Visformer: The Vision-friendly Transformer

Zhengsu Chen[1], Lingxi Xie[2], Jianwei Niu[1,5,6*], Xuefeng Liu[1], Longhui Wei[3], Qi Tian[4]

[1]Beihang University, [2]Johns Hopkins University, [3]University of Science and Technology of China,
[4]Xidian University, [5]Hangzhou Innovation Institute of Beihang University, [6]Zhengzhou University

danczs@buaa.edu.cn, 198808xc@gmail.com, niujianwei@buaa.edu.cn,

liu_xuefeng@buaa.edu.cn, longhuiwei@pku.edu.cn

Abstract

The past year has witnessed the rapid development of applying the Transformer module to vision problems. While some researchers have demonstrated that Transformer-based models enjoy a favorable ability of fitting data, there are still growing number of evidences showing that these models suffer over-fitting especially when the training data is limited. This paper offers an empirical study by performing step-by-step operations to gradually transit a Transformer-based model to a convolution-based model. The results we obtain during the transition process deliver useful messages for improving visual recognition. Based on these observations, we propose a new architecture named Visformer, which is abbreviated from the 'Vision-friendly Transformer'. With the same computational complexity, Visformer outperforms both the Transformer-based and convolution-based models in terms of ImageNet classification accuracy, and the advantage becomes more significant when the model complexity is lower or the training set is smaller. The code is available at `https://github.com/danczs/Visformer`.

1. Introduction

In the past decade, convolution used to play a central role in the deep learning models [22, 29, 31, 15] for visual recognition. This situation starts to change when the Transformer [35], a module that originates from natural language processing [35, 13, 25], is transplanted to the vision scenarios. It was shown in the ViT model [14] that an image can be partitioned into a grid of patches and the Transformer is directly applied upon the grid as if each patch is a visual word. ViT requires a large amount of training data (*e.g.*, the ImageNet-21K [12] or the JFT-300M dataset), arguably because the Transformer is equipped with long-range attention and interaction and thus is prone to over-fitting. The

---

*Corresponding author: Jianwei Niu

Network		ResNet-50	DeiT-S	Visformer-S
FLOPs (G)		4.1	4.6	4.9
Parameters (M)		25.6	21.8	40.2
Full data	base setting	77.43	63.12	77.20
	elite setting	78.73	80.07	82.19
Part of data	10% labels	58.37	40.41	58.74
	10% classes	89.90	80.06	90.06

Table 1. The comparison among ResNet-50, DeiT-S, and the proposed Visformer-S model in ImageNet classification. Although DeiT-S performs well under the elite setting, its performance drops dramatically when the base setting is used or when fewer data are used for training. In comparison, Visformer-S is friendly to both the base and elite settings, and reports smaller accuracy drops using a limited number of training data. Please refer to the main texts for the detailed settings.

follow-up efforts [34] improved ViT to some extent, but these models still perform badly especially under limited training data or moderate data augmentation compared with convolution-based models.

On the other hand, vision Transformers can achieve much better performance than convolution-based models when trained with large amount of data. Namely, vision Transformers have higher 'upper-bound' while convolution-based models are better in 'lower-bound'. Both upper-bound and lower-bound are important properties for neural networks. Upper-bound is the potential to achieve higher performance and lower-bound enables networks to perform better when trained with limited data or scaled to different complexity.

Based on the observation of lower-bound and upper-bound on Transformer-based and convolution-based networks, the main goal of this paper is to identify the reasons behind the difference, by which we can design networks with higher lower-bound and upper-bound. The gap between Transformer-based and convolution-based networks can be revealed with two different training settings on ImageNet. The first one is the base setting. It is the standard set-

ting for convolution-based models, *i.e.*, the training schedule is shorter and the data augmentation only contains basic operators such as random-size cropping [32] and flipping. The performance under this setting is called **base performance** in this paper. The other one is the training setting used in [34]. It is carefully tuned for Transformer-based models, *i.e.*, the training schedule is longer and the data augmentation is stronger (*e.g.*, RandAugment [11], CutMix [41], *etc.*, have been added). We use the **elite performance** to refer to the accuracy produced by it.

We take DeiT-S [34] and ResNet-50 [15] as the examples of Transformer-based and convolution-based models. As shown in Table 1, Deit-S and ResNet-50 employ comparable FLOPs and parameters. However, they behave very differently trained on the full data under these two settings. Deit-S has higher elite performance, but changing the setting from elite to base can cause a $10\%+$ accuracy drop for DeiT-S. ResNet-50 performs much better under the base setting, yet the improvement for the elite setting is merely 1.3%. This motivates us to study the difference between these models. With these two settings, we can roughly estimate the lower-bound and upper-bound of the models. The methodology we use is to perform step-by-step operations to gradually transit one model into another, by which we can identify the properties of modules and designs in these two networks. The entire transition process, taking a total of 8 steps, is illustrated in Figure 1.

