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-

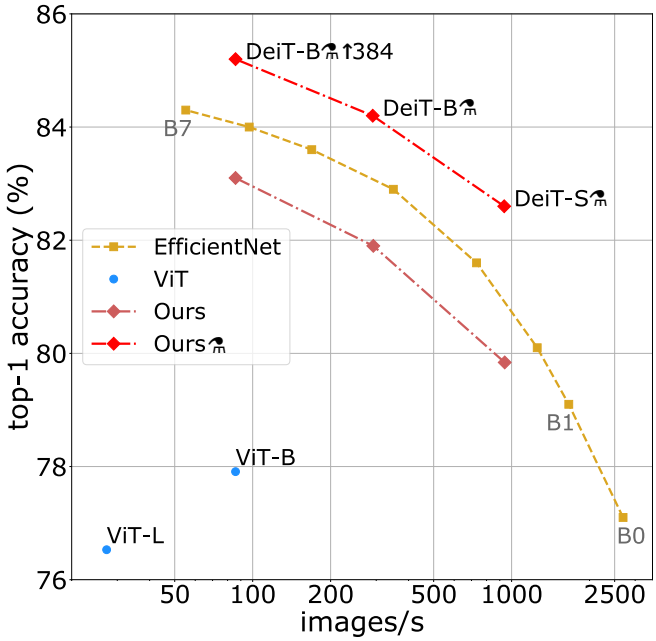

Figure 1: Throughput and accuracy on Imagenet of our methods compared to EfficientNets, trained on Imagenet1k only. The throughput is measured as the number of images processed per second on a V100 GPU. DeiT-B is identical to VIT-B, but the training is more adapted to a data-starving regime. It is learned 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 $N \times d$):

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

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 $X \in \mathbb{R}^{N\times D}$): $Q = XW_\text{Q}$, $K = XW_\text{K}$, $V = XW_\text{V}$, using linear transformations $W_\text{Q}, W_\text{K}, W_\text{V}$ 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 ($\mathcal{L}_\mathrm{CE}$) on ground truth labels y, and ψ the softmax function. The distillation objective is

$$\mathcal{L}_\mathrm{global} = (1-\lambda)\mathcal{L}_\mathrm{CE}(\psi(Z_\mathrm{s}), y) + \lambda\tau^2\mathrm{KL}(\psi(Z_\mathrm{s}/\tau), \psi(Z_\mathrm{t}/\tau)). \tag{2}$$

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

$$\mathcal{L}^\mathrm{hardDistill}_\mathrm{global} = \frac{1}{2}\mathcal{L}_\mathrm{CE}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_\mathrm{CE}(\psi(Z_s), y_\mathrm{t}). \tag{3}$$

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.

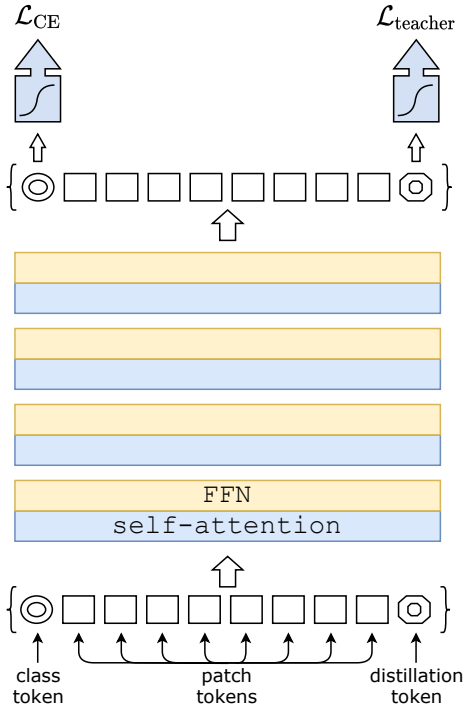

Figure 2: Our distillation procedure: we simply include a new *distillation token*. It interacts with the class and patch tokens through the self-attention layers. This distillation token is employed in a similar fashion as the class token, except that on output of the network its objective is to reproduce 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 pretraining. 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.

	Supervision		ImageNet top-1 (%)			
method ↓	label	teacher	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

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.*

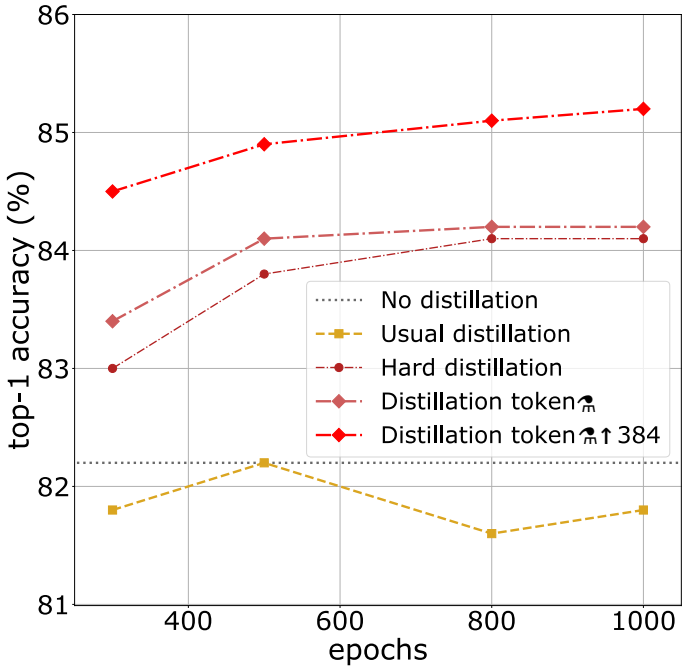

Figure 3: Distillation on ImageNet [42] with DeiT-B: performance as a function of the number of training epochs. We provide the performance without distillation (horizontal dotted line) as it saturates after 400 epochs.

