Navigating Scaling Laws: Compute Optimality in Adaptive Model Training

Sotiris Anagnostidis [*1] **Gregor Bachmann** [*1] **Imanol Schlag** [2] **Thomas Hofmann** [1]

Abstract

In recent years, the state-of-the-art in deep learning has been dominated by very large models that have been pre-trained on vast amounts of data. The paradigm is very simple: investing more computational resources (optimally) leads to better performance, and even predictably so; neural scaling laws have been derived that accurately forecast the performance of a network for a desired level of compute. This leads to the notion of a 'compute-optimal' model, i.e. a model that allocates a given level of compute during training optimally to maximize performance. In this work, we extend the concept of optimality by allowing for an 'adaptive' model, i.e. a model that can change its shape during training. By doing so, we can design adaptive models that optimally traverse between the underlying scaling laws and outpace their 'static' counterparts, leading to a significant reduction in the required compute to reach a given target performance. We show that our approach generalizes across modalities and different shape parameters.

1. Introduction

Deep learning has gradually undergone a paradigm shift, where instead of training specialized models for a given task, a so-called frontier model is prompted/few-shoted/fine-tuned for different desired downstream tasks. Frontier models are typically defined by their large-scale architectures, often rooted in the Transformer architecture (Vaswani et al., 2017), and their exposure to extensive and diverse data during their pre-training process, yielding remarkable advancements in both natural language understanding (OpenAI, 2023; Köpf et al., 2023) and computer vision tasks (Dehghani et al., 2023a; Chen et al., 2023b). An inherent and pivotal feature of such models lies in their scalability, whereby their performance can be reliably predicted as a power law across the number of parameters and the volume of data or computational resources utilized (Cortes et al., 1993; Hestness et al., 2017; Rosenfeld et al., 2019; Kaplan et al., 2020). These principles are succinctly encapsulated by the *neural scaling laws* that motivate the choice of a particular model and dataset size given a fixed budget of training compute (Hoffmann et al., 2022). The ability to accurately predict performance offers an undeniable reassurance in the often uncertain world of deep learning. It nevertheless, introduces an intimidating realization;

Given a training scheme, a fixed further improvement in performance requires exponentially more compute or parameters.

Finding solutions to address this issue becomes increasingly paramount, as staying competitive in the realm of deep learning increasingly depends on the availability of substantial computational resources. Delving deeper into the preceding statement, we highlight a pivotal assumption: *the shape of the model*, and therefore the number of FLOPs for a forward pass, remains *fixed throughout the training process*. By 'shape' we refer to any characteristic of a model that can be smoothly varied throughout training without leading to strong deterioration in performance. Such a static approach (i.e. where every model shape remains fixed) may however not always be optimal. For example, it has already been observed that the optimal model size grows smoothly with the loss target and the compute budget (Kaplan et al., 2020).

This paper challenges the assumption of a static model outlined above and explores adaptable training methodologies designed to *surpass conventional scaling laws*. In other words, our aim is to achieve equivalent performance for a specified model with fewer computational resources (FLOPs) than initially projected. To that end, we adapt the shape of the model throughout training, allowing the optimal traversal between different scaling laws. This enables us to leverage the optimality of all shape configurations in

[*]Equal contribution [1]Department of Computer Science, ETH Zürich [2]ETH AI Center. Correspondence to: <{sanagnos,gregorb}@ethz.ch>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

different regions of compute, leading to a more efficient scaling of the model.

We train Vision Transformers (Dosovitskiy et al., 2020) and Language Models (Radford et al., 2019) and showcase how an adaptive training scheme can lead to substantial training FLOPs reduction, in some cases more than 50%. Our contributions can be summarized as follows:

- We introduce a simple and effective strategy to choose when to adapt a model and traverse scaling laws, opting for the one that leads to the faster descent, i.e. maximum performance gain for the same amount of compute.
- We showcase the efficiency of our approach by optimally scheduling the patch size for ViTs as well as the context size for language models, leading to significant reductions in the required amount of compute to reach optimal performance.
- We further confirm the validity of our approach by adapting other shape parameters such as width, batch size, and overall training objective of a Vision Transformer.

2. Related Work

Neural scaling laws (Cortes et al., 1993), describe how a neural network's performance varies as a power law $E = a(P + d)^b + c$ where P can be either the number of parameters in the model, the number of training samples or simply the number of FLOPs used for training (Rosenfeld et al., 2019). Recently, scaling laws have been successfully demonstrated in a range of different applications, including language (Kaplan et al., 2020; Hoffmann et al., 2022) and vision (Zhai et al., 2022; Bachmann et al., 2023), as well as numerous learning settings, including supervised training, generative modeling (Henighan et al., 2020) and transfer learning (Hernandez et al., 2021). The predictive power of scaling laws has also been leveraged to determine compute-optimal models before training; the size of the *Chinchilla* model and the number of training tokens were chosen based on the underlying scaling law and indeed, *Chinchilla* outperformed its larger but sub-optimally trained counterpart *Gopher* (Hoffmann et al., 2022). The training of state-of-the-art models has also been guided by scaling laws built from training runs of smaller models (OpenAI, 2023; Team et al., 2023).

In this paper, we focus on the Transformer architecture and evaluate our methodology on both vision and natural language tasks. While the Transformer has been the default model in language for years, ViTs have more recently established themselves as the predominant vision architecture for large-scale pretraining tasks (Dehghani et al., 2023a). Different from convolutions, a ViT initially partitions the input image into patches and processes these through self-attention and MLP blocks. This lack of inductive bias (Smith et al., 2023) can be partially overcome through the introduction of 'soft' inductive bias, which proves to be beneficial, especially during the early phase of their training (d'Ascoli et al., 2021). Similarly to their counterparts in natural language processing, ViTs also exhibit predictable scaling behavior (Zhai et al., 2022; Dehghani et al., 2023a; Alabdulmohsin et al., 2023).

In this work, we delve into models that feature adaptive 'shape' parameters, specifically focusing on the patch size for image processing and model width. The concept of training with different patch sizes, which Beyer et al. (2023) have explored, leads to a model robust to various patch sizes. Another common approach involves pre-training a ViT model at a lower resolution, followed by fine-tuning at a higher resolution while maintaining the same patch size (Dosovitskiy et al., 2020; Zhai et al., 2022; Alabdulmohsin et al., 2023). Analogously, large language models (LLMs) can be pre-trained with a shorter and fixed context length and subsequently fine-tuned on longer ones (Chen et al., 2023c;a; Tworkowski et al., 2023).

Another way to change the model is by adding new parameters. Expanding a model under composable function-preserving operations has been a case of study for a long time in machine learning (Ash, 1989; Mitchell et al., 2023). The principal objective in this case is to accelerate training (Kaddour et al., 2023; Geiping & Goldstein, 2023). Such expansion operations have also been proposed for the Transformer architecture (Gesmundo & Maile, 2023; Chen et al., 2022) and have exhibited notable training speed-ups (Gong et al., 2019; Yao et al., 2023; Wang et al., 2023; Lee et al., 2022; Shen et al., 2022; Li et al., 2022). Apart from determining *how* and *where* in the model this expansion should occur, a primary challenge is to resolve *when* to add new neurons. We advocate that an effective strategy for adjustments to the model shape should be informed by considerations of scaling laws and the performance gains achieved per additional unit of computational resources.

Orthogonal to our approach, various techniques have been proposed to accelerate both inference and training, particularly in the context of Transformer models. These methods encompass a spectrum of strategies, including weight quantization (Dettmers et al., 2022; Frantar et al., 2022) and pruning weights and context (Frantar & Alistarh, 2023; Anagnostidis et al., 2023) among others. Specifically for ViTs, Bolya et al. (2022) propose to merge tokens at different layers in the architecture and Dehghani et al. (2023b) propose to pack sequences of tokens together to optimize hardware utilization. Additionally, d'Ascoli et al. (2021) proposes to initialize ViTs differently, making them look more like convolutions. Other methods have also been proposed to beat scaling laws, including data

(a) Patch sizes affect *how* ViTs process input images.

(b) *ViT-B* FLOPs.

Figure 1. Patch sizes define (left) how images are processed, while (right) impacting the compute of a forward pass.

pruning (Sorscher et al., 2022) or shaping models (depth vs width) more optimally (Alabdulmohsin et al., 2023). These approaches are supplementary to our methodology and can be effectively employed in conjunction to further enhance the efficiency of the training process.

3. ViTs and Optimal Patch Sizes

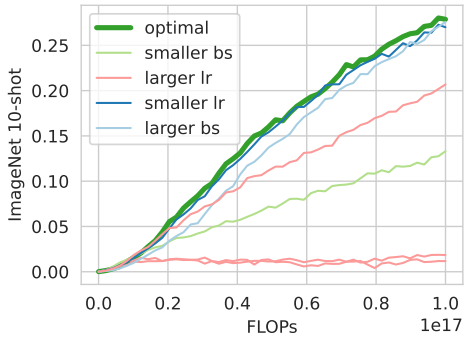
We first focus the discussion on Vision Transformers — the de-facto dominant architecture for vision — and the choice of patch size to introduce the notion of an adaptive model. In Sec. 5 we will showcase how our strategy can also be leveraged to train language models with adaptive context lengths. ViTs process images $\mathbf{x} \in \mathbb{R}^{h \times w \times c}$ where h and w are the height and width of the image in pixels and c is the number of channels. Images are 'patchified' into a sequence of n tokens based on a specified patch size $p \in \mathbb{N}$, where $n = \lfloor w/p \rfloor \times \lfloor h/p \rfloor$, leading to a representation $\mathbf{x}_{\text{patched}} \in \mathbb{R}^{n \times p^2 c}$. We illustrate the effect of different patch sizes in Fig. 1a. Each token is linearly embedded with learnable parameters $\boldsymbol{W}_{emb} \in \mathbb{R}^{p^2 c \times d}$ where $d \in \mathbb{N}$ is the embedding dimension or width of the ViT. These embeddings are further enhanced with learnable positional encodings $\boldsymbol{W}_{pos} \in \mathbb{R}^{n \times d}$ which enable a ViT to learn the spatial structure of the tokens. The resulting embeddings are then processed by L transformer blocks, consisting of a self-attention layer followed by an MLP that is shared across tokens. This specific structure of the architecture allows a ViT to generate predictions for token sequences of variable lengths, as is the case when dealing with images of different patch sizes.

