Investigating Bidimensional Downsampling in Vision Transformer Models

Paolo Bruno, Roberto Amoroso(✉), Marcella Cornia, Silvia Cascianelli, Lorenzo Baraldi, and Rita Cucchiara

University of Modena and Reggio Emilia, Modena, Italy
`225975@studenti.unimore.it`, {`roberto.amoroso,marcella.cornia,silvia.cascianelli,lorenzo.baraldi,rita.cucchiara`}`@unimore.it`

Abstract. Vision Transformers (ViT) and other Transformer-based architectures for image classification have achieved promising performances in the last two years. However, ViT-based models require large datasets, memory, and computational power to obtain state-of-the-art results compared to more traditional architectures. The generic ViT model, indeed, maintains a full-length patch sequence during inference, which is redundant and lacks hierarchical representation. With the goal of increasing the efficiency of Transformer-based models, we explore the application of a 2D max-pooling operator on the outputs of Transformer encoders. We conduct extensive experiments on the CIFAR-100 dataset and the large ImageNet dataset and consider both accuracy and efficiency metrics, with the final goal of reducing the token sequence length without affecting the classification performance. Experimental results show that bidimensional downsampling can outperform previous classification approaches while requiring relatively limited computation resources.

Keywords: Vision Transformer · ViT · Bidimensional downsampling

1 Introduction

Computer vision tasks such as image classification [14,31], object detection [28,29], or semantic segmentation [5,13] have been tackled for years by employing convolutional neural networks (CNNs), which combine the use of a local operator (*i.e.* the convolution) and of strategies for building hierarchical representations through spatial downsampling, which is usually carried out either via pooling layers or by adopting strided convolutions. Recently, the Transformer architecture [35] has received a relevant interest from the natural language processing community [10,27] and is now being adopted to solve computer vision tasks as well [4,11,34,44]. In the case of a Transformer, the architecture features an infinite receptive field, which is achieved through the computation of content-based pairwise similarities and, differently from CNNs, does not feature a hierarchical structure.

https://doi.org/10.1007/978-3-031-06430-2_24

While a Transformer can achieve non-locality and infinite receptive field without requiring a significant increase in the number of parameters, its overall architecture usually features an increased computational cost in terms of multiply-add operations [11,34]. This can be explained by the fact that Transformers maintain a full-length sequence across all layers so that the computational cost of running a single layer does not decrease with the layer depth. While this appears to be an issue from a computational point of view, it also hinders the fact that Transformers lacks a structure that is specifically designed for images.

In this paper, we address both issues and investigate the use of bidimensional pooling in Vision Transformers. Our proposal is in line with recent literature which has tackled either the optimization of the Vision Transformer model [6, 19,33] or the application of one-dimensional pooling [26]. In our approach, the sequence of visual tokens is re-arranged into its original spatial configuration at each architectural block, and bidimensional pooling is then applied to connect different patches together and downsample the sequence length. In this way, we both decrease the computational requirements of the architecture and create a hierarchy in the feature extraction process which resembles that of a CNN.

Experiments are carried out on both a small scale scenario, that of CIFAR-100 [21], and in a larger scale setup, that of ImageNet [30]. By comparing with both a baseline Vision Transformer and with the usage of one-dimensional pooling, we demonstrate the effectiveness of our proposal, both in terms of accuracy and reduction of the number of operations.

2 Related Work

Despite having been initially proposed for machine translation and natural language processing, Transformer-based architectures [10,27,35] have recently demonstrated their effectiveness also in many computer vision tasks [20,32], either combining self-attention with convolutional layers or using pure attention-based solutions. Approaches based on these architectures reached state-of-the-art results in several tasks, including image classification [11,34], object detection [4,46], semantic segmentation [1,2,17,44], image and video captioning [7,8,42], and image generation [18].

