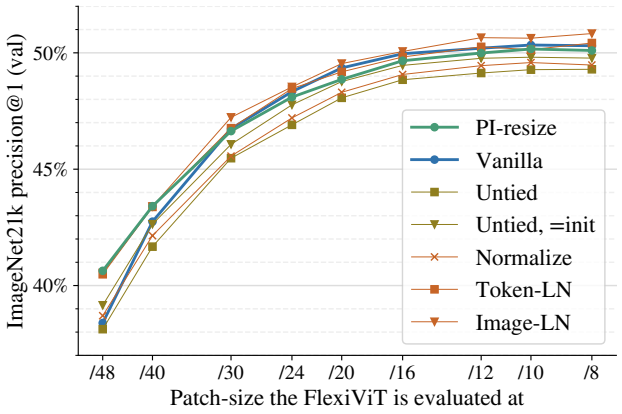

Figure 14. **Options for dealing with patch-embeddings**. Supervised training from-scratch on ImageNet-21k. Vanilla (bilinear) is the simplest method and works well. PI-resize further improves the large-patch case and provides other advantages (see text). Untied means learning separate patch-embedding kernels for each size, and does not work too well. Normalize, Token-LN, and Image-LN all but require modifications to the model making it incompatible with standard ViT.

Besides providing more details and results on various sections as mentioned in the main paper, we also provide full numerical (tabular) results of all figures in Appendix R in order to facilitate reproduction/comparison in future work.

Finally, more details about the alternative ways of flexibility discussed in Section 7 are provided in Appendix O.

A. More details on flexible patch-sizes

In this section, we further elaborate on many details of flexible patch-sizes. We provide results for alternative ways of dealing with flexible patch-sizes when one does not care about preserving model architecture in Appendix A.1. We provide a detailed derivation of PI-resize in Appendix A.2, and show some PI-resize matrices in Appendix A.3. We further show some visualizations of patch-embedding weights, both raw and resized, in Appendix A.4

A.1. Alternatives for dealing with flexible patch-size

Besides bilinear resizing (called **Vanilla**) or **PI-resizing** the patch-embedding weights to deal with variable patch-sizes, there are a few other alternatives which we discuss and compare here.

Untied weights for each size, i.e. having separate trainable parameter buffers for each patch-size.

Untied, =init is the same as above, but initializing all patch embedding weights to the same (PI-resized) values as a reference initialization. In this setting, the model is still compatible with standard ViT models at initialization time, however, the parameters can, and do, diverge during training, resulting in a non-standard ViT architecture.

Normalize simply l2-normalizes the tokens computed by the patch-embedding individually to unit-norm. This solves any norm-related issues in a simple, parameter-free way, but is incompatible with pre-trained standard ViT models.

Token-LN and Image-LN add a LayerNorm [2] right after the patch-embedding and differ only in which axis they perform the normalization. Again, this solves the norm-related issues, but does add learnable parameters and is incompatible with pre-trained standard ViT models.

A comparison of all these variants is performed in the label-supervised training setup on ImageNet-21k following [51], but training for 90 epochs. The result, presented in Figure 14, indicates that plain resizing and PI-resizing are among the best solutions, but have the added benefit of resulting in standard ViT models. Furthermore, not visible in this figure, both *Untied* variants displayed slightly unstable training curves in the first half of training, while all other variants train smoothly.

A.2. PI-resize derivation

We can rewrite the objective function in Eq. (2) as follows:

$$\begin{aligned}
&\mathbb{E}_{x\sim\mathcal{X}}\left[(\langle x,\omega\rangle - \langle Bx,\hat{\omega}\rangle)^2\right] = \\
&\mathbb{E}_{x\sim\mathcal{X}}\left[(x^T(\omega - B^T\hat{\omega}))^2\right] = \\
&\mathbb{E}_{x\sim\mathcal{X}}\left[((\omega - B^T\hat{\omega})^T x)(x^T(\omega - B^T\hat{\omega}))\right] = \\
&(\omega - B^T\hat{\omega})^T\mathbb{E}_{x\sim\mathcal{X}}\left[xx^T\right](\omega - B^T\hat{\omega}) = \\
&\|\omega - B^T\hat{\omega}\|^2_\Sigma,
\end{aligned} \tag{6}$$

where $\|v\|^2_\Sigma = v^T\Sigma v$ and $\Sigma = \mathbb{E}_{x\sim\mathcal{X}}xx^T$ is the (uncentered) covariance matrix of $\mathcal{X}$. In case when $\mathcal{X} = \mathcal{N}(0, I)$, we recover the standard euclidean norm $\|\omega - B^T\hat{\omega}\|^2$.

Finally, we note that the pseudoinverse matrix recovers a least squares solution to a linear system of equations:

$$(B^T)^+\omega \in \arg\min_{\hat{\omega}} \|\omega - B^T\hat{\omega}\|^2. \tag{7}$$

We can also derive an analytic solution for an arbitrary $\Sigma = \mathbb{E}_{x\sim\mathcal{X}}xx^T$. Note that $\|v\|^2_\Sigma = v^T\Sigma v = (\sqrt{\Sigma}v)^T\sqrt{\Sigma}v = \|\sqrt{\Sigma}v\|^2$. Then, we have

$$\|\omega - B^T\hat{\omega}\|^2_\Sigma = \|\sqrt{\Sigma}\omega - \sqrt{\Sigma}B^T\hat{\omega}\|^2. \tag{8}$$

The optimal solution is then given by

$$(\sqrt{\Sigma}B^T)^+\sqrt{\Sigma}\omega \in \arg\min_{\hat{\omega}} \|\omega - B^T\hat{\omega}\|^2. \tag{9}$$

A.3. Visualization of some PI-resize matrices

We visualize an upscaling and a downscaling matrix for both bilinear and PI-resize operations for a visual comparison in Figure 15.

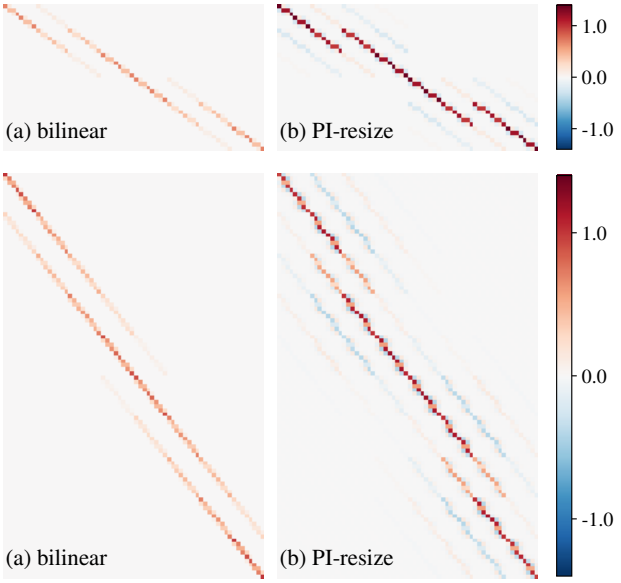

Figure 15. **Visualization of resize matrices.** Each row corresponds to the weights that are used in the computation of one output pixel. Off-diagonals correspond to pixels below/above the current one. We can see that PI-resize does include negative weights, has a larger *receptive field*, and uses overall larger weights than bilinear resizing.

A.4. Visualization of patch-embedding weights

PCA of patch embeddings (see [16]) in Figs. 32 to 36.

B. The "underlying" parameter shapes

FlexiViT does introduce two new hyper-parameters which were not present in the original ViT architecture: the size of the *underlying* patch-embedding weights and position-embeddings (i.e. learned params, before resize).

However, both of these parameters have (maybe surprisingly) little influence on the final performance of the model, as long as they are in a "reasonable" range. We verified these in two different settings.

For the patch-size parameter, since it is affected by the resize method used, we perform the ablation in the full FlexiViT setup. Figure 16 (b) shows that there is no notable difference across all evaluation sizes, and hence we stick to the (initially arbitrary) default of 32 across all experiments.

For the position embedding parameter, we ran an early experiment with ViT-S/16 trained from scratch on ImageNet-1k following [4]. Figure 16 (c) shows that in general, even for plain ViT training, this approach could be taken and training curves are mostly unaffected by in-graph bilinear resizing of position embeddings.

C. Distribution of patch-size sampling

During roughly the first half of this project, we sampled patch-size from a distribution which is not uniform, but samples patch-sizes between 16 and 30 up to three times more than patch-sizes outside this range. This "triangular" distribution was based purely on gut-feeling, and once we verified that it is no better than uniform sampling (the experiment is shown in Figure 16 (a)), we decided to use a uniform distribution for simplicity. We further decided to avoid the costly re-running of all experiments we did so far, and thus some experiments in the main paper were done with the "triangular" distribution. However, in all direct comparisons presented throughout the paper, the curves being directly compared always were trained in the exact same way.

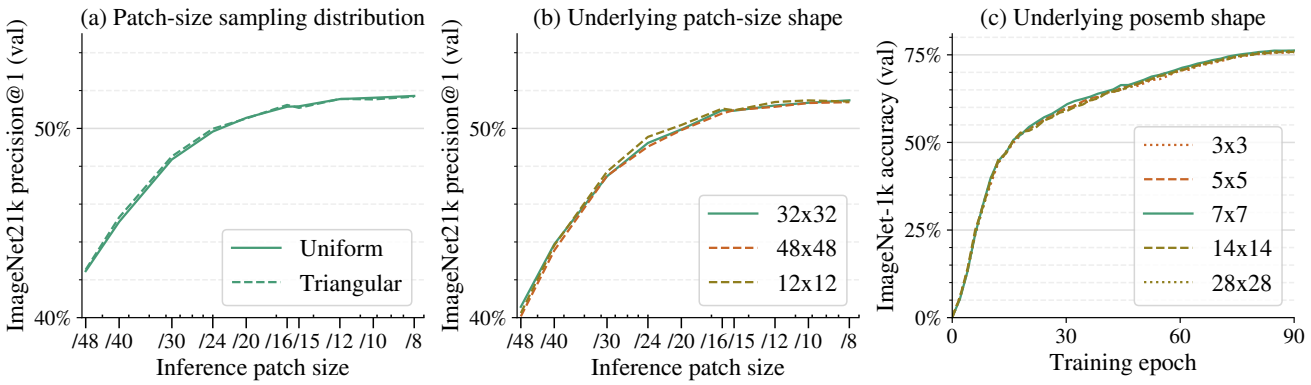

Figure 16. **Ablation of a few unimportant hyperparameters,** since changing their value shows no noteworthy difference in the result. **(a)** Sampling distribution of patch sizes. Uniform is preferable for its simplicity, but we use "Triangular" (more weight on mid-sized patches, less weight on extremely large or small ones) sometimes for legacy reasons. **(b)** Varying the shape of the underlying patch embedding parameter in the full FlexiViT training setup. **(c)** Varying the shape of the underlying position embeddings in a smaller supervised training of a ViT-S/16 baseline following [4].