Fixed patch size training. Different patch sizes come at different computational costs; the number of tokens n scales with $\mathcal{O}(1/p^2)$ and thus processing inputs scales with $\mathcal{O}(1/p^4)$ due to quadratic dependence on the input sequence length of the self-attention operation[1]. Consequently, a reduction in the patch size results in a substantial increase in the computational requirements for a forward pass. We present our empirical analysis in Fig. 1b. Using smaller patch sizes is often desirable as it yields superior model performance when paired with enough compute. To explore this trade-off, we pre-train Vision Transformers of different sizes (see Fig. 2(b) for a summary) on the public *ImageNet-21k* dataset (Ridnik et al., 2021), employing various patch sizes that remain *fixed* throughout training. To enhance computational efficiency and prevent bottlenecks caused by data transfer, we resize images to dimensions where $h = w = 120$, utilizing the FFCV data loader (Leclerc et al., 2023) [2]. This approach enables the application of a variety of patch sizes $p \in \{120, 60, 30, 24, 20, 15, 12, 10, 8, 6, 4, 3, 2, 1\}$, each of which perfectly divides the input resolution. During the training phase, we perform data augmentation, specifically random cropping, and horizontal flipping, and measure the 10-shot error (denoted as E) on the *ImageNet-1k* dataset (Deng et al., 2009). This is because upstream performance metrics may not reliably indicate model effectiveness (Tay et al., 2022; Zhai et al., 2022). While multiple epochs over the same dataset have been identified as less effective in language modeling tasks (Xue et al., 2023; Muennighoff et al., 2023), the use of augmentations in this work supports the feasibility of multi-epoch training without significant degradation in performance. This observation holds for the data and computational scales (up to 10 EFLOPs) that we consider (Zhai et al., 2022).

When calculating compute C, we exclude the computations associated with the 'head' of the network that maps the embedding dimension to the number of classes (Kaplan et al., 2020). Additionally, we adopt the approximation of previous work that the FLOPs required for the backward pass are approximately equivalent to twice the FLOPs in-

[1] The exact complexity is $\mathcal{O}(1/p^4 \times d + 1/p^2 \times d^2)$ for patch sizes $d > n$ where, unless packing is performed, $\mathcal{O}(1/p^2 \times d^2)$ is the dominant term.

[2] We expect a small decrease in performance due to this decreased resolution. Our most compute-intensive models (ViT Base variant) achieve top-1 accuracy on *ImageNet-1k* 79.2 % when fine-tuned and 77.2 % when linear probed on top of the extracted embeddings. Steiner et al. (2021) report 80.42% fine-tuning performance for a *ViT-B*/16 model on 224×224 images trained for 30 epochs on *ImageNet-21k*, which already surpasses our maximum compute budget.



(a) We optimize batch size, learning rate, and weight decay for each model configuration by running a greed search, for a small compute budget. More details are presented in the Appendix.

Name	Width	Depth	Heads	Param (M)	GFLOPs	
					$X = 8$	$X = 24$
V256-6/X	256	6	8	5.1	1.22	0.120
V192-12/X	192	12	3	5.6	1.43	0.136
V256-12/X	256	12	4	9.9	2.44	0.240
V384-12/X	384	12	6	21.8	5.25	0.538
V512-12/X	512	12	8	38.6	9.13	0.953
V640-12/X	640	12	10	60.0	14.1	1.49
V768-12/X	768	12	12	86.2	20.1	2.14

(b) Details on the ViT models we are training. We use the standard *s*, *S*, *Ti*, *B* model sizes, as well as other intermediate model sizes. To simplify and unify notation, we adopt the naming convention *Vd-L/X* for a Vision Transformer of depth L and embedding dimension d. Here *X* refers to the patch size.

Figure 2. (Left) Hyperparameters are optimized across model classes. (Right) The ViT models used for this study.

curred during the forward pass. Here, we are optimizing for FLOPs, and do not account for different types of hardware accelerators. For highly parallel neural architectures, FLOPs in general exhibit a strong correlation with accelerator time (see e.g. Fig. 4 (right) by Alabdulmohsin et al. (2023) for ViTs specifically and our results in App. F). In our study, we focus exclusively on Transformer models which are very hardware-efficient (Dosovitskiy et al., 2020). More details regarding the experimental setup are provided in App. B.

For a fixed model shape, we fit power laws for every patch size in terms of compute (which is proportional to the number of examples seen in this case). The power law takes the form[3]

$$E_P = f_P(C) = a_P(C + d_P)^{-b_P} + c_P. \tag{1}$$

where the exponent b_P dictates the decay speed and c_P corresponds to the maximal reachable performance given infinite compute. After fitting the parameters $a_P, d_P, b_P, c_P \in \mathbb{R}^+$, we can predict downstream performance E_P (*ImageNet-1k* 10-shot top-1 unless otherwise stated) as a function of compute C measured in FLOPs. We display the results for the $V640$-12 model in Fig. 4 and Fig. 5. We provide analogous plots for all model sizes in App. C. From those scaling trends, it is evident that different patch sizes are optimal for different amounts of compute. In other words, *given the same compute, different patch sizes yield different improvements at specific levels of performance.* Given that insight, a very natural question emerges:

Can we traverse between the scaling laws more efficiently by allowing for adaptive patch sizes?

[3] As aforementioned, our models are bound by data rather than the number of parameters.

4. Adaptive Patch Sizes and Traversing Scaling Laws

Adaptive patch size. To allow for a smooth traversal of different laws, we first need a mechanism that enables mapping a ViT f_P with patch size P to a ViT f_Q with patch size Q, while ideally not degrading performance, i.e. $f_P \approx f_Q$. *FlexiViT* introduced by Beyer et al. (2023) achieves this. It redefines both the patch embedding $\boldsymbol{W}_{emb}$ and the positional encodings $\boldsymbol{W}_{pos}$ for a fixed base patch size. In every forward pass, depending on the patch size, the base embedding parameters $\boldsymbol{W}_{emb}$ are resized based on the pseudo inverse of the resizing matrix. Similarly, the base positional encodings $\boldsymbol{W}_{pos}$ are bi-linearly interpolated, enabling the model to change the patch size without a strong performance degradation. For further information, we direct readers to the work by Beyer et al. (2023).

Traversing scaling laws. Let the set of scaling laws be $\{f_P\}$ with P denoting the patch size. Each law maps a given level of compute C to the predicted downstream performance $E_P = f_P(C)$. Consider the inverted laws $f_P^{-1}(E)$ which predict for a given level of desired performance E, how much compute C needs to be invested. For a given error level E^*, we aim to reach a lower error $E^* - \epsilon$ for $\epsilon > 0$, while spending the minimal amount of compute C to achieve this, i.e. we want the least change in f_P^{-1}. To solve this problem, we simply compute the partial derivatives

$$q_P(E^*) := \frac{\partial f_P^{-1}(E)}{\partial E}\bigg|_{E=E^*} \quad \forall P. \tag{2}$$

Maximising q_P over the patch size P, partitions the error space disjointly (e.g. if we assume E is the classification error taking values in $[0, 1]$),

$$[0, 1] := \bigcup_P E_P,$$

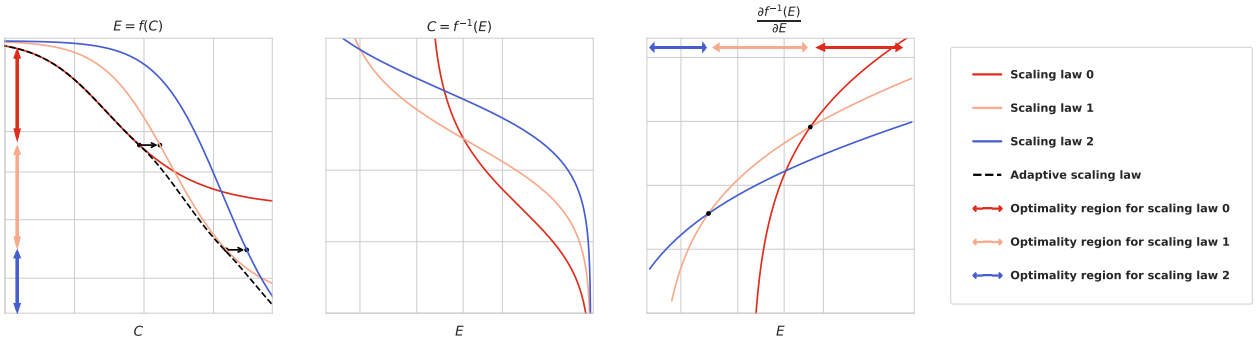


Figure 3. (Left) Different scaling law curves (function f in Eq. 1) corresponding to different training configurations. Black arrows indicate points of transition between scaling laws. (Middle) We illustrate the inverse of the above function f^{-1} for the same scaling law curves. (Right) We visualize the gradient of the inverse $\partial f^{-1}(E)/\partial E$ for the same scaling laws. Taking the curve that maximizes the aforementioned gradient, leads to a partition of the space. From this partition, we can deduce a strategy determining which scaling law to 'follow' for each performance level.

where $E_P \subset [0, 1]$ denotes the set where the patch size P achieves the most efficient improvement. This partition naturally gives rise to a scheduler for the patch size, which empirically turns out to be monotonic (i.e. starting from the largest patch size for large classification error values and ending with the smallest for small classification errors), which is expected based on the observations in Fig. 4. We visualize the strategy in Fig. 3. Required assumptions for our scheduler to work – which might hold or not in practice – are that (1) we can traverse between models of different shapes without performance degradation and (2) the power laws only depend on the current performance, and not the history of training so far.