Specifically, from DeiT-S to ResNet-50, one should (i) use global average pooling (not the classification token), (ii) introduce step-wise patch embeddings (not large patch flattening), (iii) adopt the stage-wise backbone design, (iv) use batch normalization [20] (not layer normalization [1]), (v) leverage 3×3 convolutions, (vi) discard the position embedding scheme, (vii) replace self-attention with convolution, and finally (viii) adjust the network shape (*e.g.*, depth, width, *etc.*). After a thorough analysis on the reasons behind the results, we absorb all the factors that are helpful to visual recognition and derive the **Visformer**, *i.e.*, the Vision-friendly Transformer.

Evaluated on ImageNet classification, Visformer claims better performance than the competitors, DeiT and ResNet, as shown in Table 1. With the elite setting, the Visformer-S model outperforms DeiT-S and ResNet-50 by 2.12% and 3.46%, respectively, under a comparable model complexity. Different from Deit-S, Visformer-S also survives two extra challenges, namely, when the model is trained with 10% labels (images) and 10% classes. Visformer-S even performs better than ResNet-50, which reveals the high lower-bound of Visformer-S. Additionally, for tiny models, Visformer-Ti significantly outperforms Deit-Ti by more than 6%.

The contribution of this paper is three-fold. **First**, for the first time, we introduce the lower-bound and upper-bound to investigate the performance of Transformer-based vision models. **Second**, we close the gap between the Transformer-based and convolution-based models by a gradual transition process and thus identify the properties of the designs in the Transformer-based and convolution-based models. **Third**, we propose the Visformer as the final model that achieves satisfying lower-bound and upper-bound and enjoys good scalability at the same time.

2. Related work

Image classification is a fundamental task in computer vision. In the deep learning era, the most popular method is to use deep neural networks [21, 29, 15]. One of the fundamental units to build such networks is convolution, where a number of convolutional kernels are used to capture repeatable local patterns in the input image and intermediate data. To reduce the computational costs as well as alleviate the risk of over-fitting, it was believed that the convolutional kernels should be of a small size, *e.g.*, 3×3. However, this brings the difficulty for faraway contexts in the image to communicate with each other – this is partly the reason that the number of layers has been increasing. Despite stacking more and more layers, researchers consider another path which is to use attention-based approaches to ease the propagation of visual information.

Since Transformers achieved remarkable success in natural language processing (NLP) [35, 13, 25], many efforts have been made to introduce Transformers to vision tasks. These works mainly fall into two categories. The first category consists of pure attention models [27, 18, 43, 10, 14, 34, 36]. These models usually only utilize self-attention and attempt to build vision models without convolutions. However, it is computationally expensive to relate all pixels with self-attention for realistic full-sized images. Thus, there has some interest in forcing self-attention to only concentrate on the pixels in local neighborhoods (*e.g.*, SASA [27], LR-Net [18], SANet [43]). These methods replace convolutions with local self-attentions to learn local relations and achieve promising results. However, it requires complex engineering to efficiently apply self-attention to every local region in an image. Another way to solve the complexity problem is to apply self-attention to reduced resolution. These methods either reduce the resolution and color space first [10] or regard image patches rather pixels as tokens (*i.e.*, words) [14, 34]. However, resolution reduction and patch flattening usually make it more difficult to utilize the local prior in natural images. Thus, these methods usually obtain suboptimal results [10] or require huge dataset [14] and heavy augmentation [34].

The second category contains the networks built with not only self-attentions but also convolutions. Self-attention was first introduced to CNNs by non-local neural networks [37]. These networks aim to capture global dependencies in images and videos. Note that non-local neural

networks are inspired by the classical non-local method in vision tasks [5] and unlike those in Transformers, the self-attentions in non-local networks are usually not equipped with multi-heads and position embedding [37, 6, 23]. Afterwards, Transformers achieve remarkable success in NLP tasks [13, 25] and, therefore, self-attentions that inherits NLP settings (*e.g.*, multi-heads, position encodings, classification token, *etc.*) are combined with convolutions to improve vision tasks [27, 3, 2]. A common combination is to utilize convolutions first and apply self-attention afterwards [14, 30]. [14] builds hybrids of self-attention and convolution by adding a ResNet backbone before Transformers. Besides utilizing convolution in early layers, BotNet [30] designs bottleneck cells for self-attention. Additionally, self-attention has been used in many downstream vision tasks (detection [7], segmentation [9]) and low vision tasks [8]. These methods mostly utilize both self-attentions and convolutions.