Although many improvements of convolutional neural networks for image classification are inspired by Transformers (*e.g.* squeeze and excitation [16], selective kernel [22], and split-attention networks [40]), the first work that transfers a pure Transformer-based architecture to the classification task has been proposed by Dosovitskiy *et al.* [11] with the introduction of the Vision Transformer model. This architecture takes as input a sequence of square image patches and directly applies Transformer layers over raw pixels. While it has achieved promising results on ImageNet [30] bridging the gap with state-of-the-art convolutional neural networks, a pre-training stage on a large amount of data is required to achieve these remarkable performances. To solve this issue and to manage the high computational requirements typical of Transformer-based models, many different solutions have been proposed including the use of low-bit quantization [43], network pruning [25,33], and knowledge distillation [34].

Other solutions, more specific to Transformer models, tackle the quadratic complexity of the self-attention operator. For example, Child *et al.* [6] proposed to factorize the self-attention matrix into smaller and faster attention operators, thus obtaining a $\mathcal{O}(n\sqrt{n})$ complexity. Further, linear complexity can be obtained via a kernel-based formulation of self-attention, as proposed by Katharopoulos *et al.* [19], or by performing the self-attention operator on non-overlapping local windows, as proposed in [23].

On a different line, some works go in the direction of limiting the length of the input sequence to process [12,26,37,38]. For example, the approach proposed by Yuan *et al.* [38] consists in structuring the image into tokens for capturing local patterns. Other strategies consist of downsampling the sequence length, either merging 2D patches, as done in [37], or applying 1D pooling to the intermediate tokens, as done in [26]. This work follows this direction and proposes to apply bidimensional downsampling to visual tokens at the intermediate blocks of the Vision Transformer architecture, which is closer to what is commonly done with 2D max-pooling layers in convolutional neural networks.

3 Proposed Method

We first revisit the Vision Transformer (ViT, in the following) architecture [11] and, in the following sections, introduce our proposal which applies 2D downsampling in Vision Transformers.

3.1 Preliminaries

The ViT model [11] has shown that attention and feed-forward mechanisms can be employed to solve image classification tasks. Given an input image $\mathcal{I} \in \mathbb{R}^{H \times W \times C}$, where H, W and C denote height, width, and the number of channels, this is transformed into a sequence of N square patches $[\boldsymbol{x}_p^1; \boldsymbol{x}_p^2; ..., \boldsymbol{x}_p^N]$, where $\boldsymbol{x}_p^i \in \mathbb{R}^{P^2C}$ is the i-th patch of the input image. Being $P \times P$ the resolution of square patches, H and W usually are multiple of P and the number of patches is $N = (H \cdot W)/P^2$. A linear layer is then applied to each flattened patch to project it to the input dimensionality of the model so that patches can be employed as input tokens for a Transformer encoder.

An additional classification token $\boldsymbol{x}_{\text{class}}$ is usually added to the sequence of patch embeddings. This is implemented as a trainable vector that goes through the Transformer layers and is then projected with a linear layer to predict the final output class. Positional embeddings are then added to the patch embeddings to inject information about the position of patches inside the image. Formally, the model input can be written as:

$$\boldsymbol{z}_0 = [\boldsymbol{x}_{\text{class}}, \boldsymbol{x}_p^1\boldsymbol{E}, \boldsymbol{x}_p^2\boldsymbol{E}, ..., \boldsymbol{x}_p^N\boldsymbol{E}] + \boldsymbol{E}_{\text{pos}}, \tag{1}$$

where $\boldsymbol{E}$ indicates the token embedding matrix and $\boldsymbol{E}_{\text{pos}}$ the positional embedding matrix.

