TRAPE: COMPACT AXIAL POSITION TAPES FOR CROSS-RESOLUTION VISION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision transformers trained at one resolution often degrade at larger resolutions or other aspect ratios. We propose TRAPE (Tabulated Relative Axial Positional Encoding), a relative attention bias trained as a continuous function and deployed as compact tables. Along each axis, TRAPE adds a local correction table to a continuous function of log-spaced offsets with a fixed reference; a shared learnable amplitude bounds the sum of both axis responses. The fixed reference makes a smaller grid's table an exact centre slice of a larger one. Fine-tuning only positional parameters at a few square resolutions yields a *resolution tape* of per-axis tables that height and width slice and blend independently, serving unseen resolutions and aspect ratios with one batched lookup and no positional network. On ImageNet-1k, TRAPE-DeiT-S trained at 224^2 retains 70.2% top-1 at 1024^2 without fine-tuning, against 36.4% for resized absolute embeddings and 12.1% for 2D RoPE-Mixed. As a zero-initialized branch, TRAPE raises five frozen public ViTs by 18–56 points at 1024^2 with under 0.1 ImageNet epoch of position-only adaptation. On one A800 GPU, assembling tables for a new shape takes about 0.1 ms. Fused attention avoids materializing the $N \times N$ bias matrix, enabling over $10\times$ the batch size of dense attention at 1024^2. Hierarchical TRAPE backbones with axial attention improve ADE20K segmentation by 3.0–3.1 mIoU over Swin with fewer FLOPs.

1 INTRODUCTION

Vision transformers (ViTs) are commonly pre-trained at a fixed input resolution but deployed across image sizes and aspect ratios (Dosovitskiy et al., 2021; Touvron et al., 2021); dense prediction in particular needs larger and often non-square inputs (Liu et al., 2021). Such changes alter the number of patch tokens and the offsets between them. Self-attention accepts the new sequences, but the learned positional representation may not transfer, so adapting it while reusing the pre-trained backbone is a practical problem.

Absolute position embeddings (APE) attach learned vectors to the training grid and must be interpolated when the grid changes. Relative position biases encode query–key offsets, but free tables remain tied to the offset range seen during training (Shaw et al., 2018; Liu et al., 2021). The log-spaced continuous position bias (Log-CPB) of Swin V2 generates biases from offset coordinates and can be evaluated at new offsets (Liu et al., 2022). Rotary position embeddings (RoPE) introduce relative position through query and key rotations, and two-dimensional variants improve resolution extrapolation in ViTs (Su et al., 2024; Heo et al., 2024). These mechanisms accept new grids, yet accuracy can still collapse far beyond the training range: a DeiT-S with 2D RoPE-Mixed peaks at 81.6% top-1 at 336^2 but falls to 12.1% at 1024^2, and bicubic APE reaches 36.4% (Table 1).

Supporting many resolutions also requires adapting positional representations on small budgets and reusing them across shapes. Rather than changing training or tokenization (§2), we adapt only the positional parameters of a backbone pre-trained at one resolution, with content weights and head frozen. A continuous bias must then keep its generator at inference or cache its outputs per shape; a joint 2D table stores $(2H-1)(2W-1)$ entries per block and head for an $H \times W$ grid and couples the axes, so square-fitted responses cannot be recombined per axis.

We propose TRAPE (**T**abulated **R**elative **A**xial **P**ositional **E**ncoding), a relative attention bias with distinct training and deployment representations (Figure 1). During training, each axis combines a continuous function of log-spaced offsets, which models smooth trends over the whole offset range,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

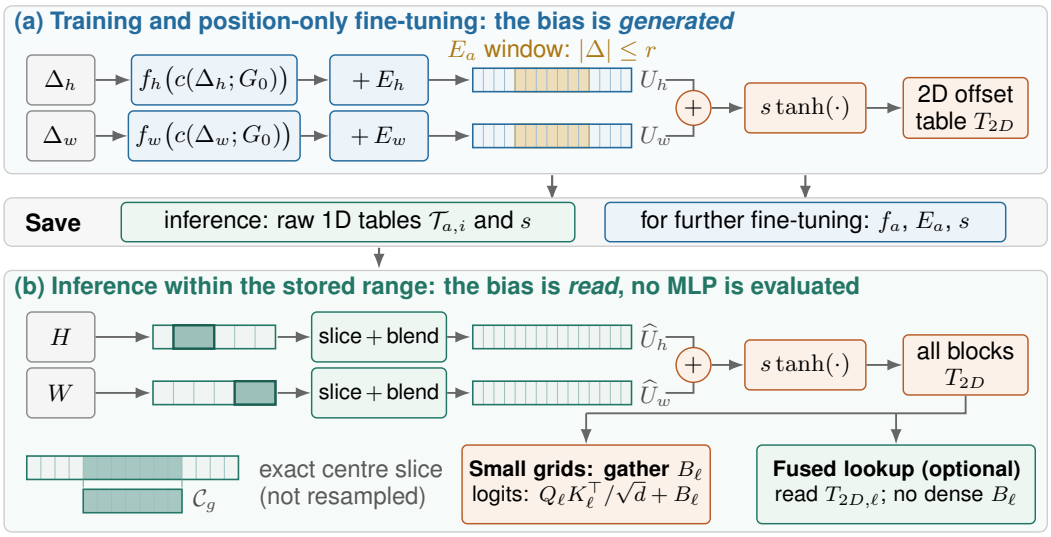


Figure 1: **TRAPE generates its bias with trainable functions and serves it from exported tables.** (a) Axis functions of the fixed-reference coordinate $c(\cdot; G_0)$ plus central tables E_a give raw tables U_h, U_w; one shared $s\tanh$ bounds their sum. **Save:** each anchor i exports raw tables $\mathcal{T}_{a,i}$ (the resolution tape). (b) Each axis blends exact centre slices $\mathcal{C}_g$ of its own bracketing anchors (highlighted; sides $\{144, \ldots, 1024\}$): a 320×960 input reads the $224/336$ pair for height and the $704/1024$ pair for width (Eq. 6). Dense and fused lookups are execution options, both timed in Section 4.6. Blue: trainable; green: exported; orange: composition.

with a local correction table that adds flexibility near the origin (Figure 1(a)). A single $\tanh$ with a shared learnable amplitude bounds the sum of the two axis responses. At export, each correction is merged into its axis response, leaving two raw 1D tables per block whose stored lengths add rather than multiply.

Because the coordinate reference stays fixed as the grid changes, a token offset receives the same coordinate at every resolution, and for a fixed parameter state the table of any smaller grid is an *exact centre slice* of a larger one. We use this to build a *resolution tape*: a bank of axis tables obtained by fine-tuning only the positional parameters at a few square anchors. At inference, height and width independently select their bracketing anchors and blend the required slices (Figure 1(b)): unseen rectangles within the stored range need no rectangular adaptation, no positional network runs, and a fused kernel reads the composed offset tables without forming the $N \times N$ bias. A zero-initialized TRAPE branch also adapts frozen public ViTs while retaining their native encoding and initial predictions.

We evaluate direct transfer, position-only adaptation and inference cost separately, and distinguish fitted anchors from unseen shapes. Our contributions are:

1. **An axial positional encoding for resolution transfer.** TRAPE combines continuous axis functions, local correction tables and a shared output amplitude; trained only at 224^2, TRAPE-DeiT-S retains 70.2% top-1 at 1024^2 without fine-tuning, where resized APE and 2D RoPE-Mixed reach 36.4% and 12.1% (§4.2).
2. **Parameter-efficient adaptation and table-only deployment.** Exact centre slicing and independent axis blending turn square adaptation runs into 132–527 KiB tapes for DeiT-Ti/S/B (§4.3). Grafted onto five frozen public ViTs, the branch raises 1024^2 accuracy by 18–56 points with under 0.1 ImageNet epoch, e.g. from 16.6% to 72.6% on a RoPE ViT-S, and unseen 320×960 inputs by up to 12.5 points (§4.4). On one A800 GPU, serving a new shape takes about 0.1 ms, and a fused kernel that reads the tables instead of an $N\times N$ bias fits over $10\times$ larger batches at 1024^2 (§4.6).
3. **Evaluation across classification and dense prediction.** With the official code and configurations of Swin and CSWin, hierarchical backbones combining TRAPE with axial attention improve ADE20K by 3.0–3.1 mIoU over Swin with fewer FLOPs, and position-only adaptation improves them at larger test scales (§4.5).

2 Related Work

Positional representations and resolution flexibility. ViTs use absolute embeddings (Dosovitskiy et al., 2021; Touvron et al., 2021), context-conditioned encodings (Chu et al., 2023), or relative representations of query–key displacement. Relative mechanisms include free offset tables (Shaw et al., 2018; Raffel et al., 2020; Liu et al., 2021; Bao et al., 2022; Wu et al., 2021), fixed distance penalties (Press et al., 2022), continuous Log-CPB (Liu et al., 2022), and rotary embeddings (Su et al., 2024; Heo et al., 2024). TRAPE builds on Log-CPB's fixed-reference log coordinate and adds axial functions, local correction, a shared output bound and independently routed tables. Decomposed relative embeddings in ViTDet and MViTv2 also split offsets along height and width, but enter the logits through query-dependent terms (Li et al., 2022a;b); TRAPE adds a query-independent scalar bias, and the resampled axial free-table row of Table 1 is its closest scalar-bias counterpart. FlexiViT, NaViT and ResFormer modify pre-training or tokenization to support multiple input sizes (Beyer et al., 2023; Dehghani et al., 2023; Tian et al., 2023), and FixRes adapts a classifier to its test size (Touvron et al., 2019); TRAPE is complementary, updating only the positional parameters of a backbone pre-trained at one resolution and storing the fitted responses as per-axis tables.

Axial attention and execution. Axial and factorized attention are established designs (Ho et al., 2019; Huang et al., 2019; Wang et al., 2020; Dong et al., 2022; Tu et al., 2022; Chu et al., 2021). Factorizing a positional response does not itself factorize attention: our plain ViTs retain full spatial attention, while the hierarchical variant uses single-axis attention. Merging local corrections into exported tables is a simple re-parameterization, related in purpose to Ding et al. (2021). Memory-efficient and fused attention avoid materializing the $N \times N$ score matrix (Rabe & Staats, 2021; Dao, 2024); a bias given by a compact offset table can be read inside such kernels, e.g. in Triton (Tillet et al., 2019) or FlexAttention (Dong et al., 2025).

3 Method

3.1 An axial continuous positional bias

Setting and notation. An image is split into an $H \times W$ grid of patch tokens, $N = HW$, and token p sits at row h_p and column w_p (row-major order). For spatial tokens, one head with queries, keys and values $Q, K, V \in \mathbb{R}^{N \times d}$ computes (Vaswani et al., 2017)

$$A = \mathrm{softmax}\left(QK^\top/\sqrt{d} + B\right)V, \qquad B_{pq} = b(\Delta_h, \Delta_w), \quad \Delta_h = h_q - h_p, \;\; \Delta_w = w_q - w_p, \tag{1}$$

so the expanded bias $B \in \mathbb{R}^{N \times N}$ depends only on key-minus-query offsets, and b is evaluated on the $(2H-1)(2W-1)$ offset pairs and then gathered. Each of the n_h heads of each of the L blocks has its own b (indices stay implicit), and bias entries that involve the class token are zero. Swin V2 maps an offset to the log-spaced coordinate

$$c(\Delta; G_0) = \mathrm{sign}(\Delta)\, \frac{\log_2\big(1 + 8|\Delta|/(G_0 - 1)\big)}{\log_2 8}, \tag{2}$$

where $G_0 > 1$ is the fixed side of the *pre-training* grid (G_0=14 for 224^2 inputs and patch size 16), and its Log-CPB sets $b = 16\,\sigma\big(g(c(\Delta_h; G_0), c(\Delta_w; G_0))\big)$ with a two-layer MLP $g : \mathbb{R}^2 \to \mathbb{R}^{n_h}$ and $\sigma(u) = 1/(1 + e^{-u})$ (Liu et al., 2022; Microsoft, 2022). The denominator $G_0 - 1$ stays fixed, so an offset has the same coordinate at every resolution. The logarithm compresses the extrapolation range: the largest coordinate grows from 1.06 at 14×14 to 1.77 at 64×64, while the maximum token offset grows $4.85\times$.

The TRAPE bias. TRAPE keeps Eq. 2 and replaces the 2D function g by one function per axis $a \in \{h, w\}$. The *axis function* $f_a : \mathbb{R} \to \mathbb{R}^{n_h}$ is a two-layer MLP with one ReLU hidden layer of width $d_c = 2G_0 - 1$, a hidden-layer bias and no output-layer bias. A free *central table* $E_a \in \mathbb{R}^{K \times n_h}$, initialized to zero, has $K = 2r + 1$ rows (K=13, r=6). Define its zero extension by $e_a(\Delta) = E_a[\Delta + r, :]$ for $|\Delta| \le r$ and $e_a(\Delta) = \mathbf{0}$ otherwise. The raw response and the bias of head j are

$$U_a(\Delta) = f_a\big(c(\Delta; G_0)\big) + e_a(\Delta) \;\in \mathbb{R}^{n_h}, \tag{3}$$

$$b_j(\Delta_h, \Delta_w) = s\, \tanh\big([U_h(\Delta_h)]_j + [U_w(\Delta_w)]_j\big), \tag{4}$$

with a separate pair of functions and tables in each block (Figure 1(a)). We omit the head index when it is clear. The amplitude s is one learnable scalar shared by all blocks and heads, initialized to 8 and excluded from weight decay, so $|b| \le |s|$. For a side-g axis, $U_a^{(g)} \in \mathbb{R}^{n_h \times (2g-1)}$ denotes the table of U_a over the offsets $-(g-1), \dots, g-1$, so entry k belongs to $\Delta = k - (g-1)$ and an $H \times W$ grid uses $U_h^{(H)}$ and $U_w^{(W)}$. Generating the bias thus takes $(2H-1) + (2W-1)$ evaluations of the axis functions instead of $(2H-1)(2W-1)$ evaluations of g (54 instead of 729 at 224^2), with 4.2k instead of 36.9k position parameters on DeiT-Ti.

3.2 Two key components

A shared learnable amplitude: s in Eq. 4. Since $16\,\sigma(u) = 8 + 8\tanh(u/2)$, the Swin V2 bound is a $\tanh$ of fixed amplitude 8 and slope 4 at the origin, plus a constant that cancels in the softmax only when it is added to every logit of a row, special tokens included (Appendix B.5). TRAPE uses a learnable amplitude: training yields $s = 12.18$, 11.94 and 14.69 on DeiT-Ti/S/B, i.e. 3.0–3.7$\times$ the slope of 16σ at the origin (Figure 5a). Row (c) of Table 1 makes this change on the factorized form and adds 2.8 points at 560^2 and 10.9 at 1024^2; it changes the input scale of the $\tanh$ as well as its amplitude, so it does not isolate one from the other. Per-block amplitudes reduce transfer accuracy in most measured shapes (Appendix C). For TRAPE adaptation, s stays frozen, so all anchors use the same output map when their raw tables are blended.

