MSViT: Dynamic Mixed-scale Tokenization for Vision Transformers

Jakob Drachmann Havtorn[†*]
Technical University of Denmark & Corti.ai
Copenhagen, Denmark
jakobhavtorn.github.io

Amélie Royer[*]
Qualcomm AI Research
Amsterdam, The Netherlands
aroyer@qti.qualcomm.com

Tijmen Blankevoort
Qualcomm AI Research
Amsterdam, The Netherlands
tijmen@qti.qualcomm.com

Babak Ehteshami Bejnordi
Qualcomm AI Research
Amsterdam, The Netherlands
behtesha@qti.qualcomm.com

Abstract

The input tokens to Vision Transformers carry little semantic meaning as they are defined as regular equal-sized patches of the input image, regardless of its content. However, processing uniform background areas of an image should not necessitate as much compute as dense, cluttered areas. To address this issue, we propose a dynamic mixed-scale tokenization scheme for ViT, MSViT. Our method introduces a conditional gating mechanism that selects the optimal token scale for every image region, such that the number of tokens is dynamically determined per input. In addition, to enhance the conditional behavior of the gate during training, we introduce a novel generalization of the batch-shaping loss. We show that our gating module is able to learn meaningful semantics despite operating locally at the coarse patch-level. The proposed gating module is lightweight, agnostic to the choice of transformer backbone, and trained within a few epochs with little training overhead. Furthermore, in contrast to token pruning, MSViT does not lose information about the input, thus can be readily applied for dense tasks. We validate MSViT on the tasks of classification and segmentation where it leads to improved accuracy-complexity trade-off.

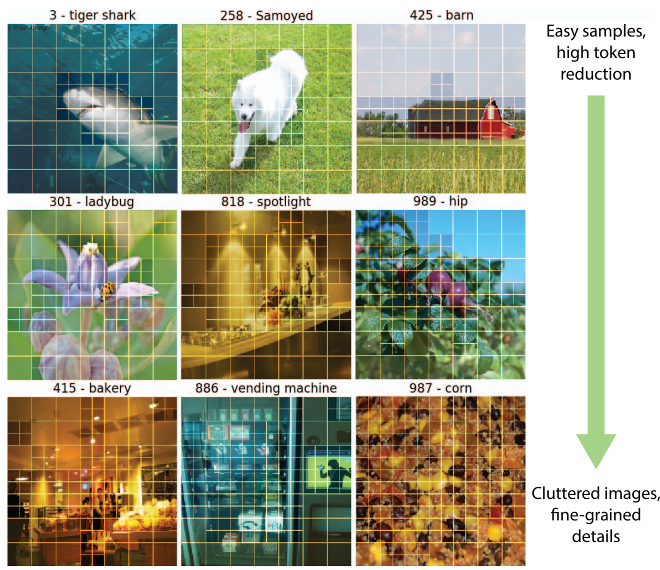


Figure 1. We introduce a learnable module to dynamically select the optimal token scale for each region. This module can be plugged in as a preprocessing step to any Vision Transformer. Here we illustrate some mixed-scale masks on ImageNet samples with varying levels of clutter, output by the scale selection module, trained alongside a pretrained ViT-S/16 for 20 epochs to choose between a coarse (32px, ☐) and a fine (16px, ☐) token scale.

1. Introduction

The Transformer architecture [51] has seen widespread success across Natural Language Processing (**NLP**) tasks and more recently in Computer Vision (**CV**) [11, 27, 49]. However, the quadratic time and memory complexity of transformers poses a challenge when deploying such models on compute constrained devices. In particular, the number of input tokens and the tokenization method are defining aspects of the computational complexity of transformers. In NLP, it is generally straightforward to use semantic units, such as words or sentences, as input tokens: This leads to little redundancy in the information carried by individual tokens. Conversely, in CV, tokenization is usually achieved by slicing an image into equal-sized, square patches without considering their content. This introduces redundant infor-

[*]equal contribution
[†]work done as an intern at Qualcomm AI Research
Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

mation across tokens, leading to computational waste: For instance, trivial background regions (e.g. sky and grass) are often expressed by a large number of tokens, dominating the bulk of compute in the model. Nonetheless, it remains unclear how to design a more efficient tokenization that reduces input redundancy compared to such uniform patching. In fact, most successful token reduction methods in the literature, such as token pruning [56, 34, 57, 29, 20, 30] or token merging [35, 42], only act on intermediate layers of the transformer, while earlier layers still inefficiently operate with a large number of redundant tokens.

In this work, we propose a novel, orthogonal approach: We predict the *tokenization scale* for each image region as a pre-processing step before the transformer. Intuitively, uninformative image regions such as background can be processed at a coarser scale than the foreground, without loss of information, leading to a smaller total number of tokens. To capture this behavior, we introduce a lightweight conditional gating MLP trained to select the optimal tokenization scale for every coarse local image region, as illustrated in Figure 1, leading to a dynamic number of tokens per image. Because it operates at the input level, the gate is agnostic to the choice of transformer backbone. Furthermore, mixed-scale tokenization is lossless, as every input region is covered by a token, making it well suited for dense prediction tasks in contrast to other methods such as pruning. Nevertheless, learning such a scale selection module raises several issues: **(i)** Current multi-scale ViT architectures are often trained with extra parameters for each scale or have cumbersome training pipelines with multiple stages [6, 62, 7]. Instead, we design a unified, single-stage model by maximizing parameter sharing across scales. **(ii)** The gating module may learn a bad local minimum such as always outputting the same trivial static pattern. To combat this, we introduce a novel training loss that enables finer control over the learned gating distribution, enhancing the dynamic behavior of the mixed-scale tokenization. Finally, **(iii)** the cost of training grows with the total number of fine and coarse tokens. To reduce training costs, we employ an adaptive trimming strategy at training time which relies on the underlying mapping between coarse and fine tokens. The main contributions of this work are as follows:

1. We design a dynamic scale selection gating mechanism that acts as a *preprocessing* stage, agnostic to the choice of transformer backbone, and trained jointly with the transformer *in a single stage* with mixed-scale tokens as inputs. We show in experiments that this dynamic tokenization process leads to improved computational costs by reducing the number of input tokens.
2. We propose a generalization of batch-shaping [13] to better handle *multi-dimensional distributions* when training dynamic gates: The resulting loss provides better control over the learned scale distribution, and allows for easier and better initialization of the gates.
3. We reduce the training overhead incurred from handling a set of tokens for each scale by (i) defining the gate locally at the coarse token level only and (ii) employing an adaptive trimming strategy during training.