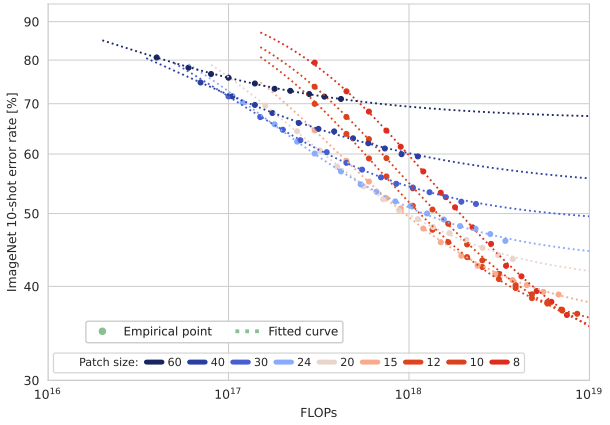


Figure 4. Downstream performance as a function of compute for the $V640$-12 model and different patch sizes. We use a log-log scale.

Scheduled training. We now test the devised strategy in a practical setting by pre-training variously-sized ViTs on *ImageNet-21k* using our patch size scheduler. We use the same training setup as for the fixed patch size experiments and let the scheduler consider patch sizes $P \in \{60, 40, 30, 24, 20, 15, 12, 10, 8\}$. We display *ImageNet-1k* 10-shot error rate as a function of compute C for the model $V640$-12 in Fig. 5 and provide plots for all other models in the App. C. The crosses denote the points where the scheduler switches patch size. We observe a significant improvement in terms of compute efficiency, allowing for up to -60% FLOPs to achieve the same performance. While switching patch sizes may initially result in a slight decrease, attributed to changes in the entropy within the self-attention layers, this minor setback is rapidly offset as the image is parsed in a more fine-grained manner. Such degradation is thus not even visible in Fig. 5[4]. To facilitate comparison across all model sizes at once, we further visualize the compute-optimal barrier for both fixed and scheduled training in Fig. 6. By compute-optimal, we refer to a model that optimally trades off model size, patch size, and

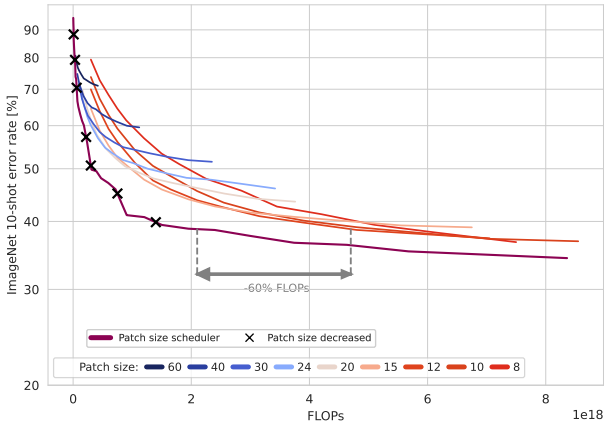


Figure 5. Downstream performance of the $V640$-12 trained with our patch size scheduler, and its potential benefits.

[4]Differences in the effective receptive field of each patch are typically mitigated by cropping as a component of the training procedure.

number of samples seen for a given level of compute C, i.e. achieving the lowest error E. We observe that the optimally scheduled models significantly outperform the optimal static models, halving the required compute to optimally train a ViT-Base model (for our compute budget).

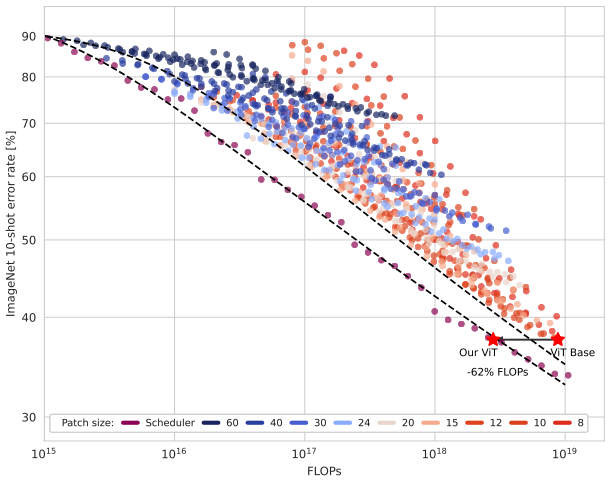


Figure 6. Compute-optimal static and scheduled models for various patch and model sizes. We plot using a log-log scale.

Is the schedule optimal? While our scheduler improves over the individual models, it is not clear yet that it does so in an optimal sense, i.e. can other schedules achieve similar benefits? Beyer et al. (2023) also employ a patch size scheduler but use a uniformly random sampling of the patch size at every step. We compare against their model *FlexiViT* in Fig. 7 and observe that our scheduler indeed outperforms *FlexiViT* as expected; *FlexiViT* targets a lower inference cost by making the model robust to many patch sizes (hence the random scheduler). Compute-optimality is not their objective. Additionally, we conduct comparisons with straightforward patch size scheduling strategies—both linear and logarithmic. Specifically, for a predetermined total computational budget, we distribute the transition points evenly or according to a logarithmic scale across the training process. This way, we assess whether simply any monotonic scheduler leads to the same improvements, or whether the position of the transition points matter. We display the results in Fig. 7. We again observe that our scheduler remains more efficient, carefully determining the transition points based on the scaling laws, thus indeed leading to a significant improvement.

Smaller patch sizes. Undeniably, the choice of patch size affects the inductive bias of ViTs — in general, the mechanism of 'tokenization' in the input affects the inductive bias of any Transformer model — by controlling the amount of compute and the level of details we are interested in extracting from an image. The patch size also controls the overall sequence length n processed by the Transformer model, and therefore the degree of weight sharing between the parameters. Our previous laws clearly show that smaller patch sizes lead to better performance in high-compute areas. But does this trend also extend to even smaller patch sizes? We explore this question empirically by using the same experimental setup and pre-training on even smaller patch sizes $P \in \{6, 4\}$ in addition to the previous results. We display the results in Fig. 8.

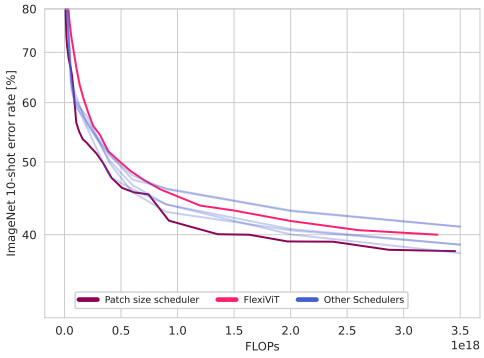


Figure 7. We compare performance as a function of training compute against the scheduler of *FlexiViT* and other schedulers. Irrespective of the current patch size in the scheduler, we use the smallest patch size (i.e. 8), when evaluating *FlexiViT*.

We observe that while some absolute gains in performance can still be achieved with patch size 6, the additional required amount of compute is extremely high. For the even smaller patch size 4, one actually starts to lose in performance, as can be seen from plotting the intercepts c_P of the corresponding scaling laws. The behavior of performance with respect to patch size is thus only monotonic up to a certain point, and performance may actually worsen beyond that. This is in contrast to other scaling parameters such as the number of samples or the model size that usually offer a monotonic behavior in performance when

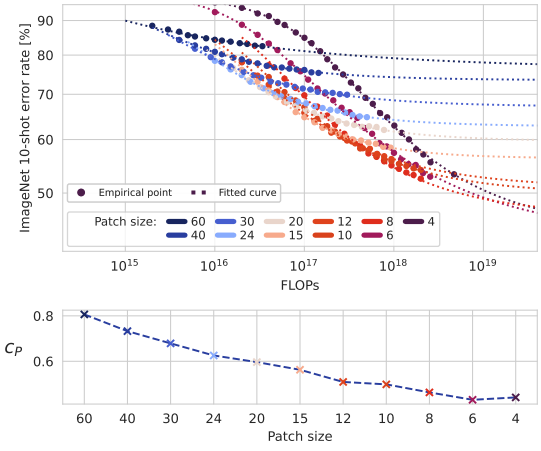


Figure 8. We train the $V256$-6 with smaller patch sizes. This does not lead to a monotonically better performance.

scaled appropriately.

5. Adaptive Context Size of an LLM

While the previous chapter focused on Vision Transformers, the aforementioned intuitions and results generalize also to other domains. In this section we present a compelling case for the training of a Transformer-based language model. A critical consideration in training such a model is determining the optimal context size, as most Transformer models do not inherently support context extrapolation out of the box (Kazemnejad et al., 2023). The context size determines the amount of information the model can use when making predictions, with an expanded context being essential for enhanced performance in certain downstream tasks (Dao et al., 2022; Tay et al., 2020). A larger context size, however, comes with increased computational requirements, similar to the patch size in ViTs.

To mitigate the significant training overhead associated with long contexts — where the quadratic complexity of self-attention becomes a bottleneck — a common strategy involves initially training with shorter contexts. Subsequently, the model is fine-tuned on longer contexts, employing a modest computational budget to balance performance and efficiency (Chen et al., 2023c). Here, we demonstrate how different context sizes lead to different learning improvements at different scales of compute, and navigating across the optimal scaling law can lead to *substantial performance gains* given the same amount of compute. We train Transformer models (Touvron et al., 2023a;b) on English Wikpedia and Books (more details in App. B.5).

The findings illustrated in Fig. 9 align with our hypothesis: smaller context sizes prove to be more optimal at the beginning of the training process, while larger contexts yield greater efficiency as training progresses. Employing a similar approach to that used with patch sizes, we introduce a scheduler that is adjusted in accordance with scaling laws. This strategy, akin to our adjustments in patch size, results in substantial computational savings, enabling a reduction in FLOPs of up to 40%.

6. Other Shape Parameters

To further verify the efficiency of our approach, we study different 'shape' parameters when training a Vision Transformer.