3. Methodology

3.1. Transformer-based and convolution-based visual recognition models

Recognition is the fundamental task in computer vision. This work mainly considers image classification, where the input image is propagated through a deep network to derive the output class label. Most deep networks are designed in a hierarchical manner and composed of a series of layers.

We consider two popular layers named convolution and Transformer. Convolution originates from the intuition to capture local patterns which are believed more repeatable than global patterns. It uses a number of learnable kernels to compute the responses of the input to different patterns, for which a sliding window is moved along both axes of the input data and the inner-product between the data and kernel is calculated. In this paper, we constrain our study in the scope of residual blocks, a combination of 2 or 3 convolutional layers and a skip-connection. Non-linearities such as activation and normalization are inserted between the neighboring convolutional layers.

On the other hand, Transformer originates from natural language processing and aims to frequently formulate the relationship between *any* two elements (called tokens) even when they are far from each other. This is achieved by generating three features for each token, named the query, key, and value, respectively. Then, the response of each token is calculated as a weighted sum over all the values, where the weights are determined by the similarity between its query and the corresponding keys. This is often referred to as multi-head self-attention (MHSA), followed by other operations including normalization and linear mapping.

Throughout the remaining part, we consider DeiT-S [34] and ResNet-50 [15] as the representative of Transformer-based and convolution-based models, respectively. Besides the basic building block, there are also differences in design, *e.g.*, ResNet-50 has a few down-sampling layers that partition the model into stages, but the number of tokens remains unchanged throughout DeiT-S. The impact of these details will be elaborated in Section 3.3.

3.2. Settings: The base and elite performance

Although DeiT-S reports a 80.1% accuracy which is higher than 78.7% of ResNet-50, we notice that DeiT-S has changed the training strategy significantly, *e.g.*, the number of epochs is enlarged by more than $3\times$ and the data augmentation becomes much stronger. Interestingly, DeiT-S seems to heavily rely on the carefully-tuned training strategy, and other Transformer-based models including ViT [14] and PIT [8] also reported their dependency on other factors, *e.g.*, a large-scale training set. In what follows, we provide a comprehensive study on this phenomenon.

We evaluate all classification models on the ImageNet dataset [28] which has 1K classes, 1.28M training images and 50K testing images. Each class has roughly the same number of training images. This is one of the most popular datasets for visual recognition.

There are two settings to optimize each recognition model. The first one is named the **base setting** which is widely adopted by convolution-based networks. Specifically, the model is trained for 90 epochs with the SGD optimizer. The learning rate starts with 0.2 for batch size 512 and gradually decays to 0.00001 following the cosine annealing function. A moderate data augmentation strategy with random-size cropping [32] and flipping is used. The second one is named the **elite setting** which has been verified effective to improve the Transformer-based models. The Adamw optimizer with an initial learning rate of 0.0005 for batch size 512 is used. The data augmentation and regularization strategy is made much stronger to avoid overfitting, for which intensive operations including RandAugment [11], Mixup [42], CutMix [41], Random Erasing [44], Repeated Augmentation [4, 17] and Stochastic Depth [19] are used. Correspondingly, the training lasts 300 epochs, much longer than that of the base setting.

Throughout the remaining part of this paper, we refer to the classification accuracy under the base and elite settings as **base performance** and **elite performance**, respectively. We expect the numbers to provide complementary views for us to understand the studied models.