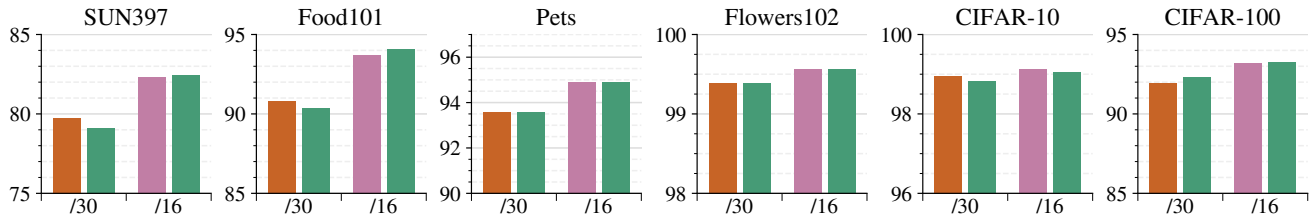

Figure 17. **More transfer results.** The legend is the same as in Figure 7. The main take-away is that the results are qualitatively the same across a wide range of image classification tasks and it is safe to use a pre-trained FlexiViT model in lieu of a ViT model even when one only uses it at a fixed patch size after transfer.

D. Detais on resource-efficient transfer

For finetuning FlexiViT models on the Imagenet-1k dataset we generally follow the transfer learning setup from [16]. We use SGD momentum optimizer, with the initial learning rate of 0.03 and cosine learning rate decay. We also reduce the learning rate for the pretrained parameters by a factor of 10. We optimize for 20000 steps with inception crop and flip left-right augmentation, using batch size 512 and input image size of 480×480.

E. Using pre-trained FlexiViT models: more details and results

In this section, we provide more details and full results for the scenario described in Section 4: using pre-trained FlexiViT models.

For more details on flexified training procedures discussed in Section 5, we redirect to Appendix M for flexified transfer learning, Appendix G for flexified contrastive image-text learning (LiT and CLIP), and Appendix N for flexified open-vocabulary detection (OWL-ViT), including results on ELEVATER.

E.1. Using FlexiViT models for transfer

For transfer, we follow the simple BiT-HyperRule [27]. In short, we transfer for a relatively brief number of steps (500 for flowers and pets, 2500 for food101 and sun, and 10 000 for CIFAR), using the SGD optimizer with a momentum of 0.9, no weight decay, no dropout, and no other augmentations besides flips and random crops. We initialize the new classification layer to all-zeros and use a short learning-rate warmup, both of these having as effect to better preserve the pre-trained weights. The only setting which differs from [27] is that we do sweep the learning-rate across $\{0.03, 0.01, 0.003, 0.001\}$ for each task individually. We show results for all 6 datasets we used in Figure 17.

E.2. Using FlexiViT models in LiT for image-text tasks

We use the same 4B image-text pairs dataset as in [67] to train the LiT models, and use identical hyper parameters as LiT models. The only difference is to use the FlexiViT model here, instead of a standard ViT model. FlexiViT models are transferred at a fixed sequence length, i.e. 30^2 or 16^2, with 240×240 image resolution. We report zero-shot classification results on ImageNet [47], zero-shot image-to-text / text-to-image retrieval results on MS-COCO [11] and Flickr30K [43], in Figure 18.

E.3. Using FlexiViT models in OWL-ViT for zero-shot detection

To evaluate FlexiViT backbones for open-vocabulary detection (Figure 7), we compare OWL-ViTs initialized with either fixed or flexible LiT-B models. Specifically, the backbones start with either a fixed or flexibly pre-trained ViT image model, which is then then frozen and LiT-tuned [67] with a text model at a fixed patch size (the same as final evaluation, i.e. 30^2 and 16^2 respectively). OWL-ViTs using these backbones are then trained at a fixed patch size (30^2 or 16^2) at a resolution of 720×720 on Objects365 [49] and Visual Genome [30] as in the original paper [38]. We report mean average precision (AP) on LVIS [20].

E.4. Using FlexiViT models in UViM for panoptic segmentation

Apart from using FlexiViT model weights for the initialization, we follow the setup of the original UViM setup [28] as close as possible. In particular, we train the model on the COCO panoptic dataset [11] and report the standard PQ metric [26] on the official validation split. We train the model for 200 epochs, using the custom adafactor optimizer variant [66] with the base learning rate of 0.001. The learning rate for the pretrained part of the model is decreased by a factor of 10. The input image size is 512×512. More details on the training setting can be found in the UViM paper [28] and official repository of the UViM model [5].

E.5. Using FlexiViT models in Segmenter for semantic segmentation

We follow the experimental setup of Segmenter [52] for end-to-end finetuning of Vision Transformer with linear decoder. For data augmentation during training, we apply ran-

[5] https://github.com/google-research/big_vision/tree/main/big_vision/configs/proj/uvim

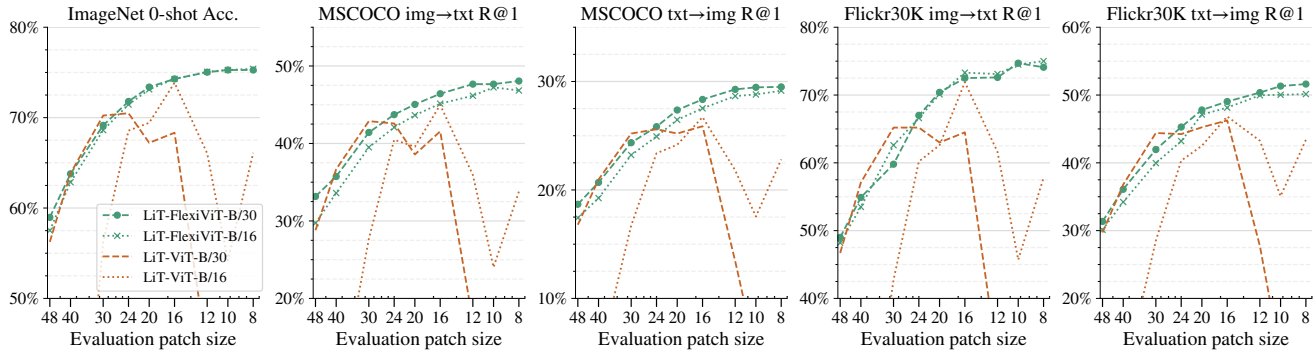

Figure 18. **Transfer of FlexiViT at a fixed patch size.** The LiT-ViT baselines match LiT-FlexiViT on the sequence length it has been trained for (16^2 or 30^2), but performance drops quickly when using a different inference sequence length. We observe that the LiT-FlexiViT models work well across different inference sequence lengths, even though only a fixed sequence length is used during LiT transfer. This effect is similar to that described in Section 4.2 and further explored in Figure 19.

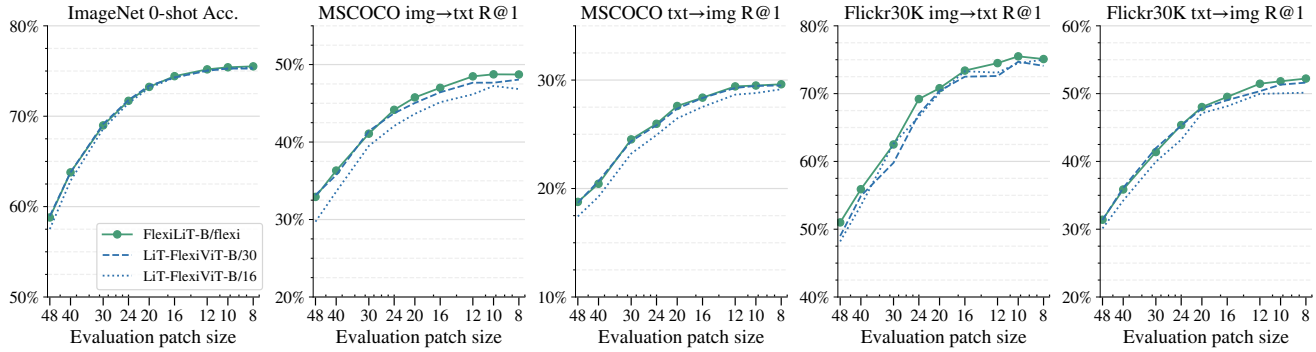

Figure 19. **FlexiLiT.** We observe consistent but marginal boost when flexifying the LiT training by randomizing patch size during LiT-tuning. This again shows that the LiT-FlexiViT baseline performs strongly and it allows fast transfer: transferred cheaply using a large patch size and served at smaller patch sizes for free.

