Simple MViT: A Hierarchical Vision Transformer without the Bells-and-Whistles

Chay Ryali [*] [1] Yuan-Ting Hu [*] [1] Daniel Bolya [*] [1] [2] Chen Wei [1] [3] Haoqi Fan [1] Po-Yao Huang [1]
Vaibhav Aggarwal [1] Arkabandhu Chowdhury [1] Omid Poursaeed [1] Judy Hoffman [2] Jitendra Malik [1]
Yanghao Li [*] [1] Christoph Feichtenhofer [*] [1]

Abstract

Modern multi-stage vision transformers have added several vision-specific components in the pursuit of supervised classification performance. While these components lead to effective accuracies and attractive FLOP counts, this added complexity actually makes these transformers *slower* than their vanilla ViT counterparts. In this paper, we argue that this additional bulk is *unnecessary*. By pretraining with a strong visual pretext task (MAE), we can strip out all the bells-and-whistles from a state-of-the-art multi-stage vision transformer *without losing accuracy*. In the process, we create Simple MViT, an extremely simple multi-stage vision transformer that is *more accurate* than previous models while being *significantly faster* both at inference and during training. Simple MViT exceeds or matches the performance of the state-of-the-art in several image and video recognition tasks *while being much faster*.

1. Introduction

Since their introduction by Dosovitskiy et al. (2021) a few years ago, Vision Transformers (ViTs) have dominated several tasks in computer vision. While architecturally simple, their accuracy (Touvron et al., 2022) and ability to scale (Zhai et al., 2021) make them still a popular choice today. Moreover, their simplicity unlocks the use of powerful pretraining strategies such as MAE (He et al., 2022), which make ViTs computationally and data efficient to train.

However, this simplicity comes at a cost: by using the same spatial resolution and number of channels throughout the network, ViTs make inefficient use of their parameters. This is in contrast to prior "hierarchical" or "multi-scale" models (e.g., Krizhevsky et al. (2012); He et al. (2016)), which use fewer channels but higher spatial resolution in early stages with simpler features, and more channels but lower spatial resolution later in the model with more complex features.

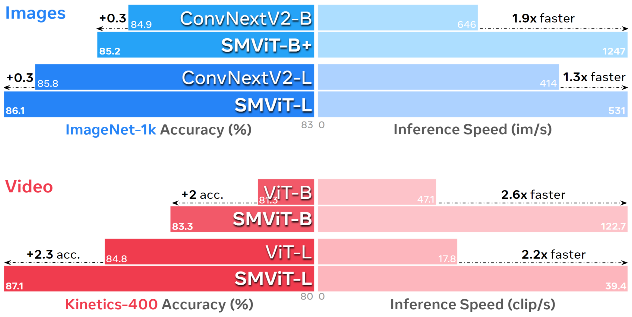


Figure 1. **Simple MViT (SMViT)** cuts out expensive specialized operations (e.g., convs) from hierarchical transformers to create a simple, efficient, and accurate model that is fast across many image and video tasks. Here compared to recent MAE-based works (Woo et al., 2023; Feichtenhofer et al., 2022). All speeds on A100 fp16.

Several domain specific vision transformers have been introduced that employ this hierarchical design, such as Swin (Liu et al., 2021) or MViT (Fan et al., 2021). However, in the pursuit of state-of-the-art results using fully supervised training on ImageNet-1K (an area where ViT has historically struggled), these models have become more and more complicated as they add specialized modules (e.g., cross-shaped windows in CSWin (Dong et al., 2022), decomposed relative position embeddings in MViTv2 (Li et al., 2022c)). While these changes produce effective models with attractive floating point operation (FLOP) counts, under the hood the added complexity makes these models *slower* overall.

We argue that a lot of this bulk is actually *unnecessary*. Because ViTs lack inductive bias after their initial patchify operation, many of the changes proposed by subsequent vision specific transformers serve to manually add spatial biases. But why should we slow down our architecture to add these biases, if we could just train the model to learn them instead? In particular, MAE pretraining has shown to be a very effective tool to teach ViTs spatial reasoning, allowing pure vision transformers to obtain good results on detection (Li et al., 2022b), which was a task previously dominated by models like Swin or MViT. Moreover, MAE pretraining is *sparse* and can be 4-10× as fast as normal supervised training, making it an already desirable alternative across many domains for more than just accuracy (He et al., 2022; Feichtenhofer et al., 2022; Huang et al., 2022b).

[*] Equal contribution [1] Meta AI, FAIR [2] Georgia Tech [3] Johns Hopkins University. Correspondence to: Chay Ryali <chayryali@meta.com>.

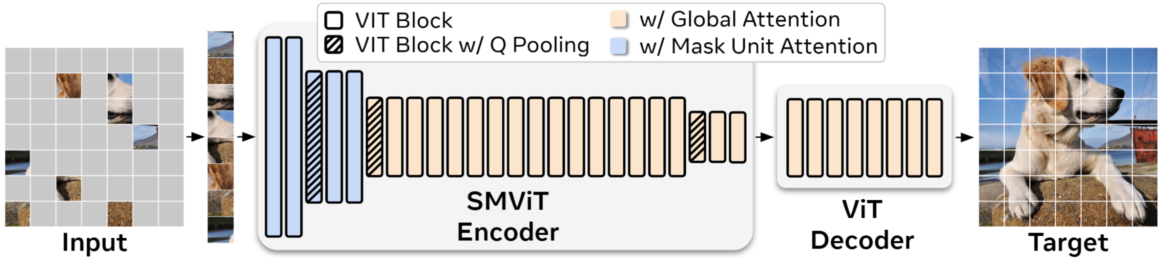


Figure 2. **SMViT Setup.** Modern hierarchical transformers like Swin (Liu et al., 2021) or MViT (Li et al., 2022c) are more parameter efficient than vanilla ViTs (Dosovitskiy et al., 2021), but end up slower due to overhead from adding spatial bias through vision-specific modules like shifted windows or convs. In contrast, we design SMViT to be as simple as possible. To add spatial bias, we opt to *teach* it to the model using a strong pretext task like MAE (pictured here) instead. SMViT consists entirely of standard ViT blocks. For efficiency, we use local attention within "mask units" (Fig. 4, 5) for the first two stages and global attention for the rest. At each stage transition, Q and the skip connection have their features doubled by a linear layer and spatial dimension pooled by a 2 × 2 maxpool. SMViT-B is shown here (see Tab. 2 for other configs).