A re-parameterized central table: E_a in Eq. 3. The continuous f_a are piecewise linear in c; E_a supplies additional capacity for local variation. E_a is indexed by the offset in tokens and anchored at $\Delta{=}0$, so the same values apply at every resolution, and outside its window the response is the continuous function alone. Because the correction is added before the nonlinearity, export merges it into U_a with no separate correction lookup or extra table storage. It adds 0.8 points at 560^2 and 2.1 on $320{\times}960$ (row (d)); Appendix H visualizes the learned corrections and their effect on the bias. Squashing each axis separately, $s\tanh U_h + s\tanh U_w$ with bound $2|s|$, defines a different function; it is within 0.2 points on DeiT-Ti but 0.6 points worse on DeiT-S at 560^2 (Table 20).

3.3 Deployment: exported axis tables and the resolution tape

Training uses functions and local corrections; deployment uses their evaluated responses (Figure 1). By *deployment* we mean three things: *storage*, a bank of raw per-axis tables and the scalar s; *inference*, which reads these tables and evaluates no positional network inside the stored range; and *serving*, which routes height and width independently. Training builds differentiable tables from the current parameters through Eqs. 3–4; after fitting we export detached raw axis tables, so the nonlinear composition remains part of inference but the positional MLPs do not.

Centre-slice identity. For a table T of length $2S-1$ and $1 \le g \le S$, define $\mathcal{C}_g(T)$ as entries $S-g, \dots, S+g-2$ (zero-based). This slice contains exactly the offsets $-(g-1), \dots, g-1$. Neither Eq. 2 nor E_a depends on the grid size, so for one fixed parameter state

$$U^{(g)} = \mathcal{C}_g\big(U^{(S)}\big) \qquad \text{for all } S \ge g, \tag{5}$$

in exact arithmetic, including for averages of raw tables (Appendix A). No resampling is needed. The identity applies to one parameter state; it does not equate responses from differently fitted anchors.

Anchors, each stored up to the next. A function read far outside its trained range still extrapolates, so we fine-tune the encoding at a few *anchor* sides, spaced approximately evenly in $x(g) = c(g-1; G_0)$ over $g \in [9, 64]$: $\mathcal{G} = \{9, 14, 21, 30, 44, 64\}$, or image sides $\{144, 224, 336, 480, 704, 1024\}$. Side 14 is the pre-training grid and is not fine-tuned. Only f_h, f_w, E_h, E_w are trained; the backbone, the classifier and s are frozen. Anchor g_i exports its raw tables up to the side $S_i = g_{i+1}$ of the next anchor ($S_i = g_i$ for the last one), so any side between two anchors can be sliced from both. For each anchor i and axis a, the tables of all L blocks form one array $\mathcal{T}_{a,i} \in \mathbb{R}^{L \times n_h \times (2S_i - 1)}$ with E_a merged in.

Serving an arbitrary shape, one axis at a time. For a side g with $g_i \le g \le g_{i+1}$,

$$\alpha_g = \frac{x(g) - x(g_i)}{x(g_{i+1}) - x(g_i)}, \qquad \widehat{U}_a^{(g)} = (1-\alpha_g)\,\mathcal{C}_g(\mathcal{T}_{a,i}) + \alpha_g\,\mathcal{C}_g(\mathcal{T}_{a,i+1}), \tag{6}$$

Table 1: **Cross-resolution generalization without fine-tuning.** Each encoding is trained once at 224^2 with the same DeiT recipe and evaluated unchanged at other shapes (ImageNet-1k top-1, %, 50,000 images; height×width in pixels). APE: resized bicubically; 2D RoPE-Mixed: Heo et al. (2024); free tables replace the axis functions and are resampled. Rows (a)–(d) add axial factorization, the $s\tanh$ output map and the central table; 2D-Tanh adds both components to (a) without factorization. Bold: best per model size. All shapes: Appendix C.

Position encoding	Train	Square generalization ($H{=}W$)							Rectangular generalization ($H{\neq}W$)				
	224^2	336^2	384^2	480^2	560^2	704^2	800^2	1024^2	384×512	512×384	448×672	672×448	320×960
DeiT-Ti: baselines and component ablations													
APE, bicubic	72.1	71.9	71.2	68.1	64.9	58.1	53.1	41.8	69.4	68.8	65.1	64.1	60.9
axial free tables, resampled	72.1	70.5	69.9	66.8	63.1	55.9	50.4	39.1	67.8	67.5	63.1	62.8	53.4
(a) Log-CPB, $16\sigma(\cdot)$	72.2	73.4	73.2	72.4	71.2	68.2	65.2	52.6	72.5	72.3	71.1	70.3	64.3
(b) + axial factorization	72.1	73.0	72.8	71.6	70.0	65.9	62.5	52.0	71.9	71.7	69.8	68.8	61.9
(c) + shared $s\tanh$ output	**72.8**	74.2	74.1	73.9	72.8	70.6	68.7	62.9	73.7	73.6	72.6	72.3	66.0
(d) + central table (= TRAPE)	**72.8**	**74.4**	**74.7**	**74.2**	**73.6**	**71.5**	**69.5**	**63.4**	**74.4**	**74.0**	**73.3**	**72.6**	**68.1**
DeiT-S													
APE, bicubic	79.8	79.4	78.4	75.1	71.0	61.5	54.0	36.4	76.4	76.0	71.4	70.4	65.5
2D RoPE-Mixed	**80.8**	81.6	81.1	78.9	74.8	59.7	43.4	12.1	79.9	79.3	74.8	73.8	65.7
2D-Tanh (not factorized)	80.6	**81.8**	**81.8**	81.3	80.6	78.2	75.1	23.6	**81.5**	**81.2**	80.3	79.9	76.2
TRAPE	80.6	81.7	**81.8**	**81.4**	**80.7**	**78.8**	**76.9**	**70.2**	**81.5**	**81.2**	**80.7**	**80.0**	**76.3**
DeiT-B													
APE, bicubic	81.8	81.9	81.4	79.6	77.5	72.6	61.3	59.6	80.2	80.1	77.6	77.0	73.6
TRAPE	**82.1**	**82.8**	**82.9**	**82.3**	**81.5**	**79.7**	**78.2**	**73.3**	**82.5**	**82.1**	**81.4**	**80.8**	**78.5**

applied to all blocks and heads at once; the bias then follows Eq. 4 with $\widehat{U}_h^{(H)}, \widehat{U}_w^{(W)}$ in place of $U_h^{(H)}, U_w^{(W)}$. At an anchor, including both range endpoints, the centre slice is returned directly. Between anchors, Eq. 6 is an empirical interpolation rule, not an accuracy guarantee. Blending acts on raw tables, before the $\tanh$, and the two axes are routed independently: a 320×960 input reads its rows from the $224/336$ pair and its columns from the $704/1024$ pair, so serving this rectangle requires no rectangular anchor. The stored lengths sum to 468 per head and axis, i.e. $2Ln_h \cdot 468$ fp32 values or $131.6/263.3/526.5$ KiB for DeiT-Ti/S/B, excluding metadata and the retained trainable state.

Assembly and attention execution. For a fixed shape, *path A* composes the compact 2D offset tables of all blocks once, $T_{2D} = s\tanh(\widehat{U}_h \oplus \widehat{U}_w) \in \mathbb{R}^{L\times n_h\times(2H-1)\times(2W-1)}$ with $(\widehat{U}_h \oplus \widehat{U}_w)[i,k] = \widehat{U}_h[i] + \widehat{U}_w[k]$, and block ℓ reads $B_{pq} = T_{2D,\ell}[\Delta_h{+}H{-}1, \Delta_w{+}W{-}1]$. Small grids can gather B_ℓ before the softmax of Eq. 1; large grids can read $T_{2D,\ell}$ inside a fused attention kernel instead: at 1024^2 the compact fp32 tables of all ViT-B blocks, a per-shape cache separate from the tape, take 8.9 MiB, against 768 MiB for one block's B_ℓ. Our Triton kernel follows the online softmax of FlashAttention-2 and reduces the lookup of a query–key pair to one subtraction of per-token integer codes; FlexAttention (Dong et al., 2025) offers the same lookup as a score modification, and an equivalent *path B* indexes the raw axes first (Appendix A.4). Reported accuracies use exact query-chunked attention (Rabe & Staats, 2021).

3.4 Grafting a tape onto a pre-trained ViT

We add Eqs. 3–4 to every attention block of a public ViT, keeping its native position encoding, and initialize the output layer of each f_a and E_a to zero, so $B = 0$ and the initial attention is unchanged; recorded checks confirm bitwise-identical logits (Appendix A.5). The hidden layer of each f_a is a frozen bank of ReLU hinges in c; only its output layer and E_a are tuned, while $s = 10$ and the backbone stay fixed, and larger anchors add hinges with zero output weights. Export and serving follow §3.3; a held-out training subset keeps an adapted state only if it beats the neutral branch (Appendix B.4).

3.5 A hierarchical backbone with axial attention

TRAPE-Swin preserves Swin's stage widths, depths, heads and patch merging (Liu et al., 2021) and replaces shifted windows by single-axis global attention (Appendix I): alternate blocks attend along the height and the width with the corresponding 1D bias $s_i \tanh U_a$, so two successive blocks connect any two tokens at $O(CHW(H{+}W))$ cost instead of $O(C(HW)^2)$ for C channels. A

Table 2: **Position-only fine-tuning of TRAPE-DeiT** (ImageNet-1k top-1, %). No-FT: checkpoints of Table 1; TRAPE: the same frozen backbones served from their tapes, PE-FT at $\{144, 336, 480, 704, 1024\}^2$ (†). DeiT-S: mean ± s.d. over three adaptation seeds. Means: five unseen rectangles; 19 shapes with both sides ≥224 (all 22 shapes: Appendix D).

Input	DeiT-Ti			DeiT-S (3 seeds)			DeiT-B		
	No-FT	TRAPE	Δ	No-FT	TRAPE	Δ	No-FT	TRAPE	Δ
224^2	72.8	72.8	—	80.6	80.7	—	82.1	82.1	—
336^2 †	74.4	**74.7**	+0.3	81.7	**81.9** ±0.04	+0.2	82.8	**82.9**	+0.1
384^2	74.7	**75.1**	+0.4	81.8	**82.1** ±0.06	+0.3	82.9	**83.3**	+0.4
480^2 †	74.2	**75.2**	+1.0	81.4	**82.1** ±0.07	+0.7	82.3	**83.2**	+0.9
560^2	73.6	**75.1**	+1.5	80.7	**81.9** ±0.02	+1.2	81.5	**83.3**	+1.8
704^2 †	71.5	**74.0**	+2.5	78.8	**80.9** ±0.08	+2.1	79.7	**82.5**	+2.8
800^2	69.5	**73.1**	+3.6	76.9	**80.0** ±0.04	+3.1	78.2	**82.2**	+4.0
1024^2 †	63.4	**70.4**	+7.0	70.2	**77.0** ±0.16	+6.8	73.3	**81.1**	+7.8
mean, 5 rectangles			+1.4			+1.1 ±0.07			+1.5
mean, 19 shapes			+2.0			+1.7 ±0.05			+2.1

parallel depthwise 3×3 convolution on the values supplies local mixing, and each stage i has its own amplitude s_i. An axial CSWin-derived variant (Dong et al., 2022) normalizes offsets by the current axis length and overwrites the central window, so the centre-slice identity does not apply to it; we report it separately (Appendix I).

4 Experiments

4.1 Setup

Training. All models use official scripts and recipes with only the model definition changed. Plain models follow the 300-epoch DeiT recipe on ImageNet-1k (Touvron et al., 2021; Deng et al., 2009) without absolute embeddings. For a fair comparison, TRAPE-Swin and TRAPE-CSWin keep the architecture configuration of Swin (Liu et al., 2021) and CSWin (Dong et al., 2022) (stage widths, depths, heads, stem and merging layers) and use their official code and configurations unchanged, both for ImageNet-1k pre-training and for UPerNet (Xiao et al., 2018) segmentation in MMSegmentation (MMSegmentation Contributors, 2020) on ADE20K (Zhou et al., 2017) (512×512 crops, 160k iterations); only attention and position encoding differ. Appendix B gives every recipe and budget.

Evaluation. Sizes are $R_h \times R_w$ pixels (R^2 if square); feature grids are $H \times W$ tokens. Classification uses all 50,000 validation images; design sweeps use a fixed 10,000-image subset, and grafted runs select on 5,000 held-out training images. Our checkpoints and the public models use different resize protocols, and comparisons stay within one protocol (Appendix B). Each model is trained once; position-only adaptation is repeated with three seeds for TRAPE-DeiT-S and at the 704^2 anchor of a public ViT-S. Values are rounded half up to one decimal; differences and means use displayed values. *Without fine-tuning* (No-FT) evaluates a checkpoint at other sizes with no update; *position-only fine-tuning* (PE-FT) fits only position parameters at the square anchors of each caption, with backbone and head frozen. A † or a square marker denotes a fitted anchor; all other sizes and every rectangle are unseen (labels: Appendix B.2).

4.2 Cross-resolution generalization without fine-tuning

Both the output map and the local correction contribute. In Table 1, replacing the continuous axis functions by resampled free tables costs 10.5 points at 560^2, and factorizing Log-CPB while keeping 16σ loses 1.2 (row b). The $s\tanh$ output map then adds 2.8 (row c) and the central table 0.8 (row d), so TRAPE exceeds Log-CPB by 0.6 points at 224^2, 2.4 at 560^2 and 10.8 at 1024^2. The two components also make a joint 2D bias strong: on DeiT-S, 2D-Tanh, i.e. (a) with both but unfactorized, is within 0.6 points of TRAPE up to 704^2 and on every rectangle, but falls to 23.6% at 1024^2, where TRAPE keeps 70.2% with 9.6× fewer position parameters (Appendix C).

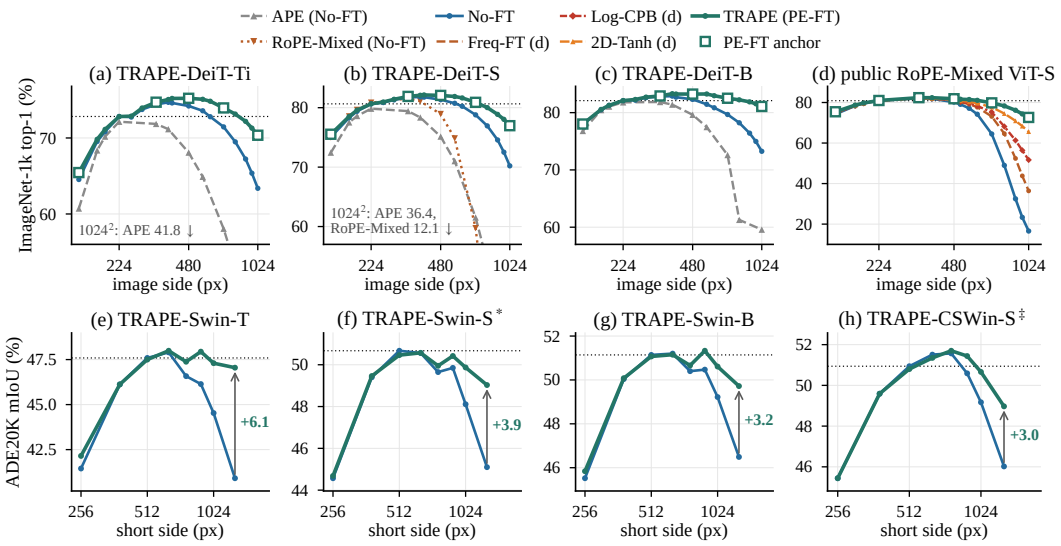


Figure 2: **Cross-resolution generalization before (No-FT) and after position-only fine-tuning (PE-FT).** Dotted lines: No-FT accuracy at the training size. **(a–c)** TRAPE-DeiT on ImageNet-1k, PE-FT at $\{144, 336, 480, 704, 1024\}^2$ (squares; DeiT-S: mean of three seeds); APE and 2D RoPE-Mixed are the No-FT baselines of Table 1. **(d)** Public RoPE-Mixed ViT-S, seed 0 (Appendix E). **(e–h)** ADE20K single-scale mIoU against the test short side (512: training scale); PE-FT fits square crops of $\{256, \ldots, 1344\}$ pixels; labels give the gain at $2.5\times$. * Adapted without the deployment gate; ‡ variant without the centre-slice identity (Appendix J).

