AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE

Alexey Dosovitskiy*,†, Lucas Beyer*, Alexander Kolesnikov*, Dirk Weissenborn*, Xiaohua Zhai*, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby*,†

*equal technical contribution, †equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

ABSTRACT

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

1 INTRODUCTION

Self-attention-based architectures, in particular Transformers (Vaswani et al., 2017), have become the model of choice in natural language processing (NLP). The dominant approach is to pre-train on a large text corpus and then fine-tune on a smaller task-specific dataset (Devlin et al., 2019). Thanks to Transformers' computational efficiency and scalability, it has become possible to train models of unprecedented size, with over 100B parameters (Brown et al., 2020; Lepikhin et al., 2020). With the models and datasets growing, there is still no sign of saturating performance.

In computer vision, however, convolutional architectures remain dominant (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2016). Inspired by NLP successes, multiple works try combining CNN-like architectures with self-attention (Wang et al., 2018; Carion et al., 2020), some replacing the convolutions entirely (Ramachandran et al., 2019; Wang et al., 2020a). The latter models, while theoretically efficient, have not yet been scaled effectively on modern hardware accelerators due to the use of specialized attention patterns. Therefore, in large-scale image recognition, classic ResNet-like architectures are still state of the art (Mahajan et al., 2018; Xie et al., 2020; Kolesnikov et al., 2020).

Inspired by the Transformer scaling successes in NLP, we experiment with applying a standard Transformer directly to images, with the fewest possible modifications. To do so, we split an image into patches and provide the sequence of linear embeddings of these patches as an input to a Transformer. Image patches are treated the same way as tokens (words) in an NLP application. We train the model on image classification in supervised fashion.

When trained on mid-sized datasets such as ImageNet without strong regularization, these models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome may be expected: Transformers lack some of the inductive biases

[1]Fine-tuning code and pre-trained models are available at https://github.com/google-research/vision_transformer

inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.

However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias. Our Vision Transformer (ViT) attains excellent results when pre-trained at sufficient scale and transferred to tasks with fewer datapoints. When pre-trained on the public ImageNet-21k dataset or the in-house JFT-300M dataset, ViT approaches or beats state of the art on multiple image recognition benchmarks. In particular, the best model reaches the accuracy of 88.55% on ImageNet, 90.72% on ImageNet-ReaL, 94.55% on CIFAR-100, and 77.63% on the VTAB suite of 19 tasks.

2 Related Work

Transformers were proposed by Vaswani et al. (2017) for machine translation, and have since become the state of the art method in many NLP tasks. Large Transformer-based models are often pre-trained on large corpora and then fine-tuned for the task at hand: BERT (Devlin et al., 2019) uses a denoising self-supervised pre-training task, while the GPT line of work uses language modeling as its pre-training task (Radford et al., 2018; 2019; Brown et al., 2020).

Naive application of self-attention to images would require that each pixel attends to every other pixel. With quadratic cost in the number of pixels, this does not scale to realistic input sizes. Thus, to apply Transformers in the context of image processing, several approximations have been tried in the past. Parmar et al. (2018) applied the self-attention only in local neighborhoods for each query pixel instead of globally. Such local multi-head dot-product self attention blocks can completely replace convolutions (Hu et al., 2019; Ramachandran et al., 2019; Zhao et al., 2020). In a different line of work, Sparse Transformers (Child et al., 2019) employ scalable approximations to global self-attention in order to be applicable to images. An alternative way to scale attention is to apply it in blocks of varying sizes (Weissenborn et al., 2019), in the extreme case only along individual axes (Ho et al., 2019; Wang et al., 2020a). Many of these specialized attention architectures demonstrate promising results on computer vision tasks, but require complex engineering to be implemented efficiently on hardware accelerators.

Most related to ours is the model of Cordonnier et al. (2020), which extracts patches of size 2×2 from the input image and applies full self-attention on top. This model is very similar to ViT, but our work goes further to demonstrate that large scale pre-training makes vanilla transformers competitive with (or even better than) state-of-the-art CNNs. Moreover, Cordonnier et al. (2020) use a small patch size of 2×2 pixels, which makes the model applicable only to small-resolution images, while we handle medium-resolution images as well.

There has also been a lot of interest in combining convolutional neural networks (CNNs) with forms of self-attention, e.g. by augmenting feature maps for image classification (Bello et al., 2019) or by further processing the output of a CNN using self-attention, e.g. for object detection (Hu et al., 2018; Carion et al., 2020), video processing (Wang et al., 2018; Sun et al., 2019), image classification (Wu et al., 2020), unsupervised object discovery (Locatello et al., 2020), or unified text-vision tasks (Chen et al., 2020c; Lu et al., 2019; Li et al., 2019).

Another recent related model is image GPT (iGPT) (Chen et al., 2020a), which applies Transformers to image pixels after reducing image resolution and color space. The model is trained in an unsupervised fashion as a generative model, and the resulting representation can then be fine-tuned or probed linearly for classification performance, achieving a maximal accuracy of 72% on ImageNet.

Our work adds to the increasing collection of papers that explore image recognition at larger scales than the standard ImageNet dataset. The use of additional data sources allows to achieve state-of-the-art results on standard benchmarks (Mahajan et al., 2018; Touvron et al., 2019; Xie et al., 2020). Moreover, Sun et al. (2017) study how CNN performance scales with dataset size, and Kolesnikov et al. (2020); Djolonga et al. (2020) perform an empirical exploration of CNN transfer learning from large scale datasets such as ImageNet-21k and JFT-300M. We focus on these two latter datasets as well, but train Transformers instead of ResNet-based models used in prior works.

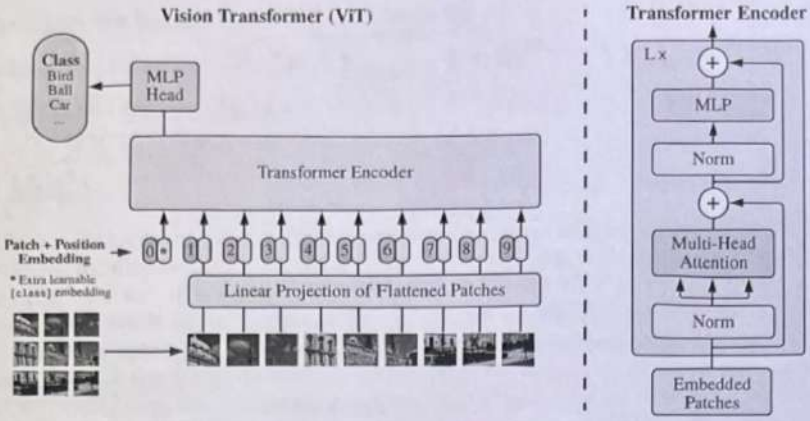

Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

3 METHOD

In model design we follow the original Transformer (Vaswani et al., 2017) as closely as possible. An advantage of this intentionally simple setup is that scalable NLP Transformer architectures – and their efficient implementations – can be used almost out of the box.

3.1 VISION TRANSFORMER (VIT)

An overview of the model is depicted in Figure 1. The standard Transformer receives as input a 1D sequence of token embeddings. To handle 2D images, we reshape the image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the original image, C is the number of channels, (P, P) is the resolution of each image patch, and $N = HW/P^2$ is the resulting number of patches, which also serves as the effective input sequence length for the Transformer. The Transformer uses constant latent vector size D through all of its layers, so we flatten the patches and map to D dimensions with a trainable linear projection (Eq. 1). We refer to the output of this projection as the patch embeddings.

Similar to BERT's `[class]` token, we prepend a learnable embedding to the sequence of embedded patches ($\mathbf{z}_0^0 = \mathbf{x}_{\text{class}}$), whose state at the output of the Transformer encoder ($\mathbf{z}_L^0$) serves as the image representation $\mathbf{y}$ (Eq. 4). Both during pre-training and fine-tuning, a classification head is attached to $\mathbf{z}_L^0$. The classification head is implemented by a MLP with one hidden layer at pre-training time and by a single linear layer at fine-tuning time.

Position embeddings are added to the patch embeddings to retain positional information. We use standard learnable 1D position embeddings, since we have not observed significant performance gains from using more advanced 2D-aware position embeddings (Appendix D.4). The resulting sequence of embedding vectors serves as input to the encoder.

The Transformer encoder (Vaswani et al., 2017) consists of alternating layers of multiheaded self-attention (MSA, see Appendix A) and MLP blocks (Eq. 2, 3). Layernorm (LN) is applied before every block, and residual connections after every block (Wang et al., 2019; Baevski & Auli, 2019).

The MLP contains two layers with a GELU non-linearity.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Inductive bias. We note that Vision Transformer has much less image-specific inductive bias than CNNs. In CNNs, locality, two-dimensional neighborhood structure, and translation equivariance are baked into each layer throughout the whole model. In ViT, only MLP layers are local and translationally equivariant, while the self-attention layers are global. The two-dimensional neighborhood structure is used very sparingly: in the beginning of the model by cutting the image into patches and at fine-tuning time for adjusting the position embeddings for images of different resolution (as described below). Other than that, the position embeddings at initialization time carry no information about the 2D positions of the patches and all spatial relations between the patches have to be learned from scratch.

Hybrid Architecture. As an alternative to raw image patches, the input sequence can be formed from feature maps of a CNN (LeCun et al., 1989). In this hybrid model, the patch embedding projection $\mathbf{E}$ (Eq. 1) is applied to patches extracted from a CNN feature map. As a special case, the patches can have spatial size 1x1, which means that the input sequence is obtained by simply flattening the spatial dimensions of the feature map and projecting to the Transformer dimension. The classification input embedding and position embeddings are added as described above.

3.2 Fine-tuning and Higher Resolution

Typically, we pre-train ViT on large datasets, and fine-tune to (smaller) downstream tasks. For this, we remove the pre-trained prediction head and attach a zero-initialized $D \times K$ feedforward layer, where K is the number of downstream classes. It is often beneficial to fine-tune at higher resolution than pre-training (Touvron et al., 2019; Kolesnikov et al., 2020). When feeding images of higher resolution, we keep the patch size the same, which results in a larger effective sequence length. The Vision Transformer can handle arbitrary sequence lengths (up to memory constraints), however, the pre-trained position embeddings may no longer be meaningful. We therefore perform 2D interpolation of the pre-trained position embeddings, according to their location in the original image. Note that this resolution adjustment and patch extraction are the only points at which an inductive bias about the 2D structure of the images is manually injected into the Vision Transformer.

4 Experiments

We evaluate the representation learning capabilities of ResNet, Vision Transformer (ViT), and the hybrid. To understand the data requirements of each model, we pre-train on datasets of varying size and evaluate many benchmark tasks. When considering the computational cost of pre-training the model, ViT performs very favourably, attaining state of the art on most recognition benchmarks at a lower pre-training cost. Lastly, we perform a small experiment using self-supervision, and show that self-supervised ViT holds promise for the future.

4.1 Setup

Datasets. To explore model scalability, we use the ILSVRC-2012 ImageNet dataset with 1k classes and 1.3M images (we refer to it as ImageNet in what follows), its superset ImageNet-21k with 21k classes and 14M images (Deng et al., 2009), and JFT (Sun et al., 2017) with 18k classes and 303M high-resolution images. We de-duplicate the pre-training datasets w.r.t. the test sets of the downstream tasks following Kolesnikov et al. (2020). We transfer the models trained on these dataset to several benchmark tasks: ImageNet on the original validation labels and the cleaned-up ReaL labels (Beyer et al., 2020), CIFAR-10/100 (Krizhevsky, 2009), Oxford-IIIT Pets (Parkhi et al., 2012), and Oxford Flowers-102 (Nilsback & Zisserman, 2008). For these datasets, pre-processing follows Kolesnikov et al. (2020).

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

We also evaluate on the 19-task VTAB classification suite (Zhai et al., 2019b). VTAB evaluates low-data transfer to diverse tasks, using 1 000 training examples per task. The tasks are divided into three groups: *Natural* – tasks like the above, Pets, CIFAR, etc. *Specialized* – medical and satellite imagery, and *Structured* – tasks that require geometric understanding like localization.

Model Variants. We base ViT configurations on those used for BERT (Devlin et al., 2019), as summarized in Table 1. The "Base" and "Large" models are directly adopted from BERT and we add the larger "Huge" model. In what follows we use brief notation to indicate the model size and the input patch size: for instance, ViT-L/16 means the "Large" variant with 16×16 input patch size. Note that the Transformer's sequence length is inversely proportional to the square of the patch size, thus models with smaller patch size are computationally more expensive.