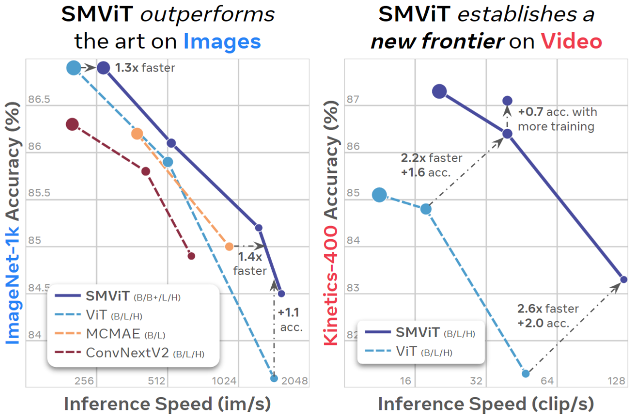


Figure 3. **Performance vs prior work.** SMViT compared to B, L, and H variants of SotA models that use MAE-like pretraining. On images, SMViT is faster and more accurate than even the most recent SotA (He et al., 2022; Gao et al., 2022; Woo et al., 2023), offering 30-40% speed-up compared to the best model at every scale. On video, SMViT represents a new class of performance, significantly improving accuracy, while *being over 2× faster* than popular ViT models. Marker size is proportional to FLOP count.

We test this hypothesis with a simple strategy: using some implementation tricks (Fig. 4), take an existing hierarchical ViT (e.g., MViTv2) and carefully *remove* non-essential components while training with MAE (Tab. 1). After tuning the MAE task to this new architecture (Tab. 3), we find that we can actually simplify or remove *all* of the non-transformer components, while *increasing in accuracy*. The result is an extremely efficient model with no bells-and-whistles: no convolutions, no shifted or cross-shaped windows, no decomposed relative position embeddings. Just a pure, simple hierarchical ViT that is both *faster and more accurate* than prior work across several model sizes, domains, and tasks.

Our Simple Multiscale Vision Transformer (SMViT) outperforms the SotA on images and *far exceeds* prior work on video while being *much faster* (Fig. 1) at every model scale (Fig. 3) and across extensive datasets and tasks (Sec. 5, 6).

2. Related Work

Vision transformers (ViTs) have attracted attention because of their massive success on several vision tasks including image classification (Dosovitskiy et al., 2021), video classification (Fan et al., 2021), semantic segmentation (Ranftl et al., 2021), object detection (Carion et al., 2020; Li et al., 2022b), video object segmentation (Duke et al., 2021), 3D object detection (Misra et al., 2021) and 3D reconstruction (Bozic et al., 2021). The key difference between vanilla ViT (Dosovitskiy et al., 2021) and prior convolutional neural networks (CNNs) (LeCun et al., 1998) is that ViT partitions images into, e.g., 16×16 pixel, *non-overlapping* patches and flattens the spatial grid into a 1D sequence, whereas CNNs maintain this grid over multiple stages of the model, reducing the resolution in each stage and introducing inductive biases such as shift equivariance. Recently, the field has shown an increased interest in hybrid methods (Fan et al., 2021; Liu et al., 2021; Li et al., 2022c; Dong et al., 2022) that combine transformers with convolution-like operations and the hierarchical stage structure of prior CNNs. This direction has shown success and has achieved state-of-the-art on various vision tasks. However, the caveat is that these models are actually slower than their vanilla ViT counterparts and convs are not easily compatible with popular self-supervised tasks such as masked image modeling. We address both of these issues in the creation of Simple MViT.

Masked pretraining has emerged as a powerful self-supervised learning pretext task for learning visual representations (Vincent et al., 2010; Pathak et al., 2016; Chen et al., 2020; He et al., 2022; Bao et al., 2022; Xie et al., 2022; Hou et al., 2022). Among previous works, Masked AutoEncoder (MAE, He et al. (2022)) takes advantage of vanilla ViTs, which allow any length of input, and thereby derives an efficient training regime using the *sparsity* of masked images. This greatly improves the training efficiency of masked pretraining, but adapting sparse training to hierarchical models is nontrivial, because the input is no longer laid out in a rigid 2D grid. There have been several attempts to enable hierarchical ViTs to use masked pretraining. MaskFeat (Wei et al., 2022) and SimMIM (Xie et al., 2022) replace masked patches with `[mask]` tokens, meaning most computation is wasted on non-visible tokens and training is incredibly slow. Huang et al. (2022a) introduce several techniques to enable sparsity in every component of the network, in the end creating a much more complicated model that doesn't improve much in accuracy. UM-MAE (Li et al., 2022a) uses a special masking strategy to allow for sparsity, but this restriction significantly hurts accuracy. MCMAE (Gao et al., 2022) uses masked convolution in the first couple of stages which obtains high accuracy but significantly reduces the efficiency of the model overall. We bypass all of these complicated techniques and restrictions by designing our architecture specifically for *sparse* MAE pretraining, thereby creating a powerful yet simple model.

3. Approach

Our goal is to create a powerful and efficient multiscale vision transformer that is, above all, *simple*. We argue that we do not need any specialized modules like convolution (Fan et al., 2021), shifted windows (Liu et al., 2021), or attention bias (Graham et al., 2021; Li et al., 2022c) to obtain high accuracy on vision tasks. This may seem difficult, as these techniques add much needed spatial (and temporal) biases that vanilla transformers (Dosovitskiy et al., 2021) lack. However, we employ a different strategy. While prior work adds spatial bias through complicated architectural changes, we opt to keep the model simple and *learn* these biases through a strong pretext task instead. To show the efficacy of this idea, we devise a simple experiment: take an existing hierarchical vision transformer and ablate its bells-and-whistles while training with a strong pretext task.

For the pretext task, we use Masked Autoencoders (MAE, He et al. (2022)), which has been shown effective in teaching ViTs localization capabilities for downstream tasks (e.g., detection (Li et al., 2022b)) by having the network reconstruct masked input patches (Fig. 2). Note that MAE pretraining is *sparse*—that is, masked tokens are *deleted* instead of being overwritten like in other masked image modeling approaches (Wei et al., 2022; Xie et al., 2022). This makes pretraining efficient, but poses a problem for existing hierarchical models as it breaks the 2D grid that they rely on (Fig. 4b). Moreover, MAE masks out individual tokens, which are large 16×16 patches for ViT, but only small 4×4 patches for most hierarchical models (Fig. 4a).

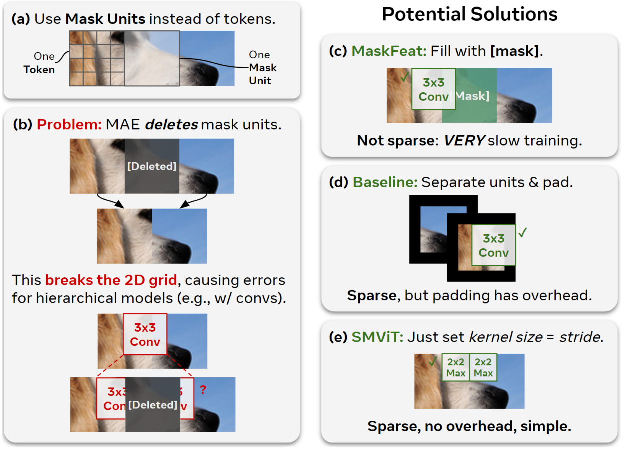

(a) Use Mask Units instead of tokens.
One Token
One Mask Unit
(b) Problem: MAE deletes mask units.
[Deleted]
This breaks the 2D grid, causing errors for hierarchical models (e.g., w/ convs).
3x3 Conv
[Deleted]
?
Potential Solutions
(c) MaskFeat: Fill with [mask].
3x3 Conv
Mask]
Not sparse: VERY slow training.
(d) Baseline: Separate units & pad.
3x3 Conv
Sparse, but padding has overhead.
(e) SMViT: Just set kernel size = stride.
2x2 Max
2x2 Max
Sparse, no overhead, simple.