6.1. Adapting Model Width

Similarly to the patch size, we need a mechanism that maps a transformer of smaller width d_1 to a transformer of larger width d_2. This is a very well-studied problem, especially in the case of natural language processing, and many schemes have been proposed (Gesmundo & Maile, 2023; Chen et al., 2022; Gong et al., 2019; Yao et al., 2023; Wang et al., 2023; Lee et al., 2022; Shen et al., 2022; Li et al., 2022). Here, we focus on the simplest approach, where we expand the initial model d_1 by adding randomly initialized weights (see App. C for details). Although our expansion is not function preserving — i.e. we can expect a small drop in performance immediately after adapting the model (see Fig. 11) — we found that the model quickly recovers, and hence conclude that while not ideal, this simple expansion mechanism suffices for our setting[5].

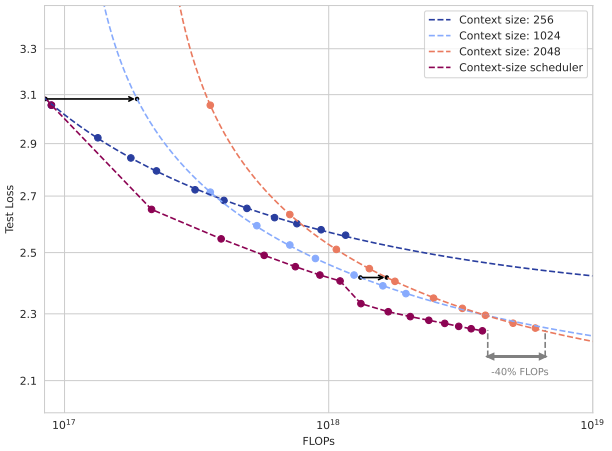


Figure 9. We compare model performance throughout training with dynamic (magenta) and fixed context sizes.

Scaling width. The role of the model width and its associated scaling properties have been long analysed in the literature (Zhai et al., 2022; Alabdulmohsin et al., 2023). We repeat the scaling study for our own experimental setup and pre-train Vision Transformers of various widths and training durations on *ImageNet-21k*, using the same experimental setup as detailed in Sec. 3. In Fig. 10 we report 10-shot *ImageNet-1k* error as a function of compute for a fixed patch size $P = 20$ (more details and results for other patch sizes are provided in the App. C). We again observe that different model widths are optimal for different levels of compute, similarly offering the potential for computational speed-ups by adapting the shape throughout training.

Scheduling width. It is worth noting that strategies for expanding models during training have been previously explored. However, the critical question of *when* this expansion should occur has largely remained unanswered. Our approach then offers a straightforward and principled solution. We consider three width settings $d \in \{192, 384, 768\}$ and devise our scheduler based on the scaling law as outlined in Sec. 4. We display the obtained optimal sched-

[5]In practice we found that proposed function preserving schemes that depend on zero-initializing weights in the network, e.g. by Gesmundo & Maile (2023), perform suboptimally.

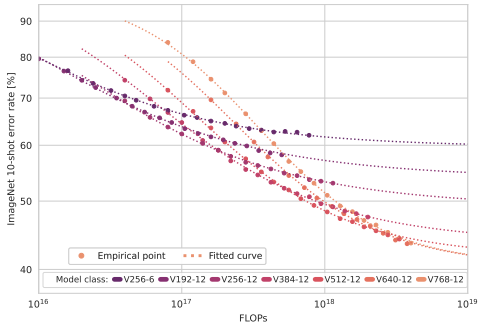


Figure 10. Downstream performance as a function of compute for ViT of different size, trained with a patch size of 20. We use a log-log scale.

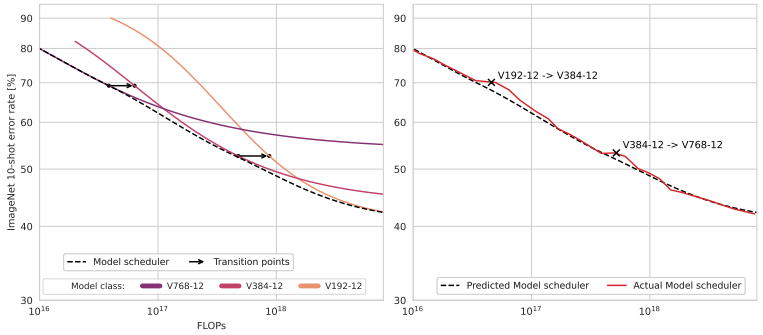


Figure 11. The theoretically predicted scheduled performance (left) and the empirical obtained (right) performance. While transitions are less smooth, the model based on the scheduler quickly recovers back to the predicted law.

ule and the actual resulting performance in Fig. 11. As remarked previously, changing the model width does lead to a momentary deterioration of the performance, but it smoothly recovers back to the predicted performance. We again observe that the scheduled model remains optimal throughout training when compared to the static models. Note that adapting multiple shape parameters — namely both patch size and model size — is possible and leads to further improvements, as we showcase in App. D.

6.2. Adapting the Training Objective

In previous experiments, we have fixed training hyperparameters and focused on adapting the model's shape. Training hyperparameters can nonetheless also be altered optimally. Previous work has already established that bigger *batch sizes* are beneficial during later stages in training (Kaplan et al., 2020; Hoffmann et al., 2022; Zhai et al., 2023). Different *training objectives* are also known to contribute differently to downstream performance at distinct stages in training (Zhai et al., 2023; Singh et al., 2023). Previous work has relied on heuristics and brute force exploration to determine when these should change. Here, we demonstrate how scaling laws can help decide when to change these parameters and descend based on the optimal one.

Batch size. We first focus on the batch size used during training. We train two Vision Transformers, one at a larger batch size and the other at a smaller batch size. We find that in terms of number of FLOPs, the smaller batch size initially dominates but again is surpassed at later stages in training by the large batch size run (Fig. 12 left). Our strategy allows us to maximally take advantage of this difference by optimally transitioning between the two batch sizes, leading to a more optimal model.

Distillation. We additionally train ViT models by distilling from a powerful teacher, a bigger pre-trained ViT (Fig. 12 right). Distilled labels naturally come at an additional computational cost, due to queries to the teacher, and lead to slower convergence in terms of FLOPs initially in training (FLOPs here include the teacher compute). Distillation objectives, however, were found to lead to increases in performance (Hinton et al., 2015; Furlanello et al., 2018; Touvron et al., 202x21). Thus, in later stages of training, such an objective will dominate the standard supervised loss. We again leverage this discrepancy and optimally switch from the standard to the distilled loss, allowing us to reach the same level of performance with fewer FLOPs.

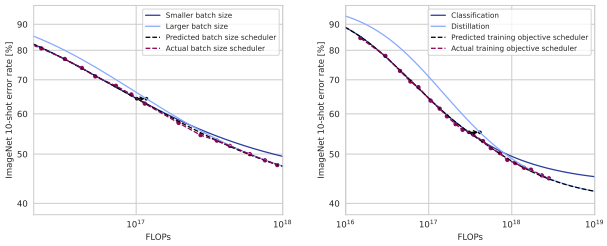


Figure 12. (Left) ViTs with small and big batch sizes and using (right) different upstream objectives. Different optimality regimes are observed for the different settings. Changing between them leads to optimal performance gain for a fixed compute budget.

7. Conclusion

We have explored strategies to train models with variable shape parameters throughout the training process. By efficiently traversing scaling laws, we have illustrated the optimal scheduling of shape parameters — including patch size, context length, model width, and other hyperparameters — yielding notable computational efficiency for a given performance level. We further observe that models with dynamically scheduled shape parameters consistently outperform their static counterparts in terms of computational efficiency during training. This underscores the effectiveness of our method. Our scheduling approach is highly adaptable and applies to any shape parameter that allows for a smooth transition between models of varying configurations. This opens up many opportunities for

future research, applying our scheduling method to other shape parameters such as model depth, sparsity, or a combination of multiple parameters. Given the increased computational demand for deep learning, we believe our findings make a crucial contribution to the field.

Impact Statement

Our method provides insights into the growing challenges associated with the exponential scaling of compute resources for the training of frontier models. By making the training of large models more accessible, our approach opens doors to a broader audience of researchers and practitioners, fostering innovation and breakthroughs in artificial intelligence. Importantly, this alternative strategy contributes to a reduction in environmental impact, showcasing a commitment to sustainable and responsible advancements in the field. In App. E we provide concrete insights on the expected reduction in CO_2 emissions for our experimental setup.

References

Alabdulmohsin, I., Zhai, X., Kolesnikov, A., and Beyer, L. Getting vit in shape: Scaling laws for compute-optimal model design. *arXiv preprint arXiv:2305.13035*, 2023.

Anagnostidis, S., Pavllo, D., Biggio, L., Noci, L., Lucchi, A., and Hofmann, T. Dynamic context pruning for efficient and interpretable autoregressive transformers, 2023.

Ash, T. Dynamic node creation in backpropagation networks. *Connection science*, 1(4):365–375, 1989.

Bachmann, G., Anagnostidis, S., and Hofmann, T. Scaling mlps: A tale of inductive bias. *arXiv preprint arXiv:2306.13575*, 2023.

Beyer, L., Izmailov, P., Kolesnikov, A., Caron, M., Kornblith, S., Zhai, X., Minderer, M., Tschannen, M., Alabdulmohsin, I., and Pavetic, F. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14496–14506, 2023.

Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Chen, S., Wong, S., Chen, L., and Tian, Y. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023a.

Chen, W., Huang, W., Du, X., Song, X., Wang, Z., and Zhou, D. Auto-scaling vision transformers without training. *arXiv preprint arXiv:2202.11921*, 2022.

Chen, X., Djolonga, J., Padlewski, P., Mustafa, B., Changpinyo, S., Wu, J., Ruiz, C. R., Goodman, S., Wang, X., Tay, Y., et al. Pali-x: On scaling up a multilingual vision and language model. *arXiv preprint arXiv:2305.18565*, 2023b.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023c.

Cortes, C., Jackel, L. D., Solla, S., Vapnik, V., and Denker, J. Learning curves: Asymptotic values and rate of convergence. *Advances in neural information processing systems*, 6, 1993.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pp. 7480–7512. PMLR, 2023a.