Transfer to larger grids and unseen aspect ratios. TRAPE-DeiT-Ti/S retain their 224^2 accuracy through 560^2, and DeiT-B loses 0.6. At 1024^2, a grid $4.6\times$ the pre-training side, TRAPE leads bicubic APE by 21.6, 33.8 and 13.7 points on DeiT-Ti/S/B and 2D RoPE-Mixed by 58.1 on DeiT-S; on the 3:1 rectangle 320×960 it leads APE by 7.2, 10.8 and 4.9 and RoPE-Mixed by 10.6, all with unchanged checkpoints. Training-size accuracy is not predictive: 2D RoPE-Mixed leads TRAPE by 0.2 points at 224^2 but trails by 19.1 at 704^2, and combining both reaches 81.2% at 224^2 yet loses 30.7 points at 1024^2 (Appendix C).

4.3 Position-only adaptation of TRAPE-DeiT

Table 2 fine-tunes only the axis functions and central tables at five anchors, each run starting from its inner neighbour. The whole tape costs 888 steps, 0.108–0.175 ImageNet epoch and 6.7–11.4 minutes of optimization, under 0.06% of the samples of pre-training (Appendix Tables 6 and 7). At the fitted 1024^2 anchor, DeiT-Ti/S/B gain 7.0/6.8/7.8 points; after adaptation they retain their 224^2 accuracy through 704^2 (Figure 2a–c). On five unseen rectangles the mean gains are 1.4/1.1/1.5 points and reach 2.3/1.7/2.3 at 320×960, although adaptation sees only square images. Over three adaptation seeds, the standard deviation of DeiT-S is at most 0.18 point at any shape, far below these gains (Appendix D). Blending two anchors per axis beats either one-sided rule in 39 of 48 non-anchor shape–model pairs and ties in 8; the rule was chosen on a validation subset, so this confirms rather than tests it (Appendix G).

4.4 Position-only adaptation of frozen public ViTs

Table 3 adapts five public ViTs of Heo et al. (2024), pre-trained at 224^2 with DeiT III (Touvron et al., 2022); Freq-FT and APE-FT tune their own frequencies or absolute embedding with the same anchors, steps, batches and selection. Each anchor is a separate 160-step run from the zero branch (Appendix B.4).

Large-resolution recovery with limited positional updates. At 704^2, TRAPE raises the three ViT-S backbones by 15.4 (RoPE-Mixed), 14.7 (RoPE-Axial) and 10.4 points (APE) and ViT-B/L by 6.1 and 2.6, with paired standard errors below 0.25 point in the original runs, while the pre-training resolution changes by -0.1 to $+0.1$. At 1024^2 it lifts RoPE-Mixed ViT-S from 16.6% to 72.6%

Table 3: **Position-only fine-tuning of frozen public ViTs** of Heo et al. (2024) (ImageNet-1k top-1, %). All methods share six anchors $\{144, 224, 336, 480, 704, 1024\}^2$ (†: those shown), 160 steps per anchor, data and held-out selection; Log-CPB and 2D-Tanh are joint 2D branches. At 704^2, RoPE-Mixed S (TRAPE, Log-CPB, 2D-Tanh): mean ± sample s.d. over three adaptation seeds $(0, 1, 2)$; § ViT-B summarizes three runs with global batches 64/128/unrecorded (Table 13). Other entries are single runs. Rect-4: mean over four unseen rectangles; Trained: fitted values at the 1024^2 anchor. All shapes: Appendix E.

Backbone	Method	224^2 †	384^2	560^2	704^2 †	960^2	1024^2 †	Rect-4	Trained
RoPE-Mixed S	No-FT	80.9	81.8	77.1	64.5	23.3	16.6	77.9	—
	Freq-FT	80.9	82.1	79.2	73.1	43.8	36.4	79.3	4.6k
	Log-CPB	80.9	82.0	79.6	75.3 ±0.06	56.4	51.7	79.5	55.3k
	2D-Tanh	**81.0**	82.0	80.4	77.8 ±0.12	68.3	65.6	80.0	67.5k
	TRAPE	**81.0**	**82.4**	**81.5**	**79.9** ±0.05	**74.4**	**72.6**	**81.0**	27.4k
RoPE-Axial S	No-FT	**80.9**	80.0	73.4	61.7	27.4	20.6	75.0	—
	TRAPE	80.8	**81.0**	**78.8**	**76.4**	**67.8**	**63.8**	**78.8**	27.4k
APE S	No-FT	80.5	79.4	73.6	67.6	54.1	50.9	74.6	—
	APE-FT	80.5	79.4	73.6	67.7	54.7	51.6	74.6	75.3k
	TRAPE	80.5	**81.6**	**79.6**	**78.0**	**73.0**	**72.1**	**79.7**	27.4k
RoPE-Mixed B	No-FT	83.8	84.4	81.9	76.8	50.8	41.7	82.2	—
	Freq-FT	83.8	84.4	82.5	79.7 ±0.17§	67.0	62.8	82.5	9.2k
	TRAPE	83.8	**84.5**	**83.7**	**82.9** ±0.17§	**77.6**	**75.4**	**83.3**	54.7k
RoPE-Mixed L	No-FT	84.8	85.6	84.2	81.4	67.4	61.3	84.2	—
	Freq-FT	84.8	85.6	84.3	82.4	74.1	70.7	84.3	24.6k
	TRAPE	84.8	85.6	**84.9**	**84.0**	**80.4**	**78.9**	**84.6**	145.9k

Table 4: **ADE20K semantic segmentation** (UPerNet, 512^2 crops, 160k iterations, official configurations of the baselines; no position-only fine-tuning). IN-1k: ImageNet-1k top-1 at 224^2; SS/MS: single-/multi-scale mIoU (%); FLOPs of the full model at 512×2048; FPS at 512^2, batch 1, fp32, one A800.

Backbone	IN-1k (%)	mIoU SS	mIoU MS	#P (M)	FLOPs (G)	FPS
Swin-T (Liu et al., 2021)	81.3	44.5	45.8	60	945	42.9
TRAPE-Swin-T (ours)	**81.5**	**47.6** +3.1	**48.5** +2.7	60	**937**	**44.7**
Swin-S (Liu et al., 2021)	**83.0**	47.6	49.5	81	1038	33.9
TRAPE-Swin-S (ours)	82.7	**50.7** +3.1	**51.7** +2.2	81	**1024**	**35.7**
Swin-B (Liu et al., 2021)	**83.5**	48.1	49.7	121	1188	28.3
TRAPE-Swin-B (ours)	83.2	**51.1** +3.0	**52.1** +2.4	121	**1166**	**30.0**
CSWin-S (Dong et al., 2022)	**83.6**	50.4	51.5	65	1027	28.1
TRAPE-CSWin-S (ours)	83.1	**50.9** +0.5	**52.1** +0.6	65	**976**	**30.7**

(Figure 2d), RoPE-Axial ViT-S from 20.6% to 63.8%, APE ViT-S from 50.9% to 72.1%, ViT-B from 41.7% to 75.4% and ViT-L from 61.3% to 78.9%. The branch updates less than 0.13% of the parameters, and a six-anchor tape costs 0.048–0.096 epoch (Appendix Table 6). Refitting the 704^2 anchor with other seeds moves RoPE-Mixed ViT-S by an s.d. of 0.05 point, and two further ViT-B runs keep the lead over Freq-FT at 3.2 points (Appendix E.1).

Rectangles and native-parameter controls. In the order of Table 3, the mean gains on Rect-4 are 3.1/3.8/5.1/1.1/0.4 points, and 9.7/9.4/12.5/3.7/1.8 at 320×960. At 1024^2, Freq-FT reaches only 36.4%, 62.8% and 70.7% on ViT-S/B/L, APE-FT gains at most 0.7 point, and TRAPE leads both on all 20 rectangles (Appendix E).

Joint 2D branches under a matched data budget. Log-CPB and 2D-Tanh replace the TRAPE branch by a non-factorized one with the same anchors, steps and data, and train 55.3k and 67.5k values per anchor against 18.7k–27.4k for TRAPE at 704^2–1024^2. Log-CPB trails TRAPE at every size, by 4.6 points at 704^2, 20.9 at 1024^2 and 1.5 on Rect-4. The two components of TRAPE lift the 2D branch by up to 13.9 points (2D-Tanh), yet it still trails TRAPE by 2.1 at 704^2, 7.0 at 1024^2 and 1.0 on Rect-4. As the optimizer parameterizations differ (Appendix B.5), this supports the adaptation

Table 5: **Inference cost of table-only deployment** (TRAPE-DeiT-S, one A800 GPU, bf16; mean±s.d. over three runs, each the median of 100 (latency) or 30 (throughput) timed passes). MLP: axis functions run in every forward pass; tape: tables served once per shape; dense: explicit softmax over the gathered $N \times N$ bias; fused: Triton kernel reading the compact table; * training implementation. Throughput batch 64 at 224^2 and 3 at 1024^2; Peak: memory at that batch; Max batch: within 80 GB. All models and shapes: Appendix K.

		Latency, batch 1 (ms)		**Throughput** (img/s)		**Memory** at 1024^2	
Bias	**Attention**	224^2	1024^2	224^2	1024^2	Peak (GiB)	Max batch
MLP	dense*	8.86 ±0.15	63.68 ±0.00	4679 ±7	24.0 ±0.0	1.74	192
tape	dense	5.81 ±0.08	62.93 ±0.00	4881 ±10	24.2 ±0.0	1.74	—
MLP	fused	9.34 ±0.08	9.41 ±0.15	6773 ±64	191.5 ±0.2	0.17	—
tape	**fused**	5.48 ±0.06	5.86 ±0.00	8681 ±7	202.3 ±0.1	0.18	2,048

recipe rather than an inability of 2D biases to fit; both share one anchor pair across the axes, and their 2D table banks are 37× larger (Appendix F).

4.5 Semantic segmentation

Hierarchical backbones. With the official configurations of the baselines, TRAPE-Swin improves segmentation at all three sizes (Table 4): $+3.1/+3.1/+3.0$ single-scale and $+2.7/+2.2/+2.4$ multi-scale mIoU over Swin-T/S/B, with fewer FLOPs, 1.04–1.06× the measured throughput and comparable ImageNet accuracy ($+0.2/-0.3/-0.3$). Single-component ablations (Table 21) show that attention and bias both contribute: without its depthwise value branch, TRAPE-Swin-T still leads Swin-T by 2.6 mIoU at 0.4 lower ImageNet accuracy, and the same branch adds only 1.3 to Swin-T; a Swin V2-style $16\sigma(f_a(c))$ bias keeps ImageNet accuracy but loses 1.0 mIoU, and 2D RoPE-Mixed in place of the bias loses 0.6. The CSWin-derived variant, trained with the official CSWin configurations, gains 0.5/0.6 with 51 fewer GFLOPs (Appendix I).

Position-only adaptation at larger test scales. The four checkpoints of Table 4 degrade when the test scale exceeds their 512^2 training crops (Figure 2e–h). We refit only the axis-function output layers and central tables with the segmentation loss at five square crop anchors: 19,200 crops, 0.75% of the 160k schedule, 13.4–18.2 minutes and at most 0.09% of the parameters (Appendix Table 6). At 2.5× the training short side, mIoU improves by 3.0–6.1 points, and by 4.4–8.8 when the long side is resized to 2048 pixels; at the standard scale, single-scale mIoU changes by -0.2 to -0.1 and multi-scale mIoU by -0.3 to $+0.2$ (Appendix J).

4.6 Inference cost of table-only deployment

Table 5 times four execution paths of the same TRAPE-DeiT-S weights. Serving the 2D tables of all 12 blocks for a new shape takes 0.06–0.13 ms, whereas the axis functions cost about 3 ms in every forward pass; skipping them lowers batch-1 latency by 1.6–1.8× with the fused kernel and raises its throughput by 6–28%. The fused kernel never forms the $N \times N$ bias: at 1024^2 the deployed path processes 8.4× as many images per second as the training implementation, peaks at 0.18 instead of 1.74 GiB, and fits 10.7× larger batches (11.3× on DeiT-B). Every timing varies by at most 2.8% (s.d./mean) across the three runs, and the served tables reproduce the logits of the training implementation to 4.5×10^{-6} in fp32 (Appendix K).

5 Conclusion

TRAPE separates an adaptable positional function from a compact inference representation: fixed-reference coordinates make centre slicing exact, and a resolution tape serves the biases of all blocks by independent height and width blending, read by a fused kernel without an $N \times N$ bias. The encoding transfers to larger grids, adapts frozen backbones on small budgets, and improves dense prediction. *Limitations:* the additive pre-activation cannot represent arbitrary 2D offset patterns, inputs beyond the stored range need an offline export, and our timings compare execution paths of TRAPE, not encodings without an attention bias.

AI USE STATEMENT

We used generative AI tools to assist with translating author-written text, with editing the text for clarity and organization, and with formatting tables and figures from the authors' experimental records. The authors conducted all experiments, reviewed and verified all AI-assisted content, including every reported number against the experimental records, and take full responsibility for the final content of the paper, including its claims and reported results.

ETHICS STATEMENT

This work studies a component of general-purpose vision backbones and collects no new data. ImageNet-1k and ADE20K are public benchmarks used under their terms, and the public checkpoints are released under the Apache-2.0 license; our models inherit any biases of these datasets. Improved resolution robustness can benefit dense prediction applications and can also be applied to surveillance. These application-level effects are not evaluated here.

REPRODUCIBILITY STATEMENT

Section 3 defines the encoding, anchor plan, export and serving rule. Appendix A gives the centre-slice proof, the deployment algorithm, the two assembly paths, the fused Triton kernel with FlexAttention sketches, and the numerical checks, distinguishing trainable checkpoints from inference buffers. Appendix B specifies preprocessing, optimization, selection and budgets for classification, and Appendix J for segmentation, including every evaluation protocol; Appendix K gives the hardware, software, timing protocol and three independent repetitions of the inference benchmark. Appendices C–I report every measured shape, the repeated runs of the DeiT-S adaptation and of the 704^2 anchor of public ViTs, the ablations and checkpoint-derived visualizations of the learned biases; every value is given with its seed and anchor set. All TRAPE models use the public DeiT, Swin, CSWin or MMSegmentation training code and configurations with modified model definitions. Training, adaptation, evaluation, benchmarking and visualization code, the Triton kernel, checkpoints and deployment tables will be released with the models. Pre-training is not bit-exact because the reference scripts enable `cudnn.benchmark`; exported tables and routing are deterministic once fitted anchors are fixed.

REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. FlexiViT: One model for all patch sizes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. Conditional positional encodings for vision transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Ekin Dogus Cubuk, Barret Zoph, Jon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver,

Piotr Padlewski, Alexey Gritsenko, Mario Lucic, and Neil Houlsby. Patch n' Pack: NaViT, a vision transformer for any aspect ratio and resolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. RepVGG: Making VGG-style ConvNets great again. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. FlexAttention: A programming model for generating fused attention variants. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025.

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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

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 *International Conference on Learning Representations (ICLR)*, 2021.

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *European Conference on Computer Vision (ECCV)*, 2024.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. CCNet: Criss-cross attention for semantic segmentation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European Conference on Computer Vision (ECCV)*, 2022a.

Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. MViTv2: Improved multiscale vision transformers for classification and detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022b.

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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Microsoft. Swin Transformer V2: Official implementation. https://github.com/microsoft/Swin-Transformer/blob/main/models/swin_transformer_v2.py, 2022. WindowAttention: continuous position bias and output scaling.

MMSegmentation Contributors. MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations (ICLR)*, 2022.

Markus N. Rabe and Charles Staats. Self-attention does not need $O(n^2)$ memory. *arXiv preprint arXiv:2112.05682*, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Rui Tian, Zuxuan Wu, Qi Dai, Han Hu, Yu Qiao, and Yu-Gang Jiang. ResFormer: Scaling ViTs with multi-resolution training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19, 2019.

Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. DeiT III: Revenge of the ViT. In *European Conference on Computer Vision (ECCV)*, 2022.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *European Conference on Computer Vision (ECCV)*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation. In *European Conference on Computer Vision (ECCV)*, 2020.

Ross Wightman. PyTorch image models. https://github.com/huggingface/pytorch-image-models, 2019.

Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision (ECCV)*, 2018.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Overview of the appendices. Appendix A proves the centre-slice identity and details table-only deployment, the two assembly paths, the fused Triton kernel with FlexAttention sketches, and our numerical checks. Appendix B gives every training, adaptation and evaluation protocol for classification, the adaptation budgets, and all method labels. Appendices C–F list the measurements behind the main tables: cross-resolution generalization without fine-tuning for the 14 reported encodings over 16 shapes (C; Tables 1 and 8), all shapes after position-only fine-tuning of TRAPE-DeiT with the three-seed repetition for DeiT-S (D; Table 2), the grafted public ViTs at 20 shapes from 144^2 to 1024^2 with repeated runs and budgets (E; Table 3), and the two-dimensional branches with their storage (F). Appendix G ablates the tape capacity and the routing rule, Appendix H analyses and visualizes the learned biases, Appendix I describes the axial backbones and their single-component ablations, Appendix J gives the protocols, budgets and every measured scale of position-only fine-tuning on ADE20K, and Appendix K the inference-cost measurements of Section 4.6.

A Centre Slicing, Deployment and Kernel Integration

A.1 Centre slicing preserves the offset function

Proposition 1 (Centre-slice identity)**.** *Fix the position parameters of one anchor, the reference side $G_0 > 1$, and a central table E supported on $|\Delta| \le (K-1)/2$. Let $U^{(S)}$ be the raw axis table over the offsets $-(S-1), \dots, S-1$. Then $\mathcal{C}_g\big(U^{(S)}\big) = U^{(g)}$ for every $1 \le g \le S$, in exact arithmetic.*

Proof. Entry k of $U^{(S)}$ belongs to the offset $\Delta = k - (S-1)$. Entry j of the slice that starts at index $S - g$ therefore belongs to $\Delta = (S-g+j) - (S-1) = j - (g-1)$, which is the offset of entry j of $U^{(g)}$. Equation 2 depends only on Δ and the fixed G_0, and E is indexed by the same Δ, so both terms of Eq. 3 agree entrywise. The argument holds for every block and head, and for averages of raw tables. □

Storing each anchor up to the next one makes the identity usable between anchors. For $g_i < g < g_{i+1}$, the lower anchor is stored to $S_i = g_{i+1} > g$ and the upper one to $S_{i+1} \ge g_{i+1} > g$, so both slices exist; the last anchor needs no larger neighbour. Because $\mathcal{C}_g$ only selects entries, it is linear, so blending and slicing commute, $\mathcal{C}_g\big((1-\alpha)T + \alpha T'\big) = (1-\alpha)\mathcal{C}_g(T) + \alpha\mathcal{C}_g(T')$ for tables T, T' of equal length, and Eq. 6 returns the stored table exactly at $\alpha \in \{0, 1\}$. The identity concerns one fixed parameter state; it does not state that differently fine-tuned anchors agree, and blending between them is an empirical rule. It also requires the fixed reference: a coordinate normalized by the current side, $\Delta/(g-1)$, assigns different coordinates to the same offset at different sides, and its tables are not sub-arrays of one another. The same restriction property holds for a two-dimensional parameterization with a fixed coordinate reference. Factorization adds two things: the stored object has $O(g)$ instead of $O(g^2)$ entries per block and head, and the two axes can be read from different anchors (Appendix F).

A.2 Bounds, and mixing before the nonlinearity

For any s, Eq. 4 gives $|b| \le |s|$. If two raw axis tables $\widehat{U}_h, \widehat{U}_w$ differ elementwise from U_h, U_w by at most ϵ_h and ϵ_w, the biases they produce satisfy $|\widehat{b} - b| \le |s|(\epsilon_h + \epsilon_w)$ because $\tanh$ is 1-Lipschitz. This bounds the bias, not the accuracy. Mixing squashed biases would be a different operation, since $\tanh((1-\alpha)u + \alpha v) \neq (1-\alpha)\tanh u + \alpha \tanh v$ in general; the tape therefore stores the raw tables and applies the nonlinearity after mixing and composition.

A.3 Table-only deployment

Input: grid (H, W); tape $\{\mathcal{T}_{a,i} \in \mathbb{R}^{L \times n_h \times (2S_i - 1)}\}$; amplitude s; routing arrays $\mathrm{lo}[g], \alpha[g]$ for every supported side g.

1. For each axis $a \in \{h, w\}$ with side $g = H$ or $g = W$: if g lies outside the declared range, reject the input; extending the tape is an explicit offline export from the retained axis functions, not a silent fallback. Otherwise read $i = \mathrm{lo}[g]$ and $\alpha = \alpha[g]$ (Eq. 6). At an exact anchor, return that anchor's centre slice directly and skip the blend in step 2; this also handles the two range endpoints.
2. Otherwise serve $\widehat{U}_a = (1 - \alpha)\,\mathcal{C}_g(\mathcal{T}_{a,i}) + \alpha\,\mathcal{C}_g(\mathcal{T}_{a,i+1})$ for all L blocks and n_h heads in one batched operation.
3. Compose the compact 2D offset tables of all blocks, $T_{2D} = s\tanh(\widehat{U}_h \oplus \widehat{U}_w)$ (path A). These are temporary shape-specific caches, not expanded token-pair matrices.
4. For small grids, gather each block's expanded bias and add it to the scaled logits before the softmax; for large grids, a fused kernel can read the compact table directly (Listings 1–3). Special-token terms are zero in either backend.

Figure 3: **Table-only deployment algorithm.** Height and width select their anchors independently, and no position network is evaluated inside the declared range. Constructing the compact tables precedes, and is independent of, the choice of attention backend.

The routing arrays hold one anchor index and one weight per supported side. Routing and compact-table composition batch all blocks and heads once for a fixed shape and positional state; expanding every block's token-pair bias at once is a separate, much larger allocation and is optional even for dense attention.

A.4 Assembly paths and FlexAttention

There are two separate decisions: the order in which the positional bias is constructed, and how attention consumes it. Figure 1(b) shows the table-first construction (path A) with its dense and fused consumers. The alternative (path B) indexes the raw axes *before* adding and squashing; it does not mean constructing blocks one at a time, and both paths can batch every block or process blocks separately when memory is limited. Throughout, offsets are key-minus-query as in Eq. 1, tokens are numbered row by row, so a spatial index t has row $\lfloor t/W \rfloor$ and column $t \bmod W$, and the first n_s tokens of the sequence are special tokens ($\mathtt{ns} = 1$ for the class token) with zero bias.

A: compose compact tables, then index. For fixed H, W the served raw arrays have shapes $(L, n_h, 2H-1)$ and $(L, n_h, 2W-1)$. Path A composes $[T_{2D}]_{\ell jik} = s\tanh\big([\widehat{U}_h]_{\ell ji} + [\widehat{U}_w]_{\ell jk}\big)$ for block ℓ, head j, $0 \le i < 2H-1$ and $0 \le k < 2W-1$, and dense lookup gives $[B_\ell]_{jpq} = [T_{2D}]_{\ell j,\, h_q - h_p + H - 1,\, w_q - w_p + W - 1}$ for spatial tokens p (query) and q (key). A fused consumer performs this lookup on demand instead of allocating B_ℓ.

B: index each raw axis, then add and squash. Path B forms the raw pairwise responses $[R_h]_{\ell juv} = [\widehat{U}_h]_{\ell j,\, v-u+H-1}$ for query row u and key row v, and R_w likewise over columns, broadcasts them over the missing coordinate and computes $[B_\ell]_{jpq} = s\tanh\big([R_h]_{\ell j h_p h_q} + [R_w]_{\ell j w_p w_q}\big)$. Substitution recovers path A exactly in real arithmetic; B has no intermediate compact table, and its two indexed matrices are added *before* the single nonlinearity.

Listing 1: Dense attention on small grids: path A (table first) and path B (axes first) give the same bias.

```
import math, torch
import torch.nn.functional as F
# Th, Tw: served raw tables of all blocks, (L, heads, 2H-1), (L, heads, 2W-1), E merged.
T2d = s * torch.tanh(Th[..., :, None] + Tw[..., None, :])   # path A, once per shape

def dense_from_table(T, H, W, ns=1):              # T: (..., heads, 2H-1, 2W-1)
    t = torch.arange(H * W, device=T.device)
    r, c = t // W, t % W                          # row-major token coordinates
    dh = r[None, :] - r[:, None] + H - 1          # [query, key] -> Delta_h + H - 1
    dw = c[None, :] - c[:, None] + W - 1          # [query, key] -> Delta_w + W - 1
    return F.pad(T[..., dh, dw], (ns, 0, ns, 0))  # zero bias for special tokens

def dense_from_axes(Uh, Uw, s, H, W, ns=1):       # path B, no compact table
    h, w = torch.arange(H, device=Uh.device), torch.arange(W, device=Uw.device)
    Rh = Uh[..., h[None, :] - h[:, None] + H - 1]              # (..., heads, H, H)
    Rw = Uw[..., w[None, :] - w[:, None] + W - 1]              # (..., heads, W, W)
    raw = Rh[..., :, None, :, None] + Rw[..., None, :, None, :]  # (..., H, W, H, W)
    B = s * torch.tanh(raw).flatten(-4, -3).flatten(-2, -1)     # (..., heads, HW, HW)
    return F.pad(B, (ns, 0, ns, 0))

B = dense_from_table(T2d[ell], H, W, ns)              # path A
# B = dense_from_axes(Th[ell], Tw[ell], s, H, W, ns)  # path B: same values
# q, k, v of block ell: (batch, heads, ns + H*W, d)
logits = q @ k.transpose(-2, -1) / math.sqrt(q.shape[-1]) + B
out = torch.softmax(logits, dim=-1) @ v
```

Memory. Path A's compact cache has $Ln_h(2H-1)(2W-1)$ entries; B's raw pair matrices have $Ln_h(H^2+W^2)$, but its dense bias has n_hN^2 per block, as does dense A. At $H=W=64$, the compact fp32 tables of all 12 ViT-B blocks occupy 8.9 MiB, whereas one block's spatial B_ℓ occupies 768 MiB; a flattened dense lookup index adds 64 MiB in int32, while B's axis indices need only H^2+W^2 entries. Path A evaluates $\tanh$ on unique offset pairs, dense B on token pairs. No universal small/large-grid cutoff is implied: the cache and the backend are chosen to fit the available memory, and neither changes the positional function.

The fused Triton kernel. The timed fused path of Section 4.6 implements path A in Triton (Tillet et al., 2019) with the tiled online softmax of FlashAttention-2 (Dao, 2024): the bias of a query–key pair is added where its logit is formed, and no $N \times N$ tensor exists. Relative biases are Toeplitz in the token index, which makes the lookup cheap. With the per-token code $c_t = \lfloor t/W \rfloor (2W-1) + (t \bmod W)$ of spatial token t and $M = (H-1)(2W-1) + W - 1$, the entry of query p and key q is $\mathrm{flat}(T_{2D,\ell})[(c_q+M) - c_p]$, one subtraction of two integers loaded once per tile. The codes are built once per shape; special tokens and padded positions receive codes that land in a zero-padded extension of the flat table, so their bias is zero without masks, and $\log_2 e$ is folded into the stored table. A variant that computes the index from row and column arithmetic was also implemented; both pass fp64 self-tests against explicit attention, and the code lookup, 1.38 times faster per layer on the benchmark GPU, is used for all timings. The kernel is used for inference only; training uses the differentiable path of Eqs. 3–4.

Sketches. Listing 1 gives both dense constructions for small grids, Listing 2 the fused lookup of the compact table through FlexAttention, a programmable alternative to a hand-written kernel, and Listing 3 a fused variant of path B. FlexAttention (Dong et al., 2025) evaluates a user-defined `score_mod(score, b, h, q_idx, kv_idx)` inside the kernel, where `score` is the already scaled dot product, so the bias of a query–key pair is added where its logit is formed and the $N \times N$ bias is never stored; the callback therefore adds the bias without scaling it again. Special-token indices are clamped to a valid spatial index before the lookup, because `torch.where` evaluates both branches, and their result is discarded. The sketches assume a complete row-major grid with no sequence padding; other restrictions need a separate mask.