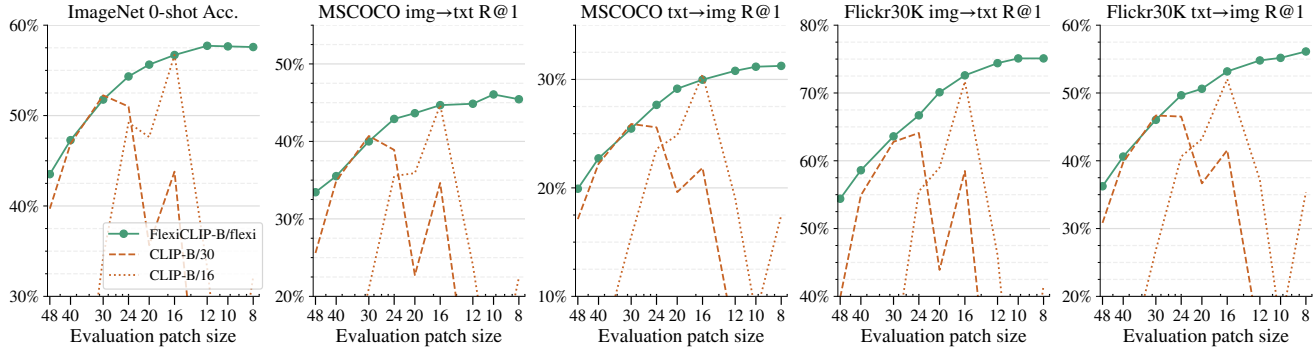

Figure 20. **FlexiCLIP.** We observe very similar conclusions as in FlexiViT, that the FlexiCLIP model works well across a large range of evaluation patch sizes during inference. It is interesting that a single fixed sequence length text tower, is able to produce embeddings aligning well with a FlexiViT image tower that allows multiple sequence lengths.

dom resizing of the image with a random ratio between 0.5 and 2.0, photometric augmentation and random horizontal flipping. We randomly crop images to 480×480 resolution with padding, therefore preserving aspect ratio. We use the 480×480 resolution for both Cityscapes and ADE20k. We train for 127 epochs with minibatch size of 16 (resulting in 160k iterations on ADE20k). We use the "poly" learning rate decay schedule and sweep the base learning rate in $\{1e-4, 3e-4, 8e-4\}$ for all of our runs. Weight-decay is kept fixed at 0.01. At evaluation time, we use the sliding-window with a resolution 480×480 to handle varying image sizes during inference. Table 3 row 6 in [52] reports 48.06

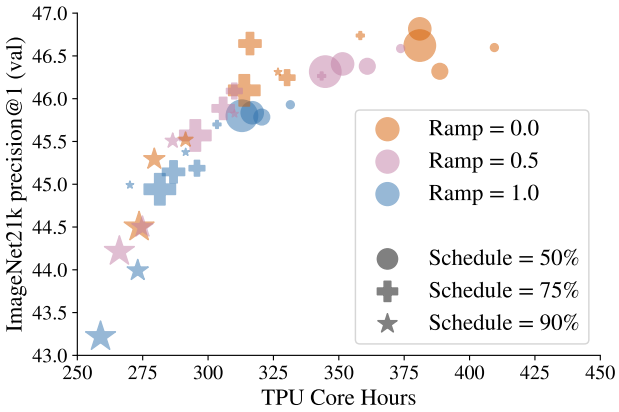

Figure 21. A detailed look into the impact of the larger patch size (size of the markers), schedule, and ramp periods on both compute and accuracy. See Appendix H for details.

mIoU. Average of 6 runs in our codebase in the same setting gives 47.6 ± 0.4 mIoU. The results on ADE20k are provided in Table 7.

F. Full numerical ImageNet-1k-only results

We provide full numerical results of the FlexiViTs trained purely on ImageNet-1k and presented in Figure 2, including on additional robustness and OOD test-sets in Tabs. 1 to 4.

G. FlexiLiT and FlexiCLIP results

FlexiLiT follows exactly the same setup as described in Section E.2, but randomizes patch sizes during LiT training. We show more FlexiLiT results in Figure 19.

For FlexiCLIP, we simply replace the pre-trained and frozen backbone in FlexiLiT with a random initialized and unfrozen backbone, which corresponds to the *uu* setting in [67] and is equivalent to CLIP [44]. Figure 20 shows the same conclusions in this setting. It is reassuring that the patch size randomization does not hinder learning both image and text representations from scratch simultaneously.

H. Accelerate pre-training

As discussed in Section 5.4, one can use FlexiViT's method of varying the patch size and resizing the embedding weights to pretrain ViTs faster. To do that, we specify a *curriculum*: a sequence $(p_k)_{k=1}^K$ of probability distributions over the patch sizes along with a mapping $c : \mathbb{N} \to [K]$ that identifies which distribution p_k to use at training step t. In this section, we use the sequence of patch sizes: $(48, 40, 30, 24, 16)$ to demonstrate the potential savings in compute.

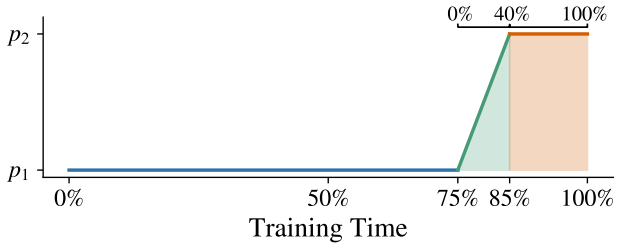

Figure 22. This figure illustrates how the distribution of patch sizes changes when accelerating pretraining (see Section 5.4) and Appendix H. Here, the first 75% of training steps use the larger patch size p_1 while the remaining 25% are used for the desired/target patch size. The ramp period in this figure is 40%.

We set 16×16 to be the desired/target patch size during evaluation. Hence, we compare a curriculum-based approach of pretraining ViTs (denoted FasterViT) with the standard ViT/B/16 architecture in which the patch size is fixed to 16×16 throughout training. Except for the variable patch sizes and the embedding layers, both architecture are otherwise identical.

We use a simple curriculum in this evaluation. Specifically, we have one large patch size (e.g. 48×48), which we denote by p_0 and the desired patch size 16×16, which we denote by p_1. We initially use the larger patch size before swtiching to the smaller patch size using FlexiViT's PI-resize. We experiment with three different schedules: $(50\%, 75\%, 90\%)$, where $\text{schedule} = 75\%$ means that 75% of the training time uses the larger patch size alone. Instead of switching immediately between patch sizes, we also include an optional ramp period as illustrated in Figure 22. A ramp period of, say, 40% means that 40% of the time allocated to the smaller patch size is used to transition gradually between the two distributions. We experiment with three ramp periods: $(0\%, 50\%, 100\%)$. We run each experiment independently and plot the resulting compute and accuracy in Figure 12(x). As shown in the figure, FasterViT achieves the same level of accuracy as standard ViTs but with less compute, although the improvement in not quite significant.

In Figure 21, we provide a detailed view into the impact of the three hyperparameters: patch size, schedule, and ramp period. Not surprisingly, increasing the fraction of time allocated to the larger patch size (e.g. by setting $\text{schedule} = 90\%$), improves compute at the expense of accuracy.

I. Further analysis of cosine similarities between token representations across scales

In Section 6, we measure cosine similarity between the representation of a seed token at one scale and representations of other tokens at other scales, demonstrating that the

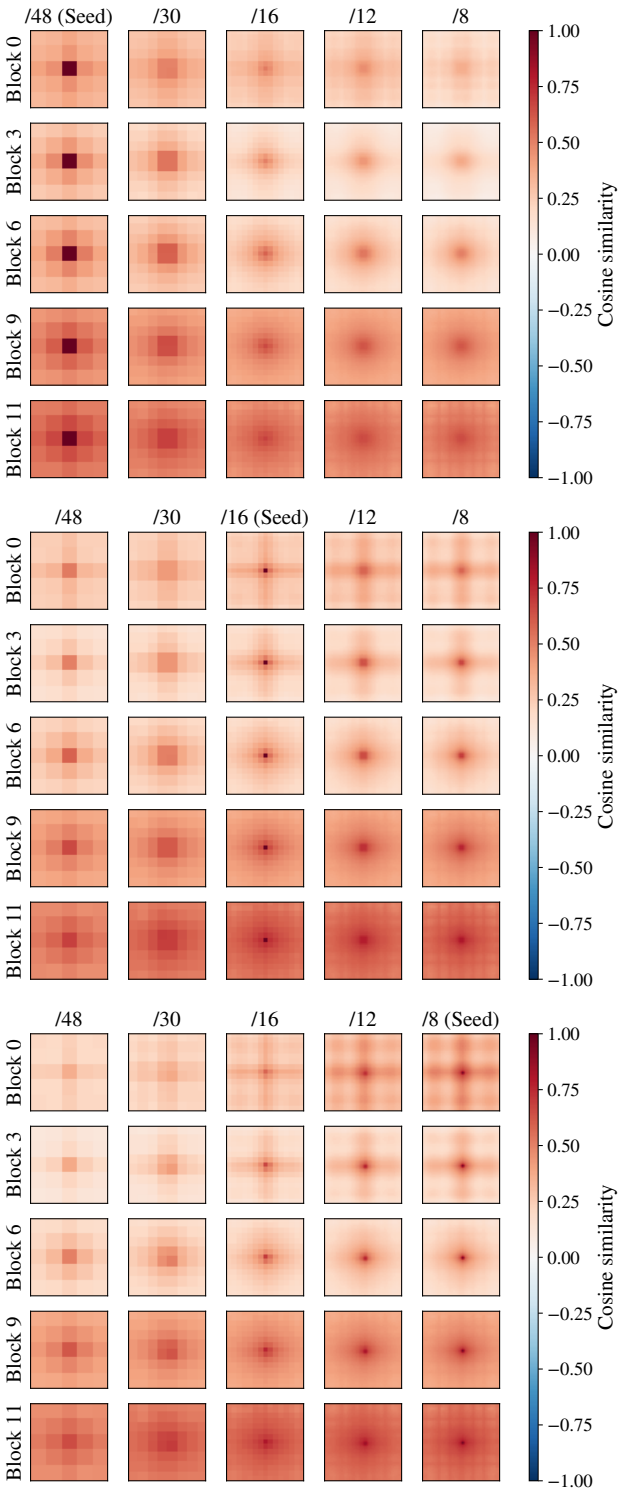

Figure 23. **Supplemental analysis of similarities of token representations across scales.** Each row shows the cosine similarity between a seed token at the center of the feature map at one scale and tokens at different scales, for a single block. In the top group of plots, the seeds are taken from FlexiViT-B/48; in the middle, from FlexiViT-B/16; and at the bottom, from FlexiViT-B/8.

most similar tokens at other scales are those that represent the same spatial location. in Figure 23, we provide additional results for seed tokens at other grid sizes and from additional blocks. Results are consistent with those in the main text.

J. Ensembling FlexiViT predictions across scales

We ensemble FlexiViT models by averaging models' logits.[6] When ensembling all models, we attain 51.7% precision@1 on our ImageNet-21K validation set, which is slightly worse than the accuracy achieved at the largest grid size/smallest patch size (52.0%).