Dehghani, M., Mustafa, B., Djolonga, J., Heek, J., Minderer, M., Caron, M., Steiner, A., Puigcerver, J., Geirhos, R., Alabdulmohsin, I., et al. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *arXiv preprint arXiv:2307.06304*, 2023b.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

d'Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., and Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pp. 2286–2296. PMLR, 2021.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Evans, T., Pathak, S., Merzic, H., Schwarz, J., Tanno, R., and Henaff, O. J. Bad students make great teachers: Active learning accelerates large-scale visual understanding. *arXiv preprint arXiv:2312.05328*, 2023.

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1607–1616. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/furlanello18a.html.

Geiping, J. and Goldstein, T. Cramming: Training a language model on a single gpu in one day. In *International Conference on Machine Learning*, pp. 11117–11143. PMLR, 2023.

Gesmundo, A. and Maile, K. Composable function-preserving expansions for transformer architectures. *arXiv preprint arXiv:2308.06103*, 2023.

Gong, L., He, D., Li, Z., Qin, T., Wang, L., and Liu, T. Efficient training of bert by progressively stacking. In *International conference on machine learning*, pp. 2337–2346. PMLR, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Kaddour, J., Key, O., Nawrot, P., Minervini, P., and Kusner, M. J. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *arXiv preprint arXiv:2307.06440*, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kazemnejad, A., Padhi, I., Ramamurthy, K. N., Das, P., and Reddy, S. The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*, 2023.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., et al. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.

Leclerc, G., Ilyas, A., Engstrom, L., Park, S. M., Salman, H., and Madry, A. Ffcv: Accelerating training by removing data bottlenecks, 2023.

Lee, Y., Lee, G., Ryoo, K., Go, H., Park, J., and Kim, S. Towards flexible inductive bias via progressive reparameterization scheduling. In *European Conference on Computer Vision*, pp. 706–720. Springer, 2022.

Li, C., Zhuang, B., Wang, G., Liang, X., Chang, X., and Yang, Y. Automated progressive learning for efficient training of vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12486–12496, 2022.

Mitchell, R., Mundt, M., and Kersting, K. Self expanding neural networks. *arXiv preprint arXiv:2307.04526*, 2023.

Muennighoff, N., Rush, A. M., Barak, B., Scao, T. L., Piktus, A., Tazi, N., Pyysalo, S., Wolf, T., and Raffel, C. Scaling data-constrained language models, 2023.

Noci, L., Anagnostidis, S., Biggio, L., Orvieto, A., Singh, S. P., and Lucchi, A. Signal propagation in transformers:

Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35: 27198–27211, 2022.

OpenAI, R. Gpt-4 technical report. *arXiv*, pp. 2303–08774, 2023.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Qin, Z., Zhong, Y., and Deng, H. Exploring transformer extrapolation. *arXiv preprint arXiv:2307.10156*, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. Imagenet-21k pretraining for the masses, 2021.

Rosenfeld, J. S., Rosenfeld, A., Belinkov, Y., and Shavit, N. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

Ruoss, A., Delétang, G., Genewein, T., Grau-Moya, J., Csordás, R., Bennani, M., Legg, S., and Veness, J. Randomized positional encodings boost length generalization of transformers. *arXiv preprint arXiv:2305.16843*, 2023.

Shen, S., Walsh, P., Keutzer, K., Dodge, J., Peters, M., and Beltagy, I. Staged training for transformer language models. In *International Conference on Machine Learning*, pp. 19893–19908. PMLR, 2022.

Singh, M., Duval, Q., Alwala, K. V., Fan, H., Aggarwal, V., Adcock, A., Joulin, A., Dollár, P., Feichtenhofer, C., Girshick, R., et al. The effectiveness of mae prepretraining for billion-scale pretraining. *arXiv preprint arXiv:2303.13496*, 2023.

Smith, S. L., Brock, A., Berrada, L., and De, S. Convnets match vision transformers at scale. *arXiv preprint arXiv:2310.16764*, 2023.

Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536, 2022.

Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Tay, Y., Dehghani, M., Abnar, S., Chung, H. W., Fedus, W., Rao, J., Narang, S., Tran, V. Q., Yogatama, D., and Metzler, D. Scaling laws vs model architectures: How does inductive bias influence scaling? *arXiv preprint arXiv:2207.10551*, 2022.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 202x21.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Wang, P., Panda, R., Hennigen, L. T., Greengard, P., Karlinsky, L., Feris, R., Cox, D. D., Wang, Z., and Kim, Y. Learning to grow pretrained models for efficient transformer training. *arXiv preprint arXiv:2303.00980*, 2023.

Xue, F., Fu, Y., Zhou, W., Zheng, Z., and You, Y. To repeat or not to repeat: Insights from scaling llm under token-crisis, 2023.

Yao, Y., Zhang, Z., Li, J., and Wang, Y. 2x faster language model pre-training via masked structural growth. *arXiv preprint arXiv:2305.02869*, 2023.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12104–12113, 2022.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023.

A. Limitations

We detail the limitations of our work to the best of our knowledge.

- We used greedy search with a small compute budget to get optimal hyper-parameters per model class. In practice, optimal parameters can change if larger levels of compute are available, as also hinted in Sec. 6.2.
- In order to determine the optimal scheduler for a given shape parameter, knowledge of its scaling behavior is needed, which comes at a high computational cost. On the other hand, the scaling behavior of many shape parameters has already been established (e.g. width, depth, MLP-dimension (Alabdulmohsin et al., 2023)) and can readily be used in our scheduler.
- Accurately predicting compute-optimal models, requires one to accurately schedule the learning rate throughout training. As we are interested in what happens during training for low-budgets of computes we do not schedule the learning rate nor embark on a cooldown phase (Zhai et al., 2022), as this would constitute a large fraction of the overall training time. We expect learning rate schedulers may shift our conclusion but not the outcome and takeaway message.
- While we observe that the scheduled models are compute-optimal throughout all of training, we observe the largest gains earlier on throughout training. Indeed, we do not expect our scheduled models to reach better performance for an infinite amount of compute.

B. Experimental Setup

We provide more details on the basis on which the experiments were conducted.

B.1. Training Details

In Tab. 1 we showcase hyper-parameters used when training on *ImageNet-21k*. We optimized each of the parameters for the different model classes by training for different configurations for a fixed, small amount of compute, namely 4×10^{17} FLOPs. Some examples of such hyper-parameter search are illustrated in Fig 13. All experiments were conducted using *bfloat16*.

B.2. Finetuning Details

In Tab. 2 we showcase hyper-parameters used when finetuning on *ImageNet-1k*. For the few-shot results, we use the *linear_model.Ridge* function from *scikit-learn* with a regularization parameter of $1e^{-2}$.

Parameter	Value
Optimizer	AdamW
Betas	$(0.9, 0.999)$
Label smoothing	0.2
Weight-decay head	0.01
Weight-decay body	0.01
Warm-up	1000 steps
Clip gradients' norm	1.0
Underlying patch-size shape	12
Underlying posemb shape	8

Table 1. Hyper-parameters during training. 'Underlying patch-size' and 'Underlying posemb shape' refer to the flexible modules when training under a flexible patch size scheduler.

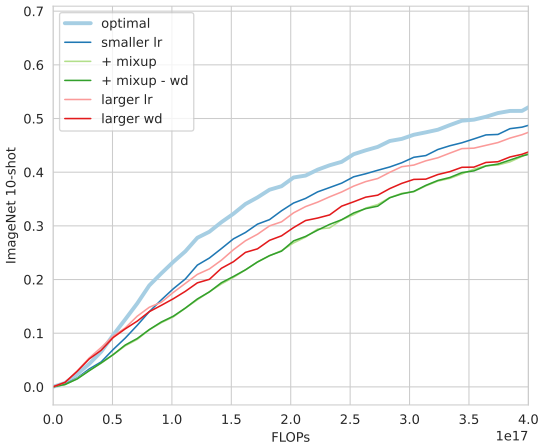


Figure 13. Hyper-parameter search for a fixed (and small) budget of compute.

Parameter	Value
Optimizer	SGD
Learning rate	0.03
Momentum	0.9
Weight decay	0.0
Number of steps	20000
Clip gradients' norm	1.0
Scheduler	Cosine

Table 2. Hyper-parameters during fine-tuning on *ImageNet-1k*.

B.3. Dataset Description

For the ViT experiments, we follow the protocol of Ridnik et al. (2021) to preprocess *ImageNet-21k*. It consists of roughly 12 million images and 11 thousand different classes. This is still considerably lower than the $\geq 29,593$ classes in the *JFT-3B* dataset. We experimented using different weight decay values for the body and the head, as proposed in Zhai et al. (2022) but found no significant dif-

ference. We attribute this to the lower number of classes in the dataset we are training in and our value of label smoothing.

B.4. Scaling Laws

We fit functions of the form

$$E = a(C + d)^{-b} + c. \tag{3}$$

Similar to previous work, we resample points to be almost equidistant in the log-domain, in terms of FLOPs. We minimize different initializations using the *minimize* function in *scipy* (Virtanen et al., 2020), and choose the one that leads to the smallest error. The function to minimize is based on the *Huber loss* with $\delta = 1e^{-3}$.

B.5. Sec. 5 Details

In Sec. 5, we train Llama (Touvron et al., 2023a;b) models, following the Transformer++ training recipe, as displayed in Tab. 3. Due to compute constraints, we train models with embedding dimension 768 and depth 12. We train on a subset of the English Wikipedia *20220301.en* and English *bookcorpus* datasets. As done for the ViT models, we select for each configuration the batch size that leads to the most efficient – in terms of FLOPs – convergence.

We evaluate on a held-out validation set. Although the validation samples are the same across different runs, we truncate them to match the context size of the respectively trained model. Models with longer contexts are thus expected to achieve lower test loss due to the enhanced context size (conditioned on longer contexts, subsequent tokens are more predictable). As during inference we are interested in the validation cross entropy loss over the longer contexts, we do not adjust for this discrepancy.

PARAMETER	VALUE
OPTIMIZER	ADAMW
BETAS	$(0.9, 0.95)$
CLIP GRADIENT'S NORM	1.0
WEIGHT DECAY	0.1
DROPOUT	0.0
WARMUP	1000
NORM	*RMSNorm*
BIAS	NO
PEAK LEARNING RATE	GPT-3 VALUES

Table 3. Hyper-parameters during training of the language models.