What was checked. For every spatial pair the three sketches add the same value as the definition of Eq. 1, and zero for a pair that involves a special token; floating-point execution need not be bitwise identical across backends. We checked the index logic of these sketches against a direct mathematical reference on random tables, for square and rectangular grids with zero, one and two special tokens, and checked FlexAttention with the two score modifications against explicit softmax attention; the

Listing 2: Large grids: the compact table of path A read inside FlexAttention; no $N \times N$ bias or index is formed.

```
from torch.nn.attention.flex_attention import flex_attention
flex_attention = torch.compile(flex_attention)

def table_score_mod(T, H, W, ns):                # T: (heads, 2H-1, 2W-1), one block
    def score_mod(score, b, h, q_idx, kv_idx):
        qi, ki = q_idx - ns, kv_idx - ns                  # spatial indices; < 0: special
        spatial = (qi >= 0) & (ki >= 0)
        qi, ki = qi.clamp(min=0), ki.clamp(min=0)         # keep the lookups in range
        dh = ki // W - qi // W + (H - 1)                  # Delta_h + H - 1
        dw = ki % W - qi % W + (W - 1)                    # Delta_w + W - 1
        return torch.where(spatial, score + T[h, dh, dw], score)
    return score_mod

out = flex_attention(q, k, v, score_mod=table_score_mod(T2d[ell], H, W, ns=1))
```

Listing 3: A fused variant of path B: each block reads its two 1D tables inside FlexAttention.

```
def axis_score_mod(Uh, Uw, s, H, W, ns):          # Uh: (heads, 2H-1), Uw: (heads, 2W-1)
    def score_mod(score, b, h, q_idx, kv_idx):
        qi, ki = q_idx - ns, kv_idx - ns
        spatial = (qi >= 0) & (ki >= 0)
        qi, ki = qi.clamp(min=0), ki.clamp(min=0)
        dh = ki // W - qi // W + (H - 1)
        dw = ki % W - qi % W + (W - 1)
        bias = s * torch.tanh(Uh[h, dh] + Uw[h, dw])
        return torch.where(spatial, score + bias, score)
    return score_mod

out = flex_attention(q, k, v, score_mod=axis_score_mod(Th[ell], Tw[ell], s, H, W, ns=1))
```

FlexAttention sketches were not timed. The reported accuracies use exact query-chunked attention (Rabe & Staats, 2021) with chunks of 128 queries. During position-only fine-tuning, the tables must instead be regenerated from the current parameters at every step through Eqs. 3–4, without detaching, and the attention backend must propagate gradients to the captured tensors; loading detached inference tables would cut the gradient path to f_h, f_w, E_h, E_w. Caches are invalidated when the shape, the position parameters, the routing, the device or the dtype changes.

Persistent parameter state versus exported buffers. For a uniform-depth ViT, the tables of anchor i are stored as arrays of shape $(L, n_h, 2S_i{-}1)$ for each axis; the first dimension counts blocks, not a second head dimension. The served arrays have shapes $(L, n_h, 2H{-}1)$ and $(L, n_h, 2W{-}1)$, and centre slicing and blending act on the last dimension while batching the first two; hierarchical models store one array per stage. The training archive retains the MLP weights and biases, the central tables, s, G_0, the capacity metadata and the frozen hinge bases where applicable, which are needed to continue optimization. The deployable file contains only the detached raw tables, s and the routing metadata; optimizing those buffers directly would define a different method, and an averaged exported table does not determine a unique checkpoint, so the adaptation code keeps the two states separate. The fixed-reference Swin adaptation code supports the same strict table-only export; the CSWin variant of Appendix I does not. Exact optimizer-step resumption additionally requires the optimizer and scheduler state; a saved position checkpoint suffices to start a further fine-tuning run.

A.5 Numerical checks

The experimental records report the following data-free checks: centre slicing against direct table construction for all anchors and all sliceable lengths; bitwise recovery of anchor tables; equality of the served table with a hand-computed slice-and-mix; independent routing of the two axes for $H \neq W$; identical outputs when the deployment file keeps only the tables; monotone, normalized routing weights; unchanged outputs when the axis functions and central tables are overwritten with `NaN`, which shows that no position network runs inside the stored range; agreement of the precomputed routing with on-the-fly bracketing; and mix $= (1{-}\alpha)\,\mathrm{lo} + \alpha\,\mathrm{hi}$. With trained weights, the tested anchors and rectangles give zero logit difference, and the float32 export path reproduces the tables to a relative error of at most 2×10^{-7}, a few float32 ULP. For the grafted models, switching the zero-initialized branch on or off leaves the logits bit-identical at 224^2, 384^2, 512^2, 1024^2 and $320{\times}960$; the table and axis paths agree exactly, also for non-zero tables; and the branch receives non-zero

gradients at the first step. For TRAPE-DeiT, the 224^2 tape differs from direct evaluation by at most 0.01 point, an execution-path difference between fp32 export and mixed-precision evaluation. On the inference benchmark of Appendix K, served tables match direct evaluation of the axis functions to 7.5×10^{-9}, and all execution paths match the logits of the training implementation to 4.5×10^{-6} in fp32.

B Training and Evaluation Protocols

B.1 Pre-training, segmentation and evaluation

Plain models use the official DeiT script (Touvron et al., 2021) with the model definition as the only changed file: 300 epochs, AdamW (Loshchilov & Hutter, 2019), cosine schedule (Loshchilov & Hutter, 2017), global batch 1024, base learning rate 5×10^{-4}, weight decay 0.05, five warm-up epochs, label smoothing 0.1 (Szegedy et al., 2016), RandAugment (Cubuk et al., 2020), mixup 0.8 (Zhang et al., 2018), CutMix 1.0 (Yun et al., 2019), random erasing (Zhong et al., 2020), and a class token without absolute embedding. The amplitude s is excluded from weight decay, and the DeiT-B run uses gradient clipping. TRAPE-Swin and TRAPE-CSWin keep the architecture configuration of the official Swin-T/S/B and CSWin-S models (stem, stage widths, depths, heads, MLP ratios and patch merging) and change only the attention and position modules of Appendix I. They are trained with the official Swin and CSWin code bases and configurations (optimizer, schedule, augmentation, stochastic depth and epochs) for ImageNet-1k pre-training, and with the official UPerNet (Xiao et al., 2018) segmentation configurations of the same code bases in MMSegmentation (MMSegmentation Contributors, 2020) (for Swin: 512×512 crops, 160k iterations, AdamW, learning rate 6×10^{-5}, poly schedule, batch 16), so every hierarchical model is trained exactly like the baseline it is compared with. In the segmentation models, the log coordinate of each stage is referenced to the grid of the 512^2 training crop (128, 64, 32 and 16 tokens, i.e. denominators 127, 63, 31 and 15). Swin and CSWin baselines are the published results obtained with these recipes.

The DeiT-S 2D RoPE-Mixed control follows Heo et al. (2024) (per-block learnable mixed frequencies with axial initialization, unrotated class token) inside the same DeiT skeleton, script and recipe; at other resolutions its rotary positions are simply extended. The TRAPE + RoPE-Mixed model keeps the TRAPE bias of every block unchanged and additionally rotates the patch-token queries and keys with these learnable mixed frequencies. The per-block variant replaces the shared s by one learnable amplitude per block, each initialized to 8 and excluded from weight decay, and changes nothing else. The axial free-table control keeps Eq. 4 and replaces $f_a(c(\Delta; G_0))$ by a free vector of length $2G_0-1$, resized bilinearly to the target length at test time. All code uses PyTorch (Paszke et al., 2019) and `timm` (Wightman, 2019); training and adaptation run on 8 A800 GPUs, and the inference benchmark of Appendix K on one.

Our own classification checkpoints are evaluated by one multi-resolution harness on all 50,000 validation images: sides up to 224 use a centre crop with crop ratio 0.875, larger inputs are resized to the target height and width. The grafted public models use the protocol of Appendix B.4 (aspect-preserving resize and centre crop). Comparisons are always made within one protocol.

B.2 Method labels used in the tables

No-FT is an unchanged checkpoint evaluated at another shape, and *PE-FT* denotes position-only fine-tuning. *TRAPE* in an adapted-model table is the position-only fine-tuned TRAPE branch, served from its exported tables. *Freq-FT* and *APE-FT* fine-tune only the backbone's own rotary frequencies or its absolute embedding, under the same anchors, steps, batches and selection. *Log-CPB* is the Swin V2 branch of Appendix B.5 and *2D-Tanh* (*2D-T* in narrow tables) the non-factorized control, a joint 2D MLP with TRAPE's shared $s\tanh$ and a 13×13 central correction, evaluated here as a grafted adaptation branch. *T,* s_ℓ is TRAPE with one amplitude per block, and *T+RoPE* is TRAPE combined with 2D RoPE-Mixed. A suffix *+Freq* or *+APE* means the bias branch and the named backbone parameters are tuned jointly, and *Joint-off* evaluates such a run with its added bias branch disabled. *Rect-4* is the mean over 384×512, 512×384, 448×672 and 672×448. A † marks a fitted anchor.

Table 6: **Cost of position-only fine-tuning** per model and run, summed over its anchors (ranges over models; eight A800 GPUs), with reference training budgets. Samples: training presentations incl. repeats, without selection or evaluation; Epochs: samples over the training set, not FLOPs; Time: [a]training only, [b]training and selection; Trained: position parameters at the largest anchor; (5): the TRAPE branches of Table 3; (4): TRAPE-Swin-T/S/B and TRAPE-CSWin-S. Details: Tables 7, 14 and 24.

Adapted models	**Data**	**Steps**	**Samples**	**Epochs**	**Time**	**Trained**
TRAPE-DeiT-Ti/S/B	IN-1k	888	138k–224k	0.108–0.175	6.7–11.4 min[a]	19.8k–60.8k
Public ViT-S/B/L (5)	IN-1k	960	61k–123k	0.048–0.096	16.8–49.3 min[a]	27.4k–145.9k
TRAPE-Swin/TRAPE-CSWin (4)	ADE20K	1,200	19.2k	0.95	13.4–18.2 min[b]	20.5k–96.0k
Ref.: DeiT training, batch 1024	IN-1k	375k	384M	300	—	all
Ref.: UPerNet, batch 16	ADE20K	160k	2.56M	127	—	all

Table 7: **Budget of position-only fine-tuning for TRAPE-DeiT** (capacity policy `both` (grow MLP width and central support)). Each anchor updates only the two axis MLPs and the two central tables; the batch is global over 8 GPUs. Presentations count processed training images including repeats; epochs are relative to the 1,281,167 ImageNet-1k training images. Anchors are listed in training order; g: token side (image side in parentheses), d_c: MLP width, K: central support, Trained: updated scalar parameters; ep.: epochs. 224^2 is not fine-tuned.

Scale	g (image)	d_c	K	**Steps**	**Batch**	**Trained**	**Presentations**
Tiny	21 (336)	41	19	166	448	6,288	74,368
Tiny	9 (144)	27	13	169	512	4,176	86,528
Tiny	30 (480)	59	29	165	216	9,168	35,640
Tiny	44 (704)	87	43	182	96	13,536	17,472
Tiny	64 (1024)	127	63	206	48	19,776	9,888
Tiny, total	—	—	—	**888**	—	—	**223,896** (0.175 ep.)
Small	21 (336)	41	19	166	336	10,608	55,776
Small	9 (144)	27	13	169	512	7,056	86,528
Small	30 (480)	59	29	165	160	15,504	26,400
Small	44 (704)	87	43	182	72	22,896	13,104
Small	64 (1024)	127	63	206	32	33,456	6,592
Small, total	—	—	—	**888**	—	—	**188,400** (0.147 ep.)
Base	21 (336)	41	19	166	168	19,248	27,888
Base	9 (144)	27	13	169	512	12,816	86,528
Base	30 (480)	59	29	165	80	28,176	13,200
Base	44 (704)	87	43	182	32	41,616	5,824
Base	64 (1024)	127	63	206	24	60,816	4,944
Base, total	—	—	—	**888**	—	—	**138,384** (0.108 ep.)

B.3 Position-only fine-tuning of TRAPE-DeiT

The optimizer updates all axis-MLP and central-table parameters; s is frozen. We use AdamW without weight decay, label smoothing 0.1, gradient clipping at 1 and mixed precision. The learning rate warms up over about 6% of the steps, stays constant, and decays with a cosine over the last 35% to 2% of its peak. The anchors are trained from the inside out, $21 \to 9$ and $21 \to 30 \to 44 \to 64$; each run starts from the parameters of its inner neighbour, and its step budget scales with the coordinate gap. The exported tables are averages of the raw tables sampled every third step over the last 25% of the run; averaging tables rather than parameters is exact for what is stored, and the centre-slice identity holds for the average. Growing the capacity preserves the function: new MLP units have zero outgoing weights and new table entries are zero. For $g > 14$, MLP growth uses $d_c = 2g-1$ and table growth uses the largest odd integer below g. Training is monitored on five images per class held out from the training split. The complete five-anchor adaptation takes 6.7, 8.1 and 11.4 minutes of optimization on 8 A800 GPUs for DeiT-Ti/S/B (Table 7). Batches shrink at large anchors because activation memory grows, so the largest anchor uses the fewest image presentations. Epochs in Tables 6 and 7 count image presentations, not equivalent FLOPs: high-resolution presentations cost more.

B.4 GRAFTED BRANCHES ON PUBLIC CHECKPOINTS

Models. We load the released ViT-S/B/L RoPE-Mixed, ViT-S RoPE-Axial and ViT-S absolute-position checkpoints of Heo et al. (2024) (Apache-2.0 license), trained at 224^2 with the DeiT III recipe (Touvron et al., 2022). Our code re-implements their forward pass with the parameter names of the release, so all backbone tensors load strictly, and adds the branch of §3.4 to every block (97 new tensors for ViT-S, all zero at the output). The *No-FT* columns are these checkpoints evaluated with our code and protocol.

Optimization. Every tape and every bank uses the six anchors $g \in \{9, 14, 21, 30, 44, 64\}$, i.e. $\{144, 224, 336, 480, 704, 1024\}^2$. Every anchor starts from the zero branch and is trained for 160 steps with global batch 128 (ViT-S models) or 64 (ViT-B/L), using gradient accumulation at large sides; AdamW with $\beta = (0.9, 0.999)$ and no weight decay, 8% warm-up, cosine decay to 2% of the peak, gradient clipping at 1, no label smoothing, fp32, and activation checkpointing. As in Section 3.3, each anchor exports its tables from its retained parameters up to the side of the next anchor, so every side between two anchors can be sliced from both, and the 144^2 anchor affects no size of at least 224^2. At anchor g the branch grows to $2g-1$ hinges, with the new knots mostly in the extrapolation band $x(G_0) < |c| \le x(g)$, and K grows to the largest odd number below g; new coefficients start at zero, so growth preserves the function. Learning rates are set as function-step budgets: the output layer of f_a uses $\eta\sqrt{12/L}/(2s\,\nu_a)$, where ν_a is the largest summed hinge activation over the anchor's coordinates, and E_a uses $\eta\sqrt{12/L}/(2s)$, with $\eta = 0.04/0.03/0.02$ for S/B/L. The loss adds 10^{-4} times the mean squares of the scaled tables sU_a and of their second differences along the offset. Training images are square and preprocessed like test images, with random horizontal flips; no run sees an anisotropic input.

Controls. Rotary frequencies use the learning rate $\eta_r \min\big(1, (G_0-1)/(g-1)\big)\sqrt{12/L}$ with $\eta_r = 1.5\times10^{-3}/8\times10^{-4}/5\times10^{-4}$ for S/B/L and a penalty 10^{-3} on the mean square of $(g-1)$ times the frequency change. The absolute embedding uses learning rate 10^{-3} and a penalty 10^{-3} on its mean-square change. At test time, the frequencies or embeddings of the two anchors are mixed with the weight of the side $\sqrt{HW}$, which a single set of frequencies or a single embedding requires; for a rectangle both axes therefore share one anchor pair. Joint runs first train only the backbone's parameters for 25% of the steps, then both, with the backbone's learning rate halved and the branch's learning rate ramped up over 10% of the steps.

Selection and evaluation. Five images per class (5,000) are held out from the training split and excluded from training. The candidates are the states after 80 and 160 steps, the average of the last 25% of the steps, and the midpoint between the starting state and the best candidate. The best candidate is kept if it beats the zero branch on the held-out images by at least 0.1 point; otherwise the zero branch is kept. For TRAPE this happened at 224^2 and 336^2 for ViT-B, at 336^2 for ViT-L, and at 224^2 for the absolute-position ViT-S. All 50,000 validation images are used only for the final evaluation, in which all configurations of a backbone are evaluated in one pass over the same images. For a rectangle, each image is resized with its aspect ratio kept until it covers $H\times W$ and then centre-cropped, and it is served from the same stored tapes and banks as the squares.