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.

training and in particular a novel distillation procedure. Convolutional neural networks have optimized, both in terms of architecture and optimization during almost a decade, including through extensive architecture search that is prone to overfiting, as it is the case for instance for EfficientNets [51]. For DeiT we have started the existing data augmentation and regularization strategies pre-existing for convnets, not introducing any significant architectural beyond our novel distillation token. Therefore it is likely that research on data-augmentation more adapted or learned for transformers will bring further gains.

Therefore, considering our results, where image transformers are on par with convnets already, we believe that they will rapidly become a method of choice considering their lower memory footprint for a given accuracy.

We provide an open-source implementation of our method. It is available at `https://github.com/facebookresearch/deit`.

Acknowledgements

Many thanks to Ross Wightman for sharing his ViT code and bootstrapping training method with the community, as well as for valuable feedback that helped us to fix different aspects of this paper. Thanks to Vinicius Reis, Mannat Singh, Ari Morcos, Mark Tygert, Gabriel Synnaeve, and other colleagues at Facebook for brainstorming and some exploration on this axis. Thanks to Ross Girshick and Piotr Dollar for constructive comments.

References

[1] Samira Abnar, Mostafa Dehghani, and Willem Zuidema. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*, 2020.

[2] Jie Hu andLi Shen and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

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

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

[5] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aaron van den Oord. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*, 2020.

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

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

[8] Yen-Chun Chen, Linjie Li, Licheng Yu, A. E. Kholy, Faisal Ahmed, Zhe Gan, Y. Cheng, and Jing jing Liu. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, 2020.

[9] J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. *International Conference on Computer Vision*, 2019.

[10] P. Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. *arXiv preprint arXiv:2008.03673*, 2020.

[11] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[12] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. *arXiv preprint arXiv:1909.13719*, 2019.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

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

[16] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019. ICLR 2020.

[17] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.

[18] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[19] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[20] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. *NIPS*, 31, 2018.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, June 2016.

[22] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

[25] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Conference on Computer Vision and Pattern Recognition*, 2020.

[26] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist challenge 2018 dataset. *arXiv preprint arXiv:1707.06642*, 2018.

[27] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist challenge 2019 dataset. *arXiv preprint arXiv:1707.06642*, 2019.

[28] H. Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Y. Wei. Relation networks for object detection. *Conference on Computer Vision and Pattern Recognition*, 2018.

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

[30] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.

[31] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, CIFAR, 2009.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[33] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: a simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[34] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. *Conference on Computer Vision and Pattern Recognition*, 2019.

[35] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, A. Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[36] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[37] Jiasen Lu, Dhruv Batra, D. Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NIPS*, 2019.

[38] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[40] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *Conference on Computer Vision and Pattern Recognition*, 2020.

[41] B. Recht, Rebecca Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International journal of Computer Vision*, 2015.

[43] Zhuoran Shen, Irwan Bello, Raviteja Vemulapalli, Xuhui Jia, and Ching-Hui Chen. Global self-attention networks for image recognition. *arXiv preprint arXiv:2010.03019*, 2020.

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[45] C. Sun, A. Myers, Carl Vondrick, Kevin Murphy, and C. Schmid. Videobert: A joint model for video and language representation learning. *Conference on Computer Vision and Pattern Recognition*, 2019.

[46] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[47] Christian Szegedy, V. Vanhoucke, S. Ioffe, Jon Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition*, 2016.

[48] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[49] Hugo Touvron, Alexandre Sablayrolles, M. Douze, M. Cord, and H. Jégou. Grafit: Learning fine-grained image representations with coarse labels. *arXiv preprint arXiv:2011.12982*, 2020.

[50] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herve Jegou. Fixing the train-test resolution discrepancy. *NIPS*, 2019.

[51] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy: Fixefficientnet. *arXiv preprint arXiv:2003.08237*, 2020.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[53] X. Wang, Ross B. Girshick, A. Gupta, and Kaiming He. Non-local neural networks. *Conference on Computer Vision and Pattern Recognition*, 2018.

[54] Longhui Wei, An Xiao, Lingxi Xie, Xin Chen, Xiaopeng Zhang, and Qi Tian. Circumventing outliers of autoaugment with knowledge distillation. *European Conference on Computer Vision*, 2020.

[55] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[56] Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arXiv preprint arXiv:2006.03677*, 2020.

[57] Qizhe Xie, Eduard H. Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves imagenet classification. *arXiv preprint arXiv:1911.04252*, 2019.

[58] L. Yuan, F. Tay, G. Li, T. Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. *Conference on Computer Vision and Pattern Recognition*, 2020.

[59] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019.

[60] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[61] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

[62] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020.