Figure 4. **MAE for Hierarchical Models.** MAE is not compatible with multi-stage models, but we can apply some simple tricks to remedy this. While MAE masks individual tokens, tokens in multi-stage transformers start very small (e.g., 4×4 pixels), doubling size in each stage. (a) Thus, we mask coarser "mask units" (32×32 pixels) instead of tokens directly. (b) For efficiency, MAE is *sparse*, meaning it *deletes* what it masks (a problem for spatial modules like convs). (c) Keeping masked tokens fixes this, but gives up the potential 4-10× training speed-up of MAE. (d) As a baseline, we introduce a trick that treats mask units as a separate entities for convs, solving the issue but requiring undesirable padding. (e) In SMViT, we side-step the problem entirely by changing the architecture so the kernels can't overlap between mask units.

To address both of these issues, we opt to distinguish tokens from "mask units". As described in Fig. 4a, mask units are at the resolution we apply MAE masking, while tokens are the internal resolution of the model (like in Wei et al. (2022); Xie et al. (2022)). In our case, we mask 32×32 pixel regions, meaning one mask unit is 8×8 tokens at the start of the network. Once we have made this distinction, we can use a clever trick (Fig. 4d) to evaluate hierarchical models by treating mask units as contiguous, separate from other tokens. Thus, we can continue with our experiments and use MAE with an existing hierarchical vision transformer.

3.1. Preparing MViTv2

We choose MViTv2 as our base architecture, as its small 3×3 kernels are affected the least by the separate-and-pad trick described in Fig. 4d. Our final model is very simple, therefore we likely could have started with a different transformer and obtained a similar end result. We briefly review MViTv2 below.

MViTv2 (Li et al., 2022c) is a *hierarchical model*. That is, it learns multi-scale representations over its four stages. It starts by modeling low level features with a small channel capacity but high spatial resolution, and then in each stage trades channel capacity for spatial resolution to model more complex high-level features in deeper layers.

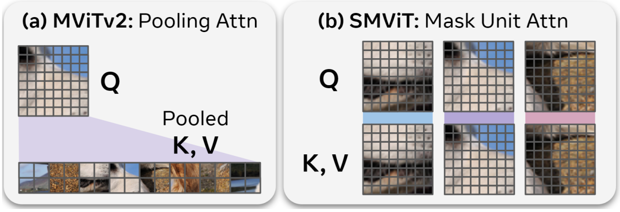


Figure 5. **Mask Unit Attention.** MViTv2 uses pooling attention (a) which performs global attention with a pooled version of K and V. This can get expensive for large inputs (e.g., for video), so we opt to replace this with "Mask Unit Attention" (b) which performs local attention within mask units (Fig. 4a). This has no overhead because we already group tokens into units for masking. We do not have to worry about shifting like in Swin (Liu et al., 2021), because we use global attention in stages 3 and 4 (Fig. 2).

A key feature of MViTv2 is *pooling attention* (Fig. 5a), wherein features are *locally aggregated*—typically using 3×3 convolution, before computing self-attention. In pooling attention, K and V are pooled to decrease computation in the first two stages, while Q is pooled to transition from one stage to the next by reducing spatial resolution. MViTv2 also features *decomposed relative position embeddings* instead of absolute ones and a *residual pooling* connection to skip between pooled Q tokens inside the attention blocks. Note that by default, pooling attention in MViTv2 contain convs with stride 1 even if no downsampling is required.

Applying MAE. Since MViTv2 downsamples by 2×2 a total of three times (Fig. 2) and because it uses a token size of 4×4 pixels, we employ a mask unit of size 32×32. This ensures that each mask unit corresponds to 8^2, 4^2, 2^2, 1^2 tokens in stages 1, 2, 3, 4 respectively, allowing each mask unit to cover at least one distinct token in each stage. Then as described in Fig. 4d, to make sure conv kernels do not bleed into deleted tokens, we shift the mask units to the batch dimension to separate them for pooling and then undo the shift afterward to ensure that self-attention is still global.

3.2. Simplifying MViTv2

In this section we *remove* non-essential components of MViTv2 while training with MAE. In Tab. 1, we find that we can remove or otherwise simplify *all* of them and still maintain high accuracy for image classification on ImageNet-1K. We use MViTv2-L to ensure our changes work at scale.

Relative Position Embeddings. MViTv2 swaps the absolute position embeddings in Dosovitskiy et al. (2021) for more powerful relative ones added to attention in *each block*. Technically, we could implement a version of this that is compatible with sparse pretraining, but doing so would add a lot of complexity. Instead, we opt to start our study here by undoing this change and using absolute position embeddings instead. As shown in Tab. 1, these relative position embeddings are not necessary when training with MAE. Further, absolute position embeddings are much faster.

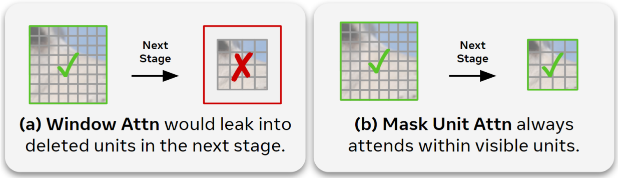
	Image		Video	
Setting	acc.	im/s	acc.	clip/s
MViTv2-L Supervised	85.3	219.8	80.5	20.5
SMViT-L MAE				
- replace rel pos with absolute *	85.6	253.3	85.3	20.7
- replace convs with maxpools *	84.4	99.9†	84.1	10.4†
- delete stride=1 maxpools *	85.4	309.2	84.3	26.2
- set kernel size equal to stride	**85.7**	369.8	**85.5**	29.4
- delete q attention residuals	85.6	374.3	**85.5**	29.8
- replace kv pooling with MU attn	85.6	**531.4**	**85.5**	**39.4**

Table 1. **Simplifying MViTv2.** MViTv2 employs several architectural tweaks to perform well on supervised training. By progressively removing them in Sec. 3.2, we find these bells-and-whistles are *unnecessary* when training with a strong pretext task such as MAE. In the process, we create an extremely simple model (Fig. 2) that is accurate while being significantly faster. We report fp16 inference speed for ImageNet-1K and Kinetics-400 on an A100. Our final SMViT-L in gray. *Requires the separate-and-pad trick described in Fig. 4d. †PyTorch's maxpool3d interacts unfavorably with this trick.

Removing Convolutions. Next, we aim to remove the convs in the model, which are vision specific modules and add potentially unnecessary overhead. We first attempt to replace every conv layer with maxpools (shown by Fan et al. (2021) to be the next best option), which itself is fairly costly. The result drops accuracy by over 1% on images, but this is to be expected as we've also replaced all of the extra stride=1 convs with maxpools, which impacts the features significantly (with padding and small mask units, this in effect performs a relu on every feature map). Once we also delete those additional stride=1 maxpools, we nearly return to the accuracy we had before, while speeding up the model by 22% for images and 27% for video. At this point, the only pooling layers that remain are for Q at stage transitions and for KV pooling in the first two stages.