Repeated runs. A fitted anchor is served from its own tables, so its accuracy depends on no other anchor. To measure run-to-run variation we therefore refit the 704^2 anchor from the zero branch with seeds 1 and 2 and evaluate it on all 50,000 images: TRAPE, Log-CPB and 2D-Tanh on RoPE-Mixed ViT-S, and Freq-FT, TRAPE and TRAPE+Freq on RoPE-Mixed ViT-B. The seed also redraws the data order and the 5,000 held-out selection images (the unadapted model scores 68.5, 66.5 and 67.9% on them for the ViT-S TRAPE seeds 0–2). For ViT-B, seed 1 uses global batch 128 instead of the original 64; the seed 2 log contains only evaluation and does not record the training batch (Table 13).

B.5 THE TWO-DIMENSIONAL BRANCHES

The baseline follows Swin V2 (Liu et al., 2022): every block owns an MLP $2 \to 512 \to n_h$ with a ReLU and a bias in the hidden layer only, evaluated on the pair $(c(\Delta_h; G_0), c(\Delta_w; G_0))$ of Eq. 2, and its output passes through $16\sigma(\cdot)$. The output layer starts at zero, so the branch adds the constant 8 to every spatial logit; the class-token rows and columns are padded with the same constant, so the

constant is added to every key of a row and the initial attention is unchanged up to rounding. For ViT-S this adds 36 tensors and 55,296 trained values (0.251% of the model) at every anchor, because the width of a Swin V2 MLP does not depend on the grid. Training uses the code of the grafted TRAPE runs with only the branch exchanged: RoPE-Mixed ViT-S, the six anchors $\{9, 14, 21, 30, 44, 64\}$, each trained from the zero branch for 160 steps with global batch 128 and the same accumulation, optimizer, schedule, clipping, precision, data order and held-out selection. The MLP has no hinge normalization, so it uses a constant learning rate of 3×10^{-4} within the same schedule, and the loss adds 10^{-4} times the mean squares of the centred bias $16\sigma(\cdot) - 8$ and of its second differences along both axes, the counterpart of the TRAPE penalty. The joint variant adds the rotary frequencies as in Appendix B.4. A non-anchor shape interpolates the MLP weights of the two anchors that bracket $\mathrm{round}(\sqrt{HW})$, with the weights of Eq. 6, and evaluates the MLP on the $(2H-1)(2W-1)$ offsets of the shape. Both code bases load the same checkpoint (identical SHA-256) and the same held-out split, and the final evaluations differ only in the branch file and the list of shapes. The frequency-only control, trained in both code bases, reproduces the same held-out accuracies and the same test accuracy at every side.

The second two-dimensional branch uses the same joint MLP, denoted f_{2D}, with output $s \tanh\big(f_{2D}(c_h, c_w) + e_{2D}(\Delta_h, \Delta_w)\big)$, where $c_a = c(\Delta_a; G_0)$ and e_{2D} is the zero extension of a free $13{\times}13{\times}n_h$ central table E_{2D} per block (indexed by $[\Delta_h + 6, \Delta_w + 6, :]$ for $|\Delta_h|, |\Delta_w| \le 6$). It has one learnable amplitude shared by all blocks (initialized to 8). Its output layer and E_{2D} start at zero, so the branch starts exactly neutral. It adds 49 tensors and 67,465 trained values (0.306%), uses the same learning rate for all of them, and penalizes the mean squares of the final bias and of its second differences with the same weight; anchors, steps, batches, selection, mixing and evaluation are those of the first branch, and its checkpoint hash, held-out split and evaluation arguments match the TRAPE run. Training the six anchors took 27.5 minutes for TRAPE, 30.8 minutes for the Swin V2 bias and 31.4 minutes for the second branch on eight GPUs (Table 14).

C Cross-Resolution Generalization Without Fine-Tuning in Full

Table 8 extends Table 1 with all reported shapes, including 144^2–192^2, the per-block amplitude ablation, the combination of TRAPE with 2D RoPE-Mixed, and position-parameter counts. Each encoding is pre-trained at 224^2 and evaluated with the same harness on 50,000 images. These are single-run comparisons. The joint 2D branches used for frozen-backbone adaptation are evaluated separately in Appendix F.

Resolution dependence of the component ablations. The contribution of each component varies with the test grid. On DeiT-Ti, factorizing Log-CPB without changing its output map reduces the mean accuracy over the seven enlarged squares (336^2–1024^2) from 68.0% to 66.8%. Replacing that map with the shared $s \tanh$ raises the mean to 71.0%, and adding the central table raises it to 71.6%. Thus the complete parameterization, rather than factorization alone, explains the improvement in this ablation. Below the pre-training size, queried offsets remain inside the coordinate range seen in training; this is distinct from evaluating large, previously unseen offsets, but does not itself guarantee accuracy preservation.

One amplitude, shared. Giving every block its own amplitude (T, s_ℓ; 11 more parameters) ties TRAPE at 224^2 but lowers accuracy at every enlarged square, by 0.1–0.4 points, and on four of the five rectangles, by up to 1.0 point at $320{\times}960$. These single-run differences support the shared default without establishing significance. A single amplitude is also what lets the anchors of a tape be blended at one output scale.

A non-factorized 2D bias with both components. The DeiT-S 2D-Tanh control applies the shared $s \tanh$ and a $13{\times}13{\times}n_h$ central table to the joint 2D MLP of Log-CPB, $2 \to 512 \to n_h$ per block, without factorizing it. Near the training range it is as accurate as TRAPE: within 0.6 points up to 704^2 and on all five rectangles, and at 336^2 even 0.04 points higher before rounding (81.788% against 81.746%). Beyond, it falls to 75.1% at 800^2 and 23.6% at 1024^2, against 76.9% and 70.2% for TRAPE, and it needs $9.6\times$ the position parameters (67.5k against 7.1k). The two components therefore yield a strong joint bias as well, as they do for the grafted 2D branch (Appendix F). This single-run comparison does not by itself establish that factorization improves a bias trained end-to-

Table 8: **Cross-resolution generalization without fine-tuning: every measured shape.** ImageNet-1k top-1 (%), 50,000 images, one evaluation harness; every encoding is trained once at 224^2. APE: absolute embedding resized bicubically; Free: resampled axial free tables; (a)–(c): rows of Table 1; T, s_ℓ: TRAPE with one amplitude per block; T: TRAPE; RoPE: 2D RoPE-Mixed; T+RoPE: both combined; 2D-T: 2D-Tanh, the joint 2D MLP of (a) with the shared $s\tanh$ and a 13×13 central table. Last row: position parameters (k = 10^3), excluding the APE class token. Bold: best for each model size.

	DeiT-Ti							DeiT-S					DeiT-B	
Input	APE	Free	(a)	(b)	(c)	T, s_ℓ	T	APE	RoPE	T+ RoPE	2D-T	T	APE	T
144^2	60.7	62.2	63.9	63.2	63.9	64.4	**64.5**	72.4	75.0	**75.5**	75.2	75.1	76.8	**78.0**
176^2	68.4	68.4	68.7	68.5	69.4	69.3	**69.5**	77.5	78.5	**79.0**	78.7	78.3	80.4	**80.5**
192^2	70.2	70.2	70.4	70.1	**70.8**	70.6	**70.8**	78.6	79.6	**79.8**	79.5	79.4	81.1	**81.3**
224^2 *(train)*	72.1	72.1	72.2	72.1	**72.8**	**72.8**	**72.8**	79.8	80.8	**81.2**	80.6	80.6	81.8	**82.1**
336^2	71.9	70.5	73.4	73.0	74.2	74.3	**74.4**	79.4	81.6	**81.9**	81.8	81.7	81.9	**82.8**
384^2	71.2	69.9	73.2	72.8	74.1	74.4	**74.7**	78.4	81.1	**81.9**	81.8	81.8	81.4	**82.9**
480^2	68.1	66.8	72.4	71.6	73.9	74.0	**74.2**	75.1	78.9	80.8	81.3	**81.4**	79.6	**82.3**
560^2	64.9	63.1	71.2	70.0	72.8	73.3	**73.6**	71.0	74.8	78.4	80.6	**80.7**	77.5	**81.5**
704^2	58.1	55.9	68.2	65.9	70.6	71.4	**71.5**	61.5	59.7	72.0	78.2	**78.8**	72.6	**79.7**
800^2	53.1	50.4	65.2	62.5	68.7	69.3	**69.5**	54.0	43.4	65.0	75.1	**76.9**	61.3	**78.2**
1024^2	41.8	39.1	52.6	52.0	62.9	63.0	**63.4**	36.4	12.1	39.5	23.6	**70.2**	59.6	**73.3**
384×512	69.4	67.8	72.5	71.9	73.7	74.3	**74.4**	76.4	79.9	81.1	**81.5**	**81.5**	80.2	**82.5**
512×384	68.8	67.5	72.3	71.7	73.6	**74.0**	**74.0**	76.0	79.3	80.9	**81.2**	**81.2**	80.1	**82.1**
448×672	65.1	63.1	71.1	69.8	72.6	73.0	**73.3**	71.4	74.8	78.4	80.3	**80.7**	77.6	**81.4**
672×448	64.1	62.8	70.3	68.8	72.3	72.3	**72.6**	70.4	73.8	77.3	79.9	**80.0**	77.0	**80.8**
320×960	60.9	53.4	64.3	61.9	66.0	67.1	**68.1**	65.5	65.7	71.3	76.2	**76.3**	73.6	**78.5**
Parameters	37.6k	1.9k	36.9k	3.2k	3.2k	4.2k	4.2k	75.3k	4.6k	11.7k	67.5k	7.1k	150.5k	12.8k

end; the case for the axial form rests on its compact table representation, independent per-axis routing and adaptation under a matched budget (Section 3.3, Appendix F).

2D RoPE-Mixed on DeiT-S. The rotary control peaks at 336^2 (81.6%) and then falls to 74.8% at 560^2, 59.7% at 704^2, 43.4% at 800^2 and 12.1% at 1024^2, the lowest DeiT-S accuracy from 704^2 on, and reaches 65.7% on 320×960, against 76.3% for TRAPE.

Combining TRAPE with a rotary encoding is not additive. TRAPE + 2D RoPE-Mixed is the best DeiT-S encoding at 224^2 (81.2%) and up to 384^2, but its rotary branch carries the finite range of RoPE: it falls 2.3 points below TRAPE at 560^2, 6.8 at 704^2 and 30.7 at 1024^2, and 2.1 points below it on the five rectangles on average. After fine-tuning, jointly tuning the tape with the backbone's rotary frequencies likewise brings no consistent benefit: it lowers the tape-only result at the largest sides of RoPE-Mixed ViT-S and ViT-L, and raises it by more than 0.2 point only for ViT-B beyond 800^2 (Appendix E). We therefore report TRAPE on its own.

D Every Measured Shape for TRAPE-DeiT

Table 9 lists the 22 shapes behind the aggregates of Section 4.3 for one adaptation run per model (seed 0). The three shapes below the pre-training grid use the 144^2 anchor and, for intermediate sizes, its blend with the unadapted 224^2 state, and gain least: +0.9 and +0.4 at 144^2 for DeiT-Ti and DeiT-S, none for DeiT-B, and at most +0.3 at 176^2 and 192^2, compared with +7.0/+6.7/+7.8 at 1024^2. Below 224^2 the coordinate $x(g)$ lies inside the range seen in pre-training, so the axis functions interpolate and there is little to recover; the gains of the tape come from replacing extrapolation. A deployment that serves only sides of at least 224 can omit this anchor, which saves one run (169 steps, 86,528 presentations) and reduces the tape to 124.0/248.1/496.1 KiB.

Three adaptation seeds. Table 10 repeats the complete DeiT-S adaptation with two further random seeds; the frozen checkpoint, the anchors, the step budgets, the capacity policy and the routing are unchanged. The per-shape standard deviation over the three runs is at most 0.18 point (960^2) and below 0.1 point for 19 of the 22 shapes, and no seed falls below the unadapted model at any shape. The mean gain is 1.06 ± 0.07 points over the five rectangles and 1.65 ± 0.05 over the 19 shapes

Table 9: **Cross-resolution generalization after position-only fine-tuning: all TRAPE-DeiT shapes.** ImageNet-1k top-1 (%), 50,000 images. No-FT: no fine-tuning; TRAPE: position-only fine-tuned TRAPE, served from stored tables. PE-FT at $\{144, 336, 480, 704, 1024\}^2$; †: fitted anchor. 224^2 is not fine-tuned. All rectangles are unseen. Δ: TRAPE minus No-FT; shapes are height $\times$ width in pixels. One adaptation run per model (seed 0); Table 10 repeats DeiT-S with three seeds.

Input	DeiT-Ti			DeiT-S			DeiT-B		
	No-FT	TRAPE	Δ	No-FT	TRAPE	Δ	No-FT	TRAPE	Δ
below the pre-training grid									
144^2 †	64.5	65.4	+0.9	75.1	75.5	+0.4	78.0	78.0	+0.0
176^2	69.5	69.8	+0.3	78.3	78.5	+0.2	80.5	80.6	+0.1
192^2	70.8	71.1	+0.3	79.4	79.6	+0.2	81.3	81.3	+0.0
squares at or above the pre-training grid									
224^2	72.8	72.8	—	80.6	80.7	—	82.1	82.1	—
256^2	72.8	72.9	+0.1	80.9	81.0	+0.1	82.2	82.3	+0.1
288^2	73.7	73.9	+0.2	81.3	81.4	+0.1	82.6	82.7	+0.1
336^2 †	74.4	74.7	+0.3	81.7	81.9	+0.2	82.8	82.9	+0.1
384^2	74.7	75.1	+0.4	81.8	82.0	+0.2	82.9	83.3	+0.4
400^2	74.6	75.2	+0.6	81.8	82.1	+0.3	82.7	83.2	+0.5
480^2 †	74.2	75.2	+1.0	81.4	82.0	+0.6	82.3	83.2	+0.9
560^2	73.6	75.1	+1.5	80.7	81.9	+1.2	81.5	83.3	+1.8
608^2	72.8	74.8	+2.0	80.2	81.6	+1.4	80.9	83.0	+2.1
704^2 †	71.5	74.0	+2.5	78.8	80.8	+2.0	79.7	82.5	+2.8
800^2	69.5	73.1	+3.6	76.9	80.0	+3.1	78.2	82.2	+4.0
896^2	67.2	72.2	+5.0	74.6	78.8	+4.2	76.4	81.9	+5.5
960^2	65.4	71.3	+5.9	72.5	78.0	+5.5	75.0	81.5	+6.5
1024^2 †	63.4	70.4	+7.0	70.2	76.9	+6.7	73.3	81.1	+7.8
rectangles									
384×512	74.4	75.1	+0.7	81.5	82.1	+0.6	82.5	83.3	+0.8
512×384	74.0	74.9	+0.9	81.2	81.8	+0.6	82.1	83.1	+1.0
448×672	73.3	74.7	+1.4	80.7	81.7	+1.0	81.4	83.1	+1.7
672×448	72.6	74.3	+1.7	80.0	81.2	+1.2	80.8	82.7	+1.9
320×960	68.1	70.4	+2.3	76.3	77.8	+1.5	78.5	80.8	+2.3
mean, 14 squares $\geq 224^2$	71.5	73.6	+2.1	78.8	80.7	+1.9	80.2	82.5	+2.3
mean, 5 rectangles	72.5	73.9	+1.4	79.9	80.9	+1.0	81.1	82.6	+1.5
mean, 19 shapes $\geq 224^2$	71.7	73.7	+2.0	79.1	80.7	+1.6	80.4	82.5	+2.1
mean, all 22 shapes	71.3	73.0	+1.7	78.9	80.3	+1.4	80.4	82.2	+1.8

with both sides of at least 224 pixels; at the fitted 1024^2 anchor it is 6.76 ± 0.16. Table 2 reports the three-seed means for DeiT-S.