We further explore ensembles of pairs of models in Figure 25. Agreement between models evaluated at large patch sizes is relatively low, with models evaluated at the largest two patch sizes (/48 and /40) agreeing on only 67.4% of examples (Figure 25 middle). Nonetheless, ensembling these models provides no accuracy improvement over simply using FlexiViT-B/40; both strategies achieve 45.8% ImageNet-21K precision@1 (Figure 25 left).

When comparing the computational cost of ensembles of FlexiViT predictions across scales to applying FlexiViT at a single scale, a single scale is nearly always better (Figure 25 right). The only configuration where the accuracy of a two-scale ensemble exceeds the accuracy of a single scale with the same computational footprint is the

[6]We have also explored ensembling based on averaging output probabilities. We find that results are nearly identical, but on average slightly worse.

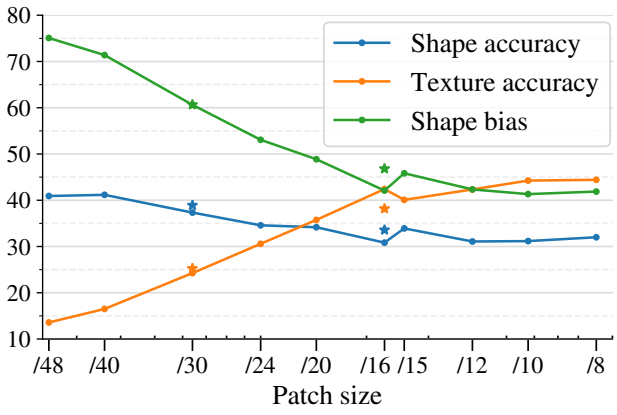

Figure 24. **Shape and texture bias of FlexiViT.** Lines reflect values for FlexiViT evaluated at different scales; stars reflect values for baseline ViT models trained at a single scale. Shape and texture accuracy are the top-1 accuracy of the model's prediction with respect to the shape and texture labels. Shape bias is the percentage of images that a classifier classifies by shape, provided that it correctly classifies them by either shape or texture.

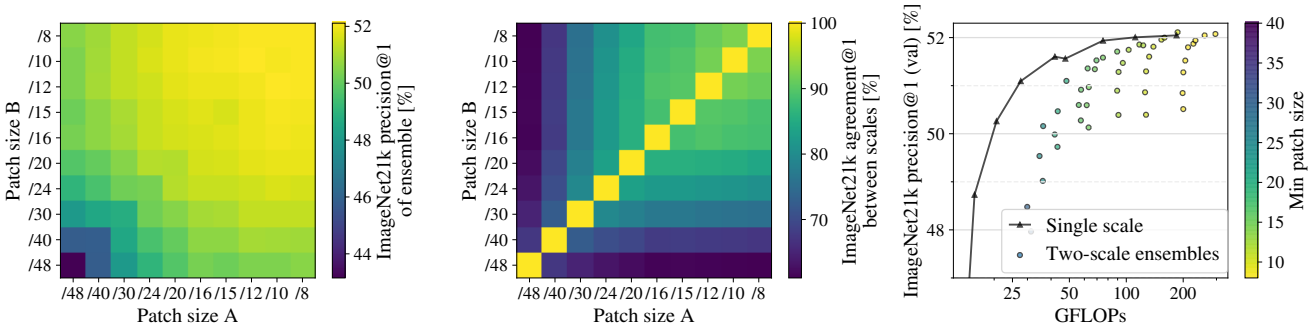

Figure 25. **Ensemble accuracy and agreement of FlexiViT-B evaluated at pairs of scales.** *Left*: Accuracy of ensembles between scales. *Middle*: Agreement of the top-1 predicted classes across scales. *Right*: Accuracy of single-scale configurations and two-scale ensembles versus computational cost. Although agreement is relatively low at small patch sizes, there is little benefit to ensembling.

ensemble of FlexiViT-B/10 and /12, which together have a similar computational cost to FlexiViT-B/8 (/10 + /12: 186.8 GFLOPs; /8: 184.5 GFLOPs). The ensemble attains marginally higher accuracy (52.1% vs. 52.0%), but this improvement is unlikely to be statistically significant nor practically meaningful.

K. Shape and texture bias of FlexiViT

When confronted with images with conflicting shape and texture, ImageNet-trained models tend to produce labels that match their textures, whereas humans instead tend to assign labels that match their shapes [18]. We evaluated FlexiViT using the same dataset as [18], which was generated using the style transfer. In the dataset constructed by [18], images have both shape and texture labels. We define the shape accuracy as the percentage of images for which the top-1 prediction matches the shape label, and texture accuracy as the percentage of images for which the top-1 prediction matches the texture label. As in [18], we define shape bias by taking the ratio of the number of images classified according to their shape label to the number of images classified correctly according to either the shape or texture label and converting this ratio to a percentage. In other words, shape bias the ratio of shape accuracy to the sum of shape and texture accuracy $\times 100\%$. To evaluate ImageNet-21K models on the dataset of [18], we use the mapping from WordNet IDs to the dataset classes provided by [18]. We take the top-1 class among the ImageNet-21K classes for which a mapping exists.

In Figure 24, we show that larger patch sizes lead to greater shape bias compared to smaller patch sizes. However, larger patch sizes have greater shape bias primarily because their texture accuracy is lower, rather than because their shape accuracy is higher. The shape biases of FlexiViT-B/16 and FlexiViT-B/30 are similar to the shape biases of ViT-B/16 and ViT-B/30 models trained at a single scale (stars).

L. Attention relevances

We provide the attention relevance maps [9] for the same image as shown in Figure 13 in the main paper, but for all three classes present in the image, in Figure 27.

We further provide the attention relevance maps for a random selection of 10 royalty-free images obtained from `unsplash.com` for the class that was predicted by all models in Figure 31 at the end of the Appendix.

M. Flexifying transfer-learning

When flexifying transfer-learning, we run the exact same setup as when transferring pre-trained FlexiViT models, described in Section E.1, except that we now randomize the patch size during transfer too.

This minor change shows good synergy when combined with using a pre-trained FlexiViT model (green bars), and even enables flexifying plain ViT models during transfer to some degree (orange and olive bars).

N. Flexifying open-vocabulary detection (OWL-ViT)

For flexifying OWL-ViT (Section 5.3), we use LiT-`uu` backbones [67], i.e. CLIP-style models in which the image and text encoders are contrastively pre-trained together (as in the OWL-ViT paper [38]). We flexify detection training as described in Algorithm 1 and use resolution 720×720. For flexible detection training, we use patch sizes from 48^2 to 12^2, i.e. omitting 10^2 and 8^2, which would use excessive memory at the higher resolution. Other training settings are as in the OWL-ViT paper [38]. We find that flexifying both the image-text pre-training and the detection training (Figure 11, pale green line) works slightly better than flexifying just the detection training.

For evaluating on the ELEVATER [34] set of datasets, we use the model for which both image-text pre-training and detection training were flexified (Figure 30).

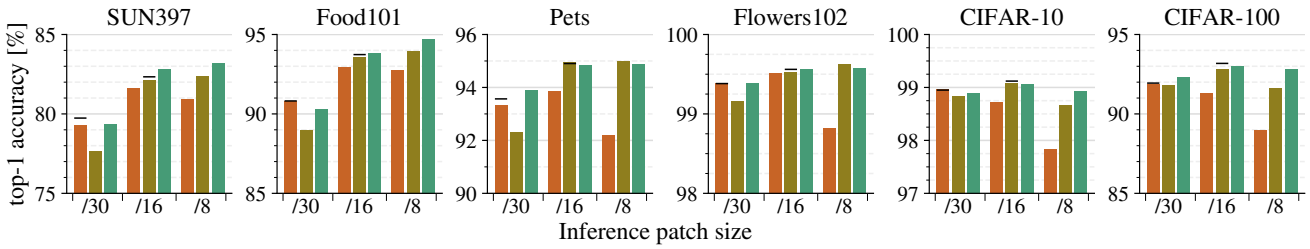

Figure 26. **More transfer results.** The legend is the same as in Figure 9. The main take-away is that the results are qualitatively the same across a wide range of image classification tasks.

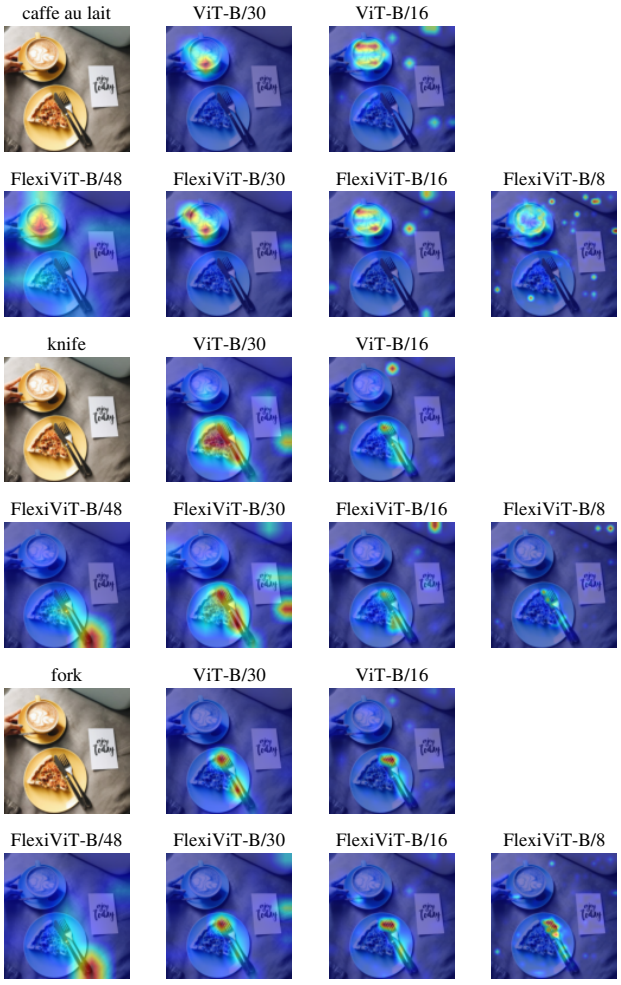

Figure 27. **Attention relevance maps** of same example as in Figure 13, with respect to three different objects present in the image ("caffe au lait", "knife" and "fork").