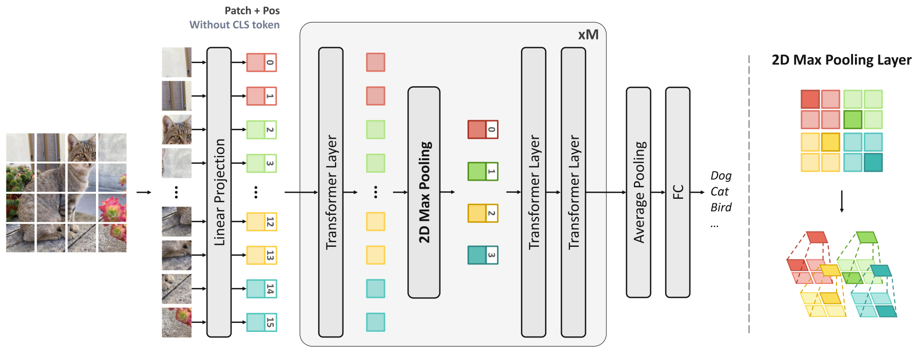

Fig. 1. Overview of the proposed architecture. To reduce redundancy and computational complexity, we progressively shrink the patches sequence length through 2D max-pooling. To this aim, we divide the ViT [11] layers into M stages. Each stage is composed of a Transformer layer, a 2D max-pooling, and a variable number of Transformer layers after the pooling operator. Instead of using the CLS token, the output of the last stage is average pooled and given as input to an FC layer to compute the final prediction. Best seen in color. (Color figure online)

The Transformer encoder [35] implemented in ViT consists of L identical layers, each being composed of a multi-head self-attention layer (MSA) and a multi-layer perceptron (MLP). Layer normalization (LN) is applied before every layer [36] and residual connections are applied after every MSA and MLP. Given the input sequence $\boldsymbol{z}_0$, the classification output of the model can be written as:

$$\boldsymbol{z}'_l = \mathsf{MSA}(\mathsf{LN}(\boldsymbol{z}_{l-1})) + \boldsymbol{z}_{l-1} \qquad l = 1, ..., L \tag{2}$$

$$\boldsymbol{z}_l = \mathsf{MLP}(\mathsf{LN}(\boldsymbol{z}'_l)) + \boldsymbol{z}'_l \qquad l = 1, ..., L \tag{3}$$

$$\boldsymbol{y} = \mathsf{LN}(\boldsymbol{z}^0_L), \tag{4}$$

where $\boldsymbol{z}^0_L$ is the first output of the last encoder layer, corresponding to $\boldsymbol{x}_{\text{class}}$.

Scaled Dot-Product Attention. The attention function performed in MSA layers can be seen as a mapping between queries and a set of key-value pairs with an output. Queries and keys have size d_k, the values dimension is d_v, and all are obtained as linear projections of the input sequence. Given the matrix of queries $\boldsymbol{Q}$, and those of keys and values $\boldsymbol{K}$ and $\boldsymbol{V}$, the output of the attention operator is:

$$\mathsf{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\intercal}{\sqrt{d_k}}\right)\boldsymbol{V}. \tag{5}$$

Multi-head Attention. The above-defined attention function is computed for h different sets of keys, values, and queries, each obtained from separate and learned linear projections. The results after the h parallel operations are concatenated and projected, as follows:

$$\mathsf{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, ...\text{head}_h)\boldsymbol{W}^O. \tag{6}$$

Each head is defined by the following equation:

$$\text{head}_i = \mathsf{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \tag{7}$$

where $\boldsymbol{W}_i^*$ indicates the parameters of each attention head.

3.2 How Pooling Layers Can Help Vision Transformers

The ViT model maintains a fixed-length sequence that passes through all the layers of the network. This choice, although simple, neglects two considerations: (i) each layer contributes differently to the accuracy and efficiency of the model, and (ii) using a sequence with a fixed length can introduce excessive redundancy, with a consequent increase in memory consumption and FLOPs without a corresponding benefit in performance. A multi-level hierarchical representation that would solve both issues is, indeed, missing. CNNs achieve this through intensive use of the pooling layer (or of the stride mechanism) to reduce the spatial dimension of the inputs [14,31] and, at the same time, significantly reduce the computational cost and increase the scalability of the model.

Bidimensional Downsampling. Inspired by Pan *et al.* [26], who investigated the usage of one-dimensional pooling in ViT-like structures, we propose to apply *bidimensional* pooling to shrink the patch embeddings sequence and create a hierarchical representation.