3.3. The transition from DeiT-S to ResNet-50

This subsection displays a step-by-step process in which we gradually transit a model from DeiT-S to ResNet-50. There are eight steps in total. The key steps are illustrated in Figure 1, and the results, including the base and elite performance and the model statistics, are summarized in Table 2.
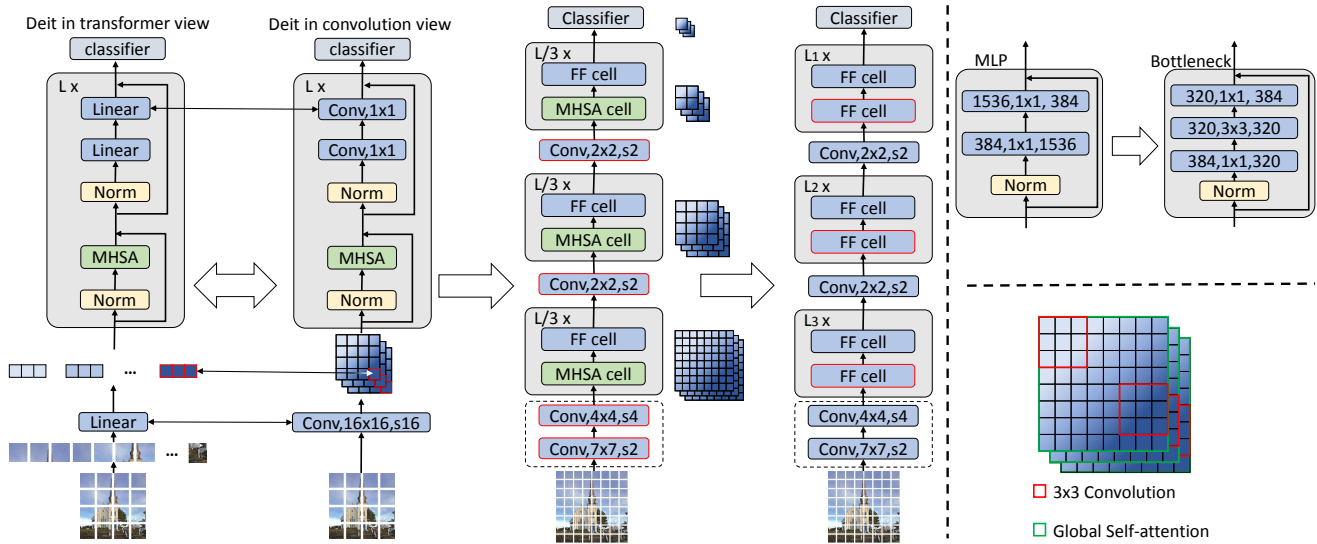
Figure 1. The transition process that starts with DeiT and ends with ResNet-50. To save space, we only show three important movements. The first movement converts DeiT from the Transformer to convolution view (Section 3.3.1). The second movement replaces the patch flattening module with step-wise patch embedding (elaborated in Section 3.3.2) and introduces the stage-wise design (Section 3.3.3) . The third movement replaces the self-attention module with convolution (Section 3.3.7). The upper-right area shows a relatively minor modifications, inserting 3×3 convolution (Section 3.3.5). The lower-right area compares the receptive fields of a 3×3 convolution and self-attention. This figure is best viewed in color.

3.3.1 Using global average pooling to replace the classification token

The first step of the transition is to remove the classification token and add global average pooling to the Transformer-based models. Unlike the convolution-based models, Transformers usually add a classification token to the inputs and utilize the corresponding output token to perform classification, which is inherited from NLP tasks [13]. As a contrast, the classification features in convolution-based models are obtained by conducting global average pooling in the space dimension.

By removing the classification token, the Transformer can be equivalent translated to the convolutional version as shown in Figure 1. Specifically, the patch embedding operation is equivalent to a convolution whose kernel size and stride is the patch size [14]. The shape of the intermediate features can be naturally converted from a sequence of tokens (i.e., words) to a bundle feature maps and the tokens become the vector in channel dimension (illustrated in Figure 1. The linear layers in MHSA and MLP blocks are equivalent to 1×1 convolutions.

The performance of the obtained network (Net1) is shown in Table 2. As can be seen, this transition can substantially improve the base performance. Our further experiments show that adding global pooling itself can improve the base performance from 64.17% to 69.44%. In other words, the global average pooling operation which is widely used in convolution-based models since NIN [24], enables the network to learn more efficiently under moderate augmentation. Furthermore, this transition can slightly improve the elite performance.

3.3.2 Replacing patch flattening with step-wise patch embedding

DeiT and ViT models directly encode the image pixels with a patch embedding layer which is equivalent to a convolution with large kernel size and stride (e.g., 16). This operation flattens the image patches to a sequence of tokens so that Transformers can handle images. However, patch flattening impairs the position information within each patch and makes it more difficult to extract the patterns within patches. To solve this problem, existing methods usually attach a preprocessing module before patch embedding. The preprocessing module can be a feature extraction convnet [14] or a specially designed Transformer [40].

We found that there is a rather simple solution, which is factorizing the large patch embedding to step-wise small patch embeddings. Specifically, We first add the stem layer in ResNet to the Transformer, which is a 7×7 convolution layer with a stride of two. The stem layer can be seen as a 2×2 patching embedding operation with pixel overlap (i.e., 7×7 kernel size). Since the patch size in the original DeiT model is 16, we still need to embed 8×8 patches after the stem. We further factorize the 8×8 patch embedding