Extrapolating to longer contexts is a very active area of research with exciting work published recently (Qin et al., 2023; Ruoss et al., 2023; Press et al., 2021; Peng et al., 2023). In our case, we are training using RoPE positional encodings (Su et al., 2024), which are known to extrapolate easier compared to other ones, such as absolute positional encodings.

The exact number for the most compute-intensive point in Fig. 9 is 17825792000 tokens. The exact model size including the embedding parameters is 137841408 parameters. That leads to an approximate token per parameter value of 129.3, past the optimal "Chinchilla" point, leading thus to model that are more inference efficient.

B.6. Sec. 6.2 Details

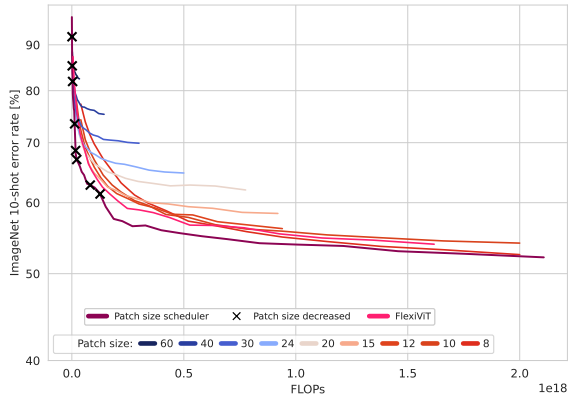
In Sec. 6.2, we train ViT models using the same hyperparameters found as described above. For the batch-size experiments, we train *V*384-20/*12* models with batch size in $\{256, 2048\}$ and navigate across the scaling laws corresponding to the same values. For the different training objective experiments, we train a *V*384-20/*12* model using either supervised training or distillation from a powerful teacher. We use as a teacher a pre-trained *V*640-10/*12* model and train using only distillation loss as in Hinton et al. (2015), with a temperature of $T = 2$. When calculating the FLOPs of the single step, we include the FLOPs of the teacher only for the forward pass.

C. Additional Experiments

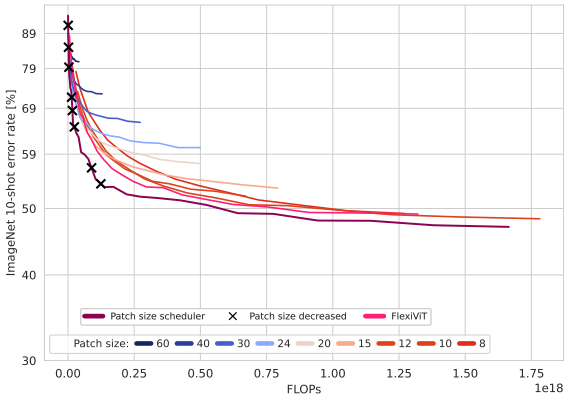
Patch size scheduler. We present additional experiments on patch size schedulers in Fig. 14. For *FlexiViT* – similar to the original paper – we sample uniformly at every step a patch size from the set $\{8, 10, 12, 15, 20, 24\}$. We did not use smaller patch sizes due to computational constraints. Note that our patch size scheduler leads to significantly faster convergence across the model classes we are analyzing. We also present in Fig. 15, the fitted scaling curves and the points where changing the patch size leads to the steepest descent for different scaling laws.

Model width scheduler. Supplementary to the results in Sec. 6.1, we provide additional examples of width examples in Fig. 16. Note that we do not touch on the (1) *where* to add the new weights and (2) *how* to initialize these new weights. Our approach simly defines a strategy on the *when* to expand the model and can be used in conjunction with any related works that provide answers to the previous (1) and (2) questions.

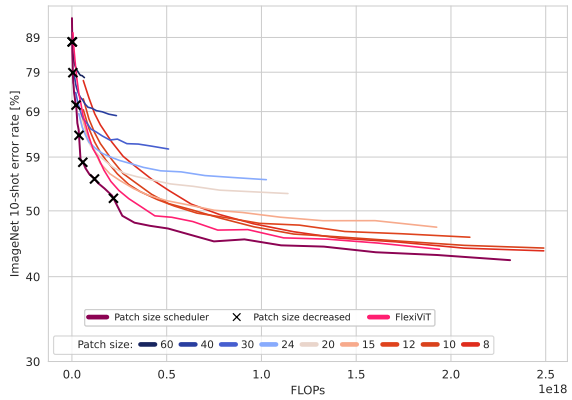
Regarding (1), we focus on models with constant depth (remember we are using the established *Ti*, *S*, and *B* Vision Transformer sizes). Therefore, we do not add new layers but merely expand the weight matrices to the new embedding dimension. Our method is agnostic to *where* these weights are added, just on the final form of the scaling law. Note that there exist more optimal ways to expand the dif-



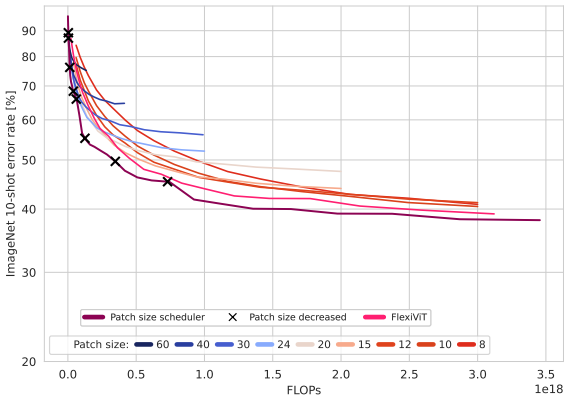
(a) Model $V256$-6.



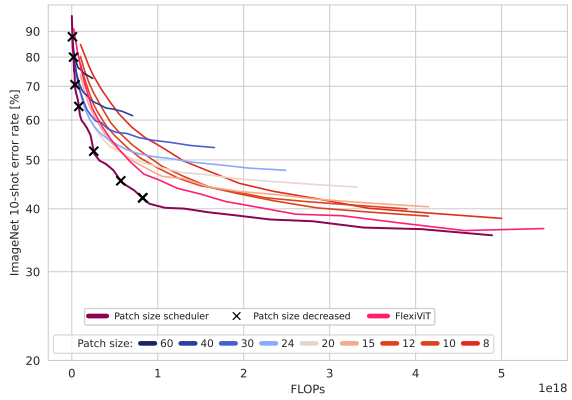
(b) Model $V192$-12.



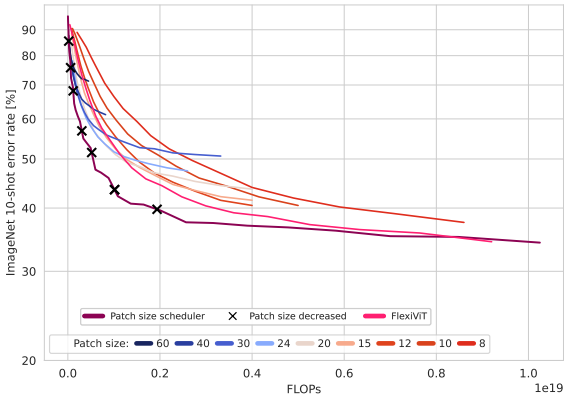
(c) Model $V256$-12.



(d) Model $V384$-12.

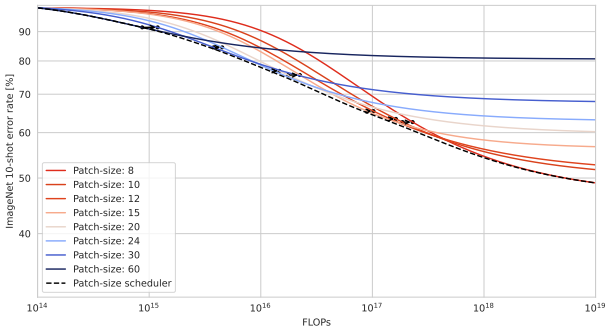


(e) Model $V512$-12.

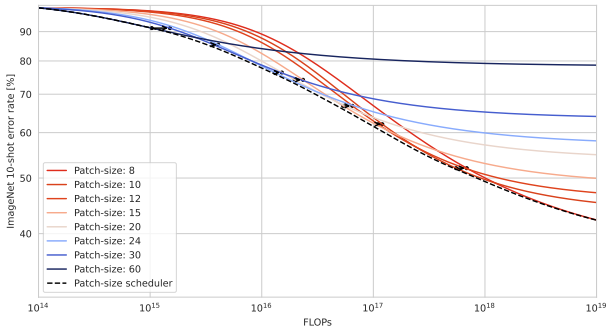


(f) Model $V768$-12.

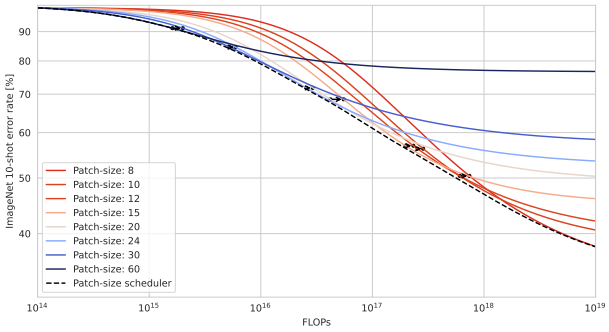
Figure 14. Patch size schedulers for all the remaining model classes analysed.



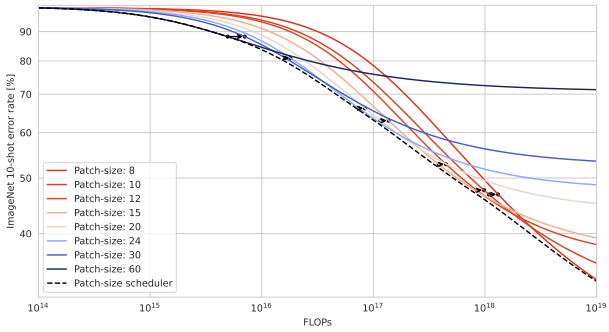
(a) Model $V256$-6.