For the baseline CNNs, we use ResNet (He et al., 2016), but replace the Batch Normalization layers (Ioffe & Szegedy, 2015) with Group Normalization (Wu & He, 2018), and used standardized convolutions (Qiao et al., 2019). These modifications improve transfer (Kolesnikov et al., 2020), and we denote the modified model "ResNet (BiT)". For the hybrids, we feed the intermediate feature maps into ViT with patch size of one "pixel". To experiment with different sequence lengths, we either (i) take the output of stage 4 of a regular ResNet50 or (ii) remove stage 4, place the same number of layers in stage 3 (keeping the total number of layers), and take the output of this extended stage 3. Option (ii) results in a 4x longer sequence length, and a more expensive ViT model.

Training & Fine-tuning. We train all models, including ResNets, using Adam (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, a batch size of 4096 and apply a high weight decay of 0.1, which we found to be useful for transfer of all models (Appendix D.1 shows that, in contrast to common practices, Adam works slightly better than SGD for ResNets in our setting). We use a linear learning rate warmup and decay, see Appendix B.1 for details. For fine-tuning we use SGD with momentum, batch size 512, for all models, see Appendix B.1.1. For ImageNet results in Table 2, we fine-tuned at higher resolution: 512 for ViT-L/16 and 518 for ViT-H/14, and also used Polyak & Juditsky (1992) averaging with a factor of 0.9999 (Ramachandran et al., 2019; Wang et al., 2020b).

Metrics. We report results on downstream datasets either through few-shot or fine-tuning accuracy. Fine-tuning accuracies capture the performance of each model after fine-tuning it on the respective dataset. Few-shot accuracies are obtained by solving a regularized least-squares regression problem that maps the (frozen) representation of a subset of training images to $\{-1, 1\}^K$ target vectors. This formulation allows us to recover the exact solution in closed form. Though we mainly focus on fine-tuning performance, we sometimes use linear few-shot accuracies for fast on-the-fly evaluation where fine-tuning would be too costly.

4.2 Comparison to State of the Art

We first compare our largest models – ViT-H/14 and ViT-L/16 – to state-of-the-art CNNs from the literature. The first comparison point is Big Transfer (BiT) (Kolesnikov et al., 2020), which performs supervised transfer learning with large ResNets. The second is Noisy Student (Xie et al., 2020), which is a large EfficientNet trained using semi-supervised learning on ImageNet and JFT-300M with the labels removed. Currently, Noisy Student is the state of the art on ImageNet and BiT-L on the other datasets reported here. All models were trained on TPUv3 hardware, and we report the number of TPUv3-core-days taken to pre-train each of them, that is, the number of TPU v3 cores (2 per chip) used for training multiplied by the training time in days.

Table 2 shows the results. The smaller ViT-L/16 model pre-trained on JFT-300M outperforms BiT-L (which is pre-trained on the same dataset) on all tasks, while requiring substantially less computational resources to train. The larger model, ViT-H/14, further improves the performance, especially on the more challenging datasets – ImageNet, CIFAR-100, and the VTAB suite. Interestingly, this

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

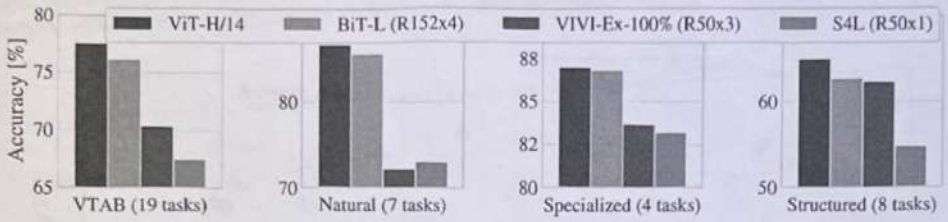

Figure 2: Breakdown of VTAB performance in *Natural*, *Specialized*, and *Structured* task groups.

model still took substantially less compute to pre-train than prior state of the art. However, we note that pre-training efficiency may be affected not only by the architecture choice, but also other parameters, such as training schedule, optimizer, weight decay, etc. We provide a controlled study of performance vs. compute for different architectures in Section 4.4. Finally, the ViT-L/16 model pre-trained on the public ImageNet-21k dataset performs well on most datasets too, while taking fewer resources to pre-train: it could be trained using a standard cloud TPUv3 with 8 cores in approximately 30 days.

Figure 2 decomposes the VTAB tasks into their respective groups, and compares to previous SOTA methods on this benchmark: BiT, VIVI – a ResNet co-trained on ImageNet and Youtube (Tschannen et al., 2020), and S4L – supervised plus semi-supervised learning on ImageNet (Zhai et al., 2019a). ViT-H/14 outperforms BiT-R152x4, and other methods, on the *Natural* and *Structured* tasks. On the *Specialized* the performance of the top two models is similar.

4.3 Pre-training Data Requirements

The Vision Transformer performs well when pre-trained on a large JFT-300M dataset. With fewer inductive biases for vision than ResNets, how crucial is the dataset size? We perform two series of experiments.

First, we pre-train ViT models on datasets of increasing size: ImageNet, ImageNet-21k, and JFT-300M. To boost the performance on the smaller datasets, we optimize three basic regularization parameters – weight decay, dropout, and label smoothing. Figure 3 shows the results after fine-tuning to ImageNet (results on other datasets are shown in Table 5)[2]. When pre-trained on the smallest dataset, ImageNet, ViT-Large models underperform compared to ViT-Base models, despite (moderate) regularization. With ImageNet-21k pre-training, their performances are similar. Only with JFT-300M, do we see the full benefit of larger models. Figure 3 also shows the performance

[2]Note that the ImageNet pre-trained models are also fine-tuned, but again on ImageNet. This is because the resolution increase during fine-tuning improves the performance.

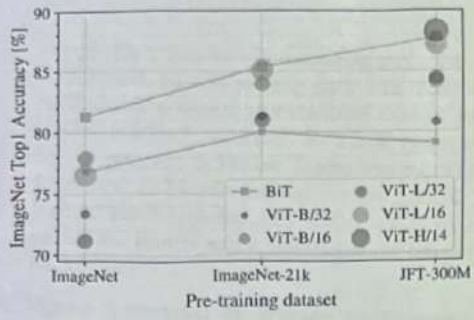

Figure 3: Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows.

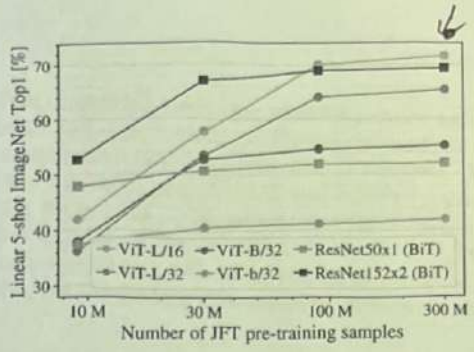

Figure 4: Linear few-shot evaluation on ImageNet versus pre-training size. ResNets perform better with smaller pre-training datasets but plateau sooner than ViT, which performs better with larger pre-training. ViT-b is ViT-B with all hidden dimensions halved.

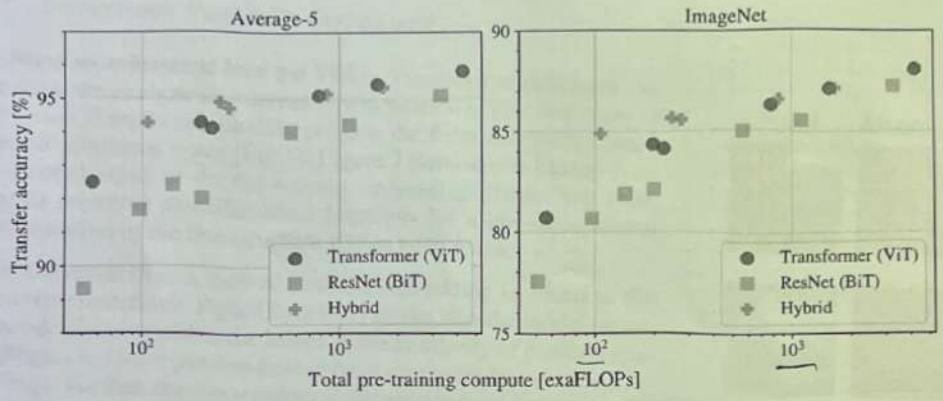

Figure 5: Performance versus pre-training compute for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

region spanned by BiT models of different sizes. The BiT CNNs outperform ViT on ImageNet, but with the larger datasets, ViT overtakes.

Second, we train our models on random subsets of 9M, 30M, and 90M as well as the full JFT-300M dataset. We do not perform additional regularization on the smaller subsets and use the same hyper-parameters for all settings. This way, we assess the intrinsic model properties, and not the effect of regularization. We do, however, use early-stopping, and report the best validation accuracy achieved during training. To save compute, we report few-shot linear accuracy instead of full fine-tuning accuracy. Figure 4 contains the results. Vision Transformers overfit more than ResNets with comparable computational cost on smaller datasets. For example, ViT-B/32 is slightly faster than ResNet50; it performs much worse on the 9M subset, but better on 90M+ subsets. The same is true for ResNet152x2 and ViT-L/16. This result reinforces the intuition that the convolutional inductive bias is useful for smaller datasets, but for larger ones, learning the relevant patterns directly from data is sufficient, even beneficial.

Overall, the few-shot results on ImageNet (Figure 4), as well as the low-data results on VTAB (Table 2) seem promising for very low-data transfer. Further analysis of few-shot properties of ViT is an exciting direction of future work.

4.4 SCALING STUDY

We perform a controlled scaling study of different models by evaluating transfer performance from JFT-300M. In this setting data size does not bottleneck the models' performances, and we assess performance versus pre-training cost of each model. The model set includes: 7 ResNets, R50x1, R50x2 R101x1, R152x1, R152x2, pre-trained for 7 epochs, plus R152x2 and R200x3 pre-trained for 14 epochs; 6 Vision Transformers, ViT-B/32, B/16, L/32, L/16, pre-trained for 7 epochs, plus L/16 and H/14 pre-trained for 14 epochs; and 5 hybrids, R50+ViT-B/32, B/16, L/32, L/16 pre-trained for 7 epochs, plus R50+ViT-L/16 pre-trained for 14 epochs (for hybrids, the number at the end of the model name stands not for the patch size, but for the total dowsampling ratio in the ResNet backbone).

Figure 5 contains the transfer performance versus total pre-training compute (see Appendix D.5 for details on computational costs). Detailed results per model are provided in Table 6 in the Appendix. A few patterns can be observed. First, Vision Transformers dominate ResNets on the performance/compute trade-off. ViT uses approximately $2 - 4\times$ less compute to attain the same performance (average over 5 datasets). Second, hybrids slightly outperform ViT at small computational budgets, but the difference vanishes for larger models. This result is somewhat surprising, since one might expect convolutional local feature processing to assist ViT at any size. Third, Vision Transformers appear not to saturate within the range tried, motivating future scaling efforts.

4.5 INSPECTING VISION TRANSFORMER

To begin to understand how the Vision Transformer processes image data, we analyze its internal representations. The first layer of the Vision Transformer linearly projects the flattened patches into a lower-dimensional space (Eq. 1). Figure 7 (left) shows the top principal components of the the learned embedding filters. The components resemble plausible basis functions for a low-dimensional representation of the fine structure within each patch.

After the projection, a learned position embedding is added to the patch representations. Figure 7 (center) shows that the model learns to encode distance within the image in the similarity of position embeddings, i.e. closer patches tend to have more similar position embeddings. Further, the row-column structure appears; patches in the same row/column have similar embeddings. Finally, a sinusoidal structure is sometimes apparent for larger grids (Appendix D). That the position embeddings learn to represent 2D image topology explains why hand-crafted 2D-aware embedding variants do not yield improvements (Appendix D.4).

Figure 6: Representative examples of attention from the output token to the input space. See Appendix D.7 for details.

Self-attention allows ViT to integrate information across the entire image even in the lowest layers. We investigate to what degree the network makes use of this capability. Specifically, we compute the average distance in image space across which information is integrated, based on the attention weights (Figure 7, right). This "attention distance" is analogous to receptive field size in CNNs. We find that some heads attend to most of the image already in the lowest layers, showing that the ability to integrate information globally is indeed used by the model. Other attention heads have consistently small attention distances in the low layers. This highly localized attention is less pronounced in hybrid models that apply a ResNet before the Transformer (Figure 7, right), suggesting that it may serve a similar function as early convolutional layers in CNNs. Further, the attention distance increases with network depth. Globally, we find that the model attends to image regions that are semantically relevant for classification (Figure 6).