E GRAFTED TAPES IN FULL

Tables 11 and 12 list every measured shape of the grafted experiments, squares and rectangles together, Figure 4 plots the squares, Table 13 repeats the 704^2 anchor with further seeds, and Table 14 gives the budgets of the six-anchor tapes. Four observations complement Section 4.4. First, from 224^2 upward the gain of the tape never decreases as the input side grows, for every backbone: from $+0.3$ at 336^2 to $+56.0$ at 1024^2 for RoPE-Mixed ViT-S, and to $+43.2$, $+21.2$, $+33.7$ and $+17.6$ at 1024^2 for RoPE-Axial ViT-S, APE ViT-S, ViT-B and ViT-L; up to 704^2 the paired standard error of these gains is below 0.25 point for every backbone. Below 224^2, where the coordinates stay inside the pre-training range, the gains are 0.0–1.3 points and largest at the fitted 144^2 anchor, as for the TRAPE-DeiT tapes (Appendix D). Second, the *Joint-off* columns show that when the branch and the backbone's parameters are trained together, the branch carries the gain: the jointly tuned frequencies alone reach 70.9% at 704^2 on ViT-S, below the 73.1% of frequencies tuned on their own, and 56.1% against 62.8% on ViT-B and 67.9% against 70.7% on ViT-L at 1024^2, while the jointly tuned absolute embedding alone stays within 0.3 point of the original model at every shape. Joint training changes the tape-only result by -4.2 to $+1.7$ points: it lowers it at every square from 336^2 on for RoPE-Mixed ViT-S and APE ViT-S, and the only gains above 0.2 point are those of ViT-B

Table 10: **Position-only fine-tuning of TRAPE-DeiT-S with three adaptation seeds** (ImageNet-1k top-1, %, 50,000 images). Seed 0 is the run of Table 9; seeds 1 and 2 repeat the same recipe, anchors, capacity policy, tail averaging and blend routing on the same frozen checkpoint. Mean ± sample standard deviation over the three seeds; Δ: displayed mean minus No-FT. Summary rows average the shapes within each seed first; their Δ gives the mean ± standard deviation of the per-seed gains. †: fitted anchor.

Input	No-FT	seed 0	seed 1	seed 2	mean ± s.d.	Δ
below the pre-training grid						
144^2 †	75.1	75.5	75.5	75.5	75.5 ±0.01	+0.4
176^2	78.3	78.5	78.5	78.6	78.5 ±0.04	+0.2
192^2	79.4	79.6	79.6	79.6	79.6 ±0.01	+0.2
squares at or above the pre-training grid						
224^2	80.6	80.7	80.7	80.7	80.7 ±0.00	—
256^2	80.9	81.0	81.0	80.9	81.0 ±0.02	+0.1
288^2	81.3	81.4	81.4	81.5	81.4 ±0.04	+0.1
336^2 †	81.7	81.9	81.9	81.9	81.9 ±0.04	+0.2
384^2	81.8	82.0	82.1	82.1	82.1 ±0.06	+0.3
400^2	81.8	82.1	82.2	82.1	82.1 ±0.01	+0.3
480^2 †	81.4	82.0	82.1	82.1	82.1 ±0.07	+0.7
560^2	80.7	81.9	81.9	81.9	81.9 ±0.02	+1.2
608^2	80.2	81.6	81.7	81.7	81.6 ±0.04	+1.4
704^2 †	78.8	80.8	81.0	80.9	80.9 ±0.08	+2.1
800^2	76.9	80.0	80.1	80.0	80.0 ±0.04	+3.1
896^2	74.6	78.8	78.9	78.9	78.9 ±0.06	+4.3
960^2	72.5	78.0	78.2	77.9	78.0 ±0.18	+5.5
1024^2 †	70.2	76.9	77.2	76.8	77.0 ±0.16	+6.8
rectangles						
384×512	81.5	82.1	82.1	82.1	82.1 ±0.04	+0.6
512×384	81.2	81.8	81.9	81.9	81.9 ±0.04	+0.7
448×672	80.7	81.7	81.7	81.8	81.7 ±0.05	+1.0
672×448	80.0	81.2	81.3	81.3	81.3 ±0.06	+1.3
320×960	76.3	77.8	78.1	78.1	78.0 ±0.16	+1.7
mean, 5 rectangles	79.9	80.9	81.0	81.0	81.0	+1.06 ±0.07
mean, 19 shapes $\geq 224^2$	79.1	80.7	80.8	80.8	80.8	+1.65 ±0.05
mean, all 22 shapes	78.9	80.3	80.4	80.4	80.4	+1.47 ±0.04

beyond 800^2. Third, rectangles follow the squares but depend on their elongation. At 320×960 every backbone gains more than at 560^2, a square of similar area (9.7, 9.4, 12.5, 3.7 and 1.8 points against 4.4, 5.4, 6.0, 1.8 and 0.7 in the order of Table 3), because the long side lies far outside the pre-training range; and every RoPE backbone loses more accuracy on the tall inputs (672×448) than on the wide ones (448×672), where the tape also recovers more. The own-parameter controls share one anchor pair between the two axes and trail the tape on all 20 rectangles where they exist. Fourth, the budget is dominated by the largest anchors, where batches are accumulated over single images; the branch adds less than 0.13% of the parameters, and the complete six-anchor ViT-S tape stores 67,392 fp32 values (263 KiB), fewer than one tuned absolute embedding (75,264).

E.1 Repeated runs at the 704^2 anchor

Table 13 combines the original run (seed 0) with refits of the 704^2 anchor using seeds 1 and 2 (protocol in Appendix B.4). On RoPE-Mixed ViT-S, TRAPE, Log-CPB and 2D-Tanh reach 79.87 ± 0.05%, 75.30 ± 0.06% and 77.82 ± 0.12%, respectively (mean ± sample standard deviation over three adaptation seeds, computed before rounding). Their gains over the unadapted model are much larger than these standard deviations. Table 3 reports these statistics at 704^2; the full-shape tables and figures retain seed 0. The ViT-B summary includes a batch-size change (64 in run 0, 128 in run 1; unrecorded in run 2), so it does not estimate variability under a fixed recipe. The two additional runs raise every adapted configuration by 0.24–0.32 point, while the paired gap between TRAPE and Freq-FT stays at 3.21 ± 0.05 points. Table 14 lists the original six-anchor budgets.

Table 11: **Generalization after position-only fine-tuning: RoPE-Mixed ViT-S/B/L, every measured shape.** Top-1 (%), 50,000 validation images, one run per configuration (seed 0; repeated runs at 704^2: Table 13). PE-FT anchors $\{144, 224, 336, 480, 704, 1024\}^2$ (†); rectangles are height×width and never tuned. No-FT: unchanged checkpoint; Freq: Freq-FT; TRAPE: grafted branch served from its tables; +Freq: TRAPE+Freq (joint tuning); Off: Joint-off, the jointly tuned frequencies with the branch disabled. Bold: best of Freq-FT, TRAPE and TRAPE+Freq.

	RoPE-Mixed ViT-S					**RoPE-Mixed ViT-B**					**RoPE-Mixed ViT-L**				
Input	No-FT	Freq	TRAPE	+Freq	Off	No-FT	Freq	TRAPE	+Freq	Off	No-FT	Freq	TRAPE	+Freq	Off
below the pre-training size															
144^2 †	74.2	74.8	75.4	**75.5**	74.7	79.4	79.4	**79.7**	**79.7**	79.4	81.0	81.1	**81.3**	81.2	81.1
176^2	78.4	78.5	78.7	**78.8**	78.5	82.0	82.0	**82.1**	**82.1**	82.0	83.3	83.3	**83.4**	83.3	83.3
192^2	79.6	79.6	**79.8**	**79.8**	79.6	82.8	82.8	82.8	**82.9**	82.8	83.9	83.9	83.9	83.9	83.9
224^2 †	80.9	80.9	**81.0**	**81.0**	81.0	83.8	83.8	83.8	83.8	83.8	84.8	84.8	84.8	84.8	84.8
336^2 †	82.1	82.1	**82.4**	82.3	82.2	84.6	84.6	84.6	84.6	84.6	85.6	85.6	85.6	85.6	85.6
384^2	81.8	82.1	**82.4**	82.3	81.9	84.4	84.4	**84.5**	**84.5**	84.4	85.6	85.6	85.6	85.6	85.6
480^2 †	80.2	80.8	**81.9**	81.7	80.7	83.4	83.6	83.9	**84.1**	83.6	85.0	85.0	**85.3**	85.2	85.1
512^2	79.1	80.2	**81.6**	81.3	79.9	82.9	83.3	83.9	**84.0**	83.2	84.7	84.7	**85.1**	85.0	84.8
560^2	77.1	79.2	**81.5**	81.2	78.6	81.9	82.5	**83.7**	**83.7**	82.3	84.2	84.3	**84.9**	84.8	84.3
608^2	74.2	77.7	**81.1**	80.8	76.7	80.6	81.6	**83.4**	83.3	81.3	83.5	83.8	**84.6**	84.5	83.7
704^2 †	64.5	73.1	**79.8**	79.6	70.9	76.8	79.5	**82.7**	82.6	78.6	81.4	82.4	**84.0**	83.8	82.1
800^2	49.0	64.6	**78.3**	77.3	60.2	70.1	75.4	**81.4**	**81.4**	73.6	78.0	79.9	**83.2**	82.8	79.3
896^2	32.5	52.4	**76.2**	74.2	45.2	59.5	70.8	79.5	**79.8**	67.3	72.6	76.7	**81.7**	81.4	75.4
960^2	23.3	43.8	**74.4**	71.5	34.9	50.8	67.0	77.6	**78.4**	62.1	67.4	74.1	**80.4**	80.1	72.1
1024^2 †	16.6	36.4	**72.6**	68.9	26.4	41.7	62.8	75.4	**77.1**	56.1	61.3	70.7	**78.9**	78.6	67.9
unseen rectangles															
384×512	80.5	80.9	**81.7**	81.5	80.7	83.5	83.6	**83.9**	**83.9**	83.5	85.0	85.0	**85.1**	**85.1**	84.9
512×384	80.1	80.8	**81.6**	81.4	80.6	83.3	83.4	**83.7**	**83.7**	83.4	84.9	84.9	**85.0**	**85.0**	84.9
448×672	76.2	78.1	**80.6**	80.3	77.5	81.5	81.9	**83.1**	83.0	81.8	83.6	83.7	**84.3**	84.2	83.7
672×448	74.7	77.3	**79.9**	79.6	76.6	80.5	81.2	**82.6**	82.5	81.0	83.2	83.4	**84.1**	84.0	83.4
320×960	64.4	67.1	**74.1**	72.8	66.3	74.7	75.4	**78.4**	78.1	75.1	78.4	78.6	**80.2**	79.9	78.5

Table 12: **Generalization after position-only fine-tuning: RoPE-Axial and APE ViT-S, every measured shape.** Top-1 (%), 50,000 images, seed 0; anchors as in Table 11. APE-FT: only the absolute embedding tuned; +APE: TRAPE+APE; Off: the jointly tuned embedding with the branch disabled. RoPE-Axial has fixed frequencies and no Freq-FT control. Bold: best of APE-FT, TRAPE and TRAPE+APE.

	RoPE-Axial ViT-S		**APE ViT-S**				
Input	No-FT	TRAPE	No-FT	APE-FT	TRAPE	+APE	Off
below the pre-training size							
144^2 †	73.6	74.7	73.6	73.6	**74.9**	74.8	73.6
176^2	78.0	78.5	77.8	77.8	**78.4**	78.3	77.8
192^2	79.2	79.5	79.1	79.1	**79.3**	79.2	79.1
224^2 †	80.9	80.8	80.5	80.5	80.5	80.5	80.5
336^2 †	81.2	81.7	80.3	80.3	**81.7**	81.5	80.3
384^2	80.0	81.0	79.4	79.4	**81.6**	81.1	79.4
480^2 †	77.2	79.5	76.5	76.5	**80.4**	79.7	76.5
512^2	76.1	79.2	75.4	75.4	**80.1**	79.3	75.4
560^2	73.4	78.8	73.6	73.6	**79.6**	78.6	73.6
608^2	70.5	78.3	71.7	71.8	**79.3**	77.9	71.8
704^2 †	61.7	76.4	67.6	67.7	**78.0**	76.2	67.7
800^2	49.3	74.1	62.7	63.0	**75.4**	72.5	62.8
896^2	35.8	70.6	57.4	57.9	**74.0**	70.6	57.6
960^2	27.4	67.8	54.1	54.7	**73.0**	69.1	54.3
1024^2 †	20.6	63.8	50.9	51.6	**72.1**	67.9	51.2
unseen rectangles							
384×512	78.6	80.2	76.7	76.7	**80.5**	79.7	76.7
512×384	77.1	79.4	76.9	76.9	**80.4**	79.7	76.9
448×672	73.8	78.2	72.6	72.7	**79.1**	77.9	72.7
672×448	70.3	77.3	72.1	72.2	**78.6**	77.4	72.2
320×960	63.1	72.5	59.5	59.6	**72.0**	69.4	59.5

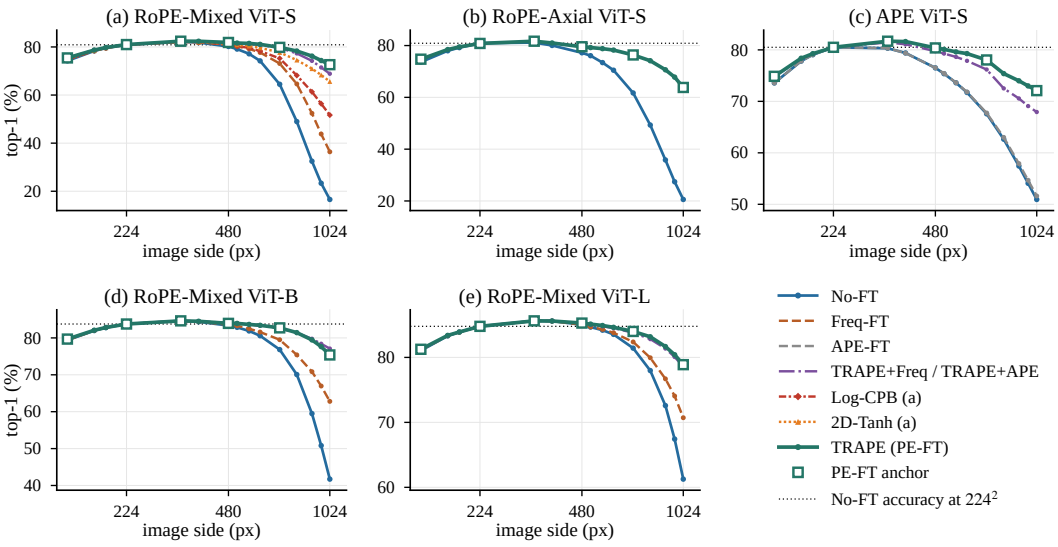


Figure 4: **Generalization without and after position-only fine-tuning on public ViTs.** ImageNet-1k top-1 on 50,000 validation images per square size (seed 0). No-FT: unchanged checkpoint; TRAPE: fitted branch; Freq-FT/APE-FT: fitted frequencies/embeddings; +Freq/+APE: joint tuning; Log-CPB and 2D-Tanh: joint 2D branches. Squares mark the fitted anchors $\{144, 224, 336, 480, 704, 1024\}^2$, shared by every method. Labels follow Appendix B.2.