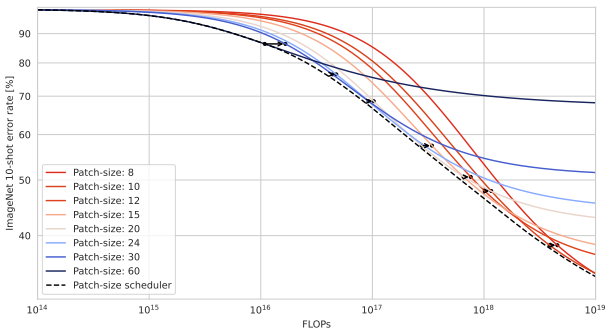
(b) Model $V192$-12.



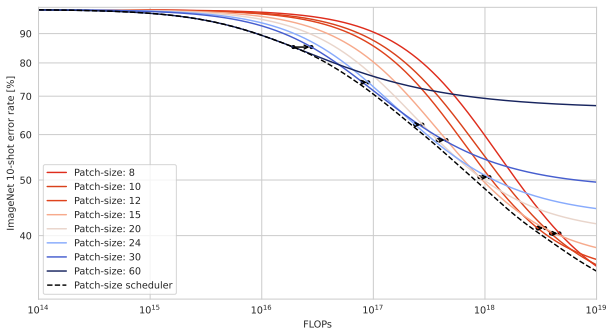
(c) Model $V256$-12.



(d) Model $V384$-12.

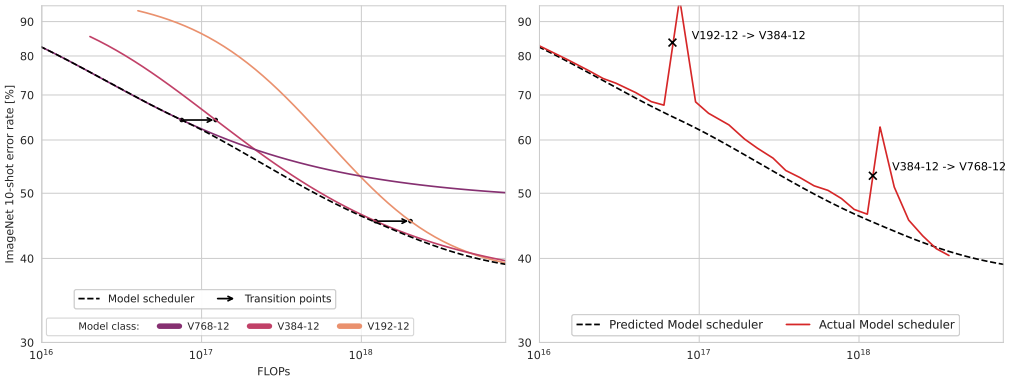


(e) Model $V512$-12.

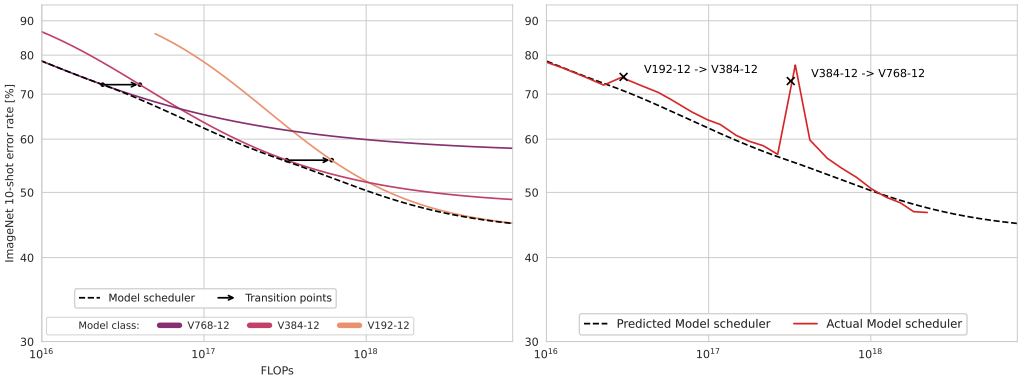


(f) Model $V640$-12.

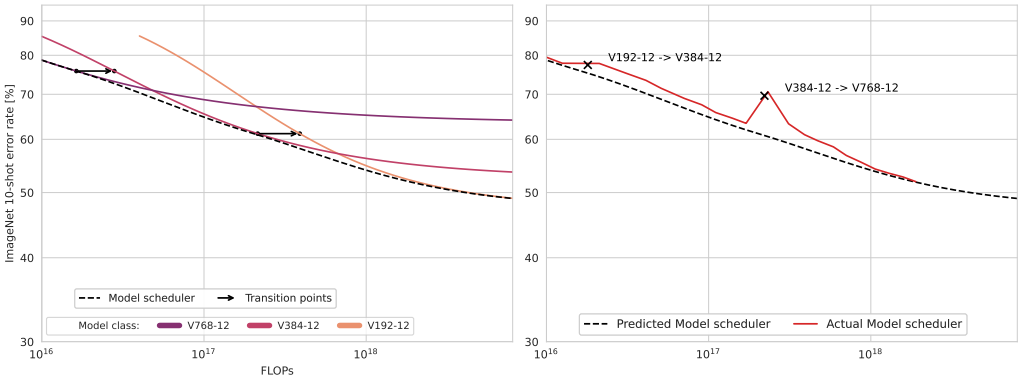
Figure 15. Fitted scaling laws and the predicted transition points that lead to the steepest descent.



(a) Patch size 15.



(a) Patch size 24.



(c) Patch size 30.

Figure 16. Width scheduler for models trained with different patch sizes. We expand the model width twice, as done in Sec. 6.1. The transition points of the expansion are found through our maximum descent rule.

ferent components of a ViT model (Alabdulmohsin et al., 2023).

Regarding (2), there are numerous works on *how* to initialize the weights under a function preservation criterion. In our case, we found that zero-initializing weights, as commonly proposed, is significantly suboptimal. In practice, we expand the weights matrices by initializing the new entries in the weight matrices randomly based on the norm of the weights of the already learned weights. In more detail, linear layers are expanded as:

$$\boldsymbol{W}' = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{W}_1 \\ \boldsymbol{W}_2 & \boldsymbol{W}_3, \end{pmatrix}$$

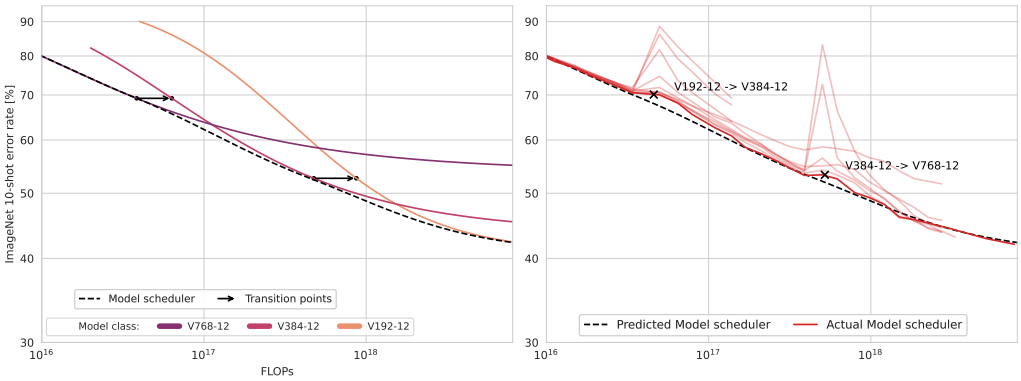


Figure 17. Different initialization schemes when expanding the width of the model. In practice, we set the variance of the new weights to be $\gamma\sigma^2$, where σ^2 is calculated from the pre-expanded weights $\boldsymbol{W}$, for different values of $\gamma \in \{0.25, 0.5, 0.75, 1, 1.25, 1.5\}$.

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3 \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$, and σ^2 is calculated from $\boldsymbol{W}$. This ensures better signal propagation in the network (He et al., 2015; Noci et al., 2022). The effect of this initialization can be important, but not detrimental, as illustrated in Fig. 17. When expanding the self-attention layers, we simply concatenate new heads, i.e. leave the heads that correspond to the previous embedding dimension unchanged. Again we stress that our method does not attempt to answer the question on *how* to initialize, and any method established in the literature can be used for this purpose.

We also include additional commonly used downstream performance metrics. In particular, we report 5/10-shot results on *ImageNet/Pets/Birds* as done in Zhai et al. (2022). Results can be seen in Fig. 18.

D. Adapting Multiple Model Shape Parameters Concurrently

We first present more results on which model configuration (number of parameters or patch size) leads to the most efficient training for different levels of performance in Fig. 20 and 21.

Motivated by these insights, we ask the question: *Can we change both the model size and patch size during training, leading to even greater training compute savings?*

We present preliminary experiments here, and more specifically in Fig. 19. We compare results when changing only the model width, only the patch size, or both the model width and patch size simultaneously. In every case, we find the transition points, when the model shape should be adapted, using our proposed methodology. Changing both patch size and model width leads to the most significant improvements. For simplicity and clarity, we here consider model sizes in the set $\{V192\text{-}12, V256\text{-}12, V384\text{-}12\}$ and patch sizes in the set $\{10, 20, 30, 40\}$.

We note that our method does not take into account momentary performance boost, when reducing the patch size and momentary performance deterioration when changing the model size, due to reasons highlighted in the main text. This justifies why changing only patch size can be better in some cases for the short term. As more compute is invested into the new model shape, these changes are counteracted.