Without loss of generality, a max-pooling operation is considered for all the experiments. Clearly, while a 1D max-pooling can only collapse adjacent tokens, in a 2D configuration the kernel window includes all the neighboring elements with respect to the application point and considers the spatial arrangement of tokens in the input image. Our pooling strategy is thus capable of summarizing intermediate features in a spatially-aware manner. The result is a better localization of relevant features inside the feature map. While the rest of the architecture is left unchanged, we also replace the class token by average pooling the entire sequence after the last encoder layer [26].

To perform a 2D operation over a mono-dimensional sequence of intermediate activations, we firstly re-arrange the sequence of activations in matrix form, thus recovering the original spatial arrangement, and then apply the 2D max-pooling and flatten the result back into a sequence of vectors. The spatial dimensions (H_{out}, W_{out}) obtained after the application of the 2D pooling operation, and before flattening, are thus:

$$H_{out} = \left\lfloor \frac{H_{in} - K}{S} + 1 \right\rfloor \qquad W_{out} = \left\lfloor \frac{W_{in} - K}{S} + 1 \right\rfloor, \tag{8}$$

where K indicates the kernel size and S the stride. We do not apply any padding or dilation. Considering that the pooling operation alters the relative spatial positions of the activations, positional embeddings are re-computed and added after each pooling stage.

3.3 Overall Architecture

To build our architecture, we conceptually divide the encoder layers into M stages, as shown in Fig. 1. Before the first stage, the input is arranged in flattened patches, linearly projected into a sequence of tokens. A learnable positional encoding, initialized as in DeiT [11], is also added to inject information about the positions of the patches. Each stage is composed of a Transformer layer, a max-pooling 2D, and a variable number of Transformer layers. Note that the sequence length is reduced only after the pooling layer. The first Transformer layer input is described by the following equation:

$$\boldsymbol{z}_0 = [\boldsymbol{x}_p^1\boldsymbol{E}, \boldsymbol{x}_p^2\boldsymbol{E}, ..., \boldsymbol{x}_p^N\boldsymbol{E}] + \boldsymbol{E}_{\text{pos}}, \tag{9}$$

where $\boldsymbol{E}_{\text{pos}}$ represents the learnable position embedding. We define the output after a layer b which precedes a max-pooling layer, with also the addition of a new positional encoding, as:

$$\hat{\boldsymbol{z}}_b = \mathsf{MaxPool2D}\,(\boldsymbol{z}_b) + \boldsymbol{E}_b, \tag{10}$$

where $\boldsymbol{z}_b$ is defined as in Eq. 2 and $\mathbf{E}_b$ is the new learnable positional embeddings applied after the application of the 2D max-pooling. An average pooling is applied after the last Transformer layer of the last stage, then a fully-connected layer (FC) is used to make the final predictions. Formally, predictions can be formulated as:

$$\mathbf{y} = \mathsf{FC}(\mathsf{AvgPool}(\mathsf{LN}(\boldsymbol{z}_L))). \tag{11}$$

4 Experiments

To evaluate the effectiveness of our proposal, we perform experiments on two image classification datasets and compare our approach with different variants and baselines. In the following, we first provide implementation and experimental details and then describe our experimental results.

4.1 Experimental Setting

In the following experiments, we consider different configurations of our ViT-based model equipped with 2D-pooling (which we refer to as VT2D) and compare against a ViT model with no pooling and a ViT-based model with 1D-pooling [26]. In the following, we refer to these baselines as VT and VT1D, respectively.

Datasets and Evaluation Protocol. We perform experiments on the CIFAR-100 dataset [21], which contains 100 classes and 60k images, and the ILSVRC-2012 ImageNet dataset [30], which has $1,000$ classes and 1.3M images. All trained models are compared in terms of FLOPs and number of parameters and evaluated in terms of top-1 and top-5 accuracy on the considered datasets.

Table 1. Experimental results on the CIFAR-100 dataset [21]. For each experiment, we indicate the indexes of network stages in which we perform 1D or 2D pooling and the max-pooling kernel size.