2. Proposed method

In this work, we enhance the standard Vision Transformer (**ViT**) formalism with mixed-scale tokens that are dynamically selected for each input image. In this section, we briefly introduce ViT, then describe how we handle tokens extracted at different scales, with a focus on keeping the architecture parameter-efficient (Section 2.1) and reducing training overhead (Section 2.3). Finally, we present the generalized batch-shaping loss for training the mixed-scale selection module (Section 2.2).

2.1. Parameter-efficient mixed-scale ViT

Given an input image of size $W \times W$, a ViT first splits the image into square *patches* of equal size, S, resulting in a total of $N_S = \lfloor W/S \rfloor^2$ tokens. These tokens are flattened, and individually embedded to the target dimension d. A position encoding is then added to each token, which is a vector capturing the initial 2D spatial location of the token. Finally, the tokens are fed to a transformer, T, which is a sequence of Multiheaded Self-Attention (**MHSA**) blocks, that compute global attention across the set of tokens, followed by FFNs, which process each token independently [51]. Our work is agnostic to the choice of the transformer backbone T, thus, in the rest of the section, we only describe changes made to the patching, token embedding, and position encoding mechanisms to handle mixed-scale tokens.

Dynamic mixed-scale ViT. An overview of the proposed mixed-scale vision transformer model (MSViT) is presented in Figure 2. In the scope of this paper, we consider the case of two scales ($S_f < S_c$). We refer to S_f (resp. S_c) as the *fine* (resp. *coarse*) scale. First, we extract square patches at both scales, for a total of $N = N_{S_f} + N_{S_c}$ tokens. We then introduce a discrete *gating* mechanism, g, which selects active tokens across both scales, for a given input image: These tokens are further sent to the transformer, while inactive ones are discarded at this stage.

In practice, we define the learned gate as a local operation, at the level of coarse tokens: The gate parses each coarse image region individually and outputs a binary decision on whether the region should be tokenized at either the coarse or fine scale. We consider the case where the fine-scale S_f evenly divides the coarse scale S_c. This way, for all i, the i-th fine token can be mapped to the unique coarse token $C(i) = j$ it belongs to. Using this mapping, we recover the complete binary mixed-scale mask at the fine

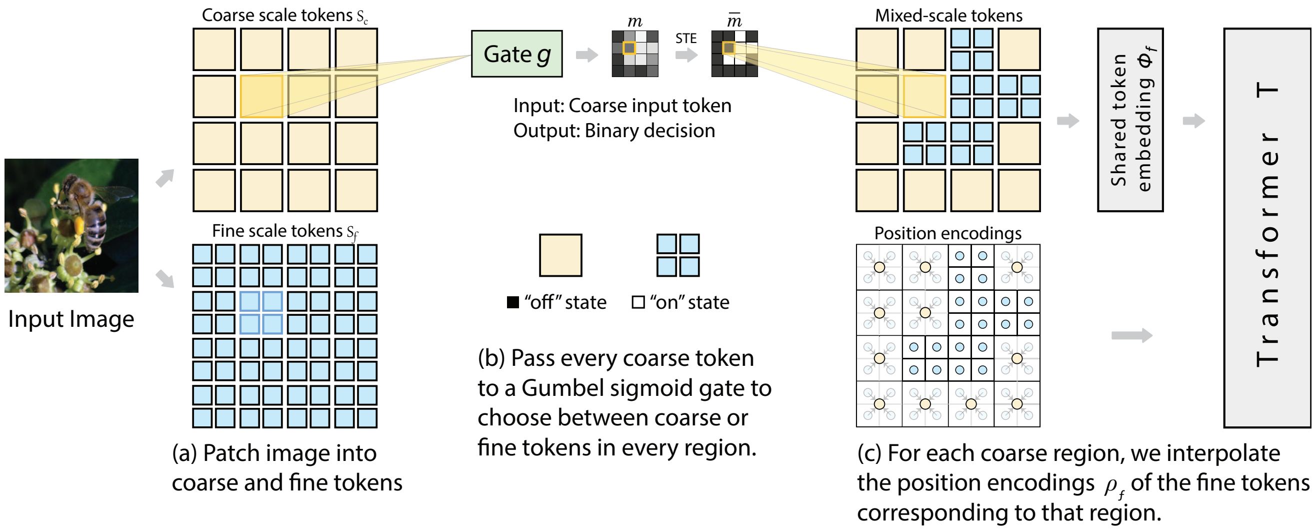


Figure 2. Overview of the proposed dynamic mixed-scale tokenization scheme for ViT, MSViT. **(a)** The input image is first patched into coarse image regions of size $S_c \times S_c$. **(b)** Each coarse region is processed by a small 4-layer MLP, the gate g, outputting a binary decision on whether the region should be processed at a coarse or fine scale. **(c)** The resulting mask, $\overline{m}$, defines the set of mixed-scale tokens for the input image. The corresponding mixed-scale position encodings are obtained by linearly interpolating the fine scale position encodings to the coarse scale, when needed. Finally, the tokens are sent to a standard transformer backbone T which outputs the task-relevant prediction.

token level, $\overline{m}$, using the coarse-level gate outputs:

$$\forall j \in [1, N_{S_c}],\ m_j = \text{GumbelSigmoid}(g(x_j)) \in [0, 1] \quad (1)$$

$$\overline{m}_j = \text{STE}(m_j) \in \{0, 1\} \quad (2)$$

$$\forall i \in [N_{S_c} + 1, N_{S_c} + N_{S_f}],\ \overline{m}_i = 1 - \overline{m}_{C(i)} \quad (3)$$

Here, we distinguish between the soft outputs of the gate, $m \in [0, 1]$, used to constrain the gate during training, and the discretized outputs $\overline{m} \in \{0, 1\}$ used during the forward pass. In order, to estimate gradients for the discrete gate operation, we use the Gumbel-Sigmoid relaxation of binary variables during training [28] with the straight-through gradient estimator (**STE**) [17, 1].