4.6 SELF-SUPERVISION

Transformers show impressive performance on NLP tasks. However, much of their success stems not only from their excellent scalability but also from large scale self-supervised pre-training (Devlin

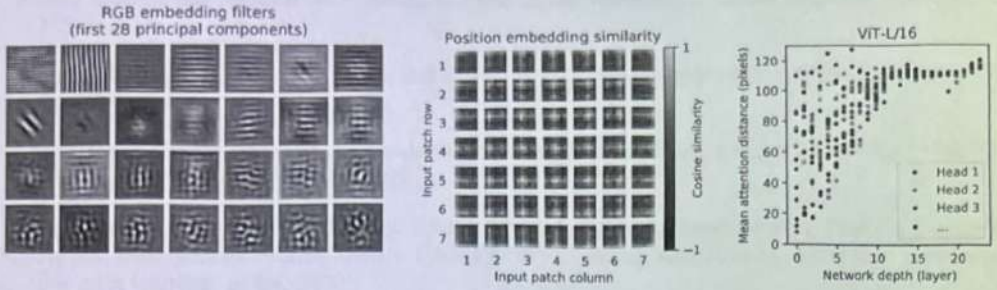

Figure 7: **Left:** Filters of the initial linear embedding of RGB values of ViT-L/32. **Center:** Similarity of position embeddings of ViT-L/32. Tiles show the cosine similarity between the position embedding of the patch with the indicated row and column and the position embeddings of all other patches. **Right:** Size of attended area by head and network depth. Each dot shows the mean attention distance across images for one of 16 heads at one layer. See Appendix D.7 for details.

et al., 2019; Radford et al., 2018). We also perform a preliminary exploration on *masked patch prediction* for self-supervision, mimicking the masked language modeling task used in BERT. With self-supervised pre-training, our smaller ViT-B/16 model achieves 79.9% accuracy on ImageNet, a significant improvement of 2% to training from scratch, but still 4% behind supervised pre-training. Appendix B.1.2 contains further details. We leave exploration of contrastive pre-training (Chen et al., 2020b; He et al., 2020; Bachman et al., 2019; Hénaff et al., 2020) to future work.

5 Conclusion

We have explored the direct application of Transformers to image recognition. Unlike prior works using self-attention in computer vision, we do not introduce image-specific inductive biases into the architecture apart from the initial patch extraction step. Instead, we interpret an image as a sequence of patches and process it by a standard Transformer encoder as used in NLP. This simple, yet scalable, strategy works surprisingly well when coupled with pre-training on large datasets. Thus, Vision Transformer matches or exceeds the state of the art on many image classification datasets, whilst being relatively cheap to pre-train.

While these initial results are encouraging, many challenges remain. One is to apply ViT to other computer vision tasks, such as detection and segmentation. Our results, coupled with those in Carion et al. (2020), indicate the promise of this approach. Another challenge is to continue exploring self-supervised pre-training methods. Our initial experiments show improvement from self-supervised pre-training, but there is still large gap between self-supervised and large-scale supervised pre-training. Finally, further scaling of ViT would likely lead to improved performance.

Acknowledgements

The work was performed in Berlin, Zürich, and Amsterdam. We thank many colleagues at Google for their help, in particular Andreas Steiner for crucial help with the infrastructure and the open-source release of the code; Joan Puigcerver and Maxim Neumann for help with the large-scale training infrastructure; Dmitry Lepikhin, Aravindh Mahendran, Daniel Keysers, Mario Lučić, Noam Shazeer, Ashish Vaswani, and Colin Raffel for useful discussions.

References

Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *ACL*, 2020.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019.

Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu[†*] Yutong Lin[†*] Yue Cao[*] Han Hu[*‡] Yixuan Wei[†]
Zheng Zhang Stephen Lin Baining Guo
Microsoft Research Asia
{v-zeliul,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

Abstract

This paper presents a new vision Transformer, called Swin Transformer, that capably serves as a general-purpose backbone for computer vision. Challenges in adapting Transformer from language to vision arise from differences between the two domains, such as large variations in the scale of visual entities and the high resolution of pixels in images compared to words in text. To address these differences, we propose a hierarchical Transformer whose representation is computed with Shifted windows. The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size. These qualities of Swin Transformer make it compatible with a broad range of vision tasks, including image classification (87.3 top-1 accuracy on ImageNet-1K) and dense prediction tasks such as object detection (58.7 box AP and 51.1 mask AP on COCO test-dev) and semantic segmentation (53.5 mIoU on ADE20K val). Its performance surpasses the previous state-of-the-art by a large margin of +2.7 box AP and +2.6 mask AP on COCO, and +3.2 mIoU on ADE20K, demonstrating the potential of Transformer-based models as vision backbones. The hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures. The code and models are publicly available at https://github.com/microsoft/Swin-Transformer.

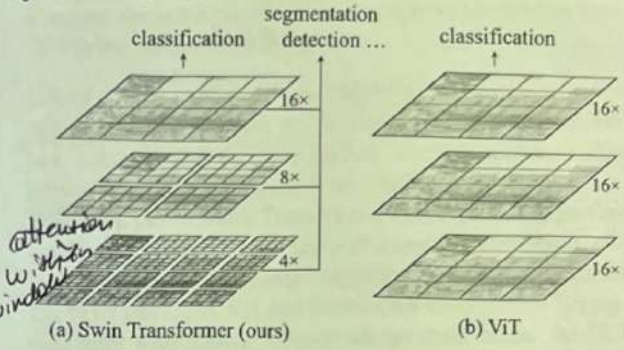

Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

1. Introduction

Modeling in computer vision has long been dominated by convolutional neural networks (CNNs). Beginning with AlexNet [39] and its revolutionary performance on the ImageNet image classification challenge, CNN architectures have evolved to become increasingly powerful through greater scale [30, 76], more extensive connections [34], and more sophisticated forms of convolution [70, 18, 84]. With CNNs serving as backbone networks for a variety of vision tasks, these architectural advances have led to performance improvements that have broadly lifted the entire field.

On the other hand, the evolution of network architectures in natural language processing (NLP) has taken a different path, where the prevalent architecture today is instead the Transformer [64]. Designed for sequence modeling and transduction tasks, the Transformer is notable for its use of attention to model long-range dependencies in the data. Its tremendous success in the language domain has led researchers to investigate its adaptation to computer vision, where it has recently demonstrated promising results on certain tasks, specifically image classification [20] and joint vision-language modeling [47].

In this paper, we seek to expand the applicability of Transformer such that it can serve as a general-purpose

*Equal contribution. †Interns at MSRA. ‡Contact person.

backbone for computer vision, as it does for NLP and as CNNs do in vision. We observe that significant challenges in transferring its high performance in the language domain to the visual domain can be explained by differences between the two modalities. One of these differences involves scale. Unlike the word tokens that serve as the basic elements of processing in language Transformers, visual elements can vary substantially in scale, a problem that receives attention in tasks such as object detection [42, 53, 54]. In existing Transformer-based models [64, 20], tokens are all of a fixed scale, a property unsuitable for these vision applications. Another difference is the much higher resolution of pixels in images compared to words in passages of text. There exist many vision tasks such as semantic segmentation that require dense prediction at the pixel level, and this would be intractable for Transformer on high-resolution images, as the computational complexity of its self-attention is quadratic to image size. To overcome these issues, we propose a general-purpose Transformer backbone, called Swin Transformer, which constructs hierarchical feature maps and has linear computational complexity to image size. As illustrated in Figure 1(a), Swin Transformer constructs a hierarchical representation by starting from small-sized patches (outlined in gray) and gradually merging neighboring patches in deeper Transformer layers. With these hierarchical feature maps, the Swin Transformer model can conveniently leverage advanced techniques for dense prediction such as feature pyramid networks (FPN) [42] or U-Net [51]. The linear computational complexity is achieved by computing self-attention locally within non-overlapping windows that partition an image (outlined in red). The number of patches in each window is fixed, and thus the complexity becomes linear to image size. These merits make Swin Transformer suitable as a general-purpose backbone for various vision tasks, in contrast to previous Transformer based architectures [20] which produce feature maps of a single resolution and have quadratic complexity.

A key design element of Swin Transformer is its *shift* of the window partition between consecutive self-attention layers, as illustrated in Figure 2. The shifted windows bridge the windows of the preceding layer, providing connections among them that significantly enhance modeling power (see Table 4). This strategy is also efficient in regards to real-world latency: all *query* patches within a window share the same *key* set[1], which facilitates memory access in hardware. In contrast, earlier *sliding window* based self-attention approaches [33, 50] suffer from low latency on general hardware due to different *key* sets for different *query* pixels[2]. Our experiments show that the proposed

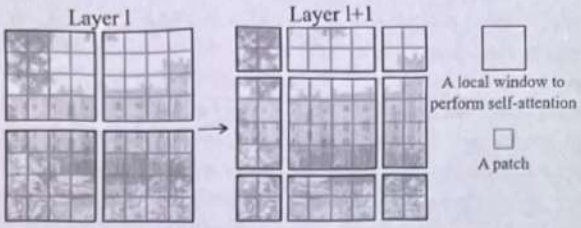

Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l+1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

shifted window approach has much lower latency than the *sliding window* method, yet is similar in modeling power (see Tables 5 and 6). The shifted window approach also proves beneficial for all-MLP architectures [61].

The proposed Swin Transformer achieves strong performance on the recognition tasks of image classification, object detection and semantic segmentation. It outperforms the ViT / DeiT [20, 63] and ResNe(X)t models [30, 70] significantly with similar latency on the three tasks. Its 58.7 box AP and 51.1 mask AP on the COCO test-dev set surpass the previous state-of-the-art results by +2.7 box AP (Copy-paste [26] without external data) and +2.6 mask AP (DetectoRS [46]). On ADE20K semantic segmentation, it obtains 53.5 mIoU on the val set, an improvement of +3.2 mIoU over the previous state-of-the-art (SETR [81]). It also achieves a top-1 accuracy of 87.3% on ImageNet-1K image classification.

It is our belief that a unified architecture across computer vision and natural language processing could benefit both fields, since it would facilitate joint modeling of visual and textual signals and the modeling knowledge from both domains can be more deeply shared. We hope that Swin Transformer's strong performance on various vision problems can drive this belief deeper in the community and encourage unified modeling of vision and language signals.

2. Related Work

CNN and variants CNNs serve as the standard network model throughout computer vision. While the CNN has existed for several decades [40], it was not until the introduction of AlexNet [39] that the CNN took off and became mainstream. Since then, deeper and more effective convolutional neural architectures have been proposed to further propel the deep learning wave in computer vision, e.g., VGG [52], GoogleNet [57], ResNet [30], DenseNet [34],

[1]The *query* and *key* are projection vectors in a self-attention layer.

[2]While there are efficient methods to implement a sliding-window based convolution layer on general hardware, thanks to its shared kernel weights across a feature map, it is difficult for a sliding-window based self-attention layer to have efficient memory access in practice.

HRNet [65], and EfficientNet [58]. In addition to these architectural advances, there has also been much work on improving individual convolution layers, such as depth-wise convolution [70] and deformable convolution [18, 84]. While the CNN and its variants are still the primary backbone architectures for computer vision applications, we highlight the strong potential of Transformer-like architectures for unified modeling between vision and language. Our work achieves strong performance on several basic visual recognition tasks, and we hope it will contribute to a modeling shift.

Self-attention based backbone architectures Also inspired by the success of self-attention layers and Transformer architectures in the NLP field, some works employ self-attention layers to replace some or all of the spatial convolution layers in the popular ResNet [33, 50, 80]. In these works, the self-attention is computed within a local window of each pixel to expedite optimization [33], and they achieve slightly better accuracy/FLOPs trade-offs than the counterpart ResNet architecture. However, their costly memory access causes their actual latency to be significantly larger than that of the convolutional networks [33]. Instead of using sliding windows, we propose to *shift* windows between consecutive layers, which allows for a more efficient implementation in general hardware.

Self-attention/Transformers to complement CNNs Another line of work is to augment a standard CNN architecture with self-attention layers or Transformers. The self-attention layers can complement backbones [67, 7, 3, 71, 23, 74, 55] or head networks [32, 27] by providing the capability to encode distant dependencies or heterogeneous interactions. More recently, the encoder-decoder design in Transformer has been applied for the object detection and instance segmentation tasks [8, 13, 85, 56]. Our work explores the adaptation of Transformers for basic visual feature extraction and is complementary to these works.