O. Details on flexible depth, stride alternatives

Common setup The setup largely follows that introduced in Section 3.5: distilling the ViT-B/8 model from [51] while simultaneously using it for initialization of the student. Distillation is performed on ImageNet-21k, the labels are ignored and the KL-divergence between student and teacher is the only loss, following [5]. Training is performed for 90 epochs with uniform sampling of patch sizes.

Flexible stride We train a FlexiViT model variant, where we flexibly change window stride when extracting image patches, but keep the patch size fixed at 32×32. In order to perfectly match default grid sizes, and perfectly cover the whole image and avoid padding, we perform minimally required image resize. For example, to get grid size 8×8 we resize the image to size 242×242 (from 240×240) and apply stride 30, or to get grid size 24×24 we resize the image to size 239×239 and apply stride 9.

Flexible depth For every batch, we sample a depth d uniformly from $\{3, 5, 9, 12\}$ and perform a forward pass up to layer d. We then apply the classification head, which is shared across all depths, to the class token of layer d and compute the loss. We also explored sampling a depth per-example, but this led to unstable training except when sampling d from all layers $\{3, 4, \ldots, 12\}$. Finally, we also tried using a head per depth as well as a class token per depth, which did not lead to any significant improvement.

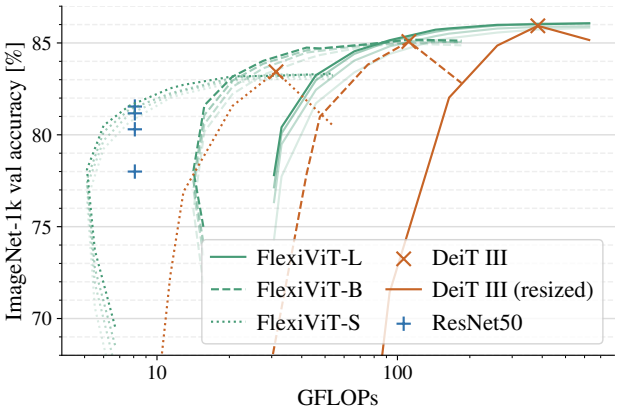

Figure 28. **Fig 2 using GFLOPs.**

P. Figure 2 with GFLOPs

Following the Efficiency Misnomer [15], we also provide a copy of Figure 2 using GFLOPs as the x-axis instead

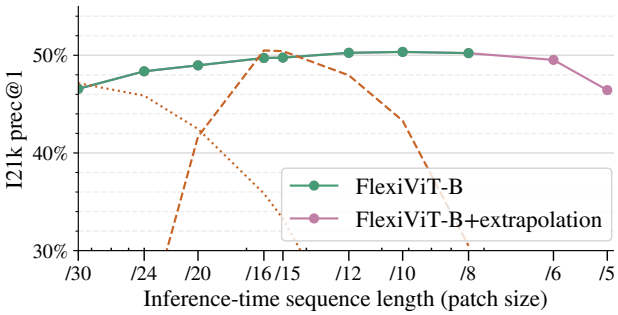

Figure 29. **Patch-size extrapolation.** We evaluate the model trained on the green patch-sizes at patch-sizes outside the range that was seen during training, and plot these in pink. The performance slowly deteriorating means that the model is not able to extrapolate beyond patchsizes or sequence lengths seen during training.

of inference time as Figure 28. This confirms that FLOPs do not always directly translate to wall-clock time in all situations.

Q. Extrapolating patch-size

We take the model from the main paper's Section 3.3 and run inference at even smaller patchsizes, see Figure 29. We observe that performance slowly starts to deteriorate. This means that the model does not learn to generalize to patch-sizes or sequence lengths beyond those seen during the training. Note that we do not claim extrapolation capabilities, but rather that training many patchsizes into a single model works without loss of quality.

R. Full tabular results

We provide Tabs. 1 to 13 which contain the numerical results from all plots from the main paper.

S. Configuration file for Fig 2

Algorithm S shows the `big_vision`[7] config for training the FlexiViT models from Figure 2.

[7] https://github.com/google-research/big_vision

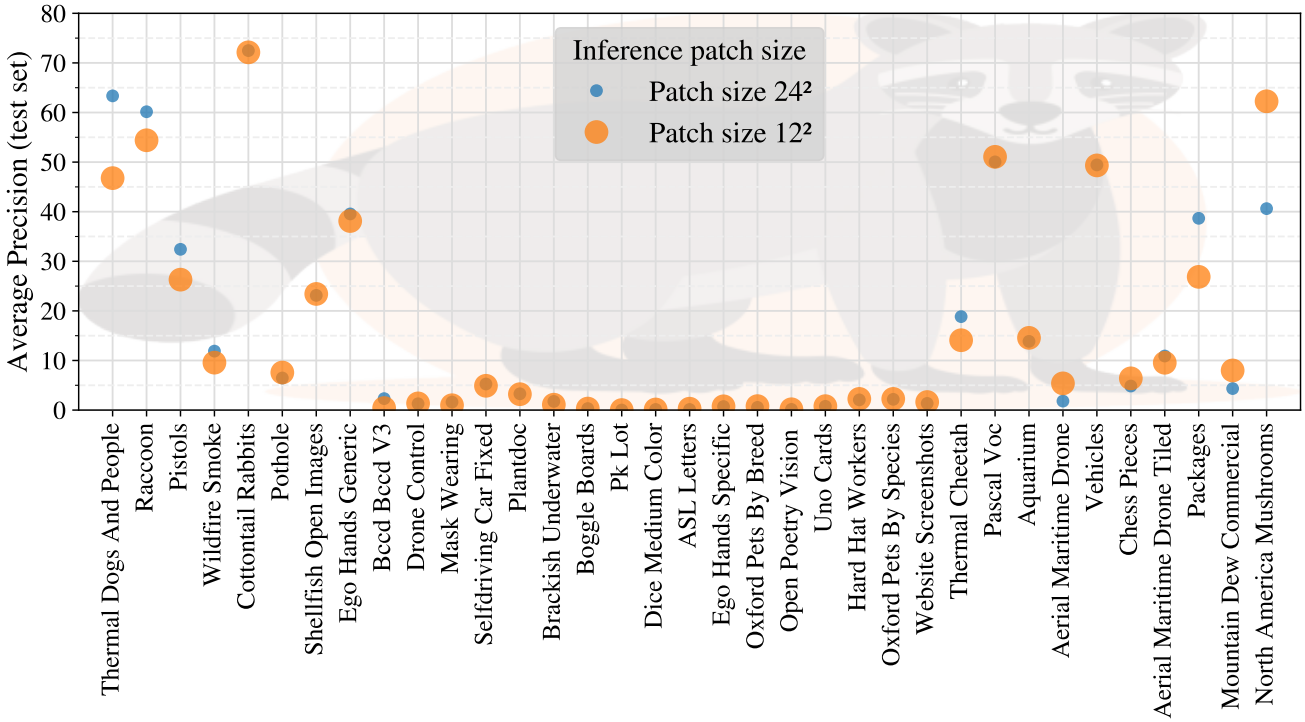

Figure 30. **Inference-time tuning of patch size can improve performance.** A single OWL-FlexiViT-B model is evaluated at two different patch sizes (30^2 and 16^2) on the 35 different detection tasks of the ELEVATER benchmark. [34]. Tasks are are ordered by the performance difference between patch size 30^2 and 16^2 on the *validation* set; the y-axis shows performance on the *test* set. The optimal patch size varies by task, such that tuning the patch size on the validation set improves mean test performance from 15.63 AP (at patch size 16^2) or 16.19 AP (at patch size 30^2) to 16.62 AP.

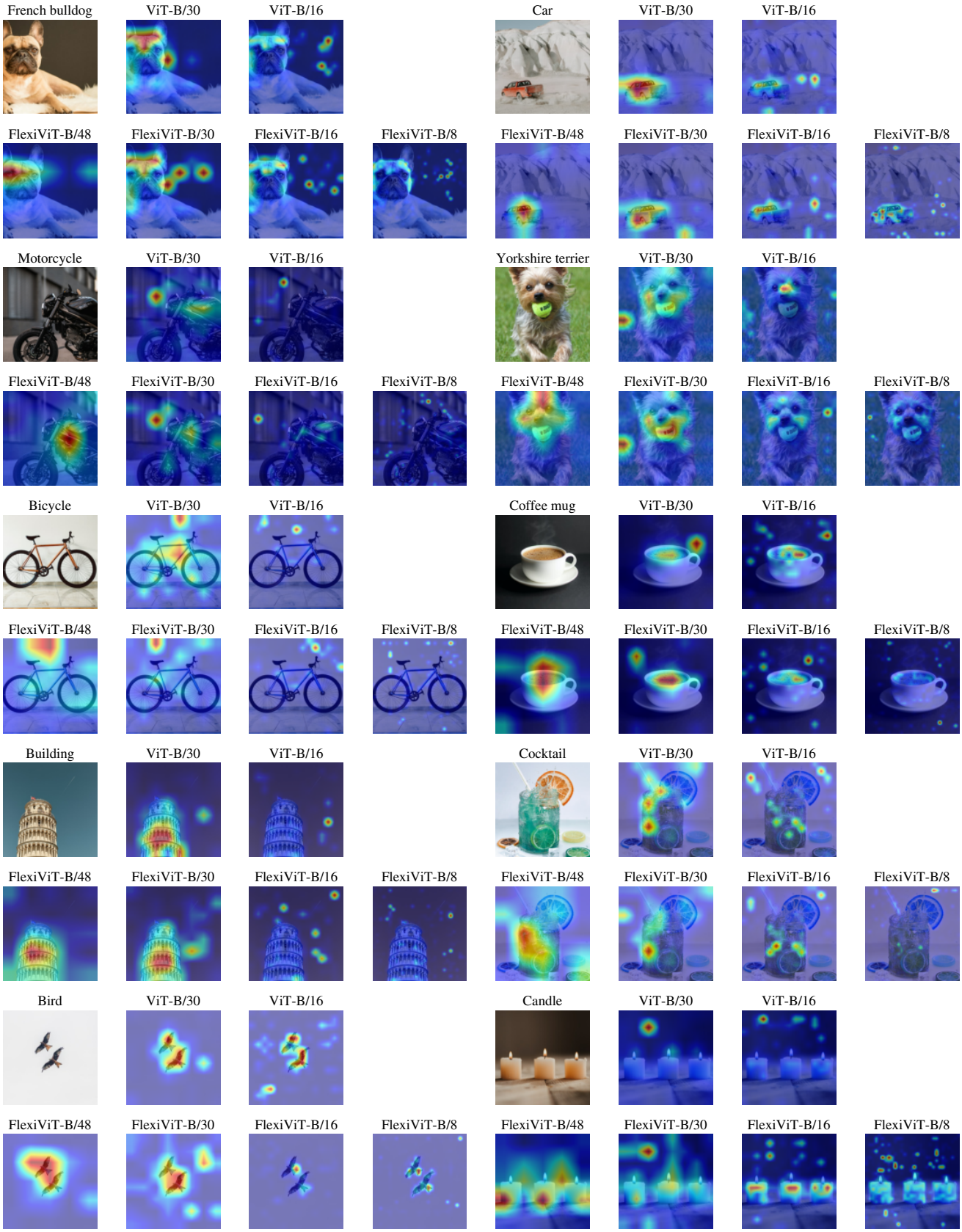

Figure 31. **Attention relevance** (as in [9]) can significantly change at different patch sizes for both ViT and FlexiViT.