Model Name	added	removed	base perf.	elite perf.	FLOPs (G)	Params (M)
DeiT-S	–		64.17	80.07	4.60	22.1
Net1	global average pooling	classification token	69.81 (+5.64)	80.16 (+0.09)	4.57	22.0
Net2	step-wise embeddings	large patch embedding	73.01 (+3.20)	81.35 (+1.19)	4.77	23.9
Net3	stages-wise design	–	75.76 (+2.75)	80.19 (-1.14)	4.79	39.5
Net4	batch norm	layer norm	76.49 (+0.73)	80.97 (+0.78)	4.79	39.5
Net5	3×3 convolution	–	77.37 (+0.88)	80.15 (-0.82)	4.76	39.2
Net6	–	position embedding	77.31 (-0.06)	79.86 (-0.29)	4.76	39.0
Net7	convolution	self-attention	76.24 (-1.07)	79.01 (-0.85)	4.83	45.0
ResNet-50	network shape adjustment		77.43 (+1.19)	78.73 (-0.28)	4.09	25.6

Table 2. The classification accuracy on ImageNet during the transition procedure from DeiT-S to ResNet-50. Both the base setting and the elite setting are considered (for the details, see Section 3.2), and we mark the positive modifications in red and the negative modifications in blue. Note that a modification can impact the base and elite performance differently. Though the number of parameters increases considerably at the intermediate status, the computational costs measured by FLOPs does not change significantly.

to a 4×4 embedding and a 2×2 embedding, which are 4×4 and 2×2 convolution layers with stride 4 and 2 in the perspective of convolution. Additionally, we add an extra 2×2 convolution to further upgrade the patch size from 16×16 to 32×32 before classification. These patch embedding layers can also be seen as the down-sampling layers and we double the channel numbers after embedding following the practice in convolution-based models.

By utilizing step-wise embeddings, the position prior within patches is encoded into features. As a result, the model can learn patterns more efficiently. As can be seen in Table 2, this transition can significantly improve the base performance and elite performance of the network. It indicates that step-wise embedding is a better choice than larger patch embedding in Transformer-based models. Additionally, this transition is computationally efficient and only introduces about 4% extra FLOPs.

3.3.3 Stage-wise design

In this section, we split networks into stages like ResNets. The blocks in the same stage share the same feature resolution. Since step-wise embeddings in the last transition have split the network into different stages, the transition in this section is to reassign the blocks to different stages as shown in Figure 1. However, unlike convolution blocks, the complexity of self-attention blocks increases by $O(N^4)$ with respect to the feature size. Thus we only insert blocks to the 8×8, 16×16 and 32×32 patch embedding stages, which correspond to 28×28, 14×14 and 7×7 feature resolutions respectively for 224×224 inputs. Additionally, we halve the head dimension and feature dimension before self-attention in 28×28 stage to ensure that the blocks in different stages utilize similar FLOPs.

This transition leads to interesting results. The base performance is further improved. It is conjectured that the stage-wise design leverages the image local priors and thus

can perform better under moderate augmentation. However, the elite performance of the network decreases markedly. To study reasons, we conduct ablation experiments and find that self-attention does not work well in very large resolutions. We conjecture that large resolution contains too many tokens and it is much more difficult for self-attention to learn relations among them. We will detail it in section 3.4.

3.3.4 Replacing LayerNorm with BatchNorm

Transformer-based models usually normalize the features with LayerNorm [1], which is inherited from NLP tasks [35, 13]. As a contrast, convolution-based models like ResNets usually utilize BatchNorm [20] to stabilize the training process. LayerNorm is independent of batch size and more friendly for specific tasks compared with BatchNorm, while BatchNorm usually can achieve better performance given appropriate batch size [38]. We replace all the LayerNorm layers with BatchNorm layers and the results show that BatchNorm performs better than LayerNorm. It can improve both the base performance and elite performance of the network.

In addition, we also try to add BatchNorm to Net2 to further improve the elite performance. However, this Net2-BN network suffers from convergence problems. This may explain why BatchNorm is not widely used in the pure self-attention models. But for our mixed model, BatchNorm is a reliable method to advance performance.

3.3.5 Introducing 3×3 convolutions

Since the tokens of the network are present as feature maps, it is natural to introduce convolutions with kernel sizes larger than 1×1. The specific meaning of large kernel convolution is illustrated at the bottom right of Figure 1. When global self-attentions attempt to build the relations among all the tokens (*i.e.*, pixels), convolutions focus on relating

the tokens within local neighborhoods. We chose to insert 3×3 convolutions between the 1×1 convolutions in feed-forward blocks, which transforms the MLP blocks into bottleneck blocks as exhibited at the top right of Figure 1. Note that the channel numbers of the 3×3 convolution layers are tuned to ensure that the FLOPs of the feed-forward blocks are nearly unchanged. The obtained bottleneck blocks are similar to the bottleneck blocks in ResNet-50, although they have different bottleneck ratios (*i.e.*, the factor of reducing the channel numbers before the 3×3 convolution). We replace the MLP blocks with bottleneck blocks in all three stages.