Transformer based vision backbones Most related to our work is the Vision Transformer (ViT) [20] and its follow-ups [63, 72, 15, 28, 66]. The pioneering work of ViT directly applies a Transformer architecture on non-overlapping medium-sized image patches for image classification. It achieves an impressive speed-accuracy trade-off on image classification compared to convolutional networks. While ViT requires large-scale training datasets (i.e., JFT-300M) to perform well, DeiT [63] introduces several training strategies that allow ViT to also be effective using the smaller ImageNet-1K dataset. The results of ViT on image classification are encouraging, but its architecture is unsuitable for use as a general-purpose backbone network on dense vision tasks or when the input image resolution is high, due to its low-resolution feature maps and the quadratic increase in complexity with image size. There are a few works applying ViT models to the dense vision tasks of object detection and semantic segmentation by direct upsampling or deconvolution but with relatively lower performance [2, 81]. Concurrent to our work are some that modify the ViT architecture [72, 15, 28] for better image classification. Empirically, we find our Swin Transformer architecture to achieve the best speed-accuracy trade-off among these methods on image classification, even though our work focuses on general-purpose performance rather than specifically on classification. Another concurrent work [66] explores a similar line of thinking to build multi-resolution feature maps on Transformers. Its complexity is still quadratic to image size, while ours is linear and also operates locally which has proven beneficial in modeling the high correlation in visual signals [36, 25, 41]. Our approach is both efficient and effective, achieving state-of-the-art accuracy on both COCO object detection and ADE20K semantic segmentation.

3. Method

3.1. Overall Architecture

An overview of the Swin Transformer architecture is presented in Figure 3, which illustrates the tiny version (Swin-T). It first splits an input RGB image into non-overlapping patches by a patch splitting module, like ViT. Each patch is treated as a "token" and its feature is set as a concatenation of the raw pixel RGB values. In our implementation, we use a patch size of 4×4 and thus the feature dimension of each patch is $4 \times 4 \times 3 = 48$. A linear embedding layer is applied on this raw-valued feature to project it to an arbitrary dimension (denoted as C).

Several Transformer blocks with modified self-attention computation (*Swin Transformer blocks*) are applied on these patch tokens. The Transformer blocks maintain the number of tokens ($\frac{H}{4} \times \frac{W}{4}$), and together with the linear embedding are referred to as "Stage 1".

To produce a hierarchical representation, the number of tokens is reduced by patch merging layers as the network gets deeper. The first patch merging layer concatenates the features of each group of 2×2 neighboring patches, and applies a linear layer on the $4C$-dimensional concatenated features. This reduces the number of tokens by a multiple of $2 \times 2 = 4$ ($2\times$ downsampling of resolution), and the output dimension is set to $2C$. Swin Transformer blocks are applied afterwards for feature transformation, with the resolution kept at $\frac{H}{8} \times \frac{W}{8}$. This first block of patch merging and feature transformation is denoted as "Stage 2". The procedure is repeated twice, as "Stage 3" and "Stage 4", with output resolutions of $\frac{H}{16} \times \frac{W}{16}$ and $\frac{H}{32} \times \frac{W}{32}$, respectively. These stages jointly produce a hierarchical representation,

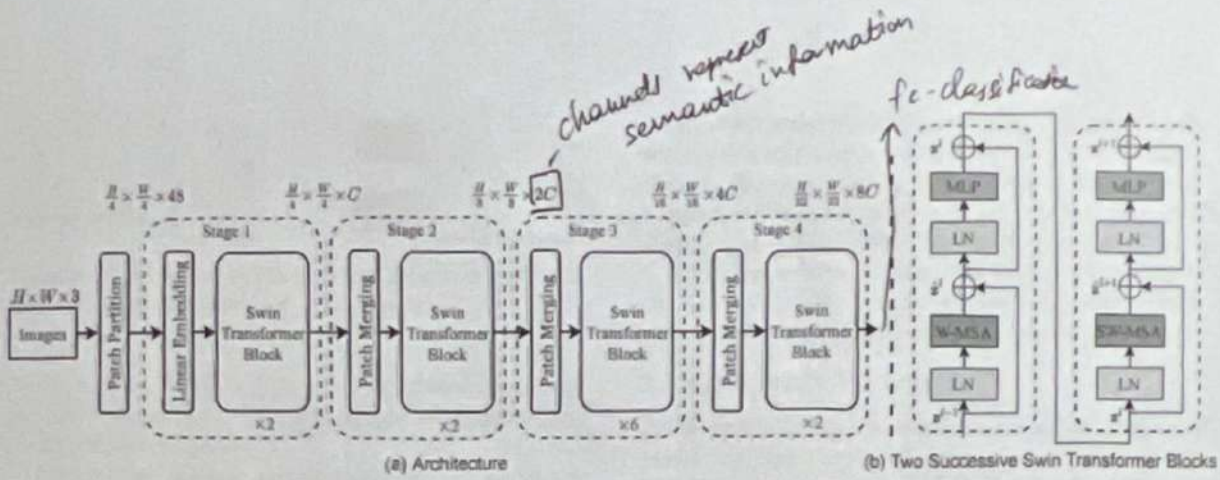

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

with the same feature map resolutions as those of typical convolutional networks, e.g., VGG [52] and ResNet [30]. As a result, the proposed architecture can conveniently replace the backbone networks in existing methods for various vision tasks.

Swin Transformer block Swin Transformer is built by replacing the standard multi-head self attention (MSA) module in a Transformer block by a module based on shifted windows (described in Section 3.2), with other layers kept the same. As illustrated in Figure 3(b), a Swin Transformer block consists of a shifted window based MSA module, followed by a 2-layer MLP with GELU nonlinearity in between. A LayerNorm (LN) layer is applied before each MSA module and each MLP, and a residual connection is applied after each module.

3.2. Shifted Window based Self-Attention

The standard Transformer architecture [64] and its adaptation for image classification [20] both conduct global self-attention, where the relationships between a token and all other tokens are computed. The global computation leads to quadratic complexity with respect to the number of tokens, making it unsuitable for many vision problems requiring an immense set of tokens for dense prediction or to represent a high-resolution image.

Self-attention in non-overlapped windows For efficient modeling, we propose to compute self-attention within local windows. The windows are arranged to evenly partition the image in a non-overlapping manner. Supposing each window contains $M \times M$ patches, the computational complexity of a global MSA module and a window based one on an image of $h \times w$ patches[3]:

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \tag{1}$$

$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \tag{2}$$

where the former is quadratic to patch number hw, and the latter is linear when M is fixed (set to 7 by default). Global self-attention computation is generally unaffordable for a large hw, while the window based self-attention is scalable.

Shifted window partitioning in successive blocks The window-based self-attention module lacks connections across windows, which limits its modeling power. To introduce cross-window connections while maintaining the efficient computation of non-overlapping windows, we propose a shifted window partitioning approach which alternates between two partitioning configurations in consecutive Swin Transformer blocks.

As illustrated in Figure 2, the first module uses a regular window partitioning strategy which starts from the top-left pixel, and the 8×8 feature map is evenly partitioned into 2×2 windows of size 4×4 ($M = 4$). Then, the next module adopts a windowing configuration that is shifted from that of the preceding layer, by displacing the windows by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels from the regularly partitioned windows.

With the shifted window partitioning approach, consecutive Swin Transformer blocks are computed as

$$\begin{aligned}
\hat{\mathbf{z}}^l &= \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1}, \\
\mathbf{z}^l &= \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^l\right)\right) + \hat{\mathbf{z}}^l, \\
\hat{\mathbf{z}}^{l+1} &= \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^l\right)\right) + \mathbf{z}^l, \\
\mathbf{z}^{l+1} &= \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},
\end{aligned} \tag{3}$$

where $\hat{\mathbf{z}}^l$ and $\mathbf{z}^l$ denote the output features of the (S)W-MSA module and the MLP module for block l, respectively;

[3] We omit SoftMax computation in determining complexity.

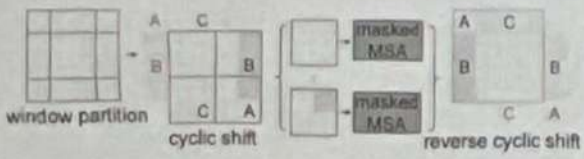

Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

W-MSA and SW-MSA denote window based multi-head self-attention using regular and shifted window partitioning configurations, respectively.

The shifted window partitioning approach introduces connections between neighboring non-overlapping windows in the previous layer and is found to be effective in image classification, object detection, and semantic segmentation, as shown in Table 4.

Efficient batch computation for shifted configuration An issue with shifted window partitioning is that it will result in more windows, from $\lceil \frac{h}{M} \rceil \times \lceil \frac{w}{M} \rceil$ to $(\lceil \frac{h}{M} \rceil + 1) \times (\lceil \frac{w}{M} \rceil + 1)$ in the shifted configuration, and some of the windows will be smaller than $M \times M$[4]. A naive solution is to pad the smaller windows to a size of $M \times M$ and mask out the padded values when computing attention. When the number of windows in regular partitioning is small, e.g. 2×2, the increased computation with this naive solution is considerable ($2 \times 2 \rightarrow 3 \times 3$, which is 2.25 times greater). Here, we propose a *more efficient batch computation approach* by cyclic-shifting toward the top-left direction, as illustrated in Figure 4. After this shift, a batched window may be composed of several sub-windows that are not adjacent in the feature map, so a masking mechanism is employed to limit self-attention computation to within each sub-window. With the cyclic-shift, the number of batched windows remains the same as that of regular window partitioning, and thus is also efficient. The low latency of this approach is shown in Table 5.

Relative position bias In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; d is the *query/key* dimension, and M^2 is the number of patches in a window. Since the relative position along each axis lies in the range $[-M+1, M-1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$, and values in B are taken from $\hat{B}$.

[4]To make the window size (M, M) divisible by the feature map size of (h, w), bottom-right padding is employed on the feature map if needed.

We observe significant improvements over counterparts without this bias term or that use absolute position embedding, as shown in Table 4. Further adding absolute position embedding to the input as in [20] drops performance slightly, thus it is not adopted in our implementation.

The learnt relative position bias in pre-training can be also used to initialize a model for fine-tuning with a different window size through bi-cubic interpolation [20, 63].

3.3. Architecture Variants

We build our base model, called Swin-B, to have of model size and computation complexity similar to ViT-B/DeiT-B. We also introduce Swin-T, Swin-S and Swin-L, which are versions of about $0.25\times$, $0.5\times$ and $2\times$ the model size and computational complexity, respectively. Note that the complexity of Swin-T and Swin-S are similar to those of ResNet-50 (DeiT-S) and ResNet-101, respectively. The window size is set to $M = 7$ by default. The query dimension of each head is $d = 32$, and the expansion layer of each MLP is $\alpha = 4$, for all experiments. The architecture hyper-parameters of these model variants are:

- Swin-T: $C = 96$, layer numbers $= \{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers $= \{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers $= \{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers $= \{2, 2, 18, 2\}$

where C is the channel number of the hidden layers in the first stage. The model size, theoretical computational complexity (FLOPs), and throughput of the model variants for ImageNet image classification are listed in Table 1.

4. Experiments

We conduct experiments on ImageNet-1K image classification [19], COCO object detection [43], and ADE20K semantic segmentation [83]. In the following, we first compare the proposed Swin Transformer architecture with the previous state-of-the-arts on the three tasks. Then, we ablate the important design elements of Swin Transformer.

4.1. Image Classification on ImageNet-1K

Settings For image classification, we benchmark the proposed Swin Transformer on ImageNet-1K [19], which contains 1.28M training images and 50K validation images from 1,000 classes. The top-1 accuracy on a single crop is reported. We consider two training settings:

- *Regular ImageNet-1K training*. This setting mostly follows [63]. We employ an AdamW [37] optimizer for 300 epochs using a cosine decay learning rate scheduler and 20 epochs of linear warm-up. A batch size of 1024, an initial learning rate of 0.001, and a

weight decay of 0.05 are used. We include most of the augmentation and regularization strategies of [63] in training, except for repeated augmentation [31] and EMA [45], which do not enhance performance. Note that this is contrary to [63] where repeated augmentation is crucial to stabilize the training of ViT.

- *Pre-training on ImageNet-22K and fine-tuning on ImageNet-1K.* We also pre-train on the larger ImageNet-22K dataset, which contains 14.2 million images and 22K classes. We employ an AdamW optimizer for 90 epochs using a linear decay learning rate scheduler with a 5-epoch linear warm-up. A batch size of 4096, an initial learning rate of 0.001, and a weight decay of 0.01 are used. In ImageNet-1K fine-tuning, we train the models for 30 epochs with a batch size of 1024, a constant learning rate of 10^{-5}, and a weight decay of 10^{-8}.

Results with regular ImageNet-1K training Table 1(a) presents comparisons to other backbones, including both Transformer-based and ConvNet-based, using regular ImageNet-1K training.