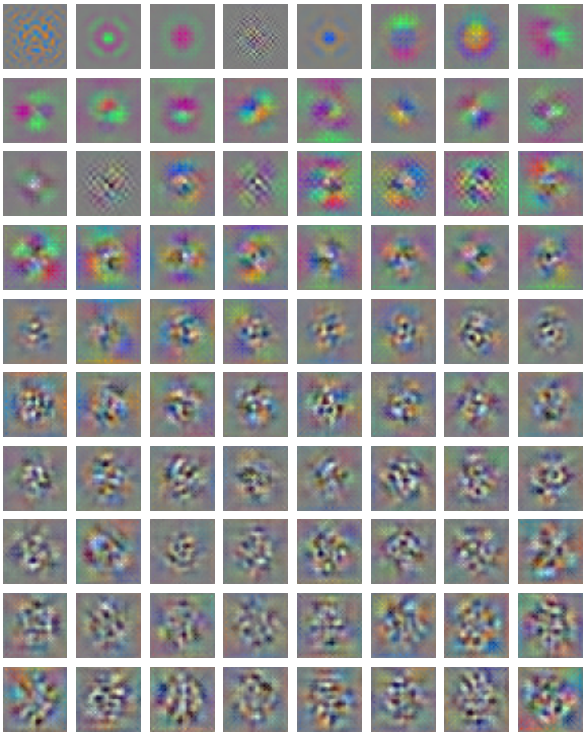

Figure 32. First 80 PCA components of the raw underlying 32x32 patch embedding weights of FlexiViT.

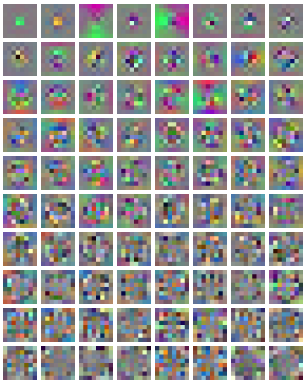

Figure 33. First 80 PCA components of the patch embedding weights from Figure 32 bilinearly resized to 8x8.

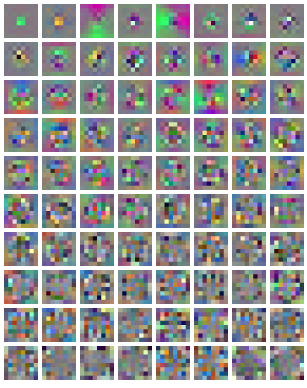

Figure 34. First 80 PCA components of the patch embedding weights from Figure 32 PI-resized to 8x8.

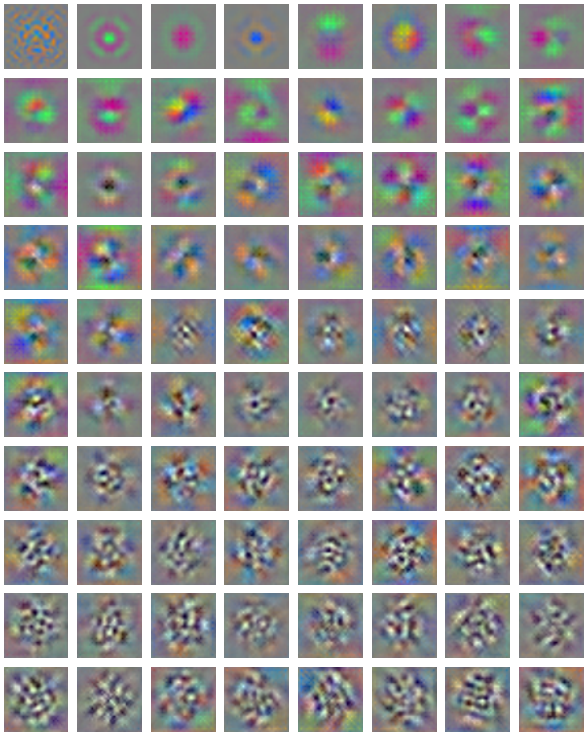

Figure 35. First 80 PCA components of the patch embedding weights from Figure 32 bilinearly resized to 48x48.

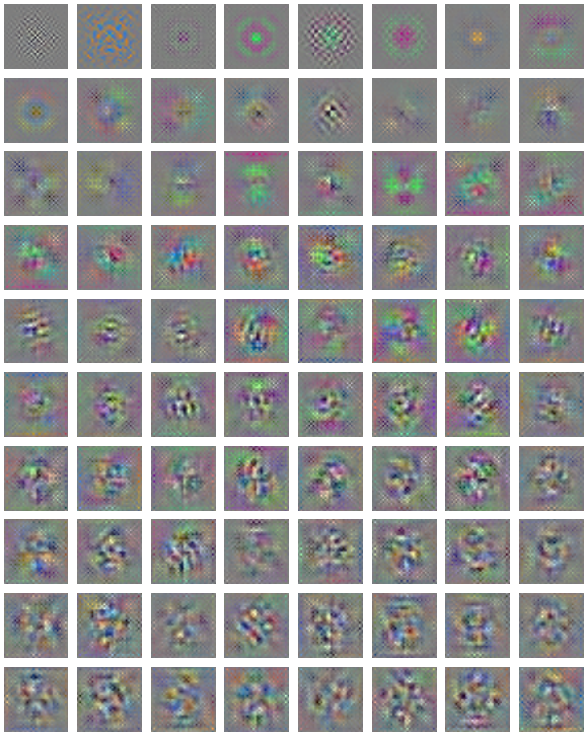

Figure 36. First 80 PCA components of the patch embedding weights from Figure 32 PI-resized to 48x48.

```python
def get_config(arg=None):
  """Config for training FlexiViT on ImageNet1k.
    """
  c = bvcc.parse_arg(arg, variant='B')
  c.total_epochs = 90
  c.num_classes = 1000
  c.loss = 'softmax_xent'

  c.input = {}
  c.input.data = dict(
      name='imagenet2012',
      split='train[:99%]',
  )
  c.input.batch_size = 1024
  c.input.shuffle_buffer_size = 250_000

  c.log_training_steps = 50
  c.ckpt_steps = 1000

  # Model section
  c.student_name = 'proj.flexi.vit'
  c.student_init = f'deit_3_{c.variant}_384_1k'
  c.student = dict(variant=c.variant,
                   pool_type='tok',
                   patch_size=(16, 16))

  c.teachers = ['prof']
  c.prof_name = 'vit'
  c.prof_init = f'deit_3_{c.variant}_384_1k'
  c.prof = dict(variant=c.variant,
                pool_type='tok',
                patch_size=(16, 16))

  pp_label = (
      '|onehot(1000, key="{lbl}", key_result="
    labels")'
      '|keep("image", "prof", "labels")')
  c.input.pp = (
      'decode|inception_crop|flip_lr'
      '|copy("image", "prof")'
      '|resize(240)'
      '|vgg_value_range'
      '|resize(384, key="prof")'
      '|vgg_value_range(key="prof")'
      + pp_label.format(lbl='label'))
  pp_eval_both = (
      'decode|copy("image", "prof")|'
      f'|resize({240//7*8})'
      '|central_crop(240)'
      '|vgg_value_range'
      f'|resize({384//7*8}, key="prof")'
      '|central_crop(384, key="prof")'
      '|vgg_value_range(key="prof")|')
  pp_eval_student = (
      'decode'
      f'|resize({240//7*8})|central_crop({240})'
      '|value_range(-1, 1)')
  pp_eval_prof = (
      'decode'
      f'|resize({384//7*8})|central_crop(384)'
      '|vgg_value_range(outkey="prof")')

  # Distillation settings
  c.mixup = dict(p=1.0, n=2)
  c.distance = 'kl'
  c.distance_kw = dict(t=1.0)

  # Optimizer section
  c.grad_clip_norm = 1.0
  c.optax_name = 'scale_by_adam'
  c.optax = dict(mu_dtype='bfloat16')

  c.lr = 1e-4
  c.wd = 1e-5
  c.schedule = dict(
      warmup_steps=5000,
      decay_type='cosine')

  # Define the flexible model params:
  c.flexi = dict()
  c.flexi.seqhw = dict(
      # The settings to sample from.
      # Corresponding patch-sizes at 240px:
      # 48, 40, 30, 24, 20, 16, 15, 12, 10, 8
      v=(5, 6, 8, 10, 12, 15, 16, 20, 24, 30),
      # The probs/weights of them (uniform):
      p=(1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
  )

  ####
  # All the rest is just evaluations.
  minitrain = 'train[:2%]'
  minival = 'train[99%:]'

  def get_eval(s, split, dataset='imagenet2012'):
    return dict(
        type='classification',
        pred=f'student_seqhw={s}',
        data=dict(name=dataset, split=split),
        pp_fn=(pp_eval_student +
               pp_label.format(lbl='label')),
        loss_name='sigmoid_xent',
        log_percent=0.05,
        cache_final=False,
    )

  c.evals = {}
  for s in c.flexi.seqhw.v:
    c.evals[f'student_minitrain_{s:02d}'] = \
        get_eval(s, minitrain)
    c.evals[f'student_minival_{s:02d}'] = \
        get_eval(s, minival)
    c.evals[f'student_val_{s:02d}'] = \
        get_eval(s, 'validation')
    c.evals[f'student_v2_{s:02d}'] = \
        get_eval(s, 'test', 'imagenet_v2')
    c.evals[f'student_a_{s:02d}'] = \
        get_eval(s, 'test', 'imagenet_a')
    c.evals[f'student_r_{s:02d}'] = \
        get_eval(s, 'test', 'imagenet_r')
    c.evals[f'student_real_{s:02d}'] = \
        get_eval(s, 'validation',
                 'imagenet2012_real')
    c.evals[f'student_real_{s:02d}'].pp_fn = (
        pp_eval_student +
        pp_label.format(lbl='real_label'))

  # A bunch more evals here ...

  return c
```

Table 1. Scores for 1200ep ImageNet-1k-only runs from Figure 2.

