of our experiments. We discussed the role of architectural elements such as local and global interactions, receptive field, classification token, support for multiple types of attention, normalization and similarity of branch architecture with the backbone layers, on the performance of multi-exit ViT architectures.

A potential future research direction would be to check whether other recent architectures for computer vision that operate on a sequence of image patches such as Perceiver [24], gMLP [25] and FNet [26] produce suitable early exits for Vision Transformer backbones.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *arXiv:2102.04906*, 2021.

[3] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, Sep 2020.

[4] A. Sabet, J. Hare, B. Al-Hashimi, and G. V. Merrett, "Temporal early exits for efficient video object detection," *arXiv:2106.11208*, 2021.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[6] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *arXiv:2101.01169*, 2021.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[8] M.-H. Guo, Z.-N. Liu, T.-J. Mu, D. Liang, R. R. Martin, and S.-M. Hu, "Can attention enable mlps to catch up with cnns?" *arXiv:2105.15078*, 2021.

[9] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "Mlp-mixer: An all-mlp architecture for vision," *arXiv:2105.01601*, 2021.

[10] H. Touvron, P. Bojanowski, M. Caron, M. Cord, A. El-Nouby, E. Grave, A. Joulin, G. Synnaeve, J. Verbeek, and H. Jégou, "Resmlp: Feed-forward networks for image classification with data-efficient training," *arXiv:2105.03404*, 2021.

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

[12] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *ICPR*, 2016.

[13] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis, "Single-layer vision transformers for more accurate early exits with less overhead," *arXiv:2105.09121*, 2021.

[14] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *arXiv:2103.13413*, 2021.

[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[16] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017.

[17] G. Gao, J. Gao, Q. Liu, Q. Wang, and Y. Wang, "Cnn-based density estimation and crowd counting: A survey," *arXiv:2003.12783*, 2020.

[18] D. Hu, L. Mou, Q. Wang, J. Gao, Y. Hua, D. Dou, and X. X. Zhu, "Ambient sound helps: Audiovisual crowd counting in extreme conditions," *arXiv:2005.07097*, 2020.

[19] D. Liang, X. Chen, W. Xu, Y. Zhou, and X. Bai, "Transcrowd: Weakly-supervised crowd counting with transformer," *arXiv:2104.09116*, 2021.

[20] F. Dai, H. Liu, Y. Ma, J. Cao, Q. Zhao, and Y. Zhang, "Dense scale network for crowd counting," *arXiv:1906.09707*, 2019.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[22] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis, "Improving the accuracy of early exits in multi-exit architectures via curriculum learning," *arXiv:2104.10461*, 2021.

[23] G. Hacohen and D. Weinshall, "On the power of curriculum learning in training deep networks," in *ICML*, 2019.

[24] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention," *arXiv:2103.03206*, 2021.

[25] H. Liu, Z. Dai, D. R. So, and Q. V. Le, "Pay attention to mlps," *arXiv:2105.08050*, 2021.

[26] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, "Fnet: Mixing tokens with fourier transforms," *arXiv:2105.03824*, 2021.