Compared to the previous state-of-the-art Transformer-based architecture, i.e. DeiT [63], Swin Transformers noticeably surpass the counterpart DeiT architectures with similar complexities: +1.5% for Swin-T (81.3%) over DeiT-S (79.8%) using 224^2 input, and +1.5%/1.4% for Swin-B (83.3%/84.5%) over DeiT-B (81.8%/83.1%) using 224^2/384^2 input, respectively.

Compared with the state-of-the-art ConvNets, i.e. RegNet [48] and EfficientNet [58], the Swin Transformer achieves a slightly better speed-accuracy trade-off. Noting that while RegNet [48] and EfficientNet [58] are obtained via a thorough architecture search, the proposed Swin Transformer is adapted from the standard Transformer and has strong potential for further improvement.

Results with ImageNet-22K pre-training We also pre-train the larger-capacity Swin-B and Swin-L on ImageNet-22K. Results fine-tuned on ImageNet-1K image classification are shown in Table 1(b). For Swin-B, the ImageNet-22K pre-training brings 1.8%∼1.9% gains over training on ImageNet-1K from scratch. Compared with the previous best results for ImageNet-22K pre-training, our models achieve significantly better speed-accuracy trade-offs: Swin-B obtains 86.4% top-1 accuracy, which is 2.4% higher than that of ViT with similar inference throughput (84.7 vs. 85.9 images/sec) and slightly lower FLOPs (47.0G vs. 55.4G). The larger Swin-L model achieves 87.3% top-1 accuracy, +0.9% better than that of the Swin-B model.

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

4.2. Object Detection on COCO

Settings Object detection and instance segmentation experiments are conducted on COCO 2017, which contains 118K training, 5K validation and 20K test-dev images. An ablation study is performed using the validation set, and a system-level comparison is reported on test-dev. For the ablation study, we consider four typical object detection frameworks: Cascade Mask R-CNN [29, 6], ATSS [79], RepPoints v2 [12], and Sparse RCNN [56] in mmdetection [10]. For these four frameworks, we utilize the same settings: multi-scale training [8, 56] (resizing the input such that the shorter side is between 480 and 800 while the longer side is at most 1333), AdamW [44] optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs). For system-level comparison, we adopt an improved HTC [9] (denoted as HTC++) with instaboost [22], stronger multi-scale training [7], 6x schedule (72 epochs), soft-NMS [5], and ImageNet-22K pre-trained model as initialization.

We compare our Swin Transformer to standard Con-

(a) Various frameworks

Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade Mask R-CNN	R-50	46.3	64.3	50.5	82M	739G	18.0
	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse R-CNN	R-50	44.5	63.4	48.2	106M	166G	21.0
	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN

	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S†	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison

Method	mini-val AP^{box}	mini-val AP^{mask}	test-dev AP^{box}	test-dev AP^{mask}	#param.	FLOPs
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Table 2. Results on COCO object detection and instance segmentation. †denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

vNets, i.e. ResNe(X)t, and previous Transformer networks, e.g. DeiT. The comparisons are conducted by changing only the backbones with other settings unchanged. Note that while Swin Transformer and ResNe(X)t are directly applicable to all the above frameworks because of their hierarchical feature maps, DeiT only produces a single resolution of feature maps and cannot be directly applied. For fair comparison, we follow [81] to construct hierarchical feature maps for DeiT using deconvolution layers.

Comparison to ResNe(X)t Table 2(a) lists the results of Swin-T and ResNet-50 on the four object detection frameworks. Our Swin-T architecture brings consistent +3.4~4.2 box AP gains over ResNet-50, with slightly larger model size, FLOPs and latency.

Table 2(b) compares Swin Transformer and ResNe(X)t under different model capacity using Cascade Mask R-CNN. Swin Transformer achieves a high detection accuracy of 51.9 box AP and 45.0 mask AP, which are significant gains of +3.6 box AP and +3.3 mask AP over ResNeXt101-64x4d, which has similar model size, FLOPs and latency. On a higher baseline of 52.3 box AP and 46.0 mask AP using an improved HTC framework, the gains by Swin Transformer are also high, at +4.1 box AP and +3.1 mask AP (see Table 2(c)). Regarding inference speed, while ResNe(X)t is built by highly optimized Cudnn functions, our architecture is implemented with built-in PyTorch functions that are not all well-optimized. A thorough kernel optimization is beyond the scope of this paper.

Method	Backbone	ADE20K val mIoU	ADE20K test score	#param.	FLOPs	FPS
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large‡	50.3	61.7	308M	-	-
UperNet	DeiT-S†	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B‡	51.6	-	121M	1841G	8.7
UperNet	Swin-L‡	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. † indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

Comparison to DeiT The performance of DeiT-S using the Cascade Mask R-CNN framework is shown in Table 2(b). The results of Swin-T are +2.5 box AP and +2.3 mask AP higher than DeiT-S with similar model size (86M vs. 80M) and significantly higher inference speed (15.3 FPS vs. 10.4 FPS). The lower inference speed of DeiT is mainly due to its quadratic complexity to input image size.

Comparison to previous state-of-the-art Table 2(c) compares our best results with those of previous state-of-the-art models. Our best model achieves 58.7 box AP and 51.1 mask AP on COCO test-dev, surpassing the previous best results by +2.7 box AP (Copy-paste [26] without external data) and +2.6 mask AP (DetectoRS [46]).

4.3. Semantic Segmentation on ADE20K

Settings ADE20K [83] is a widely-used semantic segmentation dataset, covering a broad range of 150 semantic

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

categories. It has 25K images in total, with 20K for training, 2K for validation, and another 3K for testing. We utilize UperNet [69] in mmseg [16] as our base framework for its high efficiency. More details are presented in the Appendix.

Results Table 3 lists the mIoU, model size (#param), FLOPs and FPS for different method/backbone pairs. From these results, it can be seen that Swin-S is +5.3 mIoU higher (49.3 vs. 44.0) than DeiT-S with similar computation cost. It is also +4.4 mIoU higher than ResNet-101, and +2.4 mIoU higher than ResNeSt-101 [78]. Our Swin-L model with ImageNet-22K pre-training achieves 53.5 mIoU on the val set, surpassing the previous best model by +3.2 mIoU (50.3 mIoU by SETR [81] which has a larger model size).

4.4. Ablation Study

In this section, we ablate important design elements in the proposed Swin Transformer, using ImageNet-1K image classification, Cascade Mask R-CNN on COCO object detection, and UperNet on ADE20K semantic segmentation.

Shifted windows Ablations of the *shifted window* approach on the three tasks are reported in Table 4. Swin-T with the shifted window partitioning outperforms the counterpart built on a single window partitioning at each stage by +1.1% top-1 accuracy on ImageNet-1K, +2.8 box AP/+2.2 mask AP on COCO, and +2.8 mIoU on ADE20K. The results indicate the effectiveness of using shifted windows to build connections among windows in the preceding layers. The latency overhead by *shifted window* is also small, as shown in Table 5.

Relative position bias Table 4 shows comparisons of different position embedding approaches. Swin-T with relative position bias yields +1.2%/+0.8% top-1 accuracy on ImageNet-1K, +1.3/+1.5 box AP and +1.1/+1.3 mask AP on COCO, and +2.3/+2.9 mIoU on ADE20K in relation to those without position encoding and with absolute position embedding, respectively, indicating the effectiveness of the relative position bias. Also note that while the inclusion of absolute position embedding improves image classification accuracy (+0.4%), it harms object detection and semantic segmentation (-0.2 box/mask AP on COCO and -0.6 mIoU on ADE20K).

While the recent ViT/DeiT models abandon translation invariance in image classification even though it has long been shown to be crucial for visual modeling, we find that inductive bias that encourages certain translation invariance is still preferable for general-purpose visual modeling, particularly for the dense prediction tasks of object detection and semantic segmentation.

method	MSA in a stage (ms)				Arch. (FPS)		
	S1	S2	S3	S4	T	S	B
sliding window (naive)	122.5	38.3	12.1	7.6	183	109	77
sliding window (kernel)	7.6	4.7	2.7	1.8	488	283	187
Performer [14]	4.8	2.8	1.8	1.5	638	370	241
window (w/o shifting)	2.8	1.7	1.2	0.9	770	444	280
shifted window (padding)	3.3	2.3	1.9	2.2	670	371	236
shifted window (cyclic)	3.0	1.9	1.3	1.0	755	437	278

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.

Different self-attention methods The real speed of different self-attention computation methods and implementations are compared in Table 5. Our cyclic implementation is more hardware efficient than naive padding, particularly for deeper stages. Overall, it brings a 13%, 18% and 18% speed-up on Swin-T, Swin-S and Swin-B, respectively.

The self-attention modules built on the proposed *shifted window* approach are 40.8×/2.5×, 20.2×/2.5×, 9.3×/2.1×, and 7.6×/1.8× more efficient than those of *sliding windows* in naive/kernel implementations on four network stages, respectively. Overall, the Swin Transformer architectures built on *shifted windows* are 4.1/1.5, 4.0/1.5, 3.6/1.5 times faster than variants built on *sliding windows* for Swin-T, Swin-S, and Swin-B, respectively. Table 6 compares their accuracy on the three tasks, showing that they are similarly accurate in visual modeling.

Compared to Performer [14], which is one of the fastest Transformer architectures (see [60]), the proposed *shifted window* based self-attention computation and the overall Swin Transformer architectures are slightly faster (see Table 5), while achieving +2.3% top-1 accuracy compared to Performer on ImageNet-1K using Swin-T (see Table 6).

5. Conclusion

This paper presents Swin Transformer, a new vision Transformer which produces a hierarchical feature repre-

Training data-efficient image transformers & distillation through attention

Hugo Touvron*,† Matthieu Cord† Matthijs Douze*
Francisco Massa* Alexandre Sablayrolles* Hervé Jégou*
*Facebook AI †Sorbonne University

Abstract

Recently, neural networks purely based on attention were shown to address image understanding tasks such as image classification. These high-performing vision transformers are pre-trained with hundreds of millions of images using a large infrastructure, thereby limiting their adoption.

In this work, we produce competitive convolution-free transformers by training on Imagenet only. We train them on a single computer in less than 3 days. Our reference vision transformer (86M parameters) achieves top-1 accuracy of 83.1% (single-crop) on ImageNet with no external data.

More importantly, we introduce a teacher-student strategy specific to transformers. It relies on a distillation token ensuring that the student learns from the teacher through attention. We show the interest of this token-based distillation, especially when using a convnet as a teacher. This leads us to report results competitive with convnets for both Imagenet (where we obtain up to 85.2% accuracy) and when transferring to other tasks. We share our code and models.

1 Introduction

Convolutional neural networks have been the main design paradigm for image understanding tasks, as initially demonstrated on image classification tasks. One of the ingredient to their success was the availability of a large training set, namely Imagenet [13, 42]. Motivated by the success of attention-based models in Natural Language Processing [14, 52], there has been increasing interest in architectures leveraging attention mechanisms within convnets [2, 34, 61]. More recently several researchers have proposed hybrid architecture transplanting transformer ingredients to convnets to solve vision tasks [6, 43].

The vision transformer (ViT) introduced by Dosovitskiy et al. [15] is an architecture directly inherited from Natural Language Processing [52], but ap-

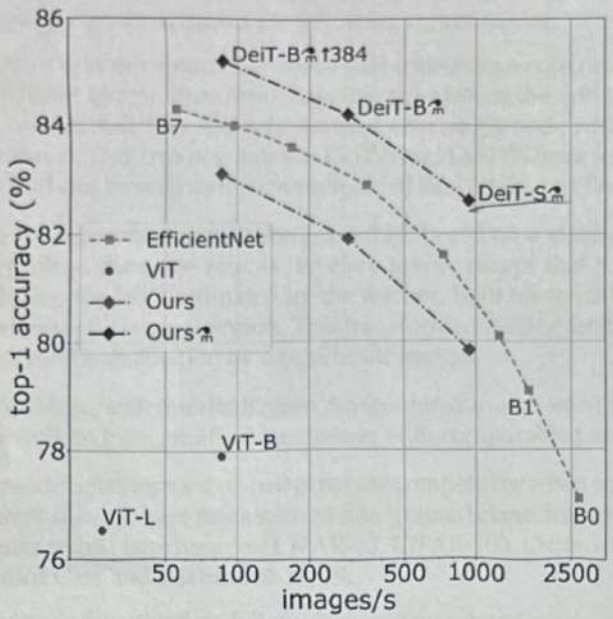

Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to VIT-B, but the training is more adapted to a data-starving regime. It is learned in a few days on one machine. The symbol ⚗ refers to models trained with our transformer-specific distillation. See Table 5 for details and more models.

plied to image classification with raw image patches as input. Their paper presented excellent results with transformers trained with a large private labelled image dataset (JFT-300M [46], 300 millions images). The paper concluded that transformers *"do not generalize well when trained on insufficient amounts of data"*, and the training of these models involved extensive computing resources.