Model	Eps	PS	Val	ReaL	v2	-A	-R
FlexiViT-S	1200	48^2	69.6	76.1	55.5	3.3	24.1
FlexiViT-S	1200	40^2	73.7	80.2	60.3	4.7	26.4
FlexiViT-S	1200	30^2	78.1	84.3	65.1	7.4	29.2
FlexiViT-S	1200	24^2	80.5	86.3	68.4	9.4	30.5
FlexiViT-S	1200	20^2	81.6	87.2	70.3	12.2	31.8
FlexiViT-S	1200	16^2	82.5	87.9	71.7	15.0	31.4
FlexiViT-S	1200	15^2	82.7	88.1	71.8	15.7	32.9
FlexiViT-S	1200	12^2	83.2	88.4	72.7	17.8	32.9
FlexiViT-S	1200	10^2	83.2	88.4	72.9	19.4	33.0
FlexiViT-S	1200	8^2	83.3	88.5	72.9	19.3	32.6
FlexiViT-B	1200	48^2	75.0	80.5	61.1	5.7	28.1
FlexiViT-B	1200	40^2	78.0	83.3	64.7	7.5	30.0
FlexiViT-B	1200	30^2	81.6	86.5	69.7	11.4	33.0
FlexiViT-B	1200	24^2	83.2	87.7	71.7	15.5	34.5
FlexiViT-B	1200	20^2	84.0	88.4	73.0	18.4	35.6
FlexiViT-B	1200	16^2	84.7	88.8	74.0	21.7	35.8
FlexiViT-B	1200	15^2	84.7	88.8	74.3	22.7	36.7
FlexiViT-B	1200	12^2	84.9	89.1	74.8	25.3	37.1
FlexiViT-B	1200	10^2	85.2	89.2	75.0	26.7	37.2
FlexiViT-B	1200	8^2	85.1	89.2	74.9	27.1	37.2
FlexiViT-L	1200	48^2	77.8	83.3	63.9	7.1	30.0
FlexiViT-L	1200	40^2	80.4	85.6	67.2	9.9	32.7
FlexiViT-L	1200	30^2	83.2	87.9	70.8	14.8	35.9
FlexiViT-L	1200	24^2	84.5	88.8	73.6	19.9	38.1
FlexiViT-L	1200	20^2	85.1	89.4	74.9	23.5	39.4
FlexiViT-L	1200	16^2	85.7	89.7	76.0	28.6	39.6
FlexiViT-L	1200	15^2	85.8	89.9	76.0	29.1	40.6
FlexiViT-L	1200	12^2	86.0	90.0	76.5	32.0	40.8
FlexiViT-L	1200	10^2	86.0	90.0	76.8	33.6	40.9
FlexiViT-L	1200	8^2	86.1	90.0	76.7	34.1	41.2

Table 2. Scores for 600ep ImageNet-1k-only runs from Figure 2.

Model	Eps	PS	Val	ReaL	v2	-A	-R
FlexiViT-S	600	48^2	68.6	75.1	54.2	3.2	23.9
FlexiViT-S	600	40^2	72.7	79.4	59.3	4.4	26.3
FlexiViT-S	600	30^2	77.6	83.8	64.6	6.9	29.0
FlexiViT-S	600	24^2	80.2	86.0	67.8	9.2	30.4
FlexiViT-S	600	20^2	81.4	87.0	69.9	11.5	31.5
FlexiViT-S	600	16^2	82.3	87.7	71.2	14.2	31.2
FlexiViT-S	600	15^2	82.5	87.9	71.5	15.1	32.7
FlexiViT-S	600	12^2	83.1	88.3	72.5	17.5	32.7
FlexiViT-S	600	10^2	83.3	88.5	72.7	19.5	32.7
FlexiViT-S	600	8^2	83.3	88.5	72.8	19.4	32.4
FlexiViT-B	600	48^2	74.1	80.0	60.2	5.3	27.7
FlexiViT-B	600	40^2	77.5	83.0	64.4	7.5	30.0
FlexiViT-B	600	30^2	81.1	86.1	68.9	11.2	32.7
FlexiViT-B	600	24^2	82.9	87.5	71.6	15.0	34.2
FlexiViT-B	600	20^2	83.9	88.2	72.6	17.5	35.4
FlexiViT-B	600	16^2	84.6	88.7	73.9	22.1	35.7
FlexiViT-B	600	15^2	84.7	88.8	73.9	22.6	36.6
FlexiViT-B	600	12^2	84.9	89.0	74.7	25.4	36.9
FlexiViT-B	600	10^2	85.1	89.2	74.8	26.9	37.2
FlexiViT-B	600	8^2	85.0	89.2	74.8	27.0	36.9
FlexiViT-L	600	48^2	77.1	82.6	62.7	7.2	30.1
FlexiViT-L	600	40^2	80.1	85.2	66.6	9.4	32.6
FlexiViT-L	600	30^2	83.0	87.7	71.0	14.6	36.0
FlexiViT-L	600	24^2	84.4	88.8	73.4	19.3	38.1
FlexiViT-L	600	20^2	85.1	89.4	74.7	22.5	39.3
FlexiViT-L	600	16^2	85.6	89.7	76.0	27.7	39.7
FlexiViT-L	600	15^2	85.7	89.8	75.9	28.3	40.8
FlexiViT-L	600	12^2	85.9	89.9	76.4	31.0	41.0
FlexiViT-L	600	10^2	86.1	90.0	76.6	32.8	41.1
FlexiViT-L	600	8^2	86.1	90.0	76.6	33.2	41.3

Table 3. Scores for 300ep ImageNet-1k-only runs from Figure 2.

Model	Eps	PS	Val	ReaL	v2	-A	-R
FlexiViT-S	300	48^2	67.4	74.0	53.2	2.8	23.5
FlexiViT-S	300	40^2	71.9	78.6	58.2	3.9	26.0
FlexiViT-S	300	30^2	77.2	83.5	64.2	6.5	29.0
FlexiViT-S	300	24^2	79.7	85.6	67.6	8.8	30.2
FlexiViT-S	300	20^2	81.1	86.7	69.6	11.1	31.4
FlexiViT-S	300	16^2	82.1	87.6	71.1	13.9	31.1
FlexiViT-S	300	15^2	82.4	87.9	71.3	14.9	32.7
FlexiViT-S	300	12^2	83.0	88.2	72.3	17.6	32.5
FlexiViT-S	300	10^2	83.2	88.3	72.9	19.3	32.4
FlexiViT-S	300	8^2	83.2	88.3	73.0	19.4	32.2
FlexiViT-B	300	48^2	73.4	79.3	59.7	4.9	27.4
FlexiViT-B	300	40^2	77.0	82.6	63.7	7.0	29.6
FlexiViT-B	300	30^2	80.6	85.8	68.4	10.4	32.7
FlexiViT-B	300	24^2	82.6	87.3	71.2	14.6	34.1
FlexiViT-B	300	20^2	83.6	88.1	72.5	17.3	35.1
FlexiViT-B	300	16^2	84.5	88.6	73.8	21.8	35.4
FlexiViT-B	300	15^2	84.6	88.7	73.9	22.5	36.5
FlexiViT-B	300	12^2	84.9	89.0	74.6	25.5	36.9
FlexiViT-B	300	10^2	85.0	89.1	74.8	27.3	37.0
FlexiViT-B	300	8^2	85.1	89.1	75.0	27.1	36.9
FlexiViT-L	300	48^2	76.3	81.8	62.4	6.5	30.2
FlexiViT-L	300	40^2	79.5	84.7	66.4	9.0	32.6
FlexiViT-L	300	30^2	82.4	87.3	70.5	13.7	35.9
FlexiViT-L	300	24^2	84.0	88.5	72.8	18.0	37.7
FlexiViT-L	300	20^2	84.8	89.1	74.3	21.7	39.2
FlexiViT-L	300	16^2	85.4	89.6	75.7	26.7	39.4
FlexiViT-L	300	15^2	85.5	89.7	75.8	28.1	40.6
FlexiViT-L	300	12^2	85.8	89.9	76.4	30.9	40.8
FlexiViT-L	300	10^2	85.9	89.9	76.8	32.7	41.0
FlexiViT-L	300	8^2	85.9	90.0	76.7	33.4	41.2

Table 4. Scores for 90ep ImageNet-1k-only runs from Figure 2.