Removing Overlap. The remaining maxpool layers still have a kernel size of 3×3, necessitating the use of the separate-and-pad trick in Fig. 4d during both training and inference. However, as shown in Fig. 4e, we can avoid this problem entirely if we just do not let these maxpool kernels overlap. That is, if we set the kernel size equal to stride for each maxpool, we can use sparse MAE pretraining *without* the separate-and-pad trick. This speeds up the model by 20% on image and 12% on video while increasing accuracy, likely due to not having pad.

Removing the Attention Residual. MViTv2 adds a residual connection in the attention layer between Q and the output to assist in learning its pooling attention. However, at this point we've minimized the number of layers, making attention easier to learn. Thus, we can safely remove it.



(a) Window Attn would leak into deleted units in the next stage.

(b) Mask Unit Attn always attends within visible units.

Figure 6. **Mask Unit Attn vs. Window Attn.** Window attention (a) performs local attention within a *fixed* size window. Doing so would potentially overlap with deleted tokens during sparse MAE pretraining. In contrast, Mask Unit attention (b) performs local attention within individual mask units, no matter their size.

Mask Unit Attention. At this point, the only specialized module left is pooling attention. Pooling Q is necessary to maintain a hierarchical model, but KV pooling is only there to reduce the size of the attention matrix in the first two stages. We can remove this outright, but it would considerably increase the computational cost of the network. Instead, we replace it with an implementationally trivial alternative: local attention within a mask unit.

During MAE pretraining, we already have to separate out mask units at the start of the network (see Fig. 2). Thus the tokens are already neatly grouped by units once they arrive at attention. We can then simply perform local attention within these units with no overhead. While this "Mask Unit attention" is local instead of global like pooling attention (Fig. 5), K and V were only pooled in the first two stages, where global attention isn't as useful. Thus, as shown in Tab. 1, this change has no impact on accuracy but increases throughput by quite a lot—up to 32% on video.

Note that mask unit attention is distinct from window attention because it adapts the window size to the size of mask units at the current resolution. Window attention would have a fixed size throughout the network, which would leak into deleted tokens after a downsample (see Fig. 6).

Simple MViT. The result of these changes is an extremely simple and efficient model, which we denote "Simple MViT". SMViT is *2.4× faster* on images and *5.1× faster* on video than the MViTv2 we started with and is actually *more accurate* because of MAE. Furthermore, because SMViT supports sparse pretraining, the results in Tab. 1 are extremely fast to obtain. In fact, to obtain superior accuracy on images, SMViT-L is *3× faster* to train than a supervised MViTv2-L (Fig. 7). For video, Wei et al. (2022) report 80.5% using a *cut down* version of MViTv2 with double the KV stride in the first 3 stages. Compared to this model, our SMViT-L obtains 84.0% in 400 pretrain epochs (Tab. 3e) while being *3.6× faster* to train (Fig. 7). All benchmarks throughout the paper are on an A100 with fp16 (as this setting is most useful in practice) unless noted otherwise.

While we used SMViT-L for the experiments in this section, we can of course instantiate it in different sizes, e.g. Tab. 2.

model	#Channels	#Blocks	#Heads	FLOPs	Param
SMViT-B	[96-192-384-768]	[2-3-16-3]	[1-2-4-8]	9G	51M
SMViT-B+	[112-224-448-896]	[2-3-16-3]	[2-4-8-16]	13G	70M
SMViT-L	[144-288-576-1152]	[2-6-36-4]	[2-4-8-16]	40G	213M
SMViT-H	[256-512-1024-2048]	[2-6-36-4]	[4-8-16-32]	125G	672M

Table 2. **Configuration for Simple MViT variants**. #Channels, #Blocks and #Heads specify the channel width, number of SMViT blocks and heads in each block for the four stages, respectively. FLOPs are measured for image classification with 224 × 224 input. The stage resolutions are [56^2, 28^2, 14^2, 7^2]. We introduce B+ for more direct comparison against prior work with slower B models.

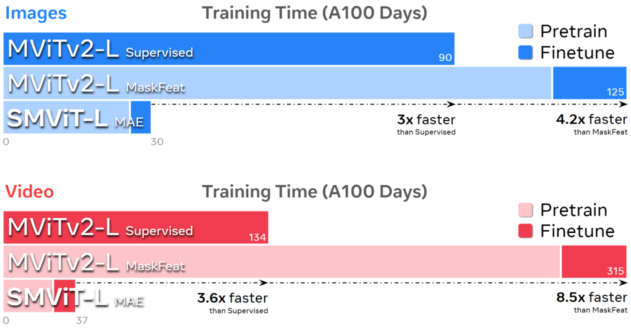


Figure 7. **Training time.** Measured in half precision A100 days. Our SMViT is *significantly* faster to train than MViTv2 due to being more efficient and benefiting from sparse pretraining (as opposed to MaskFeat). Here, supervised uses 300 epochs for ImageNet-1K and 200 epochs for Kinetics-400, while MaskFeat and MAE use 400 epochs for pretraining and 50 for finetuning.

4. MAE Ablations

In this section, we ablate MAE pretraining settings in SMViT for both images and video, using ImageNet-1K (IN1K, Deng et al. (2009)) and Kinetics-400 (K400, Kay et al. (2017)). Like in He et al. (2022); Feichtenhofer et al. (2022), we abkate using our large model, SMViT-L, to ensure that our method works at scale. We evaluate performance by finetuning. All metrics are top-1 accuracies using standard evaluation protocols—a single (resized) center crop on IN1K and 3 spatial × 5 temporal views on K400.

Multi-Scale decoder. While He et al. (2022); Feichtenhofer et al. (2022) use the tokens from the *last* block of the encoder as the input to the decoder, SMViT being *hierarchical* permits more flexibility: as in Gao et al. (2022), we can use *multi-scale* information by fusing representations from all stages, which brings large gains in both modalities (Tab. 3a).

Masking ratio. Feichtenhofer et al. (2022) find video to require a much higher masking ratio than images, suggesting higher information redundancy. We observe a similar trend in Tab. 3b with optimal masking ratios of 0.6 for images but 0.9 for video. Our optimal ratio for images, 0.6, is slightly lower than the 0.75 used in He et al. (2022). We expect this is due to increased difficulty of the pretext task from

multi-scale	image	video
✗	85.0	83.8
✔	**85.6**	**85.5**

(a) **Multi-Scale Decoder.** SMViT being *hierarchical*, using multi-scale information for decoding brings significant gains.

mask	image	mask	video
0.5	85.5	0.75	84.9
0.6	**85.6**	0.9	**85.5**
0.7	85.3	0.95	84.4

(b) **Mask ratio.** High masking ratios lead to good performance, with video benefiting from higher masking than image modality.

dpr	image	video
0.0	85.2	84.5
0.1	85.6	85.4
0.2	**85.6**	**85.5**
0.3	85.5	85.2

(c) **Drop path rate.** Surprisingly, we find drop path important during MAE pretraining, especially for video, unlike in He et al. (2022); Feichtenhofer et al. (2022).

depth	image	video
4	85.5	84.8
8	**85.6**	**85.5**
12	85.5	85.4

(d) **Decoder depth.** We find that a deeper decoder than in Feichtenhofer et al. (2022) works better for video.

epochs	image	video
400	85.6	84.0
800	85.8	85.5
1600	**86.1**	86.4
3200	-	**87.1**

(e) **Pretraining schedule.** Our pretraining follows the same trend as He et al. (2022), benefiting significantly from longer training.