Not surprisingly, 3×3 convolutions which can leverage the local priors in images further improve the network base performance. The base performance (77.37%) becomes comparable with ResNet-50 (77.43%). However, the elite performance decreases by 0.82%. We conduct more experiments to study the reasons. Instead of adding 3×3 convolutions to all stages, we insert 3×3 convolutions to different stages separately. We observe that 3×3 convolutions only work well on the high-resolution features. We conjecture that leveraging local relations is important for the high-resolution features in natural images. For the low-resolution features, however, local convolutions become unimportant when equipped with global self-attention. We will detail it in section 3.4.

3.3.6 Removing position embedding

In Transformer-based models, position embedding is proposed to encode the position information inter tokens. In the transition network, we utilize learnable position embedding as in [13] and add them to features after patch embeddings. To approaching ResNet-50, position embedding should be removed.

The results are exhibited in Table 2. The base performance is almost unchanged and the elite performance declines slightly (0.29%). As a comparison, We test to remove the position embedding of DeiT-S and elite performance decreases significantly by 3.95%. It reveals that position embedding is less important in the transition model than that in the pure Transformer-based models. It is because that the position prior inter tokens is preserved by the feature maps and convolutions with spatial kernels can encode and leverage it. Consequently, the harm of removing position embedding is remarkably reduced in the transition network. It also explains why convolution-based models do not need position embedding.

3.3.7 Replacing self-attention with feed-forward

In this section, we remove the self-attention blocks in each stage and utilize a feed-forward layer instead, so that the network becomes a pure convolution-based network. To keep the FLOPs unchanged, several bottleneck blocks are added to each stage. After the replacement, the obtained network consists of bottleneck blocks like ResNet-50.

The performance of the obtained network (Net7) is shown in Table 2. The pure convolution-based network performs much worse both in base performance and elite performance. *It indicates that self-attentions do drive neural networks to higher elite performance and is not responsible for the poor base performance in ViT or DeiT. It is possible to design a self-attention network with high base performance and elite performance.*

3.3.8 Adjusting the shape of network

There are still many differences between Net7 and ResNet-50. First, the shape of Net7 is different from ResNet-50. Their depths, widths, bottleneck ratios and block numbers in network stages are different. Second, they normalize the features in different positions. Net7 only normalizes input features in a block, while ResNet-50 normalizes features after each convolutional layer. Third, ResNet-50 down-samples the features with bottleneck blocks but Net7 utilizes a single convolution layer (*i.e.*, patch embedding layer). In addition, Net7 employs a few more FLOPs. Nevertheless, both these two networks are convolution-based networks. The performance gap between these two networks can be attributed to architecture design strategy.

As shown in Table 2, the base performance is improved after transition. It demonstrates that ResNet-50 has better network architecture and can perform better with fewer FLOPs. However, ResNet-50 obtains worse elite performance. It indicates that the inconsistencies between base performance and elite performance exist not only in self-attention models but also in pure convolution-based networks.

3.4. Summary: the Visformer model

We aim to build a network with high base performance and elite performance. The transition study has shown that there are some inconsistencies between base performance and elite performance. The first problem is the stage-wise design, which increases the base performance but decreases the elite performance. To study the reasons, we replace the self-attention blocks with bottleneck blocks in each stage separately for Net5, by which we can estimate the importance of self-attention in different stages. The results are shown in Table 3. The replacement of self-attention in all three stages reduces both the base performance and the elite performance. There is a trend that self-attentions in lower resolutions play more important roles than those in higher resolutions. Additionally, replacing the self-attentions in the first stage almost has no effect on the network performance. Larger resolutions contain much more tokens and

Network	base perf.(%)	elite perf.(%)
Net5	77.37	80.15
Net5-DS1	77.29 (-0.08)	80.13 (-0.02)
Net5-DS2	77.34 (-0.02)	79.75 (-0.40)
Net5-DS3	77.05 (-0.32)	79.59 (-0.56)

Table 3. Impact of replacing the self-attention blocks with the bottleneck blocks in each stage of Net5. These experiments are performed individually.

Network	base perf.(%)	elite perf.(%)
Net4	76.49	80.97
Net4-S1	77.02 (+0.53)	81.10 (+0.13)
Net4-S2	76.55 (+0.06)	80.50 (-0.47)
Net4-S3	76.82 (+0.33)	80.44 (-0.53)
Net5	77.37 (+0.88)	80.15 (-0.82)

Table 4. Impact of replacing the MLP layers with the bottleneck blocks in each stage of Net4. These experiments are performed individually.

we conjecture that it is more difficult for self-attentions to learn relations among them.