In this paper, we train a vision transformer on a single 8-GPU node in two to three days (53 hours of pre-training, and optionally 20 hours of fine-tuning) that is competitive with convnets having a similar number of parameters and efficiency. It uses Imagenet as the sole training set. We build upon the visual transformer architecture from Dosovitskiy et al. [15] and improvements included in the timm library [55]. With our Data-efficient image Transformers (DeiT), we report large improvements over previous results, see Figure 1. Our ablation study details the hyper-parameters and key ingredients for a successful training, such as repeated augmentation.

We address another question: how to distill these models? We introduce a token-based strategy, specific to transformers and denoted by DeiT⚗, and show that it advantageously replaces the usual distillation.

In summary, our work makes the following contributions:

- We show that our neural networks that contains no convolutional layer can achieve competitive results against the state of the art on ImageNet with no external data. They are learned on a single node with 4 GPUs in three days[1]. Our two new models DeiT-S and DeiT-Ti have fewer parameters and can be seen as the counterpart of ResNet-50 and ResNet-18.
- We introduce a new distillation procedure based on a distillation token, which plays the same role as the class token, except that it aims at reproducing the label estimated by the teacher. Both tokens interact in the transformer through attention. This transformer-specific strategy outperforms vanilla distillation by a significant margin.
- Interestingly, with our distillation, image transformers learn more from a convnet than from another transformer with comparable performance.
- Our models pre-learned on Imagenet are competitive when transferred to different downstream tasks such as fine-grained classification, on several popular public benchmarks: CIFAR-10, CIFAR-100, Oxford-102 flowers, Stanford Cars and iNaturalist-18/19.

This paper is organized as follows: we review related works in Section 2, and focus on transformers for image classification in Section 3. We introduce our distillation strategy for transformers in Section 4. The experimental section 5 provides analysis and comparisons against both convnets and recent transformers, as well as a comparative evaluation of our transformer-specific distillation. Section 6 details our training scheme. It includes an extensive ablation of our data-efficient training choices, which gives some insight on the key ingredients involved in DeiT. We conclude in Section 7.

2 Related work

Image Classification is so core to computer vision that it is often used as a benchmark to measure progress in image understanding. Any progress usually translates to improvement in other related tasks such as detection or segmentation. Since 2012's AlexNet [32], convnets have dominated this benchmark and have become the de facto standard. The evolution of the state of the art on the ImageNet dataset [42] reflects the progress with convolutional neural network architectures and learning [32, 44, 48, 50, 51, 57].

Despite several attempts to use transformers for image classification [7], until now their performance has been inferior to that of convnets. Nevertheless hybrid architectures that combine convnets and transformers, including the self-attention mechanism, have recently exhibited competitive results in image classification [56], detection [6, 28], video processing [45, 53], unsupervised object discovery [35], and unified text-vision tasks [8, 33, 37].

[1] We can accelerate the learning of the larger model DeiT-B by training it on 8 GPUs in two days.

Recently Vision transformers (ViT) [15] closed the gap with the state of the art on ImageNet, without using any convolution. This performance is remarkable since convnet methods for image classification have benefited from years of tuning and optimization [22, 55]. Nevertheless, according to this study [15], a pre-training phase on a large volume of curated data is required for the learned transformer to be effective. In our paper we achieve a strong performance without requiring a large training dataset, i.e., with Imagenet1k only.

The Transformer architecture, introduced by Vaswani et al. [52] for machine translation are currently the reference model for all natural language processing (NLP) tasks. Many improvements of convnets for image classification are inspired by transformers. For example, Squeeze and Excitation [2], Selective Kernel [34] and Split-Attention Networks [61] exploit mechanism akin to transformers self-attention (SA) mechanism.

Knowledge Distillation (KD), introduced by Hinton et al. [24], refers to the training paradigm in which a *student* model leverages "soft" labels coming from a strong *teacher* network. This is the output vector of the teacher's softmax function rather than just the maximum of scores, wich gives a "hard" label. Such a training improves the performance of the student model (alternatively, it can be regarded as a form of compression of the teacher model into a smaller one – the student). On the one hand the teacher's soft labels will have a similar effect to labels smoothing [58]. On the other hand as shown by Wei et al. [54] the teacher's supervision takes into account the effects of the data augmentation, which sometimes causes a misalignment between the real label and the image. For example, let us consider image with a "cat" label that represents a large landscape and a small cat in a corner. If the cat is no longer on the crop of the data augmentation it implicitly changes the label of the image. KD can transfer inductive biases [1] in a soft way in a student model using a teacher model where they would be incorporated in a hard way. For example, it may be useful to induce biases due to convolutions in a transformer model by using a convolutional model as teacher. In our paper we study the distillation of a transformer student by either a convnet or a transformer teacher. We introduce a new distillation procedure specific to transformers and show its superiority.

3 Vision transformer: overview

In this section, we briefly recall preliminaries associated with the vision transformer [15, 52], and further discuss positional encoding and resolution.

Multi-head Self Attention layers (MSA). The attention mechanism is based on a trainable associative memory with (key, value) vector pairs. A *query* vector $q \in \mathbb{R}^d$ is matched against a set of k *key* vectors (packed together into a matrix $K \in \mathbb{R}^{k \times d}$) using inner products. These inner products are then scaled and

normalized with a softmax function to obtain k weights. The output of the attention is the weighted sum of a set of k value vectors (packed into $V \in \mathbb{R}^{k \times d}$). For a sequence of N query vectors (packed into $Q \in \mathbb{R}^{N \times d}$), it produces an output matrix (of size $N \times d$):

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^\top / \sqrt{d})V. \quad (1)$$

where the Softmax function is applied over each row of the input matrix and the $\sqrt{d}$ term provides appropriate normalization.

In [52], a Self-attention layer is proposed. Query, key and values matrices are themselves computed from a sequence of N input vectors (packed into $X \in \mathbb{R}^{N \times D}$): $Q = XW_Q$, $K = XW_K$, $V = XW_V$, using linear transformations W_Q, W_K, W_V with the constraint $k = N$, meaning that the attention is in between all the input vectors.

Finally, Multi-head self-attention layer (MSA) is defined by considering h attention "heads", *ie* h self-attention functions applied to the input. Each head provides a sequence of size $N \times d$. These h sequences are rearranged into a $N \times dh$ sequence that is reprojected by a linear layer into $N \times D$.

Transformer block for images. To get a full transformer block as in [52], we add a Feed-Forward Network (FFN) on top of the MSA layer. This FFN is composed of two linear layers separated by a GeLu activation [23]. The first linear layer expands the dimension from D to $4D$, and the second layer reduces the dimension from $4D$ back to D. Both MSA and FFN are operating as residual operators thank to skip-connections, and with a layer normalization [3].

In order to get a transformer to process images, our work builds upon the ViT model [15]. It is a simple and elegant architecture that processes input images as if they were a sequence of input tokens. The fixed-size input RGB image is decomposed into a batch of N patches of a fixed size of 16×16 pixels ($N = 14 \times 14$). Each patch is projected with a linear layer that conserves its overall dimension $3 \times 16 \times 16 = 768$.

The transformer block described above is invariant to the order of the patch embeddings, and thus does not consider their relative position. The positional information is incorporated as fixed [52] or trainable [18] positional embeddings. They are added before the first transformer block to the patch tokens, which are then fed to the stack of transformer blocks.

The class token is a trainable vector, appended to the patch tokens before the first layer, that goes through the transformer layers, and is then projected with a linear layer to predict the class. This class token is inherited from NLP [14], and departs from the typical pooling layers used in computer vision to predict the class. The transformer thus process batches of $(N+1)$ tokens of dimension D, of which only the class vector is used to predict the output. This architecture forces the self-attention to spread information between the patch tokens and the class token: at training time the supervision signal comes only from the class embedding, while the patch tokens are the model's only variable input.

Fixing the positional encoding across resolutions. Touvron et al. [50] show that it is desirable to use a lower training resolution and fine-tune the network at the larger resolution. This speeds up the full training and improves the accuracy under prevailing data augmentation schemes. When increasing the resolution of an input image, we keep the patch size the same, therefore the number N of input patches does change. Due to the architecture of transformer blocks and the class token, the model and classifier do not need to be modified to process more tokens. In contrast, one needs to adapt the positional embeddings, because there are N of them, one for each patch. Dosovitskiy et al. [15] interpolate the positional encoding when changing the resolution and demonstrate that this method works with the subsequent fine-tuning stage.

4 Distillation through attention

In this section we assume we have access to a strong image classifier as a teacher model. It could be a convnet, or a mixture of classifiers. We address the question of how to learn a transformer by exploiting this teacher. As we will see in Section 5 by comparing the trade-off between accuracy and image throughput, it can be beneficial to replace a convolutional neural network by a transformer. This section covers two axes of distillation: hard distillation versus soft distillation, and classical distillation versus the distillation token.

Soft distillation [24, 54] minimizes the Kullback-Leibler divergence between the softmax of the teacher and the softmax of the student model.

Let Z_t be the logits of the teacher model, Z_s the logits of the student model. We denote by τ the temperature for the distillation, λ the coefficient balancing the Kullback–Leibler divergence loss (KL) and the cross-entropy ($\mathcal{L}_{\text{CE}}$) on ground truth labels y, and ψ the softmax function. The distillation objective is

$$\mathcal{L}_{\text{global}} = (1-\lambda)\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \lambda\tau^2 \text{KL}(\psi(Z_s/\tau), \psi(Z_t/\tau)). \quad (2)$$

Hard-label distillation. We introduce a variant of distillation where we take the hard decision of the teacher as a true label. Let $y_t = \text{argmax}_c Z_t(c)$ be the hard decision of the teacher, the objective associated with this hard-label distillation is:

$$\mathcal{L}^{\text{hardDistill}}_{\text{global}} = \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_{\text{CE}}(\psi(Z_s), y_t). \quad (3)$$

For a given image, the hard label associated with the teacher may change depending on the specific data augmentation. We will see that this choice is better than the traditional one, while being parameter-free and conceptually simpler: The teacher prediction y_t plays the same role as the true label y.

Note also that the hard labels can also be converted into soft labels with label smoothing [47], where the true label is considered to have a probability of $1-\varepsilon$, and the remaining ε is shared across the remaining classes. We fix this parameter to $\varepsilon = 0.1$ in our all experiments that use true labels.

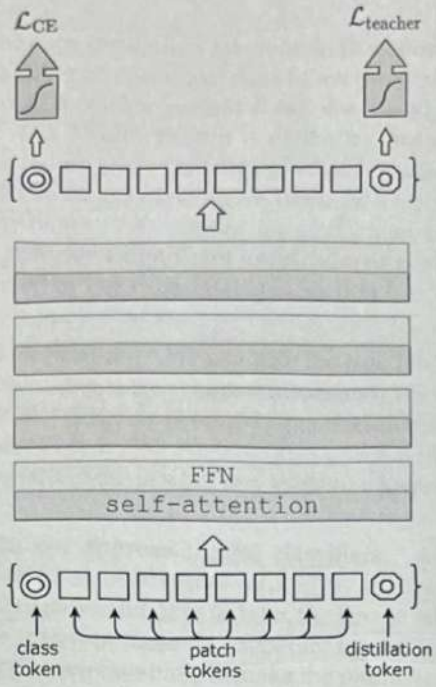

Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce the (hard) label predicted by the teacher, instead of true label. Both the class and distillation tokens input to the transformers are learned by back-propagation.

Distillation token. We now focus on our proposal, which is illustrated in Figure 2. We add a new token, the distillation token, to the initial embeddings (patches and class token). Our distillation token is used similarly as the class token: it interacts with other embeddings through self-attention, and is output by the network after the last layer. Its target objective is given by the distillation component of the loss. The distillation embedding allows our model to learn from the output of the teacher, as in a regular distillation, while remaining complementary to the class embedding.

Interestingly, we observe that the learned class and distillation tokens converge towards different vectors: the average cosine similarity between these tokens equal to 0.06. As the class and distillation embeddings are computed at each layer, they gradually become more similar through the network, all the way through the last layer at which their similarity is high (cos=0.93), but still lower than 1. This is expected since as they aim at producing targets that are similar but not identical.

We verified that our distillation token adds something to the model, compared to simply adding an additional class token associated with the same target label: instead of a teacher pseudo-label, we experimented with a transformer with two class tokens. Even if we initialize them randomly and independently, during training they converge towards the same vector (cos=0.999), and the output embedding are also quasi-identical. This additional class token does not bring anything to the classification performance. In contrast, our distillation strategy provides a significant improvement over a vanilla distillation baseline, as validated by our experiments in Section 5.2.