While this design choices for the gate may limit representational power, as g only sees local regions of the image as inputs, we find that it works well in practice and yields a very lightweight gating strategy. Moreover, as in the original ViT tokenization, token overlap is prevented by design, as every image region can only be captured by a unique scale.

Sharing parameters across scales. Previous mixed-scale ViTs usually introduce extra parameters to handle each scale [6, 55] or train a shared backbone stage by stage for each scale separately [7, 62]. Instead, our intention is **(i)** to fully share the token embedding, position encodings, and the transformer backbone across scales, and **(ii)** to directly train the model in one stage with batches of mixed-scale tokens, rather than treating each scale individually. This allows us to avoid extra parameter costs and makes our method architecture agnostic. In addition, due to the dynamic nature of the gating mechanism, defining separate branches per scale instead of sharing may lead to common issues of training conditional models such as imbalanced routing and data starvation [14, 43, 39].

To implement sharing across scales, we draw inspiration from ViT [11, 2]: At inference, the authors scale a trained model to a different input image size by linearly interpolating its position encodings to match the size of the new grid. We extend this idea to our setting by defining the learned embedding ϕ_f and position encoding parameters ρ_f relative to the *fine scale* only (Figure 2 (c)). We then deterministically infer their equivalent for the coarse scale as:

$$\phi_f : x \in \mathbb{R}^{S_f \times S_f \times 3} \mapsto \mathbb{R}^d,\ \ \rho_f \in \mathbb{R}^{N_{S_f} \times d} \quad (4)$$

$$\phi_c = \phi_f \circ \text{resize}(S_c \to S_f),\ \rho_c = \text{interpolate}(\rho_f) \quad (5)$$

In Appendix G.3, we show that this simple linear interpolation scheme works very well in practice, but may suffer when rescaling to a very low token resolution: For instance, directly training with the coarse patch size 32 on inputs of size 224px yields higher accuracy than the model with fine patch size 16, rescaled for 112px inputs to reach the same number of 49 tokens. Nevertheless, this is not an issue for the image and patch sizes we consider in our experiments.

2.2. Learning the mixed-scale gating

We jointly train the transformer and the gate by balancing the model performance with computational efficiency, forcing the model to only select a few tokens at fine scale:

$$\mathcal{L}(x_{1\ldots N}, m_{1\ldots N}, y) = \mathcal{L}_{task}(x, y; m) + \lambda \mathcal{L}_{gate}(m) \quad (6)$$

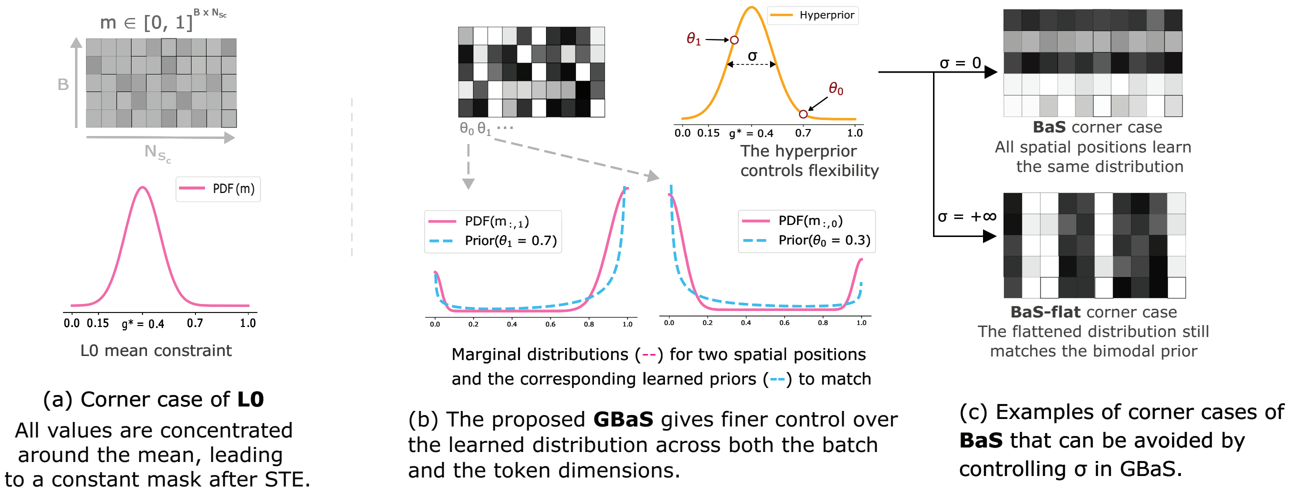


Figure 3. Our proposed generalized batch-shaping (GBaS) allows for fine control over the learned distribution via a hyperprior **(b)**: GBaS allows for learning different distributions for each token position in contrast to BaS **(c, top)**; In addition, GBaS explicitly controls this flexibility through the variance hyperparameter σ, hence avoiding corner cases of BaS-flat **(c, bottom)** or L0 **(a)**

where $\mathcal{L}_{task}$ is the task loss (e.g., cross-entropy) applied on the masked transformer outputs, $\mathcal{L}_{gate}$ is a sparsity constraint on the gate output m (before STE), which directly controls the model's computational cost, and λ is a hyperparameter balancing both losses. In the next paragraphs, we will motivate and define a novel gate loss to use for $\mathcal{L}_{gate}$.

Common gate sparsity losses. The L_0 loss is often used as sparsity loss in the conditional computing literature[52]. Given the 2-dimensional active token mask for the current batch, $m \in [0,1]^{B\times N}$, we define:

$$\mathcal{L}^{L_0}_{gate}(m) = \frac{1}{B}\sum_{b=1}^{B} \min\left(0, \frac{1}{N_{S_c}}\sum_{i=1}^{N_{S_c}} m_{b,i} - g^*\right) \quad (7)$$

where the hyperparameter $g^* \in [0,1]$ is the target rate for gate sparsity. However, L_0 only penalizes the *mean* of the distribution, and can not prevent the model from learning static patterns, such as assigning the same probability to all tokens independent of input, as illustrated in Figure 3 (a).