The second problem is adding 3×3 convolutions to the feed-forward blocks, which decreases the elite performance by 0.82%. Based on Net4, we replace MLP blocks with bottleneck blocks in each stage separately. As can be seen in Table 4, although all stages obtain improvements in base performance, only the first stage benefits from bottleneck blocks in elite performance. The 3×3 convolutions are not necessary for the other two low-resolution stages when self-attentions already have a global view in these positions. On the high-resolution stage, for which self-attentions have difficulty in handling all tokens, the 3×3 convolutions can provide improvement.

Integrating the observation above, we propose the **Visformer** as vision-friendly, Transformer-based models. The detailed architectures are shown in Table 5. Besides the positive transitions, Visformer adopts the stage-wise design for higher base performance. But self-attentions are only utilized in the last two stages, considered that self-attention in the high-resolution stage is relatively inefficient even when the FLOPs are balanced. Visformer employs bottleneck blocks in the first stage and utilizes group 3×3 convolutions in bottleneck blocks inspired by ResNeXt [39]. We also introduce BatchNorm to patch embedding modules as in CNNs. We name Visformer-S to denote the model that directly comes from DeiT-S. In addition, we can adjust the complexity by changing the output dimensionality of multi-head attentions. Here, we shrink the dimensionality by half and derive the Visformer-Ti model, which requires around $1/4$ computational costs of the Visformer-S model.

	output size	Visformer-Ti	Visformer-S
stem	112×112	7×7, 16, stride 2	7×7, 32, stride 2
emb.	28×28	4×4, 96, stride 4	4×4, 192, stride 4
s1	28×28	$\begin{bmatrix} 1 \times 1, 192 \\ 3 \times 3, 384 \\ (\text{group} = 8) \\ 1 \times 1, 96 \end{bmatrix} \times 7$	$\begin{bmatrix} 1 \times 1, 384 \\ 3 \times 3, 768 \\ (\text{group} = 8) \\ 1 \times 1, 192 \end{bmatrix} \times 7$
emb.	14×14	2×2, 192, stride 2	2×2, 384, stride 2
s2	14×14	$\begin{bmatrix} \text{MHSA, 192} \\ 1 \times 1, 768 \\ 1 \times 1, 192 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{MHSA, 384} \\ 1 \times 1, 1536 \\ 1 \times 1, 384 \end{bmatrix} \times 4$
emb.	7×7	2×2, 384, stride 2	2×2, 768, stride 2
s3	7×7	$\begin{bmatrix} \text{MHSA, 384} \\ 1 \times 1, 1536 \\ 1 \times 1, 384 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{MHSA, 768} \\ 1 \times 1, 3072 \\ 1 \times 1, 768 \end{bmatrix} \times 4$
	1×1	global average pool, 1000-d fc, softmax	
FLOPs		1.3×10^9	4.9×10^9

Table 5. The configuration for constructing the Visformer-Ti and Visformer-S models, where 'emb.' stands for feature embedding, and 's1'–'s3' indicate the three stages with different spatial resolutions.

4. Evaluating the Visformer

4.1. Comparison to the state-of-the-arts

We first compare Visformer against DeiT, the direct baseline. Results are summarized in Table 6. Using comparable computational costs, the Visformer models outperform the corresponding DeiT models significantly. Specifically, the advantages of Visformer-S and Visformer-Ti over DeiT-S and DeiT-Ti under the elite setting are 2.12% and 6.41%, while under the base setting, the numbers grow to 14.08% and 10.47%, respectively. In other words, the advantage becomes more significant under the base setting, which is more frequently used for visual recognition.

We then compare Visformer to other Transformer-based approaches in Table 7. At the tiny level, Visformer-Ti significantly outperforms other vision Transformer models. For larger models, Visformer-S performs much better than the models with similar FLOPs. Other models usually need to utilize much more FLOPs to achieve comparable performance. As for the state-of-the-art EfficientNet convnets, our models are below the EfficientNets with similar FLOPs. However, EfficientNets is computing inefficient on GPUs. The results in Table 8 show that our model is significantly faster than EfficientNet-b3 which performance is slightly worse than our model. Our model is as efficient as DeiT-S and ResNet-50 but with a rather better performance.

Network	base perf. (%)	elite perf. (%)	FLOPs (G)
Visformer-Ti	74.34	78.62	1.3
DeiT-Ti	63.87	72.21	1.3
Visformer-S	77.20	82.19	4.9
DeiT-S	63.12	80.07	4.6

Table 6. The comparison of base and elite performance as well as the FLOPs between Visformer and DeiT, the direct baseline.