	Pooling stages	Kernel size	Params (M)	FLOPs (G)	Top-1 Acc. (%)	Top-5 Acc. (%)
VT-Ti (no pooling)	–	–	5.54	1.25	72.92	92.88
VT1D-Ti-4	0,1,2,3	3	5.58	0.38	72.76	92.67
VT2D-Ti-1	0	3 × 3	5.55	0.31	71.86	92.03
VT2D-Ti-1	1	3 × 3	5.55	0.57	73.39	92.94
VT2D-Ti-1	2	3 × 3	5.55	0.82	73.04	92.78
VT2D-Ti-1	3	3 × 3	5.55	1.08	71.49	92.53
VT2D-Ti-2	0,2	3 × 3	5.55	**0.24**	70.31	91.27
VT2D-Ti-2	0,2	2 × 2	5.55	0.29	70.92	91.82
VT2D-Ti-2	1,3	3 × 3	5.55	0.54	72.25	92.32
VT2D-Ti-2	1,3	2 × 2	5.55	0.58	72.17	92.36
VT2D-Ti-4	0,1,2,3	3 × 3	5.61	0.61	72.87	92.61
VT2D-Ti-4	0,1,2,3	2 × 2	5.65	0.88	**75.31**	**93.47**
VT-S (no pooling)	–	–	21.70	4.58	75.62	93.01
VT1D-S-4	0,1,2,3	3	21.77	1.39	76.09	93.43
VT2D-S-1	0	3 × 3	21.71	1.15	75.31	92.32
VT2D-S-1	1	3 × 3	21.71	2.08	76.59	93.16
VT2D-S-1	2	3 × 3	21.71	3.02	76.18	93.35
VT2D-S-1	3	3 × 3	21.71	3.95	75.13	93.34
VT2D-S-2	0,2	3 × 3	21.71	**0.86**	73.31	91.65
VT2D-S-2	0,2	2 × 2	21.73	1.04	74.44	92.02
VT2D-S-2	1,3	3 × 3	21.71	1.97	76.26	92.91
VT2D-S-2	1,3	2 × 2	21.73	2.13	75.51	93.13
VT2D-S-4	0,1,2,3	3 × 3	21.83	2.28	75.68	92.26
VT2D-S-4	0,1,2,3	2 × 2	21.91	3.26	**77.61**	**93.57**

Implementation Details. Following recent literature on ViT-based models [11, 34], we devise two model configurations varying the model dimensionality d and the number of attention heads h: Tiny ($d = 192$, $h = 3$) and Small ($d = 384$, $h = 6$). Regardless of the model configuration, we always employ 12 layers, divided in $M = 4$ stages with 3 Transformer layers each.

For the experiments on the CIFAR-100 dataset, we use a batch size of 128 and an initial learning rate of $1.25 \cdot 10^{-4}$, while for the ImageNet dataset, the batch size is set to 1024 and the initial learning rate is equal to $5 \cdot 10^{-4}$. The input image resolution is set to 224×224 for both datasets. For training, we use the AdamW optimizer [24], with momentum and weight decay set to 0.9

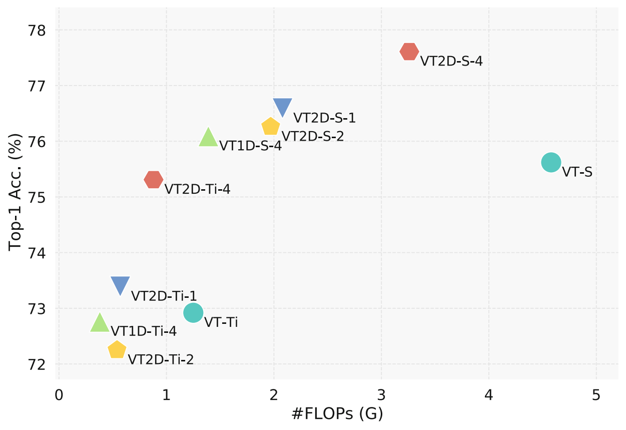

Fig. 2. Performance comparison in terms of top-1 accuracy and FLOPs on the CIFAR-100 dataset [21].

and 0.25 respectively, and train the models for 300 epochs on both the datasets. Note that, during the training phase, we use a cosine scheduler, so that the initial learning rate is reached only after 5 warm-up epochs, and a stochastic depth of 0.1 to facilitate convergence. In all experiments, model weights are initialized with a truncated normal distribution.