Table 3. **Ablating MAE pretraining with Simple MViT-L.** For each ablation, we use 400 (800) epochs of sparse MAE pretraining for IN1K (K400) and 50 epochs of dense finetuning unless otherwise noted. Our default[†] settings are marked in gray. For design choices not ablated here, we find the defaults in (He et al., 2022; Feichtenhofer et al., 2022) to be appropriate. † default pretraining length for the rest of the paper is 1600 epochs as in previous work, unless otherwise noted.

using a 32 × 32 mask unit instead of 16 × 16 as in He et al. (2022). Interestingly, we find the same 0.9 masking ratio to be appropriate for video as in Feichtenhofer et al. (2022). This could be because they actually find 0.95 to work optimally if allowed to train twice as long. With our increased task difficulty, 0.9 works out to be best.

Droppath rate. The original MAE pretraining recipes (He et al., 2022; Feichtenhofer et al., 2022) explicitly do not use drop path (Huang et al., 2016) during pretraining, instead opting to only do so while finetuning. However, our SMViT-L has twice the depth of a ViT-L model: 48 for SMViT-L *vs.* 24 for ViT-L. While each layer individually has a lower parameter count, due to the sheer depth of SMViT, there could be a significant benefit from drop path.

In Tab. 3c, we ablate applying drop path during pretraining (finetuning employs drop path by default) and find significant gains. This is surprising because it means that without drop path, Simple MViT can overfit to the MAE task.

Decoder depth. We find a significant benefit from a deeper decoder than previous work use for video (Feichtenhofer et al., 2022), see Tab. 3d. This brings the decoder for video in line with images (He et al., 2022).

Pretraining schedule. Several masked image modeling approaches (He et al., 2022; Wei et al., 2022) have found benefits from longer pretraining schedules, often using up to 1600 epochs. In Tab. 3e, we observe the same trend for SMViT, increasing +0.5% over 400 epochs on IN1K. In fact, SMViT's accuracy at 400ep is +0.7% higher than ViT-L MAE (84.9%) at the same number of epochs but only +0.2% higher at 1600 epochs—suggesting that SMViT is a more *efficient* learner. On K400, even with only 800 epochs of pretraining, SMViT outperforms the previous SotA result that uses 1600 epochs (84.8%). Gains from longer training saturate less quickly on video, with a large 0.9% gain from 800 epochs to 1600 epochs and beyond.

backbone	pretrain	acc.	FLOPs (G)	Param
ViT-B	MAE	81.5	180×3×7	87M
SMViT-B	MAE	84.0	**101**×3×5	**51M**
SMViT-B+	MAE	**84.8**	133×3×5	70M
MViTv2-L	-	80.5	**377**×1×10	218M
MViTv2-L	MaskFeat	84.3	**377**×1×10	218M
ViT-L	MAE	85.2	598×3×7	304M
SMViT-L	MAE	**87.1**	413×3×5	**213M**
ViT-H	MAE	86.6	1193×3×7	**632M**
SMViT-H	MAE	**87.2**	**1158**×3×5	672M

Table 4. **K400 results**. SMViT improves on previous SotA by a large amount, while being lighter and faster. FLOPs are reported as inference FLOPs × spatial crops × temporal clips.

5. Video Results

We report our results on video recognition. All models input 16 frames of 224^2 pixels unless otherwise specified.

Kinetics-400,-600,-700. In Tab. 4, we compare SMViT trained with MAE to the SotA on Kinetics-400 (Kay et al., 2017) at a system level. We compare to MViTv2-L (Li et al., 2022c) pretrained with MaskFeat (Wei et al., 2022) and ViT (Dosovitskiy et al., 2021) pretrained with MAE on video (Feichtenhofer et al., 2022). SMViT-L brings large

backbone	pretrain	acc.	FLOPs (G)	Param
MViTv2-L	Sup, IN-21K	85.8	377×1×10	218M
MViTv2-L	MaskFeat	86.4	**377**×1×10	218M
SMViT-L	MAE	**87.9**	413×3×5	**213M**

(a) **Kinetics-600** video classification

backbone	pretrain	acc.	FLOPs (G)	Param
MViTv2-L	Sup, IN-21K	76.7	377×1×10	218M
MViTv2-L	MaskFeat	77.5	**377**×1×10	218M
SMViT-L	MAE	**79.5**	413×3×5	**213M**

(b) **Kinetics-700** video classification

Table 5. **K600 and K700 results**. SMViT improves over SotA by a large margin. FLOPs reported as inference FLOPs × spatial crops × temporal clips. Approaches using extra data are de-emphasized.

gains (+**2.3**%) over previous SotA (Feichtenhofer et al., 2022), while using ~45% *fewer* flops, being ~43% *smaller* and **2.2**× faster (Fig. 3). In fact, SMViT-L *significantly outperforms* (+**2**%) models *one tier higher*, while being **3**× *smaller*. SMViT-L achieves a gain of +**6.6**% over the corresponding MViTv2-L supervised baseline. Going one tier up in size, SMViT-H improves performance over previous SotA by +**2.2**%, *establishing a new SotA* for 224^2 without external data. We show similarly large improvements over the art on K600 (+**1.5**%) and K700 (+**2.0**%) in Tab. 5.

Something-Something-v2 (SSv2). In Tab. 6, we compare our SMViT with the current art on SSv2 (Goyal et al., 2017b) at a system level: MViTv2-L (Li et al., 2022c) pretrained with MaskFeat (Wei et al., 2022) and ViT (Dosovitskiy et al., 2021) pretrained with MAE on video (Tong et al., 2022). When pretrained on K400, SMViT-L outperforms the runner-up method MaskFeat by +**0.6**%, but SMViT is *dramatically* more efficient, using 16 frames at 224^2 resolution *vs*. 40 frames at 312^2 resolution in MaskFeat, effectively using **3.4**× *fewer* FLOPs. When pretrained on SSv2, SMViT-L achieves **75.1**%, outperforming ViT-L pretrained with MAE, by +**0.8**%, while using ~45% *fewer* flops and being ~43% *smaller*. Our SMViT-L$_{32}$ model further achieves **76.5**%, SotA among approaches trained only on SSv2.