Model	Eps	PS	Val	ReaL	v2	-A	-R
FlexiViT-S	90	48^2	65.9	72.5	51.9	2.9	23.2
FlexiViT-S	90	40^2	70.6	77.3	56.9	3.5	26.0
FlexiViT-S	90	30^2	76.4	82.9	62.9	5.9	29.3
FlexiViT-S	90	24^2	79.2	85.3	66.8	7.8	30.6
FlexiViT-S	90	20^2	80.7	86.5	69.0	10.6	31.5
FlexiViT-S	90	16^2	82.0	87.5	70.9	13.9	31.7
FlexiViT-S	90	15^2	82.2	87.7	71.1	14.4	32.7
FlexiViT-S	90	12^2	82.8	88.1	72.0	17.3	32.5
FlexiViT-S	90	10^2	83.0	88.2	72.7	19.0	32.4
FlexiViT-S	90	8^2	83.0	88.3	72.7	19.4	32.2
FlexiViT-B	90	48^2	71.9	77.8	58.0	4.6	26.9
FlexiViT-B	90	40^2	75.9	81.6	62.2	6.1	29.3
FlexiViT-B	90	30^2	80.2	85.3	67.3	9.5	32.3
FlexiViT-B	90	24^2	82.2	87.0	70.2	13.1	34.1
FlexiViT-B	90	20^2	83.3	87.8	71.9	16.5	35.1
FlexiViT-B	90	16^2	84.1	88.4	73.2	21.3	35.2
FlexiViT-B	90	15^2	84.3	88.5	73.9	21.8	36.4
FlexiViT-B	90	12^2	84.8	88.8	74.4	25.2	36.8
FlexiViT-B	90	10^2	85.0	89.0	74.5	27.3	36.9
FlexiViT-B	90	8^2	84.9	89.0	74.7	27.7	36.6
FlexiViT-L	90	48^2	74.3	80.0	60.2	5.6	29.4
FlexiViT-L	90	40^2	77.7	83.2	64.5	7.7	32.2
FlexiViT-L	90	30^2	81.7	86.7	69.4	12.1	35.3
FlexiViT-L	90	24^2	83.4	88.0	72.4	17.1	37.3
FlexiViT-L	90	20^2	84.4	88.8	73.9	20.9	38.9
FlexiViT-L	90	16^2	85.1	89.3	75.4	26.5	39.4
FlexiViT-L	90	15^2	85.3	89.5	75.6	27.2	40.3
FlexiViT-L	90	12^2	85.6	89.7	76.3	31.2	40.5
FlexiViT-L	90	10^2	85.7	89.8	76.7	33.1	40.7
FlexiViT-L	90	8^2	85.8	89.9	76.6	33.7	40.6

Table 5. Numerical data for Figure 3.

Model	/48	/40	/30	/24	/20	/16	/15	/12	/10	/8
Flexi	39.5	43.2	46.6	48.4	49.0	49.7	49.8	50.2	50.3	50.2
B/16	0.0	0.1	2.4	21.6	41.7	50.5	50.4	47.9	43.3	30.5
B/30	14.0	30.2	47.1	45.9	42.5	35.9	33.3	21.0	11.9	2.9

Table 6. Numerical data for Figure 5.

		/5	/6	/8	/10	/12	/15	/16	/20	/24	/30
Top-1 accuracy											
T-init	90	40.8	43.8	47.9	49.4	50.5	51.3	51.4	51.9	51.9	52.0
	300	43.1	45.9	48.7	50.3	51.1	51.6	51.6	51.9	52.0	52.0
	1000	44.1	46.6	49.2	50.6	51.2	51.9	51.8	52.1	52.3	52.2
R-init		41.5	44.2	47.0	48.5	48.9	49.6	49.8	50.0	50.1	50.0
None		40.6	43.4	46.6	48.1	48.9	49.7	49.7	50.0	50.2	50.1
Teacher						52.2					
Top-1 agreement											
T-init	90	56.0	61.9	69.5	74.7	78.1	81.1	81.8	83.8	84.5	84.4
	300	59.6	65.7	73.0	77.3	80.1	82.8	82.9	84.2	84.5	84.5
	1000	62.0	67.5	74.4	78.6	81.3	83.4	83.7	84.6	85.2	85.0
R-init		56.4	60.7	66.2	69.0	70.7	72.1	72.4	72.8	73.1	73.1
CKA similarity											
T-init	90	.65	.69	.76	.80	.84	.86	.87	.88	.89	.89
	300	.68	.72	.78	.82	.85	.86	.87	.87	.88	.87
	1000	.68	.72	.78	.81	.83	.85	.85	.86	.86	.86
R-init		.41	.43	.45	.47	.48	.49	.50	.50	.50	.50

Table 7. Numerical data for Figure 7.

Method	/30→/30	F→/30	/16→/16	F→/16
Clf SUN397	79.1	79.7	82.3	82.5
Clf Food101	90.4	90.8	93.7	94.1
Clf Pets	93.6	93.6	94.9	94.9
Clf Flowers102	99.4	99.4	99.6	99.6
Clf CIFAR-10	98.8	99.0	99.1	99.1
Clf CIFAR-100	92.3	91.9	93.2	93.3
UViM coco PQ	24.8	24.1	30.5	34.3
OWL-ViT lvis AP	18.8	18.4	22.7	23.4
LiT i2t coco	42.2	41.6	44.4	45.8
Seg (lin) City mIoU	61.0	61.1	69.3	70.0
Seg (lin) ADE mIoU	43.1	43.5	46.1	47.5

T. Acknowledgments

We thank Daniel Keysers for good feedback on a draft of the paper, Geoffrey Hinton for a nudge to pursue this project early on, and our respective teams at Google for encouraging creative and independent research.

Table 8. Numerical data for Figure 8.

/40	/30	/24	/20	/15	/12	/10	/8
ViT-B/30 transferred at /30							
78.3	82.4	83.0	82.7	80.4	75.1	66.7	58.6
FlexiViT-B transferred at /30							
78.0	81.8	83.4	84.3	85.0	85.2	85.3	84.9
FlexiViT-B transferred at /16							
77.0	81.2	83.2	84.5	85.5	85.9	86.0	85.7
FlexiViT-B transferred at /8							
76.2	80.6	82.8	84.1	85.3	85.8	86.1	86.4

Table 9. Numerical data for Figure 9.

Model	Transfer	SUN	Food	Pet	Flow	C10	C100
Evaluated at /30							
ViT-B/30	/30	79.7	90.8	93.6	99.4	99.0	91.9
ViT-B/30	Flexi	79.3	90.8	93.3	99.4	99.0	91.9
ViT-B/16	Flexi	77.7	89.0	92.3	99.2	98.8	91.8
FlexiViT-B	Flexi	79.3	90.3	93.9	99.4	98.9	92.3
Evaluated at /16							
ViT-B/16	/16	82.3	93.7	94.9	99.6	99.1	93.2
ViT-B/30	Flexi	81.6	93.0	93.8	99.5	98.7	91.3
ViT-B/16	Flexi	82.1	93.6	94.9	99.5	99.1	92.8
FlexiViT-B	Flexi	82.8	93.8	94.8	99.6	99.1	93.0
Evaluated at /8							
ViT-B/30	Flexi	80.9	92.7	92.2	98.8	97.8	89.0
ViT-B/16	Flexi	82.4	94.0	95.0	99.6	98.7	91.6
FlexiViT-B	Flexi	83.2	94.7	94.9	99.6	98.9	92.8

Table 10. Numerical data for Figure 10.

Base	LiT	/48	/30	/24	/16	/12	/10	/8
ViT-B/30	/30	46.7	65.2	65.2	64.5	30.9	10.3	3.2
ViT-B/16	/16	3.2	42.7	60.2	71.9	61.6	45.7	57.8
FlexiViT-B	/30	49.0	59.8	67.0	72.5	72.6	74.7	74.1
FlexiViT-B	/16	48.2	62.6	66.6	73.3	73.1	74.5	75.0
FlexiViT-B	Flexi	51.0	62.5	69.2	73.4	74.5	75.5	75.1

Table 11. Numerical data for Figure 11.

Base	OWL	/48	/40	/30	/24	/20	/16	/12
LiT-B/30	/30	6.2	10.8	20.6	15.6	7.8	1.7	0.3
LiT-B/30	/16	0.1	0.5	4.1	9.5	16.7	26.8	15.0
LiT-B/30	Flexi	15.7	17.8	21.2	23.0	24.2	25.6	26.0
FlexiLiT-B	Flexi	16.0	18.5	21.5	23.5	24.9	26.7	27.1

Table 12. Numerical data for Figure 4.

Resize	/2	/4	/6	/8	/10	/12	/14	/16	/18	/20	/22	/24	/26	/28	/30	/32
PI	10.3	44.5	48.9	52.4	52.4	52.4	52.4	52.3	52.3	52.4	52.3	52.4	52.3	52.3	52.4	52.4
Area	24.5	42.6	46.1	52.4	47.7	48.3	48.6	48.1	48.7	48.4	48.5	48.7	48.5	48.5	48.6	48.4
Norm	10.8	40.7	46.4	52.4	45.6	44.1	42.0	39.4	37.1	33.4	29.8	26.1	22.0	18.1	14.8	11.9
Vanilla	25.5	41.6	45.6	52.4	42.5	37.2	25.5	12.8	5.4	1.9	0.6	0.2	0.1	0.0	0.0	0.0

Table 13. Numerical data for Figure **??**.

Setting	Params	GFLOPs	Speed	Prec
Patch FlexiViT-B				
Eval at patch /30	87.5M	15.7	2745	47.9
Eval at patch /24	87.5M	20.6	2022	49.4
Eval at patch /20	87.5M	27.7	1135	50.5
Eval at patch /16	87.5M	41.9	806	51.3
Eval at patch /15	87.5M	47.6	595	51.4
Eval at patch /12	87.5M	75.3	362	51.9
Eval at patch /10	87.5M	111.5	246	51.9
Eval at patch /8	87.5M	184.5	128	52.0
Stride FlexiViT-B/32				
Eval at stride 30	87.8M	11.7	2317	47.5
Eval at stride 23	87.8M	18.2	1768	49.3
Eval at stride 19	87.8M	26.3	1041	50.3
Eval at stride 15	87.8M	41.7	750	51.0
Eval at stride 14	87.8M	47.6	563	51.2
Eval at stride 11	87.8M	76.4	347	51.5
Eval at stride 9	87.8M	113.7	235	51.6
Eval at stride 7	87.8M	188.4	124	51.7
Depth FlexiViT-B/8				
Eval at depth 3	22.1M	46.3	403	35.2
Eval at depth 6	43.4M	92.2	216	44.8
Eval at depth 9	64.6M	138.2	147	48.7
Eval at depth 12	85.9M	184.2	112	51.0

We would also like to thank the following artists for making their photographs (used for visualizations) freely available through unsplash.com: Tanya Patrikeyeva, Markus Spiske, Matheus Bardemaker, Alexandru Sofronie, Chris Smith, Kajetan Sumila, Julee Juu, Mike Erskine, Piermanuele Sberni, Feyza Yıldırım and Sixteen Miles Out.

Finally, the raccoon picture used as background in Figure 30 is from publicdomainvectors.org.