E. Environmental Impact

To estimate the carbon footprint of our training, we follow the recipe detailed in Touvron et al. (2023a). Specifically, we approximate the Watt-hours (Wh) used as

$$\text{Wh} = \text{GPU-hours} \times \text{GPU-power-consumption} \times \text{PUE}$$

where PUE refers to Power Usage Effectiveness. Following Touvron et al. (2023a) we set this quantity to 1.1. In order to enable comparisons across different works, we use the national US average carbon intensity factor of $0.385\ kg\ CO_2eq/KWh$ and we thus estimate the amounts of carbon emissions as

$$tCO_2eq = MWh \times 0.385.$$

We compare our adaptively trained model against standard training of the compute-optimal model, in this case, the ViT Base model with patch size 8. The model requires $\approx$ 120 GPU-hours with an average consumption of $\approx 280W$ with the default training. Our adaptive training requires roughly 40% of GPU-hours, i.e. $\approx$ 48 GPU-hours while enjoying the same average consumption $\approx 280W$. This leads to $\approx 0.036MWh$ for ViT-Base and $\approx 0.014MWh$ for our adaptive training. Thus, the default training of the ViT Base model causes carbon emissions of $0.014tCO_2eq$ and our training $0.006tCO_2eq$.

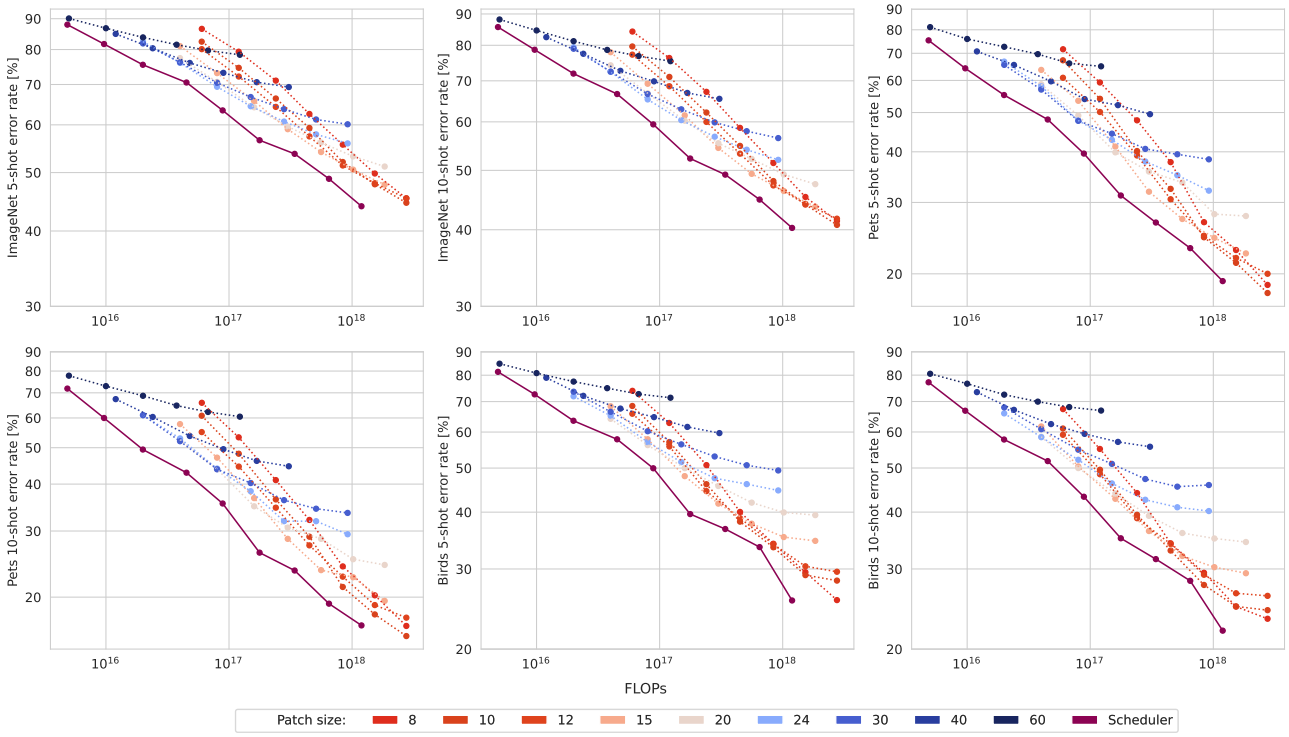


Figure 18. We include more downstream performance metrics for the $V384$-12 model.

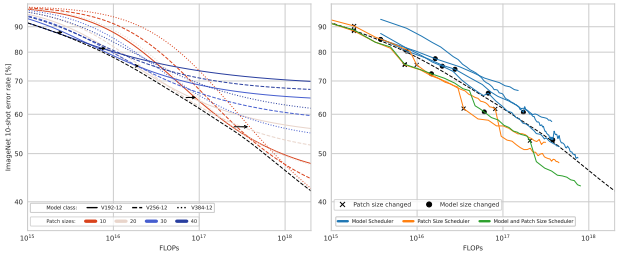


Figure 19. Changing both model width and patch size during training, further accelerates training.

F. Time Measurement

Although we focused on FLOPS, a similar hardware-aware analysis can take place, where the desired quantity to minimize is time instead of FLOPs. We note that time and FLOPs are usually highly correlated (Alabdulmohsin et al., 2023). This relationship also depends on the type of hardware and the mode it is operating in, i.e. whether we are memory-bound, whether data loading is the bottleneck etc. As an additional result, we replicate Fig. 6 from the main text but with time on the x-axis. Results can be seen in Fig. 22.

G. Discussion

Although our approach is not directly comparable or inspired by them, we discuss some further interesting connections.

Neural Architecture Search. The discovery of optimal architectures has also been explored in the line of work of neural architecture search (Elsken et al., 2019). Neural architecture search explores a collection of techniques to automate the selection of an optimal architecture. We are interested in more efficient training for a fixed architecture, the Transformer, that has established itself across different modalities.

Curriculum Learning. Curriculum learning argues that the order in which samples are presented plays a crucial role in the learning efficiency of a model (Soviany et al., 2022). The role of training data undoubtedly played a crucial role in the convergence speed (Sorscher et al., 2022). Recently, other techniques for data selection have been proposed to accelerate large-scale pre-training (Evans et al., 2023). Our technique does not filter or select data, just chooses to invest different amounts of compute to different data, based on the current stage of training. Population based training is also a related area of work (Jaderberg et al., 2017).

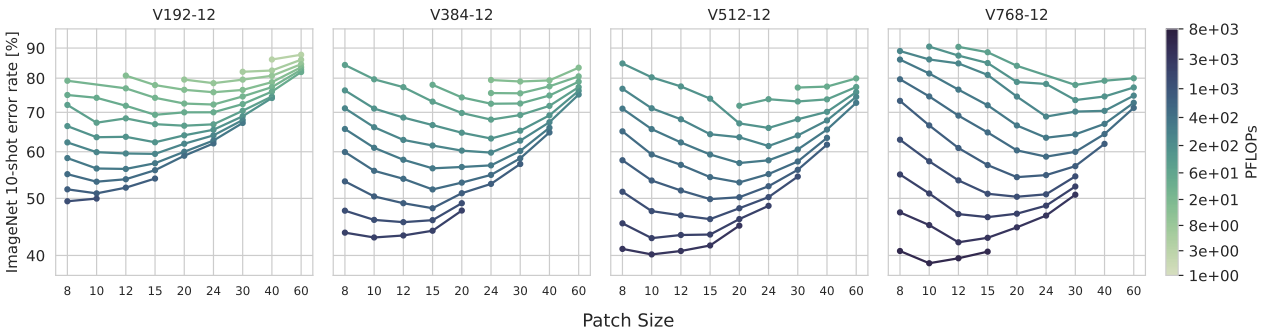


Figure 20. IsoFLOPs curves for different size ViTs trained with different constant patch sizes. Note how larger patch sizes are favored for smaller total FLOPs, while smaller patch sizes become more efficient as total FLOPs increase. Larger model sizes also become more favorable as total FLOPs increase.

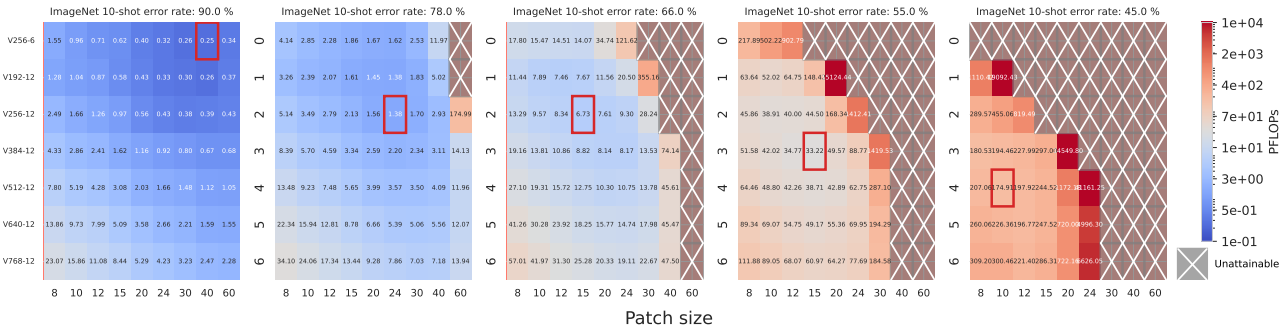


Figure 21. Values for $-\frac{\partial g_P(E)}{\partial E}$ in Eq. 2. Values indicate how many FLOPs are required for a proportionate increase in performance (i.e. drop in the error rate).

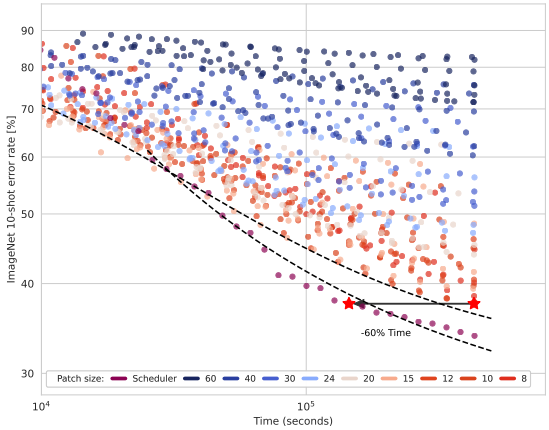


Figure 22. Same plot as Fig. 6 but with time instead of FLOPs in the x-axis.