To enhance the desired conditional behavior, the recently proposed batch-shaping loss [13] (BaS) constrains the distribution of the gate outputs, across the batch, to match a certain prior p. In our setting, this means enforcing the *same prior* across each spatial position. This lacks the necessary flexibility for our use-case, as the gate could not learn for instance that edges of the image are less likely to contain fine-scale details. As a more flexible alternative, we apply BaS directly on the flattened distribution of the gate outputs:

$$\mathcal{L}^{BaS}_{gate}(m) = [\mathrm{CDF}(\{m_{b,i},\ \forall b, i\}) - \mathrm{CDF}(p(g^*))]^2 \quad (8)$$

where CDF is the cumulative distribution function, and p is a prior with mean g^*. Unfortunately, this variant is now too flexible, e.g. it cannot prevent spatial positions from being constantly on or off regardless of the input patch. Corner cases for both variants of BaS are illustrated in Figure 3 (c).

Generalized batch-shaping loss. To address these shortcomings, we introduce the *generalized batch-shaping loss* (**GBaS**) for finer control over the learned mask distribution, across both the batch and token dimensions. Like BaS, GBaS constrains the marginal distribution at each token spatial position, $m_{:,i}\ \forall i \in [1, N_{S_c}]$, but with a dedicated independent prior instead of a shared one. Manually setting the prior for each position would be tedious; Instead, we let the model learn each of these independent prior's parameters, while controlling their distribution using a *hyperprior* $\mathcal{P}$ with mean equal to the target sparsity g^* (Figure 3 (b)):

$$\mathcal{L}^{GBaS}_{gate}(m) = \sum_{i=1}^{N_S} [\mathrm{CDF}(\{m_{b,i},\ \forall b\}) - \mathrm{CDF}(p(\theta_i))]^2 + [\mathrm{CDF}(\{\theta_i,\ \forall i\}) - \mathrm{CDF}(\mathcal{P}(g^*;\sigma))]^2 \quad (9)$$

where θ are learned parameters defining each prior, and σ is a variance hyperparameter controlling the spread of the learned priors. When $\sigma = 0$, all priors are identical; hence we recover the original BaS; When $\sigma \to +\infty$, there is little constraint on the learned θ and we may encounter the same corner cases as for BaS applied to the flattened distribution.

In summary, GBaS enables fine-grained control over the learned distribution through the hyperprior. Another benefit of GBaS is that we can easily inject prior knowledge about which spatial positions are more likely to be kept at fine/coarse scale by initializing the θ parameters accordingly. In contrast, achieving a similar initialization with BaS would require pretraining the gate to match the desired prior. For instance, in most of our experiments with

GBaS, we initialize the learned prior parameters θ with the inverse normalized distances of each spatial position to the center. We further compare BaS and GBaS in ablation experiments in Section 4.3 and Appendix G. We use the Relaxed Bernoulli [28] distribution for the prior p, as we found it easier to parametrize than the Beta distribution used in BaS. We use a Gaussian for the hyperprior $\mathcal{P}$ with mean g^* and variance given by the hyperparameter σ.

2.3. Reducing the training overhead

When executing the model with batches of data, inactive tokens ($\overline{m}_i = 0$) cannot be pruned statically, as the masking pattern output by the gate g varies across the batch. Instead, we explicitly mask the inactive tokens in the attention layers and the output of the transformer backbone; the FFN layers are applied individually to every token and hence are not affected. Given the set of tokens across all scales, $x \in \mathbb{R}^{N \times d}$ and the current binary mask output by the gate, $\overline{m} \in \{0,1\}^N$, we must apply masking in every attention block, such that the inactive tokens are ignored when updating the representations of active ones:

$$\forall i, j \in [1, N],\ A^{\text{mask}}(x_i, x_j) = \frac{\overline{m}_j\ e^{Q_i K_j^T}}{\sum_{p=1}^{N} \overline{m}_p\ e^{Q_i K_p^T}} \tag{10}$$

where $A^{\text{mask}}(x_i, x_j)$ is the normalized attention score from token i to j and Q and K denote the query and key embeddings of the tokens. Unfortunately, with this naive masking approach the increased total number of tokens, $N = N_{S_f} + N_{S_c}$, leads to higher training costs.

To address this issue, we employ an *adaptive trimming* (**AT**) strategy at training time: For each image in the batch, we first reorder the tokens in descending order of the corresponding gate outputs m, omitting the class token or any task-specific token. This reordering step takes advantage of the fact that the transformer is not affected by the order of the tokens. We then trim the token dimension to only keep k tokens for each image, where k is the maximum number of active tokens in any image in the current batch. As a result, the number of tokens (and hence the computational cost) is lower bounded by N_{S_f}, i.e., the number of fine scale tokens. This strategy does impact the gradients received by the gate, effectively making the gating module less robust to tokens flipping from the coarse to the fine scale during training (see Appendix F). Nevertheless, as we show in Appendix F.3, this only leads to a small drop in accuracy in practice but a clear reduction in training time (~1.16-1.35 times per-epoch speedup, depending on the target sparsity). For this reason, we always use AT in our training pipeline.

3. Related work

Self-Attention for computer vision. Starting from Vision Transformer (ViT) [11, 9, 4, 32], Multiheaded Self-Attention (MHSA) has been successfully applied in many vision tasks such as image classification [11, 49], object detection [5, 61] or semantic segmentation [12, 59, 27]. While ViTs are often able to match CNN-based models' performance with fewer computational resources [11], the number of input tokens remains an important bottleneck to achieve efficient transformers. Several works [48] have focused on reducing the cost of the attention operation, which scales quadratically with the number of tokens, by using low-rank approximations [8, 54, 31] or exploiting redundant or sparse structures [19, 53, 16, 21, 26, 53]. However, unlike for NLP, the cost incurred by the Feed-Forward Neural Networks (FFNs) in ViTs is often significant due in part to the generally smaller number of tokens. Hence, instead of focusing only on attention layers, a number of techniques have been proposed to reduce the total number of tokens.