Fine-tuning with distillation. We use both the true label and teacher prediction during the fine-tuning stage at higher resolution. We use a teacher with the same target resolution, typically obtained from the lower-resolution teacher by the method of Touvron et al [50]. We have also tested with true labels only but this reduces the benefit of the teacher and leads to a lower performance.

Classification with our approach: joint classifiers. At test time, both the class or the distillation embeddings produced by the transformer are associated with linear classifiers and able to infer the image label. Yet our referent method is the late fusion of these two separate heads, for which we add the softmax output by the two classifiers to make the prediction. We evaluate these three options in Section 5.

5 Experiments

This section presents a few analytical experiments and results. We first discuss our distillation strategy. Then we comparatively analyze the efficiency and accuracy of convnets and vision transformers.

5.1 Transformer models

As mentioned earlier, our architecture design is identical to the one proposed by Dosovitskiy et al. [15] with no convolutions. Our only differences are the training strategies, and the distillation token. Also we do not use a MLP head for the pre-training but only a linear classifier. To avoid any confusion, we refer to the results obtained in the prior work by ViT, and prefix ours by DeiT. If not specified, DeiT refers to our referent model DeiT-B, which has the same architecture as ViT-B. When we fine-tune DeiT at a larger resolution, we append the resulting operating resolution at the end, e.g, DeiT-B↑384. Last, when using our distillation procedure, we identify it with an alembic sign as DeiT⚗.

The parameters of ViT-B (and therefore of DeiT-B) are fixed as $D = 768$, $h = 12$ and $d = D/h = 64$. We introduce two smaller models, namely DeiT-S and DeiT-Ti, for which we change the number of heads, keeping d fixed. Table 1 summarizes the models that we consider in our paper.

Table 1: Variants of our DeiT architecture. The larger model, DeiT-B, has the same architecture as the ViT-B [15]. The only parameters that vary across models are the embedding dimension and the number of heads, and we keep the dimension per head constant (equal to 64). Smaller models have a lower parameter count, and a faster throughput. The throughput is measured for images at resolution 224×224.

Model	ViT model	embedding dimension	#heads	#layers	#params	training resolution	throughput (im/sec)
DeiT-Ti	N/A	192	3	12	5M	224	2536
DeiT-S	N/A	384	6	12	22M	224	940
DeiT-B	ViT-B	768	12	12	86M	224	292

Table 2: We compare on ImageNet [42] the performance (top-1 acc., %) of the student as a function of the teacher model used for distillation.

Teacher Models	acc.	Student: DeiT-B ⚗ pretrain	↑384
DeiT-B	81.8	81.9	83.1
RegNetY-4GF	80.0	82.7	83.6
RegNetY-8GF	81.7	82.7	83.8
RegNetY-12GF	82.4	83.1	84.1
RegNetY-16GF	82.9	83.1	84.2

5.2 Distillation

Our distillation method produces a vision transformer that becomes on par with the best convnets in terms of the trade-off between accuracy and throughput, see Table 5. Interestingly, the distilled model outperforms its teacher in terms of the trade-off between accuracy and throughput. Our best model on ImageNet-1k is 85.2% top-1 accuracy outperforms the best Vit-B model pretrained on JFT-300M at resolution 384 (84.15%). For reference, the current state of the art of 88.55% achieved with extra training data was obtained by the ViT-H model (600M parameters) trained on JFT-300M at resolution 512. Hereafter we provide several analysis and observations.

Convnets teachers. We have observed that using a convnet teacher gives better performance than using a transformer. Table 2 compares distillation results with different teacher architectures. The fact that the convnet is a better teacher is probably due to the inductive bias inherited by the transformers through distillation, as explained in Abnar et al. [1]. In all of our subsequent distillation experiments the default teacher is a RegNetY-16GF [40] (84M parameters) that we trained with the same data and same data-augmentation as DeiT. This teacher reaches 82.9% top-1 accuracy on ImageNet.

Table 3: Distillation experiments on Imagenet with DeiT, 300 epochs of pre-training. We report the results for our new distillation method in the last three rows. We separately report the performance when classifying with only one of the class or distillation embeddings, and then with a classifier taking both of them as input. In the last row (class+distillation), the result correspond to the late fusion of the class and distillation classifiers.

method ↓	Supervision label	Supervision teacher	ImageNet top-1 (%) Ti 224	S 224	B 224	B↑384
DeiT– no distillation	✓	✗	72.2	79.8	81.8	83.1
DeiT– usual distillation	✗	soft	72.2	79.8	81.8	83.2
DeiT– hard distillation	✗	hard	74.3	80.9	83.0	84.0
DeiT⚗: class embedding	✓	hard	73.9	80.9	83.0	84.2
DeiT⚗: distil. embedding	✓	hard	74.6	81.1	83.1	84.4
DeiT⚗: class+distillation	✓	hard	74.5	81.2	83.4	84.5

Comparison of distillation methods. We compare the performance of different distillation strategies in Table 3. Hard distillation significantly outperforms soft distillation for transformers, even when using only a class token: hard distillation reaches 83.0% at resolution 224×224, compared to the soft distillation accuracy of 81.8%. Our distillation strategy from Section 4 further improves the performance, showing that the two tokens provide complementary information useful for classification: the classifier on the two tokens is significantly better than the independent class and distillation classifiers, which by themselves already outperform the distillation baseline.

The distillation token gives slightly better results than the class token. It is also more correlated to the convnets prediction. This difference in performance is probably due to the fact that it benefits more from the inductive bias of convnets. We give more details and an analysis in the next paragraph. The distillation token has an undeniable advantage for the initial training.

Agreement with the teacher & inductive bias? As discussed above, the architecture of the teacher has an important impact. Does it inherit existing inductive bias that would facilitate the training? While we believe it difficult to formally answer this question, we analyze in Table 4 the decision agreement between the convnet teacher, our image transformer DeiT learned from labels only, and our transformer DeiT⚗.

Our distilled model is more correlated to the convnet than with a transformer learned from scratch. As to be expected, the classifier associated with the distillation embedding is closer to the convnet that the one associated with the class embedding, and conversely the one associated with the class embedding is more similar to DeiT learned without distillation. Unsurprisingly, the joint class+distil classifier offers a middle ground.

Table 4: Disagreement analysis between convnet, image transformers and distillated transformers: We report the fraction of sample classified differently for all classifier pairs, i.e., the rate of different decisions. We include two models without distillation (a RegNetY and DeiT-B), so that we can compare how our distilled models and classification heads are correlated to these teachers.

	groundtruth	no distillation		DeiT⚗ student (of the convnet)		
		convnet	DeiT	class	distillation	DeiT⚗
groundtruth	0.000	0.171	0.182	0.170	0.169	0.166
convnet (RegNetY)	0.171	0.000	0.133	0.112	0.100	0.102
DeiT	0.182	0.133	0.000	0.109	0.110	0.107
DeiT⚗– class only	0.170	0.112	0.109	0.000	0.050	0.033
DeiT⚗– distil. only	0.169	0.100	0.110	0.050	0.000	0.019
DeiT⚗– class+distil.	0.166	0.102	0.107	0.033	0.019	0.000

Number of epochs. Increasing the number of epochs significantly improves the performance of training with distillation, see Figure 3. With 300 epochs, our distilled network DeiT-B⚗ is already better than DeiT-B. But while for the latter the performance saturates with longer schedules, our distilled network clearly benefits from a longer training time.

5.3 Efficiency vs accuracy: a comparative study with convnets

In the literature, the image classificaton methods are often compared as a compromise between accuracy and another criterion, such as FLOPs, number of parameters, size of the network, etc.

We focus in Figure 1 on the tradeoff between the throughput (images processed per second) and the top-1 classification accuracy on ImageNet. We focus on the popular state-of-the-art EfficientNet convnet, which has benefited from years of research on convnets and was optimized by architecture search on the ImageNet validation set.

Our method DeiT is slightly below EfficientNet, which shows that we have almost closed the gap between vision transformers and convnets when training with Imagenet only. These results are a major improvement (**+6.3%** top-1 in a comparable setting) over previous ViT models trained on Imagenet1k only [15]. Furthermore, when DeiT benefits from the distillation from a relatively weaker RegNetY to produce DeiT⚗, it outperforms EfficientNet. It also outperforms by 1% (top-1 acc.) the Vit-B model pre-trained on JFT300M at resolution 384 (85.2% vs 84.15%), while being significantly faster to train.

Table 5 reports the numerical results in more details and additional evaluations on ImageNet V2 and ImageNet Real, that have a test set distinct from the ImageNet validation, which reduces overfitting on the validation set. Our results show that DeiT-B⚗ and DeiT-B⚗ ↑384 outperform, by some margin, the state of the art on the trade-off between accuracy and inference time on GPU.

Network	#param.	image size	throughput (image/s)	ImNet top-1	Real top-1	V2 top-1
Convnets						
ResNet-18 [21]	12M	224^2	4458.4	69.8	77.3	57.1
ResNet-50 [21]	25M	224^2	1226.1	76.2	82.5	63.3
ResNet-101 [21]	45M	224^2	753.6	77.4	83.7	65.7
ResNet-152 [21]	60M	224^2	526.4	78.3	84.1	67.0
RegNetY-4GF [40]⋆	21M	224^2	1156.7	80.0	86.4	69.4
RegNetY-8GF [40]⋆	39M	224^2	591.6	81.7	87.4	70.8
RegNetY-16GF [40]⋆	84M	224^2	334.7	82.9	88.1	72.4
EfficientNet-B0 [48]	5M	224^2	2694.3	77.1	83.5	64.3
EfficientNet-B1 [48]	8M	240^2	1662.5	79.1	84.9	66.9
EfficientNet-B2 [48]	9M	260^2	1255.7	80.1	85.9	68.8
EfficientNet-B3 [48]	12M	300^2	732.1	81.6	86.8	70.6
EfficientNet-B4 [48]	19M	380^2	349.4	82.9	88.0	72.3
EfficientNet-B5 [48]	30M	456^2	169.1	83.6	88.3	73.6
EfficientNet-B6 [48]	43M	528^2	96.9	84.0	88.8	73.9
EfficientNet-B7 [48]	66M	600^2	55.1	84.3	-	-
EfficientNet-B5 RA [12]	30M	456^2	96.9	83.7	-	-
EfficientNet-B7 RA [12]	66M	600^2	55.1	84.7	-	-
KDforAA-B8	87M	800^2	25.2	85.8	-	-
Transformers						
ViT-B/16 [15]	86M	384^2	85.9	77.9	83.6	-
ViT-L/16 [15]	307M	384^2	27.3	76.5	82.2	-
DeiT-Ti	5M	224^2	2536.5	72.2	80.1	60.4
DeiT-S	22M	224^2	940.4	79.8	85.7	68.5
DeiT-B	86M	224^2	292.3	81.8	86.7	71.5
DeiT-B↑384	86M	384^2	85.9	83.1	87.7	72.4
DeiT-Ti⚗	6M	224^2	2529.5	74.5	82.1	62.9
DeiT-S⚗	22M	224^2	936.2	81.2	86.8	70.0
DeiT-B⚗	87M	224^2	290.9	83.4	88.3	73.2
DeiT-Ti⚗ / 1000 epochs	6M	224^2	2529.5	76.6	83.9	65.4
DeiT-S⚗ / 1000 epochs	22M	224^2	936.2	82.6	87.8	71.7
DeiT-B⚗ / 1000 epochs	87M	224^2	290.9	84.2	88.7	73.9
DeiT-B⚗ ↑384	87M	384^2	85.8	84.5	89.0	74.8
DeiT-B⚗ ↑384 / 1000 epochs	87M	384^2	85.8	85.2	89.3	75.2

nice table

Table 5: Throughput on and accuracy on Imagenet [42], Imagenet Real [5] and Imagenet V2 matched frequency [41] of DeiT and of several state-of-the-art convnets, for models trained with no external data. The throughput is measured as the number of images that we can process per second on one 16GB V100 GPU. For each model we take the largest possible batch size for the usual resolution of the model and calculate the average time over 30 runs to process that batch. With that we calculate the number of images processed per second. Throughput can vary according to the implementation: for a direct comparison and in order to be as fair as possible, we use for each model the definition in the same GitHub [55] repository.

⋆ : *Regnet optimized with a similar optimization procedure as ours, which boosts the results. These networks serve as teachers when we use our distillation strategy.*

Regnet?