Transferring to action detection (AVA). We evaluate transfer learning of a K400 pretrained SMViT on action detection using AVA v2.2 dataset (Gu et al., 2018). In Tab. 7 we compare SMViT with SotA methods, MViTv2 with MaskFeat (Wei et al., 2022) and ViT with MAE on video (Tong et al., 2022) at system level, and report mean average precision (mAP). Our SMViT-L outperforms an MAE pretrained ViT-L by +**2.8**% and an MViTv2-L$_{40,312}$ MaskFeat by +**1.3**% mAP while SMViT-L has fewer FLOPs and parameters. Our SMViT-H outperforms an MAE pretrained ViT-H by +**1.3**% mAP, establishing a new SotA.

backbone	pretrain	acc.	FLOPs (G)	Param
K400 pretrain				
ViT-L	supervised	55.7	598×3×1	304M
MViTv2-L$_{40,312}$	MaskFeat	74.4	2828×3×1	218M
ViT-L	MAE	74.0	597×3×2	305M
SMViT-L	MAE	74.7	**413**×3×1	**213M**
SMViT-L	MAE	**75.0**	**413**×3×2	**213M**
SSv2 pretrain				
ViT-L	MAE	74.3	597×3×2	305M
SMViT-L	MAE	74.9	**413**×3×1	**213M**
SMViT-L	MAE	**75.1**	**413**×3×2	**213M**
ViT-L$_{32}$	MAE	75.4	1436×3×1	305M
SMViT-L$_{32}$	MAE	**76.5**	**1029**×3×1	**213M**

Table 6. **SSv2 results** pretrained on Kinetics-400 and SSv2. SMViT improves over SotA by a large margin. We report inference FLOPs × spatial crops × temporal clips.

backbone	pretrain	mAP	FLOPs (G)	Param
ViT-L	supervised	22.2	598	304M
MViTv2-L$_{40,312}$	MaskFeat	38.5	2828	218M
ViT-L	MAE	37.0	597	305M
SMViT-L	MAE	**39.8**	**413**	**213M**
ViT-H	MAE	39.5	1192	**633M**
SMViT-H	MAE	**40.8**	**1158**	672M

Table 7. **AVA v2.2 results** pretrained on Kinetics. SMViT improves over SotA by a large margin. All inference FLOPs reported with a center crop strategy following Fan et al. (2021).

6. Image Results

We first evaluate performance on IN1K and then transfer to other image recognition, detection, and segmentation tasks.

6.1. Performance on ImageNet-1K

In Tab. 8, we perform a system-level comparison of SMViT trained with MAE to relevant prior work. First, we observe that the supervised MViTv2 baselines are already quite strong, with MViTv2-B (L) reaching 84.4 (85.3) top-1 accuracy—better than several approaches that use pretraining (e.g. ViT-B MAE). This showcases the significant benefits that convolutions give in the supervised setting, *especially* at the *base* model size. Remarkably, even at this size, SMViT-B *without* using any bells-and-whistles (e.g., convs), is able to reach **84.5**% (slightly) outperforming MViTv2-B; MCMAE-B achieves a higher accuracy, but the model is significantly heavier. Our SMViT-B+ model handily outperforms it in both speed (Fig. 3) and accuracy.

At one tier higher, SMViT-L MAE reaches an accuracy of **86.1**%, a significant +**0.8**% gain over MViTv2-L; it also (slightly) outperforms ViT-L MAE, which is 42% larger and has 1.6× the FLOPs, by +**0.2**%. Note that while we adopted the MAE pretraining in this work due to its efficient

backbone	pretrain	acc.	FLOPs (G)	Param
ViT-B		82.3	18	87M
Swin-B		83.3	15	88M
MViTv2-B		84.4	10	52M
ViT-B	BEiT, DALLE	83.2	18	87M
ViT-B	MAE	83.6	18	87M
ViT-B	MaskFeat	84.0	18	87M
Swin-B	SimMIM	83.8	15	88M
MCMAE-B	MCMAE	85.0	28	88M
SMViT-B	MAE	84.5	**9**	**51M**
SMViT-B+	MAE	**85.2**	13	70M
ViT-L		82.6	62	304M
MViTv2-L		85.3	42	218M
ViT-L	BEiT, DALLE	85.2	62	304M
ViT-L	MAE	85.9	62	304M
ViT-L	MaskFeat	85.7	62	304M
Swin-L	SimMIM	85.4	**36**	**197M**
MCMAE-L	MCMAE	**86.2**	94	323M
SMViT-L	MAE	86.1	40	213M
ViT-H	MAE	**86.9**	167	**632M**
SMViT-H	MAE	**86.9**	**125**	672M

Table 8. **ImageNet-1K** comparison to previous MIM approaches. We de-emphasize approaches using extra data and indicate the source of extra data.

backbone	iNat17	iNat18	iNat19	Places365
ViT-B	70.5	75.4	80.5	57.9
SMViT-B	73.3	77.9	83.0	58.9
SMViT-B+	**74.7**	**79.9**	**83.1**	**59.2**
ViT-L	75.7	80.1	83.4	59.4
SMViT-L	**76.8**	**80.9**	**84.3**	**59.6**
ViT-H	79.3	83.0	**85.7**	59.8
SMViT-H	**79.6**	**83.5**	**85.7**	**60.0**
ViT-H$_{448}$	83.4	86.8	88.3	60.3
SMViT-H$_{448}$	**83.8**	**87.3**	**88.5**	**60.6**

Table 9. **Transfer learning** on iNaturalists and Places datasets.

sparse pretraining, SMViT-L is readily compatible with complementary, *orthogonal* approaches, e.g. using an EMA teacher (El-Nouby et al., 2021; Baevski et al., 2022).

6.2. Transfer learning experiments

Here, we perform transfer learning experiments on downstream classification, detection, and segmentation tasks.

Classification on iNaturalists and Places. In Tab. 9 we evaluate transfer learning performance on downstream iNaturalist (Van Horn et al., 2018) and Places (Zhou et al., 2014) datasets. We finetune the ImageNet-1K pretrained SMViT on iNaturalist 2017, 2018, and 2019, and Places 365. SMViT consistently outperforms ViT pretrained with MAE (He et al., 2022) , indicating that our SMViT-L and SMViT-H architectures are effective outside of just ImageNet.

backbone	pretrain	AP^{box}	AP^{mask}	FLOPs	params	time
Swin-B	Sup, 21K	51.4	45.4	0.7T	109M	164ms
MViTv2-B	Sup, 21K	53.1	47.4	0.6T	73M	208ms
Swin-B	Sup	50.1	44.5	0.7T	109M	**164ms**
MViTv2-B	Sup	52.4	46.7	**0.6T**	**73M**	208ms
ViTDet-B	MAE	51.6	45.9	0.8T	111M	201ms
SMViT-B	MAE	52.1	46.2	**0.6T**	**73M**	173ms
SMViT-B+	MAE	**53.3**	**47.1**	0.7T	113M	205ms
Swin-L	Sup, 21K	52.4	46.2	1.1T	218M	243ms
MViTv2-L	Sup, 21K	53.6	47.5	1.3T	239M	447ms
MViTv2-L	Sup	53.2	47.1	1.3T	239M	447ms
ViTDet-L	MAE	**55.6**	**49.2**	1.9T	331M	396ms
SMViT-L	MAE	55.0	48.6	**1.2T**	**236M**	**340ms**

Table 10. **COCO object detection and segmentation** using Mask-RCNN. All methods are following the training recipe from Li et al. (2022b) and pretrained on ImageNet-1K by default. Methods using ImageNet-21K pretraining are de-emphasized. Test time is measured on a single V100 GPU with full precision.