Token pruning and merging. Token pruning [56, 34, 57, 29, 20, 30, 23] and merging [35, 42, 3] are some of the most successful token reduction approaches in the literature. These methods usually prune away a fixed number of tokens in intermediate layers of the transformer based on their class attention score [23, 57] or on the previous layer's features [34], or merge tokens into a fixed smaller number of tokens using a cross-attention layer or projection [35, 42].

Orthogonal to these methods, our mixed-scale selection scheme outputs a dynamic number of tokens, tailored to the input image content. It is also designed as a preprocessing module acting on the token set before the first Transformer layer, and hence can be combined with methods such as token pruning or early-exiting which act on the intermediate transformer layers. Finally, in contrast to pruning, mixed-scale tokens are lossless, as every input image region is covered by a token. This is crucial for dense tasks such as segmentation where the final spatial predictions are usually directly reconstructed from the tokens.

Mixed-scale ViTs. Mixing features from different scales has shown positive results for convolutional networks [24, 25]. Following this insight, recent works have started to investigate ways to incorporate mixed-scale information in ViTs as well: For instance, Quadtree Attention [47] uses hierarchical structures to improve the efficiency of MHSA, while ReViT [62] learns a global input patch scale for each image with an EfficientNet backbone trained with precomputed proxy labels. The majority of these works treat each scale separately, either by incorporating extra parameters (entire branch [6, 55] or layernorm parameters [62]) or by training for each scale in separate stages [7, 62]. In contrast, we design a simple single-stage model which directly handles having mixed-scale tokens in one batch, for both training and inference, and learns the optimal scale selection pattern alongside the model features. Closest to our work is [40], which leverages saliency maps from a pretrained model to design a quadtree structure on token scales.

4. Experiments

4.1. ImageNet classification

We first benchmark the proposed mixed-scale tokenization on ImageNet [41]: We use publicly available SotA ViT backbones pretrained on ImageNet-21k [44, 11, 37], and DeiT backbones pretrained on ImageNet [49, 36]. We implement the gate as a lightweight 4-layer MLP with a scalar output in [0, 1], applied to every coarse token individually. After the first layer, a learned position encoding, specific to the gate, is also added to the token representations. Finally, the bias of the last layer is initialized such that the gate outputs 1: i.e., all patches are extracted at the fine scale at the beginning of training. We set all other hyperparameters to that of the original ViT (resp. DeiT) pipeline and finetune all models for 20 epochs with a batch size of 512 on a single device (see additional training details in Appendix C).

In Table 1, we evaluate the proposed mixed-scale MSViT across different backbone architectures (ViT-S and ViT-Tiny), pretraining strategies (ViT and DeiT), and input image sizes. We report top-1 accuracy results as well as MACs counts calculated via deepseed [38].

From the quantitative results, we observe that the mixed-scale models consistently reach higher accuracy at equivalent MACs across different compute budgets and input sizes. We also display qualitative examples of the mixed-scale selection patterns learned by the gate in Figure 1 and Appendix A: Despite having a limited field of view, the learned gate picks up on meaningful local features such as background/foreground distinction to select tokens' scales. Furthermore, we observe that the learned mixed-scale pattern is very similar across experimental settings: Two gates with the same number of active tokens, trained for MSViT-S/16 and MSViT-L/16 respectively, select the same scale for **78.4%** of the tokens on the ImageNet validation set. Similarly, the gates of a MSViT-S/16 model trained with 224px and 192px inputs respectively, agree for **87.1%** of the tokens. Motivated by this observation, we investigate in the next section whether the learned mixed-scale gate can be directly transferred as an off-the-shelf lightweight preprocessing module to other vision transformer-based models.

4.2. Transferring mixed-scale tokenization across tasks and backbones

4.2.1 Mixed-scale tokens for segmentation

In contrast to other token reduction methods, MSViT does not discard information about any input image regions. It can thus be readily applied to dense prediction tasks such as

DeiT-Small backbone	Avg # tokens	GMACs (avg)	accuracy top-1	accuracy top-5
DeiT-S/16 in=160	100	**2.27**	75.86	92.84
MSDeiT-S/16,32 in=224	**97**	**2.27**	**76.99**	**93.38**
DeiT-S/16 in=192	144	**3.32**	77.79	93.96
MSDeiT-S/16,32 in=224	**142**	**3.32**	**78.76**	**94.32**
DeiT-S/16 in=224	196	4.60	**79.85**	**94.57**
MSDeiT-S/16,32 in=224	**173**	**4.08**	79.38	94.38

ViT-Tiny backbone	Avg # tokens	GMACs (avg)	accuracy top-1	accuracy top-5
ViT-Ti/16 in=160	100	**0.60**	71.63	90.68
MSViT-Ti/16,32 in=224	**95**	**0.60**	**72.57**	**91.32**
ViT-Ti/16 in=192	144	0.89	74.24	92.22
MSViT-Ti/16,32 in=224	**138**	**0.88**	**74.93**	**92.54**
ViT-Ti/16 in=224	196	1.25	**76.00**	**93.26**
MSViT-Ti/16,32 in=224	**154**	**0.98**	75.51	92.98

ViT-Small backbone	Avg # tokens	GMACs (avg)	accuracy top-1	accuracy top-5
ViT-S/16 in=128	**64**	**1.44**	75.48	93.08
MSViT-S/16,32 in=224	75	1.76	**77.16**	**94.14**
ViT-S/16 in=160	100	**2.27**	78.88	94.95
MSViT-S/16,32 in=224	**98**	2.30	**79.51**	**95.33**
ViT-S/16 in=192	144	3.32	80.75	95.86
MSViT-S/16,32 in=224	**138**	**3.23**	**81.47**	**96.14**
ViT-S/16 in=224	196	4.60	**82.02**	**96.45**
MSViT-S/16,32 in=224	**187**	**4.43**	**82.02**	96.44