To obtain the required amount of data to train Transformer-based models, we follow the same data augmentation strategy used to train the DeiT model [34]. In particular, we apply rand-augment [9] and random erasing [45], together with mixup [41] and cutmix [39]. The magnitude and standard deviation of rand-augment are set to 9 and 0.5, respectively. Random erasing is applied with a probability equal to 0.25. We also employ repeated augmentation [3,15,34]. We run our experiments on 4 RTX 2080 GPUs.

4.2 Experimental Results

Experiments on CIFAR-100. To identify the best strategy to apply the 2D pooling for reducing the model complexity while maintaining competitive performance, we conduct an ablation study in which we vary the kernel size and stride of the pooling layers and the network configuration. The configurations considered differ in terms of the stages in which the pooling layer is applied. In particular, we vary the number of stages, from 1 to 4, and the depth of the stage in which 2D pooling is performed in the model. For all VT2D configurations, we use 2D pooling with stride 2 except when applying the 2D pooling in all four stages of the model, where we use stride equal to 1. As already mentioned, as our baselines, we also consider the VT1D approach, in which 1D pooling with stride 2 is applied at all four stages of the model, and the VT model, which has no pooling layers for downsampling. Results on the CIFAR-100 dataset are reported in Table 1, using Tiny and Small configurations. A noteworthy aspect

Table 2. Experimental results on the ImageNet dataset [30].

	Pooling stages	Kernel size	Params (M)	FLOPs (G)	Top-1 Acc. (%)	Top-5 Acc. (%)
VT1D-Ti-4	0,1,2,3	3	5.75	0.38	67.28	87.70
VT2D-Ti-1	0	3×3	5.72	**0.31**	65.56	86.47
VT2D-Ti-1	1	3×3	5.72	0.57	70.39	**89.84**
VT2D-Ti-4	0,1,2,3	2×2	5.82	0.88	**70.60**	89.83
VT1D-S-4	0,1,2,3	3	22.12	1.40	74.83	92.05
VT2D-S-1	0	3×3	22.06	**1.15**	73.92	91.17
VT2D-S-1	1	3×3	22.06	2.08	**78.19**	**93.87**
VT2D-S-4	0,1,2,3	2×2	22.26	3.26	78.02	93.66

that emerges from the presented experimental results is that applying downsampling at early stages brings the most noticeable saving in terms of computational complexity. Moreover, performing a finer-grained pooling by applying kernels of size 2 compared to 3 benefits the most the performance, at the cost of a slightly higher computational complexity.

Our approach with 2D pooling applied at all four stages obtains better performance compared to both the model without pooling and the 1D pooling version, with a significant reduction of the computational complexity, especially when compared to the VT model without pooling layers. Specifically, it can be noticed that some of the tested configurations with bidimensional pooling in one or two stages perform on par with the VT model in terms of accuracy but require on average 50% fewer FLOPs. The best-performing variant, the VT2D with four stages featuring pooling, brings to an accuracy increase of 2.39% and 1.99% for the Tiny and Small configurations, respectively, while reducing the FLOPs by one third. When comparing with the VT1D version, instead, it can be seen that the best configuration of the VT2D model is always significantly better in terms of accuracy but brings to an increase of the number of FLOPs. However, considering the variants with a single pooling stage, we can notice a computational complexity similar to the 1D pooling version, but with better performance in terms of accuracy, thus further demonstrating the effectiveness of our approach.