Object detection and segmentation on COCO. We finetune Mask R-CNN (He et al., 2017) with different pretrained backbones on the COCO dataset (Lin et al., 2014). We report AP^{box} and AP^{mask} for object detection and instance segmentation. We utilize the training recipe following ViTDet (Li et al., 2022b) and incorporate multi-scale features from SMViT with a Feature Pyramid Network (FPN, Lin et al. (2017)) as described in the original paper.

Our SMViTs with MAE pretraining demonstrate a strong scaling behavior when compared models with supervised pretraining such as MViTv2 (Li et al., 2022c), while being consistently faster. For example, SMViT-L is **+1.8** AP^{box} higher than MViTv2-L (55.0 *vs.* 53.2) with a **24**% reduction in inference time. Even when compared to MViTv2 using ImageNet-21K pretraining, SMViT-L still performs **+1.4** AP^{box} better than MViTv2-L.

When compared to the state-of-the-art method, ViTDet, our SMViT models achieve comparable results while having faster inference and a lower operation count. For example, SMViT-B shows +0.5 higher AP^{box} than ViTDet-B with **34**% fewer parameters and **15**% lower inference time. Additionally, SMViT-B+ achieves **+1.7** boxAP improvements when having similar inference time and model complexity to ViTDet-B. For the large model, SMViT-L is consistently faster than ViTDet-L with only a slightly lower accuracy.

7. Conclusion

In this work we create a simple hierarchical vision transformer by taking an existing one and removing all its bells-and-whistles while supplying the model with spatial bias through MAE pretraining. The resulting architecture, SMViT, is more effective than current work on image recognition tasks and surpasses the state-of-the-art on video tasks. We hope SMViT can allow future work to do more, faster.

config	value
optimizer	AdamW (Loshchilov & Hutter, 2019)
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.95$
weight decay	0.05
learning rate	8e-4
learning rate sch.	cosine decay (Loshchilov & Hutter, 2017)
warmup epochs (Goyal et al., 2017a)	120
epochs	1600
repeated sampling (Hoffer et al., 2020)	4
augmentation	hflip, crop $[0.5, 1]$
batch size	512
num. decoder blocks	8
drop path (Huang et al., 2016)	0.1 (B), 0.2 (L), 0.3 (H)

(a) Pre-training

config	SMViT-B	SMViT-L	SMViT-H
optimizer		AdamW	
optimizer momentum		$\beta_1, \beta_2{=}0.9, 0.999$	
weight decay		0.05	
learning rate	5e-5	5e-5	2.5e-5
learning rate schedule		cosine decay	
warmup epochs		10	
epochs	150	100	100
repeated sampling		2	
augmentation		RandAug (7, 0.5) (Cubuk et al., 2020)	
batch size		256	
mixup (Zhang et al., 2018)		0.8	
cutmix (Yun et al., 2019)		1.0	
label smoothing (Szegedy et al., 2016)		0.1	
drop path	0.2	0.2	0.3
dropout (Srivastava et al., 2014)	0.3	0.5	0.5
layer-wise decay (Clark et al., 2020)	0.85	0.9	0.925

(b) Fine-tuning

Table 11. Default settings on Kinetics-400, 600, 700.

config	value
optimizer	AdamW
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.95$
weight decay	0.05
learning rate	8e-4
learning rate schedule	cosine decay
warmup epochs	30
epochs	1600
augmentation	crop $[0.5, 1]$
batch size	1024
gradient clipping	0.02
num. decoder blocks	8
drop path	0.2

(a) Pre-training

config	values
optimizer	SGD
weight decay	1e-4
learning rate	0.16
learning rate schedule	cosine decay
warmup epochs	3
epochs	40
augmentation	RandAug (7, 0.5)
batch size	256
mixup	0.8
cutmix	1.0
label smoothing	0.1
drop path	0.1
dropout	0.5
layer-wise decay	0.875

(b) SMViT-L fine-tuning

config	values
optimizer	SGD
weight decay	1e-4
learning rate	0.08
learning rate schedule	cosine decay
warmup epochs	3
epochs	40
augmentation	RandAug (7, 0.5)
batch size	128
drop path	0.2
dropout	0.5
layer-wise decay	0.875

(c) SMViT-L$_{32}$ fine-tuning

Table 12. Settings on SSv2.

config	values
optimizer	SGD
weight decay	1e-8
learning rate	6.0
learning rate schedule	cosine decay
warmup epochs	5
epochs	30
batch size	128
drop path	0.4
dropout	0.5
layer-wise decay	0.875 (L), 0.9 (H)

Table 13. SMViT fine-tuning settings on AVA.

A. Implementation Details

A.1. Video Experiments

Kinetics-400, -600, -700. Our settings mainly follow Feichtenhofer et al. (2022). We report below the pre-training and fine-tuning settings for our main results on the Kinetics-400 (Kay et al., 2017), -600 (Carreira et al., 2018) and -700 (Carreira et al., 2019) human action datasets. Epochs are always reported as effective epochs (Feichtenhofer et al., 2022), i.e. accounting for repeated sampling.

Something-Something-v2 (SSv2). We evaluate SMViT-L on the SSv2 dataset (Goyal et al., 2017b). SSv2 is a dataset focusing on human-object interaction classification. We pre-train SMViT-L on either Kinetics 400 or SSv2 and fine-tune on SSv2. We report the top-1 classification accuracy in Tab. 6. We provide further details about the pre-training and fine-tuning settings on SSv2 in Tab. 12.

AVA v2.2. We perform transferring experiments on AVA v2.2 (Gu et al., 2018) for human action localization in video. We adopt a detection framework following (Feichtenhofer et al., 2019) for human action localization. We extract ROI

config	value
optimizer	AdamW
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.95$
weight decay	0.05
learning rate	8e-4
learning rate sch.	cosine decay
warmup epochs	40
epochs	1600
augmentation	RandomResizedCrop
batch size	4096
num. decoder blocks	8
drop path	0.2 (B, B+), 0.2 (L), 0.3 (H)

(a) Pre-training

config	SMViT-B, B+	SMViT-L	SMViT-H
optimizer	AdamW		
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.999$		
weight decay	0.05		
learning rate	2.5e-5		
learning rate schedule	cosine decay		
warmup epochs	5		
epochs	100	50	50
augmentation	RandAug (9, 0.5)		
batch size	1024		
mixup	0.8		
cutmix	1.0		
label smoothing	0.1		
drop path	0.2	0.2	0.3
layer-wise decay	0.7	0.85	0.85

(b) Fine-tuning

Table A1. Default settings on ImageNet-1K.

features from the feature map of the last layer in SMViT and pool the ROI features via spatial max-pooling. We then use a linear classifier trained with cross entropy loss to predict the action class. We use the center crop for SMViT in the evaluation and report the mAP in Tab. 7. We use a K400 pre-trained and fine-tuned SMViT in the experiments. We provide details about the fine-tuning setting on AVA v2.2 in Tab. 13.