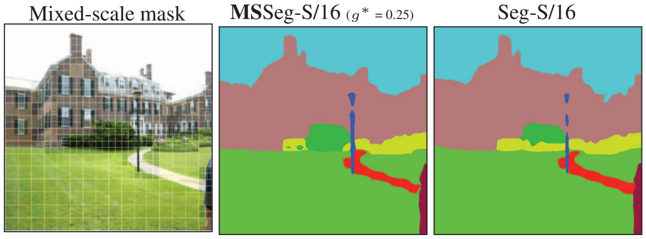
Table 1. Comparison of our dynamic mixed-scale model with the corresponding backbone baseline evaluated at different input image sizes. For ease of reading, the results are sorted by MACs, and grouped by backbones. Inside each table, we group results by comparable MAC counts or accuracy. We refer to models as "`arch/S in=X`", where `arch` is the backbone architecture, `X` is the input image size, and `S` is the patch scale(s). The prefix `MS` (Multi-Scale) refers to our mixed-scale models: We sweep over values of the gate target $g^* \in \{0.5, 0.25, 0.1\}$ and loss weight $\lambda \in \{1, 4, 20\}$ to obtain dynamic models with various MACs counts and report their GMACs and number of tokens averaged over the evaluation set (For reference, the additional computational cost induced by the gate for ViT-S is 0.017 GMACs). Additional results for all hyperparameters and different input image sizes, and including latency measurements, can be found in Appendix B.

token pruning. Following this insight, we augment the standard Segmenter training pipeline [18, 45] on ADE20k [60] with one of our gates, pretrained on ImageNet and frozen. The change is easy to implement: we replace the standard patch embedding of the ViT encoder with our own mixed-scale tokenization (Section 2.1) and keep it frozen during training. We then propagate the mixed-scale mask into further layers using masked attention (Equation (15)), and finally reshape the stream of mixed-scale tokens to the original 2D image shape before feeding it to the decoder (see Appendix E for details).

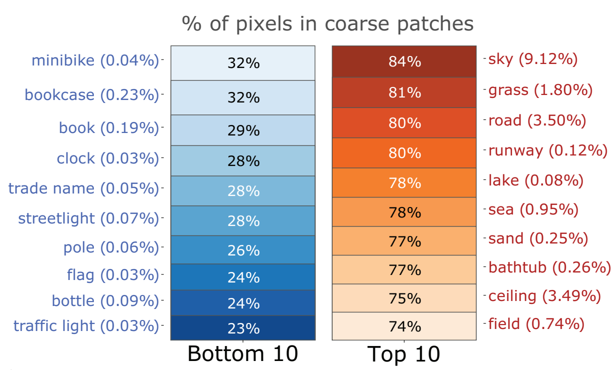
We report the results (mIOU, average MAC count, and average latency) in Figure 4 (a, b). Similar to the classifica-

Backbone	g^*	# tokens avg	MACs x 1e10	time ms	mIoU single-scale
Seg-T/16 (512px)	-	1024	1.04	113.68	38.1
MSSeg-T/16	0.5	655	0.56	86.12	37.9
	0.25	565	0.46	75.96	37.3
	0.1	525	0.42	69.13	36.8
Seg-S/16 (512px)	-	1024	3.17	252.09	45.3
MSSeg-S/16	0.5	684	1.92	184.81	44.9
	0.25	586	1.59	153.12	44.1
	0.1	552	1.48	144.02	43.3

(a) Single-scale segmentation results of our mixed-scale model with ViT-S and ViT-Ti backbones finetuned on ADE20K [60]. We measure the computational cost of the encoder, as the decoder cost is the same for both MSViT and ViT backbones; We also report the average runtime per image measured on a Geforce 2080 Ti GPU



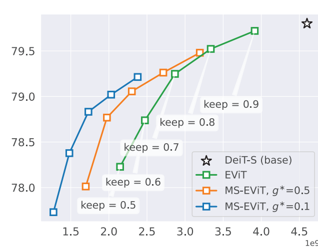
(b) Example of a mixed-scale mask and segmentation output, as well as the baseline backbone's output (*best seen zoomed*). We report additional qualitative results in Appendix D.



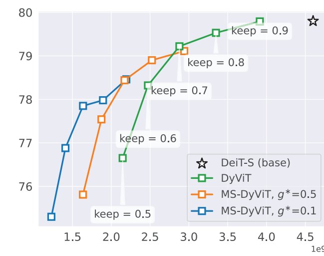
(c) ADE20K classes with the highest and lowest percentage of pixels falling in coarse patches. We also write the pixel frequency of each class in the whole dataset next to its label.

Figure 4. We train Segmenter [18, 45] on the ADE20k [60] dataset, after adding a (frozen) mixed-scale gate trained on ImageNet. We report quantitative results in Table **(a)**, a qualitative example in **(b)**, and a break-down of classes most often in coarse regions in **(c)**

tion task, we observe improved accuracy-efficiency trade-offs across different backbone sizes and gate sparsities: For instance with a ViT-S backbone, we can save roughly 40% MACs for a minor drop of 0.4 in mIoU. In addition, the scale selection pattern learned on ImageNet is still very meaningful for the images in ADE20k: In Figure 4 (c), we show that classes represented via coarse tokens often correspond to uniform regions such as sky or sea, which typically occupy large regions of the image.



(a) EViT [22] + mixed-scale



(b) DyViT [33] + mixed-scale

Figure 5. Mixed-scale tokenization combine well with token pruning methods, leading to improved efficient/accuracy trade-offs as compared to using token pruning on its own.

4.2.2 Pruning mixed-scale tokens

Token pruning methods iteratively discard a fixed ratio of the tokens in several intermediate layers of the transformer, based on their global class token attention [56, 34, 57, 29, 35, 20, 30]. In contrast, MSViT treats every local region individually and reduces the number of tokens before applying any transformer layer, using pixel-level information only, and without discarding any image region. As a result, both methods are orthogonal and select active tokens on different criteria. To verify how they interact, we augment two SotA pruning methods on DeiT-S, namely EViT [23, 22] and DyViT [34, 33], with one of our pretrained frozen gates instead of the standard ViT tokenization, and then train each model with their respective original pipeline, for different pruning ratios. We report results in Figure 5. We observe that mixed-scale tokenization followed by token pruning in the intermediate layers complement one another well, which also introduces an interesting trade-off: Rather than using very high pruning ratios, better accuracy/efficiency performance can be reached by combining mixed-scale tokenization with a token pruning ratio.