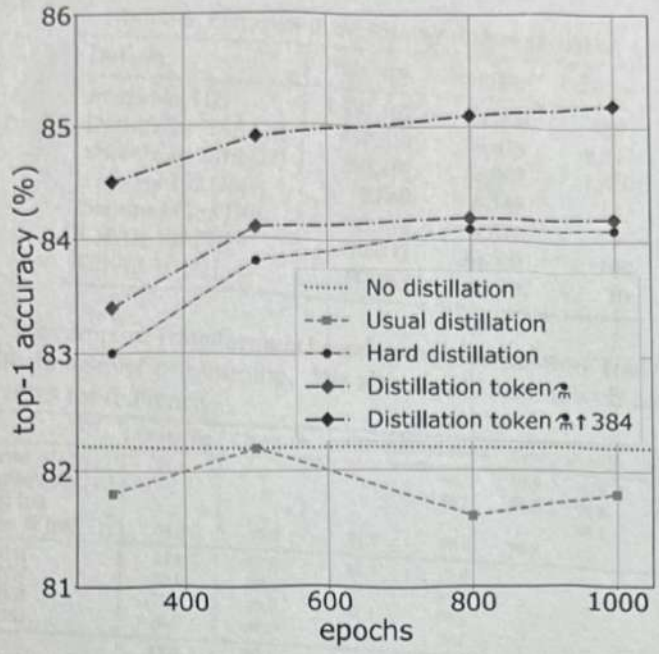

Figure 3: Distillation on ImageNet [42] with DeiT-B: performance as a function of the number of training epochs. We provide the performance without distillation (horizontal dotted line) as it saturates after 400 epochs.

5.4 Transfer learning: Performance on downstream tasks

Although DeiT perform very well on ImageNet it is important to evaluate them on other datasets with transfer learning in order to measure the power of generalization of DeiT. We evaluated this on transfer learning tasks by fine-tuning on the datasets in Table 6. Table 7 compares DeiT transfer learning results to those of ViT [15] and state of the art convolutional architectures [48]. DeiT is on par with competitive convnet models, which is in line with our previous conclusion on ImageNet.

Comparison vs training from scratch. We investigate the performance when training from scratch on a small dataset, without Imagenet pre-training. We get the following results on the small CIFAR-10, which is small both w.r.t. the number of images and labels:

Method	RegNetY-16GF	DeiT-B	DeiT-B⚗
Top-1	98.0	97.5	98.5

For this experiment, we tried we get as close as possible to the Imagenet pre-training counterpart, meaning that (1) we consider longer training sched-

Table 6: Datasets used for our different tasks.

Dataset	Train size	Test size	#classes
ImageNet [42]	1,281,167	50,000	1000
iNaturalist 2018 [26]	437,513	24,426	8,142
iNaturalist 2019 [27]	265,240	3,003	1,010
Flowers-102 [38]	2,040	6,149	102
Stanford Cars [30]	8,144	8,041	196
CIFAR-100 [31]	50,000	10,000	100
CIFAR-10 [31]	50,000	10,000	10

Table 7: We compare Transformers based models on different transfer learning task with ImageNet pre-training. We also report results with convolutional architectures for reference.

Model	ImageNet	CIFAR-10	CIFAR-100	Flowers	Cars	iNat-18	iNat-19	im/sec
Grafit ResNet-50 [49]	79.6	-	-	98.2	92.5	69.8	75.9	1226.1
Grafit RegNetY-8GF [49]	-	-	-	99.0	94.0	76.8	80.0	591.6
ResNet-152 [10]	-	-	-	-	-	69.1	-	526.3
EfficientNet-B7 [48]	84.3	98.9	91.7	98.8	94.7	-	-	55.1
ViT-B/32 [15]	73.4	97.8	86.3	85.4	-	-	-	394.5
ViT-B/16 [15]	77.9	98.1	87.1	89.5	-	-	-	85.9
ViT-L/32 [15]	71.2	97.9	87.1	86.4	-	-	-	124.1
ViT-L/16 [15]	76.5	97.9	86.4	89.7	-	-	-	27.3
DeiT-B	81.8	99.1	90.8	98.4	92.1	73.2	77.7	292.3
DeiT-B↑384	83.1	99.1	90.8	98.5	93.3	79.5	81.4	85.9
DeiT-B⚗	83.4	99.1	91.3	98.8	92.9	73.7	78.4	290.9
DeiT-B⚗↑384	84.4	99.2	91.4	98.9	93.9	80.1	83.0	85.9

ules (up to 7200 epochs, which corresponds to 300 Imagenet epochs) so that the network has been fed a comparable number of images in total; (2) we re-scale images to 224×224 to ensure that we have the same augmentation. The results are not as good as with Imagenet pre-training (98.5% vs 99.1%), which is expected since the network has seen a much lower diversity. However they show that it is possible to learn a reasonable transformer on CIFAR-10 only.

6 Training details & ablation

In this section we discuss the DeiT training strategy to learn vision transformers in a data-efficient manner. We build upon PyTorch [39] and the timm library [55][2]. We provide hyper-parameters as well as an ablation study in which we analyze the impact of each choice.

Initialization and hyper-parameters. Transformers are relatively sensitive to initialization. After testing several options in preliminary experiments, some

[2]The timm implementation already included a training procedure that improved the accuracy of ViT-B from 77.91% to 79.35% top-1, and trained on Imagenet-1k with a 8xV100 GPU machine.

Ablation on ↓	Pre-training	Fine-tuning	Rand-Augment	AutoAug	Mixup	CutMix	Erasing	Stoch. Depth	Repeated Aug.	Dropout	Exp. Moving Avg.	top-1 accuracy: pre-trained 224^2	top-1 accuracy: fine-tuned 384^2
none: DeiT-B	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8 ±0.2	83.1 ±0.1
optimizer	SGD	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✗	74.5	77.3
	adamw	SGD	✓	✗	✓	✓	✓	✓	✓	✗	✗	81.8	83.1
data augmentation	adamw	adamw	✗	✗	✓	✓	✓	✓	✓	✗	✗	79.6	80.4
	adamw	adamw	✗	✓	✓	✓	✓	✓	✓	✗	✗	81.2	81.9
	adamw	adamw	✓	✗	✗	✓	✓	✓	✓	✗	✗	78.7	79.8
	adamw	adamw	✓	✗	✓	✗	✓	✓	✓	✗	✗	80.0	80.6
	adamw	adamw	✓	✗	✗	✗	✓	✓	✓	✗	✗	75.8	76.7
regularization	adamw	adamw	✓	✗	✓	✓	✗	✓	✓	✗	✗	4.3*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✗	✓	✗	✗	3.4*	0.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✗	✗	✗	76.5	77.4
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✓	✗	81.3	83.1
	adamw	adamw	✓	✗	✓	✓	✓	✓	✓	✗	✓	81.9	83.1

Table 8: Ablation study on training methods on ImageNet [42]. The top row ("none") corresponds to our default configuration employed for DeiT. The symbols ✓ and ✗ indicates that we use and do not use the corresponding method, respectively. We report the accuracy scores (%) after the initial training at resolution 224×224, and after fine-tuning at resolution 384×384. The hyper-parameters are fixed according to Table 9, and may be suboptimal.
* indicates that the model did not train well, possibly because hyper-parameters are not adapted.

of them not converging, we follow the recommendation of Hanin and Rolnick [20] to initialize the weights with a truncated normal distribution.

Table 9 indicates the hyper-parameters that we use by default at training time for all our experiments, unless stated otherwise. For distillation we follow the recommendations from Cho et al. [9] to select the parameters τ and λ. We take the typical values $\tau = 3.0$ and $\lambda = 0.1$ for the usual (soft) distillation.

Data-Augmentation. Compared to models that integrate more priors (such as convolutions), transformers require a larger amount of data. Thus, in order to train with datasets of the same size, we rely on extensive data augmentation. We evaluate different types of strong data augmentation, with the objective to reach a data-efficient training regime.

Auto-Augment [11], Rand-Augment [12], and random erasing [62] improve the results. For the two latter we use the timm [55] customizations, and after ablation we choose Rand-Augment instead of AutoAugment. Overall our experiments confirm that transformers require a strong data augmentation: almost all the data-augmentation methods that we evaluate prove to be useful. One exception is dropout, which we exclude from our training procedure.

Methods	ViT-B [15]	DeiT-B
Epochs	300	300
Batch size	4096	1024
Optimizer	AdamW	AdamW
learning rate	0.003	$0.0005 \times \frac{\text{batchsize}}{512}$
Learning rate decay	cosine	cosine
Weight decay	0.3	0.05
Warmup epochs	3.4	5
Label smoothing ε	✗	0.1
Dropout	0.1	✗
Stoch. Depth	✗	0.1
Repeated Aug	✗	✓
Gradient Clip.	✓	✗
Rand Augment	✗	9/0.5
Mixup prob.	✗	0.8
Cutmix prob.	✗	1.0
Erasing prob.	✗	0.25

Table 9: Ingredients and hyper-parameters for our method and Vit-B.

Regularization & Optimizers. We have considered different optimizers and cross-validated different learning rates and weight decays. Transformers are sensitive to the setting of optimization hyper-parameters. Therefore, during cross-validation, we tried 3 different learning rates ($5.10^{-4}, 3.10^{-4}, 5.10^{-5}$) and 3 weight decay (0.03, 0.04, 0.05). We scale the learning rate according to the batch size with the formula: $\text{lr}_{\text{scaled}} = \frac{\text{lr}}{512} \times \text{batchsize}$, similarly to Goyal et al. [19] except that we use 512 instead of 256 as the base value.

The best results use the AdamW optimizer with the same learning rates as ViT [15] but with a much smaller weight decay, as the weight decay reported in the paper hurts the convergence in our setting.

We have employed stochastic depth [29], which facilitates the convergence of transformers, especially deep ones [16, 17]. For vision transformers, they were first adopted in the training procedure by Wightman [55]. Regularization like Mixup [60] and Cutmix [59] improve performance. We also use repeated augmentation [4, 25], which provides a significant boost in performance and is one of the key ingredients of our proposed training procedure.

Exponential Moving Average (EMA). We evaluate the EMA of our network obtained after training. There are small gains, which vanish after fine-tuning: the EMA model has an edge of is 0.1 accuracy points, but when fine-tuned the two models reach the same (improved) performance.

Fine-tuning at different resolution. We adopt the fine-tuning procedure from Touvron et al. [51]: our schedule, regularization and optimization procedure are identical to that of FixEfficientNet but we keep the training-time data aug-

image size	throughput (image/s)	Imagenet [42] acc. top-1	Real [5] acc. top-1	V2 [41] acc. top-1
160^2	609.31	79.9	84.8	67.6
224^2	291.05	81.8	86.7	71.5
320^2	134.13	82.7	87.2	71.9
384^2	85.87	83.1	87.7	72.4

Table 10: Performance of DeiT trained at size 224^2 for varying finetuning sizes on ImageNet-1k, ImageNet-Real and ImageNet-v2 matched frequency.

mentation (contrary to the dampened data augmentation of Touvron et al. [51]). We also interpolate the positional embeddings: In principle any classical image scaling technique, like bilinear interpolation, could be used. However, a bilinear interpolation of a vector from its neighbors reduces its ℓ_2-norm compared to its neighbors. These low-norm vectors are not adapted to the pre-trained transformers and we observe a significant drop in accuracy if we employ use directly without any form of fine-tuning. Therefore we adopt a bicubic interpolation that approximately preserves the norm of the vectors, before fine-tuning the network with either AdamW [36] or SGD. These optimizers have a similar performance for the fine-tuning stage, see Table 8.

By default and similar to ViT [15] we train DeiT models with at resolution 224 and we fine-tune at resolution 384. We detail how to do this interpolation in Section 3. However, in order to measure the influence of the resolution we have finetuned DeiT at different resolutions. We report these results in Table 10.

Training time. A typical training of 300 epochs takes 37 hours with 2 nodes or 53 hours on a single node for the DeiT-B.As a comparison point, a similar training with a RegNetY-16GF [40] (84M parameters) is 20% slower. DeiT-S and DeiT-Ti are trained in less than 3 days on 4 GPU. Then, optionally we fine-tune the model at a larger resolution. This takes 20 hours on a single node (8 GPU) to produce a FixDeiT-B model at resolution 384×384, which corresponds to 25 epochs. Not having to rely on batch-norm allows one to reduce the batch size without impacting performance, which makes it easier to train larger models. Note that, since we use repeated augmentation [4, 25] with 3 repetitions, we only see one third of the images during a single epoch[3].

7 Conclusion

In this paper, we have introduced DeiT, which are image transformers that do not require very large amount of data to be trained, thanks to improved

[3]Formally it means that we have 100 epochs, but each is 3x longer because of the repeated augmentations. We prefer to refer to this as 300 epochs in order to have a direct comparison on the effective training time with and without repeated augmentation.