A.2. Image Experiments

ImageNet-1K. Our settings mainly follow He et al. (2022). We report below the pre-training and fine-tuning settings for our main results.

Transfer learning experiments on iNaturalists and Places. We conduct transfer learning experiments on classification datasets including iNaturalist2017, iNaturalist2018, iNaturalist2019 (Van Horn et al., 2018) and Places365 (Zhou et al., 2014). Following (He et al., 2022), we adjust learning rate, training epochs on each dataset. We search the layer-wise decay among 0.875, 0.9 and 0.925, drop path rate between 0.1 to 0.5, and the dropout rate among 0.1, 0.2 and 0.3. For SMViT-H$_{448}$, we set the learning rate decay of the positional embedding to 0.5 instead of following the layer-wise decay rule.

COCO. We use the Mask R-CNN (He et al., 2017) framework in Detectron2 (Wu et al., 2019) for object detection and instance segmentation experiments on the COCO dataset. Similar to ViTDet (Li et al., 2022b), we use 2 hidden convolution layers for the RPN and 4 hidden convolution layers for the RoI heads for SMViT and all comparison detection methods. These layers are followed by LayerNorm layers. For the training recipe, we follow ViTDet to use input size as 1024×1024 with large-scale jittering (LSJ) (Ghiasi et al., 2021). We don't use the layer-wise decay during training. Additional hyperparamters can be found in Tab. A2.

config	values
optimizer	AdamW
optimizer momentum	$\beta_1, \beta_2{=}0.9, 0.999$
weight decay	0.1
learning rate	3.5e-5 (B), 3e-5 (B+, L)
learning rate schedule	step-wise decay
epochs	100
augmentation	LSJ [0.1, 2.0]
batch size	64
drop path	0.1 (B, B+), 0.4 (L)

Table A2. SMViT fine-tuning settings on COCO.

A.3. Speed Benchmarking

We use an NVIDIA A100 40GB GPU, PyTorch v1.12.1 and CUDA 11.4 to benchmark speed for all baselines and our approach, unless otherwise mentioned. For each of the methods, we measure purely the model inference throughput. We compute the throughput with various batch sizes, and report the throughput with the optimal batch size. We use half precision (FP16) to run speed benchmarking unless otherwise specified. We set the input resolution to 224×224×3 for image benchmarking, and 224×224×3 with 16 frames as a clip for video benchmarking. To measure the training time, we measure the speed of a forward-backward pass on a single gpu and extrapolate the total training time according to the size of the dataset and the number of training epochs, ignoring dataloading and communication overheads when training with multiple GPUs.

References

Baevski, A., Hsu, W.-N., Xu, Q., Babu, A., Gu, J., and Auli, M. data2vec: A general framework for self-supervised learning in speech, vision and language. In *ICML*, 2022.

Bao, H., Dong, L., and Wei, F. Beit: Bert pre-training of image transformers. *ICLR*, 2022.

Bozic, A., Palafox, P., Thies, J., Dai, A., and Nießner, M.

Transformerfusion: Monocular rgb scene reconstruction using transformers. *NeurIPS*, 2021.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *ECCV*, 2020.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

Carreira, J., Noland, E., Hillier, C., and Zisserman, A. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., and Sutskever, I. Generative pretraining from pixels. In *ICML*, 2020.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. RandAugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, 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. In *ICLR*, 2021.

Duke, B., Ahmed, A., Wolf, C., Aarabi, P., and Taylor, G. W. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *CVPR*, 2021.

El-Nouby, A., Izacard, G., Touvron, H., Laptev, I., Jegou, H., and Grave, E. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. Multiscale vision transformers. In *ICCV*, 2021.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. Slowfast networks for video recognition. In *ICCV*, 2019.

Feichtenhofer, C., Fan, H., Li, Y., and He, K. Masked autoencoders as spatiotemporal learners. *NeurIPS*, 2022.

Gao, P., Ma, T., Li, H., Lin, Z., Dai, J., and Qiao, Y. Mcmae: Masked convolution meets masked autoencoders. In *NeurIPS*, 2022.

Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.-Y., Cubuk, E. D., Le, Q. V., and Zoph, B. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017a.

Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017b.

Graham, B., El-Nouby, A., Touvron, H., Stock, P., Joulin, A., Jégou, H., and Douze, M. Levit: a vision transformer in convnet's clothing for faster inference. In *ICCV*, 2021.

Gu, C., Sun, C., Ross, D. A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., et al. AVA: A video dataset of spatio-temporally localized atomic visual actions. In *CVPR*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *ICCV*, 2017.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Hoffer, E., Ben-Nun, T., Hubara, I., Giladi, N., Hoefler, T., and Soudry, D. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020.

Hou, Z., Sun, F., Chen, Y.-K., Xie, Y., and Kung, S.-Y. Milan: Masked image pretraining on language assisted representation. *arXiv preprint arXiv:2208.06049*, 2022.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *ECCV*, 2016.

Huang, L., You, S., Zheng, M., Wang, F., Qian, C., and Yamasaki, T. Green hierarchical vision transformer for masked image modeling. In *NeurIPS*, 2022a.

Huang, P.-Y., Xu, H., Li, J., Baevski, A., Auli, M., Galuba, W., Metze, F., and Feichtenhofer, C. Masked autoencoders that listen. *NeurIPS*, 2022b.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

Li, X., Wang, W., Yang, L., and Yang, J. Uniform masking: Enabling mae pre-training for pyramid-based vision transformers with locality. *arXiv preprint arXiv:2205.10063*, 2022a.

Li, Y., Mao, H., Girshick, R., and He, K. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022b.

Li, Y., Wu, C.-Y., Fan, H., Mangalam, K., Xiong, B., Malik, J., and Feichtenhofer, C. Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, 2022c.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *CVPR*, 2017.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Misra, I., Girdhar, R., and Joulin, A. An end-to-end transformer model for 3d object detection. In *ICCV*, 2021.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

Ranftl, R., Bochkovskiy, A., and Koltun, V. Vision transformers for dense prediction. In *CVPR*, 2021.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Tong, Z., Song, Y., Wang, J., and Wang, L. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. In *NeurIPS*, 2022.

Touvron, H., Cord, M., and Jégou, H. Deit iii: Revenge of the vit. *ECCV*, 2022.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. The iNaturalist species classification and detection dataset. In *CVPR*, 2018.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A., and Bottou, L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 2010.

Wei, C., Fan, H., Xie, S., Wu, C.-Y., Yuille, A., and Feichtenhofer, C. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, 2022.

Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., and Xie, S. Convnext v2: Co-designing and scaling convnets with masked autoencoders. *arXiv preprint arXiv:2301.00808*, 2023.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. Detectron2, 2019.

Xie, Z., Zhang, Z., Cao, Y., Lin, Y., Bao, J., Yao, Z., Dai, Q., and Hu, H. Simmim: A simple framework for masked image modeling. In *CVPR*, 2022.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *CVPR*, 2021.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *NeurIPS*, 2014.