Methods	Top-1(%)	FLOPs (G)	Params (M)
ResNet-18 [15]	69.8	1.8	11.7
DeiT-Ti [34]	72.2	1.3	5.7
DeiT-Ti (KD) [34]	74.6	1.3	5.7
PVT-Ti [36]	75.1	1.9	13.2
Visformer-Ti (ours)	78.6	1.3	10.3
ResNet-50 [15]	76.2	4.1	25.6
ResNet-50* [15]	78.7	4.1	25.6
RegNetY-4GF [26]	79.4	4.0	20.6
RegNetY-8GF [26]	79.9	8.0	39.2
RegNetY-4GF* [26]	80.0	4.0	20.6
DeiT-S [34]	79.8	4.6	21.8
DeiT-S* [34]	80.1	4.6	21.8
DeiT-B [34]	81.8	17.4	86.3
PVT-S [36]	79.8	3.8	24.5
PVT-Medium [36]	81.2	6.7	44.2
T2T-ViT$_t$-14 [40]	80.7	5.2	21.5
T2T-ViT$_t$-19 [40]	81.4	8.4	39.0
BoTNet-S1-59 [30]	81.7	7.3	33.5
Visformer-S (ours)	82.2	4.9	40.2

Table 7. Comparison among our method and other Transformer-based vision models. '*' indicates that we re-run the model using the elite setting. 'KD' stands for knowledge distillation [16].

Methods	Top-1 (%)	FLOPs (G)	Batch Time (ms)
ResNet-50*	78.7	4.1	34.2
DeiT-S*	80.1	4.6	36.9
RegNetY-4GF*	80.0	4.0	40.2
EfficientNet-B3 [33]	81.6	1.8	48.3
Visformer-S (ours)	82.2	4.9	36.7

Table 8. Comparison of inference efficiency among Visformer-S and other models. A batch size of 32 is used for testing. Besides EfficientNet-B3, other models are trained using the elite setting.

4.2. Training with limited data

Last but not least, we evaluate the performance of Visformer in the scenario with limited training data, which we consider is an important ability of being vision-friendly,

Network	100% classes	10% classes	1% classes	10% images	1% images
Visformer-S	82.19	90.06	91.60	58.74	16.56
DeiT-S	80.07	80.06	73.40	40.41	6.94
ResNet-50	78.73	89.90	93.20	58.37	13.59
Visformer-Ti	78.62	89.48	90.60	55.14	11.79
Deit-Ti	72.33	78.72	74.40	38.44	6.53
ResNet-50-55%	72.84	87.10	91.40	51.48	10.68

Table 9. Comparison among Visformer, DeiT, and ResNet, in terms of classification accuracy (%) using limited training data. The elite setting with 300 epochs is used for all models.

while prior Transformer-based models mostly required abundant training data [14].

Four subsets of ImageNet are used, with 10% and 1% randomly chosen classes (all data), and with 10% and 1% randomly chosen images (all classes), respectively. To challenge the models, we still use the elite setting with 300 epochs (not extended). As shown in Table 9, it is observed that the DeiT-S model reports dramatic accuracy drops in all the four tests (note that the accuracy of using only 10% and 1% classes should be much higher if epochs are extended). In comparison, Visformer remains robust in these scenarios, showing its potential of being used for visual recognition with limited data.

In tiny level, ResNet-50-55% is obtained by reducing the channel numbers (like other tiny models) to 55% (so that the FLOPs, 1.3G, is similar to Visformer-Ti and Deit-Ti). The conclusion is similar: Visformer-Ti is still the best overall model, and the advantage is slightly enlarged because the risk of over-fitting has been reduced.

5. Conclusions

This paper presents Visformer, a Transformer-based model that is friendly to visual recognition. We propose to use two protocols, the base and elite setting, to evaluate the performance of each model. To study the reason why Transformer-based models and convolution-based models behave differently, we decompose the gap between these models and design an eight-step transition procedure that bridges the gap between DeiT-S and ResNet-50. By absorbing the advantages and discarding the disadvantages, we obtain the Visformer-S model that outperforms both DeiT-S and ResNet-50. Visformer also shows a promising ability when it is transferred to a compact model and when it is evaluated on small datasets.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021.

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

[4] Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

[5] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[6] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[8] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020.

[9] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[10] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020.

[11] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[18] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.

[19] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[23] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10297–10306, 2020.

[24] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[25] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[26] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[27] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Standalone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[33] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[34] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[36] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021.

[37] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[38] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[39] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[40] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[41] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[42] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. 2017.

[43] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[44] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.