4.2.3 Mixed-scale tokens for hierarchical ViTs

Hierarchical (or Multi-scale) Vision Transformers [27, 26, 15, 10, 58] is a popular family of models that draw inspiration from the inductive bias of CNNs to build efficient ViT architectures: For instance in Swin, the image is initially split in very small tokens (4x4) which interact through local attention windows (7x7 tokens) and are progressively merged into larger tokens as depth increases.

To incorporate mixed-scale tokens in this scenario, we process fine-scale regions with the standard Swin paradigm; Coarse tokens on the other hand are passed through a single linear embedding and merged back in the stream of tokens at layer ℓ, once the fine tokens stream has been merged all the way up to the coarse scale. We discuss this process in more details in Appendix E. We report results for two values of ℓ in Table 2: The value of $\ell = 3$ yields better performance than merging the coarse and fine tokens in an earlier block ($\ell = 2$, bottom table).

$\ell = 3$	Base		$g^* = 0.5$		$g^* = 0.1$	
	acc	GMACs	acc	GMACs	acc	GMACs
Swin-T	81.0	4.3	80.0	3.6	78.8	3.1
Swin-S	83.4	8.6	82.4	6.9	81.4	5.9
Swin-L	86.0	33.8	85.4	27.7	84.7	23.3

$\ell = 2$	Base		$g^* = 0.5$		$g^* = 0.1$	
	acc	GMACs	acc	GMACs	acc	GMACs
Swin-T	81.0	4.3	80.4	4.0	79.6	3.8
Swin-S	83.4	8.6	83.1	8.2	82.5	8.0
Swin-L	86.0	33.8	85.9	32.6	85.4	31.6

Table 2. We incorporate mixed-scale information in Swin [26] by keeping coarse tokens determined by the gate out from the attention mechanism until layer ℓ. We then train the models at different sizes and gate sparsities in the origina Swin training pipeline.

4.3. Ablation experiments

4.3.1 Generalized batch-shaping loss (GBaS)

In Section 2.2, we introduced the novel GBaS, which allows for more control over the conditional behavior of the gate, and enables us to easily inject prior knowledge about the spatial distribution of the selected scale at initialization. In Figure 6 (a), we confirm that the best trade-off is achieved by GBaS, further improved when the learned priors are initialized as the inverse normalized distance of each spatial position to the center (`ctr init` for short).

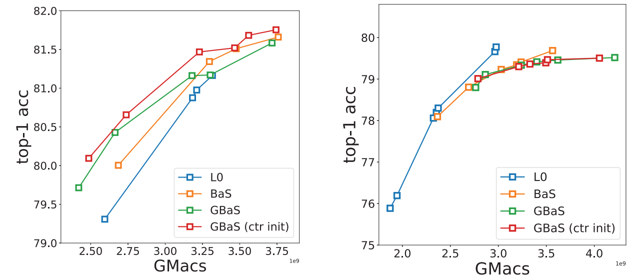
In addition, we observe that the cropping data augmentation used during training is a key factor. By default, we use the standard "Inception-style" cropping strategy[46] which leads to a shift between the tokens distributions at train and test time [50]. This behavior can be observed qualitatively in Figure 7 (a): When training with Inception crops, there is high uncertainty on the location of objects, and the L0 loss ends up stuck on a trivial static pattern early during training. On the other hand, GBaS learns more centered mixed-scale patterns, but still captures uncertainty over spatial positions through the learned priors (Fig. 7 (b) *top row*), which can be further reduced with `ctr init` (*bottom row*).

In contrast, with a lighter cropping strategy, all losses learn that, on average, fine scale tokens are more likely to appear in the center-top of the image, where the object to categorize usually lies (see Appendix G). As a result, all batch-shaping variants perform equally well, and the L0 loss even outperforms them in some cases (Figure 6 (b)).

In summary, GBaS is more robust to train/test discrepancy than other losses; Nevertheless when there is no notable distribution shift, then even a simple L0 sparsity loss can reach a similar or even better performance.

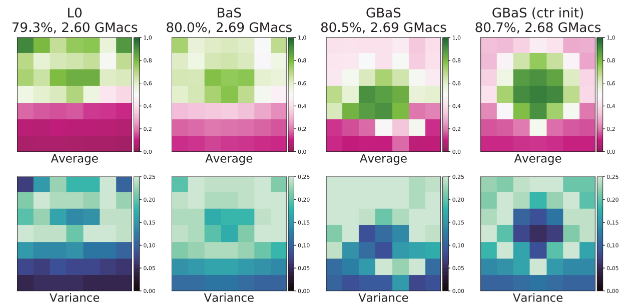
4.3.2 Benefits of learning a dynamic gate

In Figure 8, we illustrate how the learned gate module dynamically adapts the mixed-scale pattern, hence the computation cost, to the input image content. We further investi-

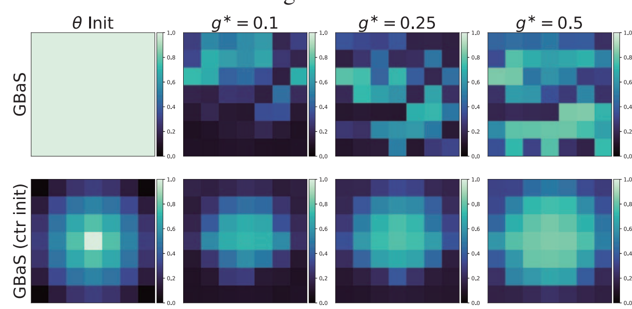


(a) Inception-style crops data augmentation (high train/test shift) **(b)** Light crops data augmentation (small train/test shift)

Figure 6. Accuracy-to-MACs comparison on MSViT-S/16,32 of the L0, batch-shaping (BaS) and generalized batch-shaping losses, with different random cropping augmentation strategies.



(a) Average (*top*) and variance (*bottom*) across the validation set of the learned masks selecting between **fine** and **coarse** scale.



(b) Prior parameters θ learned with the GBaS loss with/without `ctr init` (*top/bottom*). The first column is initial values of θ.