In Fig. 2, we show the performance comparison of our approach and the considered baselines in terms of top-1 accuracy and FLOPs, using both Tiny and Small configurations. From the graph, we can notice that our model obtains the best trade-off between overall performance and computational complexity, outperforming the VT and VT1D models in terms of accuracy while keeping FLOPs comparable or even reduced.

Experiments on ImageNet. As a further analysis, we explore the effects of applying 2D pooling in the case of a bigger and more complex dataset than CIFAR-100, and consider the ImageNet dataset. In this analysis, we include a subset of variants previously described, *i.e.* those with best accuracy/FLOPs

Table 3. Comparison with state-of-the-art models on the CIFAR-100 [21] and ImageNet [30] datasets.

	FLOPs (G)	CIFAR-100 Top-1 Acc. (%)	CIFAR-100 Top-5 Acc. (%)	ImageNet Top-1 Acc. (%)	ImageNet Top-5 Acc. (%)
DeiT-Ti [34]	1.25	–	–	72.20	91.10
DeiT-Ti+SCOP [33]	0.80	–	–	68.90	89.00
DeiT-Ti+PoWER [12]	0.80	–	–	69.40	89.20
HVT-Ti-1 [26]	0.64	–	–	69.64	89.40
HVT-Ti-4 [26]	0.38	69.51	91.78	–	–
VT2D-Ti-1	0.57	73.39	92.94	70.39	89.84
VT2D-Ti-4	0.88	75.31	93.47	70.60	89.83
DeiT-S [34]	4.60	–	–	79.80	95.00
DeiT-S+SCOP [33]	2.60	–	–	77.50	93.50
DeiT-S+PoWER [12]	2.70	–	–	78.30	94.00
HVT-S-1 [26]	2.40	74.27	93.07	78.00	93.83
HVT-S-4 [26]	1.39	75.43	93.56	75.23	92.30
VT2D-S-1	2.08	76.59	93.16	78.19	93.87
VT2D-S-4	3.26	77.61	93.57	78.02	93.66

trade-off, and the baseline model with 1D pooling. Again, for all VT2D configurations, we use 2D pooling with stride equal to 2, except for the model variant that applies bidimensional pooling in all four stages of the network. The results of this comparison are reported in Table 2.

Considering the Tiny configuration, our best model is VT2D-Ti-4 which applies a 2D pooling with kernel size 2 in all four stages, followed by VT2D-Ti-1 in which a single bidimensional downsampling is applied in the second stage of the network. Noticeably, VT2D-Ti-1 outperforms the one-dimensional pooling baseline with a slight increase in terms of FLOPs. Similar results can be also observed when turning to the Small configuration. Specifically, the VT2D-S-1 with two-dimensional pooling at the second stage of the network overcomes the VT1D baseline by 3.36% and 1.82% in terms of top-1 and top-5 accuracy, respectively.

Comparison with State-of-the-Art Models. As a final analysis, we report the comparison of our best variants and other state-of-the-art models based on the Vision Transformer architecture that apply different strategies to achieve efficiency (see Table 3). For the competitors, we use the same notation as for our models. In particular, we consider the knowledge distillation-based approach proposed in [34] (DeiT), two variants of DeiT that additionally perform pruning (one following the strategy proposed in [33] - SCOP, the other the strategy proposed in [12] - PoWER), and the monodimensional pooling-based approach proposed in [26] (HVT). From the table, it can be observed that bidimensional pooling is comparable to the considered state-of-the-art approaches both in terms of FLOPs saving and accuracy.

5 Conclusion

In this work, we proposed to apply the concept of bidimensional pooling, which is commonly used in CNNs, to the intermediate patch sequences processed in ViT architectures. As happens in CNNs, this downsampling strategy allows reducing the computational requirement and memory footprint of the model. Moreover, it benefits the performance by enforcing a hierarchical input representation. We evaluated our proposal on two commonly used classification datasets and demonstrated its effectiveness for reducing the computational complexity and increasing the classification accuracy. In fact, with respect to standard ViT, the required FLOPs can be almost halved for equal performance while the classification accuracy can be increased by around 2% with 30% FLOPs reduction in both the Tiny and Small configurations.