Figure 7. Illustration of the masks **(a)** and priors **(b)** learned by the model with Inception-style crop data augmentations: The gate is more prone to learn trivial mixed-scale patterns if not controlled properly during training using the GBaS. In addition, initializing the prior parameters in GBaS with the `ctr init` is enough to guide the gate towards a more central pattern, as illustrated in **(b)**.

gate and highlight this behavior quantitatively in Appendix G.1, in which we compare using a learned gate versus using a fixed oracle mixed-resolution pattern where all central patches are at the fine scale, and any region further than a certain radius from the center is kept at coarse scale.

5. Conclusions

In this work, we proposed a dynamic mixed-scale tokenization scheme for ViT, MSViT, via a novel conditional

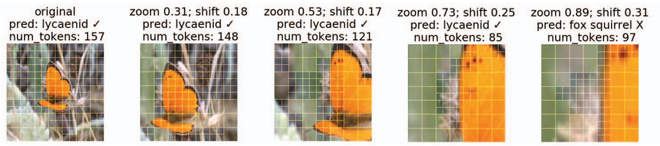


Figure 8. Example of the learned dynamic gate outputs when applied on random image zooms and shifts of the validation dataset

gating mechanism. The gate is agnostic to the choice of transformer backbone, and is trained jointly with it, in a single-stage, with mixed-scale tokens. To improve the conditional behaviour of the gate, we proposed a generalization of batch-shaping [13] to better handle *multi-dimensional distributions*. GBaS improves results and allows for easier and better initialization of the gates. Our experiments on image classification and semantic segmentation show that the proposed dynamic tokenization enhances computational efficiency by reducing the number of input tokens, with minimal impact on performance. For both tasks, the gate learns to represent uniform and background regions with coarse tokens and higher entropy regions with fine ones.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *ArXiv*, abs/1308.3432, Aug. 2013. 3

[2] Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim M. Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. *ArXiv*, abs/2212.08013, 2022. 3

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023. 5

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, 2020. 5

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 5

[6] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. CrossViT: Cross-attention multi-scale vision transformer for image classification. In *ICCV*, 2021. 2, 3, 5

[7] Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. Coarse-to-Fine vision transformer. *ArXiv*, abs/2203.03821, 2022. 2, 3, 5

[8] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with Performers. In *ICLR*, 2021. 5

[9] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the Relationship between Self-Attention and Convolutional Layers. In *ICLR*, 2020. 5

[10] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *CVPR*, pages 12114–12124, 2021. 7

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 1, 3, 5, 6

[12] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W. Taylor. SSTVOS: Sparse spatiotemporal transformers for video object segmentation. In *CVPR*, 2021. 5

[13] Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-shaping for learning conditional channel gated networks. In *ICLR*, 2019. 2, 4, 9

[14] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. In *ICLR Workshop*, 2014. 3

[15] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *ICCV*, pages 6804–6815, 2021. 7

[16] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. 5

[17] George Hinton. Neural networks for machine learning. *Coursera*, video lectures, 2012. 3

[18] Inria. Segmenter. `https://github.com/rstrudel/segmenter`. 6, 7, 14

[19] Nikita Kitaev, L. Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. 5

[20] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. SPViT: Enabling faster vision transformers via soft token pruning. *ArXiv*, abs/2112.13890, 2021. 2, 5, 7

[21] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding, June 2020. 5

[22] Youwei Liang. Evit. `https://github.com/youweiliang/evit`. 7, 13

[23] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *ArXiv*, abs/2202.07800, 2022. 5, 7

[24] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CVPR*, 2017. 5

[25] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE TPAMI*, 42:318–327, 2020. 5

[26] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer V2: Scaling up capacity and resolution. In *CVPR*, 2022. 5, 7, 8

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 5, 7

[28] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017. 3, 5

[29] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. AdaViT: Adaptive vision transformers for efficient image recognition. In *CVPR*, 2022. 2, 5, 7

[30] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Fast vision transformers with HiLo attention. *ArXiv*, abs/2205.13213, 2022. 2, 5, 7

[31] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention, Mar. 2021. 5

[32] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 5

[33] Yongming Rao. Dyvit. https://github.com/raoyongming/DynamicViT. 7, 13

[34] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *NeurIPS*, 2021. 2, 5, 7

[35] Cédric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *ArXiv*, abs/2202.12015, 2022. 2, 5, 7

[36] Facebook Research. Data-efficient architectures and training for image classification. https://github.com/facebookresearch/deit. 6, 13

[37] Google Research. Vision transformer. https://github.com/google-research/vision_transformer. 6, 13

[38] Microsoft Research. Deepspeed. https://github.com/microsoft/DeepSpeed. 6, 13

[39] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *NeurIPS*, 2021. 3

[40] Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization. *Workshops of CVPR*, abs/2304.00287, 2023. 6

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. In *IJCV*, 2015. 6

[42] Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: What can 8 learned tokens do for images and videos? *ArXiv*, abs/2106.11297, 2021. 2, 5

[43] Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017. 3

[44] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *ArXiv*, abs/2106.10270, 2021. 6

[45] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. *ICCV*, 2021. 6, 7

[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 8

[47] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. QuadTree attention for vision transformers. In *ICLR*, 2022. 5

[48] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2020. 5

[49] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 5, 6

[50] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *NeurIPS*, 2019. 8

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017. 1, 2

[52] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018. 4

[53] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21665–21674. Curran Associates, Inc., 2020. 5

[54] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768, 2020. 5

[55] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *NeurIPS*, 2021. 3, 5

[56] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-ViT: Slow-Fast token evolution for dynamic vision transformer. In *AAAI*, 2021. 2, 5, 7

[57] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *CVPR*, 2022. 2, 5, 7

[58] Xiaosong Zhang, Yunjie Tian, Lingxi Xie, Wei Huang, Qi Dai, Qixiang Ye, and Qi Tian. Hivit: A simpler and more efficient design of hierarchical vision transformer. In *ICLR*, 2023. 7

[59] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. 5

[60] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 6, 7, 14

[61] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 5

[62] Yichen Zhu, Yuqin Zhu, Jie Du, Yi Wang, Zhicai Ou, Feifei Feng, and Jian Tang. Make a long image short: Adaptive token length for vision transformers. *ArXiv*, abs/2112.01686, 2021. 2, 3, 5