Acknowledgment. This work has been partially supported by the project "ROADSTER: Road Sustainable Twins in Emilia Romagna", funded by the International Foundation Big Data and Artificial Intelligence for Human Development.

References

1. Amoroso, R., Baraldi, L., Cucchiara, R.: Assessing the role of boundary-level objectives in indoor semantic segmentation. In: CAIP (2021)
2. Amoroso, R., Baraldi, L., Cucchiara, R.: Improving indoor semantic segmentation with boundary-level objectives. In: IWANN (2021)
3. Berman, M., Jégou, H., Vedaldi, A., Kokkinos, I., Douze, M.: Multigrain: a unified image embedding for classes and instances. arXiv preprint arXiv:1902.05509 (2019)
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
5. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. PAMI **40**(4), 834–848 (2017)
6. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
7. Cornia, M., Baraldi, L., Cucchiara, R.: Explaining transformer-based image captioning models: an empirical analysis. In: AI Communications, pp. 1–19 (2021)
8. Cornia, M., Baraldi, L., Fiameni, G., Cucchiara, R.: Universal captioner: long-tail vision-and-language model training through content-style separation. arXiv preprint arXiv:2111.12727 (2021)
9. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.: RandAugment: practical automated data augmentation with a reduced search space. In: NeurIPS (2020)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL (2018)

11. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. In: ICLR (2021)
12. Goyal, S., Choudhury, A.R., Raje, S., Chakaravarthy, V., Sabharwal, Y., Verma, A.: PoWER-BERT: accelerating BERT inference via progressive word-vector elimination. In: ICML (2020)
13. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
15. Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., Soudry, D.: Augment your batch: improving generalization through instance repetition. In: CVPR (2020)
16. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR (2018)
17. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: CCNet: criss-cross attention for semantic segmentation. In: CVPR (2019)
18. Jiang, Y., Chang, S., Wang, Z.: TransGAN: two pure transformers can make one strong GAN, and that can scale up. In: NeurIPS (2021)
19. Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are RNNs: fast autoregressive transformers with linear attention. In: ICML (2020)
20. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: a survey. ACM Comput. Surv. (2021)
21. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
22. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: CVPR (2019)
23. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV (2021)
24. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
25. Michel, P., Levy, O., Neubig, G.: Are sixteen heads really better than one? In: NeurIPS (2019)
26. Pan, Z., Zhuang, B., Liu, J., He, H., Cai, J.: Scalable vision transformers With hierarchical pooling. In: ICCV (2021)
27. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
29. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NeurIPS (2015)
30. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
32. Stefanini, M., Cornia, M., Baraldi, L., Cascianelli, S., Fiameni, G., Cucchiara, R.: From show to tell: a survey on deep learning-based image captioning. IEEE Trans. PAMI 1–20 (2022)
33. Tang, Y., et al.: Scop: scientific control for reliable neural network pruning. In: NeurIPS (2020)
34. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
35. Vaswani, A., et al.: Attention is all you need. In: NeurIPS (2017)
36. Wang, Q., et al.: Learning deep transformer models for machine translation. In: ACL (2019)
37. Wang, W., et al.: Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. In: ICCV (2021)

38. Yuan, L., et al.: Tokens-to-token ViT: training vision transformers from scratch on imageNet. In: ICCV (2021)
39. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: regularization strategy to train strong classifiers with localizable features. In: ICCV (2019)
40. Zhang, H., et al.: ResNeSt: split-attention networks. arXiv preprint arXiv:2004.08955 (2020)
41. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: ICLR (2018)
42. Zhang, P., et al.: VinVL: revisiting visual representations in vision-language models. In: CVPR (2021)
43. Zhang, W., et al.: TernaryBERT: distillation-aware ultra-low bit BERT. In: EMNLP (2020)
44. Zheng, S., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR (2021)
45. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI (2020)
46. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: deformable transformers for end-to-end object detection. In: ICLR (2021)