Table 13: **Repeated position-only fine-tuning at the 704^2 anchor** (top-1, %, 50,000 images). A fitted anchor is served from its own tables, so each repetition refits the 704^2 anchor from the zero branch and evaluates it. Seed 0 is the original run in Tables 11 and 15; ViT-S seeds 1 and 2 repeat its recipe (global batch 128). § ViT-B batches are 64/128/unrecorded for runs 0/1/2; their summary does not estimate seed variation under a fixed recipe. Mean ± sample s.d. ($n = 3$), calculated before rounding; per-seed values retain three decimals. No-FT repeats the same frozen checkpoint, not independent training. Δ: displayed mean minus No-FT.

Backbone	**Method**	seed 0	seed 1	seed 2	mean ± s.d.	Δ
RoPE-Mixed ViT-S	No-FT	64.474	64.474	64.474	64.5	—
	TRAPE	79.828	79.874	79.918	79.9 ±0.05	+15.4
	Log-CPB	75.240	75.322	75.346	75.3 ±0.06	+10.8
	2D-Tanh	77.706	77.826	77.938	77.8 ±0.12	+13.3
RoPE-Mixed ViT-B§	No-FT	76.834	76.834	76.834	76.8	—
	Freq-FT	79.506	79.790	79.804	79.7 ±0.17	+2.9
	TRAPE	82.726	83.046	82.964	82.9 ±0.17	+6.1
	TRAPE+Freq	82.592	82.860	82.836	82.8 ±0.15	+6.0
	Joint-off	78.596	78.882	78.860	78.8 ±0.16	+2.0
ViT-B: TRAPE − Freq-FT		3.22	3.26	3.16	3.21 ±0.05	—

F Two-Dimensional Biases in Full

Grafted branches: accuracy and fit. Table 15 lists every shape of the comparison in Table 3, adds the held-out selection accuracy at each anchor, and includes the second two-dimensional branch with TRAPE's shared $s\tanh$ and a 13×13 central table (protocols in Appendix B.5). The Swin V2 branch does not pass the selection threshold at 224^2 and 336^2, where TRAPE improves the held-out accuracy by 0.1 and 0.2 points. At the fitted 704^2 anchor, where no mixing takes place, it reaches 80.6% on the held-out images against 85.8% for TRAPE, although it trains 2.95 times more values; the test gap at 704^2 (4.6 points in seed 0) mirrors this held-out gap (5.2 points), so under this budget the deficit is one of fitting, not of interpolation. The same holds at the 1024^2 anchor: 53.3% on the held-out images against 76.2% for TRAPE, and 51.7% against 72.6% on the test set; in between, at 800^2–960^2, the Swin V2 branch trails by 10.1–18.0 points, at 320×960 by 5.5, and below 224^2 by 0.2–0.8. Adding the rotary frequencies changes the Swin V2 result by at most 0.3 point up to 704^2 but lowers it by 2.3–4.9 points from 800^2 to 1024^2, and the jointly tuned frequencies alone (*Joint-off*)

Table 14: **Budget of position-only fine-tuning on public checkpoints.** Every configuration fits the six anchors $\{144, 224, 336, 480, 704, 1024\}^2$, each a separate 160-step run from the zero branch (global batch 128 for ViT-S, 64 for ViT-B/L). Freq-FT/APE-FT tune only existing frequencies/embeddings; TRAPE tunes the grafted branch; +Freq/+APE tune both; the 2D branches (Appendix B.5) share the RoPE-Mixed ViT-S backbone. Images: training presentations including repeats, without selection or evaluation images; Epochs divide by 1,281,167. Values: parameters trained at the 1024^2 anchor; Share: their fraction of the model. Time: training time summed over the six anchors on 8 A800 GPUs in fp32, excluding selection and evaluation.

Backbone	Trained	Steps	Images	Epochs	Values	Share	Time
RoPE-Mixed ViT-S	Freq-FT	960	122.9k	0.096	4.6k	0.021%	19.0 min
	TRAPE	960	122.9k	0.096	27.4k	0.124%	27.5 min
	TRAPE+Freq	960	122.9k	0.096	32.0k	0.145%	26.5 min
2D branches	Log-CPB	960	122.9k	0.096	55.3k	0.251%	30.8 min
	Log-CPB+Freq	960	122.9k	0.096	59.9k	0.272%	29.1 min
	2D-Tanh	960	122.9k	0.096	67.5k	0.306%	31.4 min
RoPE-Axial ViT-S	**TRAPE**	960	122.9k	0.096	27.4k	0.124%	18.7 min
APE ViT-S	APE-FT	960	122.9k	0.096	75.3k	0.341%	14.7 min
	TRAPE	960	122.9k	0.096	27.4k	0.124%	19.4 min
	TRAPE+APE	960	122.9k	0.096	102.6k	0.465%	19.1 min
RoPE-Mixed ViT-B	Freq-FT	960	61.4k	0.048	9.2k	0.011%	13.2 min
	TRAPE	960	61.4k	0.048	54.7k	0.063%	16.8 min
	TRAPE+Freq	960	61.4k	0.048	63.9k	0.074%	17.5 min
RoPE-Mixed ViT-L	Freq-FT	960	61.4k	0.048	24.6k	0.008%	36.3 min
	TRAPE	960	61.4k	0.048	145.9k	0.048%	49.3 min
	TRAPE+Freq	960	61.4k	0.048	170.5k	0.056%	50.0 min

never exceed the frequencies tuned on their own, as in the TRAPE joint runs. The second branch shows that TRAPE's output form helps a two-dimensional bias here as well: it matches or exceeds the Swin V2 form at every size, by 2.5 points at 704^2 and 13.9 at 1024^2, and reaches 83.3% and 68.5% on the held-out images at these two anchors. It still trails TRAPE at every side from 336^2 on, by 0.3–7.0 points, and on every rectangle, by 1.0 on Rect-4 and 2.6 at 320×960, while training 2.5–3.6 times more values. Because a zero-initialized 16σ branch shifts every logit by 8, it is neutral only up to rounding: at 224^2 and at 336^2 it changes the top-5 prediction of one test image, and on the held-out images its untrained accuracy differs from that of the backbone by up to two of the 5,000 images. Zero-initialized $s\tanh$ branches reproduce the backbone exactly: at 336^2, where the second branch kept its zero state, its top-1 and top-5 equal those of the backbone, and Appendix A.5 verifies this for TRAPE. Budgets are matched in anchors, steps, data and selection; learning-rate normalization, regularization and the treatment of the amplitude necessarily differ between the parameterizations.

Scope of the comparison. These results compare adaptation of a frozen public RoPE-Mixed ViT-S under a matched data and step budget. They do not establish a universal accuracy ordering between axial and joint 2D biases or reproduce training those encodings with the full backbone. The structural advantage of the axial branch is that its exported responses can be stored and routed separately for height and width.

What must be stored. Table 16 counts the stored values that serve every shape with sides between 224 and 704 pixels. A two-dimensional bias can retain MLP weights and regenerate uncached shapes, or cache 2D tables; both options are included. Two-dimensional tables obey the centre-slice identity as well, so a square input can be sliced from a larger square table, but a table fixes the anchor of both axes: a rectangle whose sides fall between different anchor pairs must share one pair, and the tables grow with the square of the side. A single 704^2 table of ViT-S has 544,968 values (2.1 MiB), one table per anchor needs 5.6 MiB, 37 times the TRAPE tape, and covering every $H\times W$ directly would take 857.6 MiB; the central 13×13 table merges into these tables and does not change their size. The 2D-Tanh parameter bank stores $4 \cdot 67{,}465 = 269{,}860$ values, including one learned amplitude per anchor. A 2D table bank is therefore feasible, but larger and tied to one anchor choice for both axes. Mixing 2D tables is not generally equivalent to the measured baseline's MLP-weight mixing, because its hidden representation is nonlinear; Table 16 reports the storage option without assigning it an

Table 15: **Position-only fine-tuning of 2D branches: full results.** RoPE-Mixed ViT-S; PE-FT anchors $\{144, 224, 336, 480, 704, 1024\}^2$ (†) for every branch. Top: validation top-1 (%), 50,000 images, seed 0; three-seed 704^2 statistics: Table 13. No-FT: unchanged checkpoint; Freq-FT: frequency-only tuning; Log-CPB: Swin V2 bias; Log-CPB+Freq: joint tuning; Joint-off: the latter with its bias disabled; 2D-Tanh: joint 2D MLP with the shared $s\tanh$ and a learned 13×13 central table; TRAPE: factorized branch. No-FT, Freq-FT and TRAPE repeat Table 11; —: not evaluated. Bottom: 5,000 held-out training images used for selection, before (Zero) and after fitting; $*$: neutral branch retained. Bold: best bias branch.

			Log-CPB				
Input	**No-FT**	**Freq-FT**	**Log-CPB**	**+Freq**	**Joint-off**	**2D-Tanh**	**TRAPE**
144^2 †	74.2	74.8	74.6	74.9	74.8	—	**75.4**
176^2	78.4	78.5	78.4	78.5	78.5	78.5	**78.7**
192^2	79.6	79.6	79.6	79.6	79.6	79.6	**79.8**
224^2 †	80.9	80.9	80.9	80.9	80.9	**81.0**	**81.0**
336^2 †	82.1	82.1	82.1	82.1	82.1	82.1	**82.4**
384^2	81.8	82.1	82.0	82.1	82.0	82.0	**82.4**
480^2 †	80.2	80.8	81.0	80.9	80.8	81.3	**81.9**
512^2	79.1	80.2	80.4	80.4	80.1	80.9	**81.6**
560^2	77.1	79.2	79.6	79.6	78.9	80.4	**81.5**
608^2	74.2	77.7	78.3	78.5	77.2	79.6	**81.1**
704^2 †	64.5	73.1	75.2	75.3	71.8	77.7	**79.8**
800^2	49.0	64.6	68.2	65.9	58.6	74.5	**78.3**
896^2	32.5	52.4	61.3	57.7	46.0	71.0	**76.2**
960^2	23.3	43.8	56.4	52.1	37.7	68.3	**74.4**
1024^2 †	16.6	36.4	51.7	46.8	30.2	65.6	**72.6**
384×512	80.5	80.9	81.0	81.0	80.8	81.2	**81.7**
512×384	80.1	80.8	80.6	80.8	80.7	80.9	**81.6**
448×672	76.2	78.1	78.7	78.5	77.8	79.4	**80.6**
672×448	74.7	77.3	77.5	77.7	77.0	78.5	**79.9**
320×960	64.4	67.1	68.6	67.6	65.8	71.5	**74.1**
Rect-4	77.9	79.3	79.5	79.5	79.1	80.0	**81.0**

held-out selection images at the anchors

Anchor	**Zero**	**Log-CPB**	**2D-Tanh**	**TRAPE**
144^2	82.9	83.4	83.9	84.7
224^2	90.2	90.2^*	90.3	90.3
336^2	90.2	90.2^*	90.2^*	90.4
480^2	87.5	88.4	88.5	88.8
704^2	68.5	80.6	83.3	85.8
1024^2	16.0	53.3	68.5	76.2

Table 16: **Inference storage for all supported shapes.** ViT-S: $L = 12$, $n_h = 6$, four PE-FT anchors $\{224, 336, 480, 704\}^2$; token sides $14 \le H, W \le 44$ (31^2 shapes). Values are fp32 scalars; KiB/MiB are $2^{10}/2^{20}$ bytes. Training checkpoints and expanded attention matrices are excluded. Log-CPB and 2D-Tanh retain MLP weights; alternative 2D cache banks are also counted. A 2D anchor bank uses shared-axis table blending, which need not reproduce MLP-weight blending.

Stored object	**Values**	**Size**	**Work for a new shape**	**Axes**
TRAPE tape	39,456	154 KiB	slice and mix per axis	independent
TRAPE, one table per side and axis	254,448	994 KiB	read per axis	independent
Log-CPB, MLP weights	221,184	864 KiB	mix weights, run MLP	shared
2D-Tanh, weights	269,860	1.0 MiB	mix weights, run MLP	shared
2D, one table per anchor	1,461,600	5.6 MiB	slice and mix	shared
2D, one table per shape	224,804,808	857.6 MiB	read	—

unmeasured accuracy. Because the TRAPE axes are independent, even one table for every side and axis, which needs neither mixing nor a network at test time, takes 994 KiB and covers all 31^2 shapes.

Table 17: **Generalization after position-only fine-tuning: capacity and routing.** TRAPE-DeiT at Ti/S/B scales, PE-FT at $\{144, 336, 480, 704, 1024\}^2$; top-1 means (%) on the same 10,000 validation images. No-FT: no adaptation. Left: mean over 19 shapes with both sides ≥ 224; the capacity policy `both` grows the MLP width and the central-table support. Right: nine unseen square sizes, all with `both`. Lower/Upper choose one anchor, Nearest chooses the closer in $x(g)$, Blend weights both. Bold: best column.

Grown with the grid	**Ti**	**S**	**B**	**Assignment rule**	**Ti**	**S**	**B**
No-FT	71.4	79.3	80.6	No-FT	71.3	79.2	80.4
neither (fixed capacity)	73.3	80.8	82.8	lower anchor only	73.4	80.8	82.6
MLP width d_c	73.4	**80.9**	82.8	upper anchor only	73.3	80.7	82.6
central table K	73.5	**80.9**	82.8	nearest anchor	73.5	80.8	82.7
both (default)	**73.6**	**80.9**	**82.9**	**blend** (default)	**73.7**	**81.0**	**82.9**

G Tape Design Choices: Capacity and Assignment

A tape involves two choices that the centre-slice identity leaves open: how much capacity each anchor run may add, and how a side between two anchors obtains its table. Both were swept at all three scales on a fixed 10,000-image validation subset, with the same images in every cell (Tables 17 and 18); the defaults were selected from these sweeps and then evaluated on all 50,000 images (Table 19), which include the images used to choose them. A single accuracy on 10,000 images has a 95% interval of at most ± 1.0 point under an independent-image binomial approximation. Every cell uses the same images, so differences should be assessed as paired; without the per-image outcomes we do not assign intervals to the small policy gaps.

Capacity. For $g > 14$, MLP growth uses $d_c = 2g-1$ and table growth uses the largest odd integer below g; the anchor below the pre-training grid keeps the initial capacity. All policies export tables of identical shape, so capacity is a training-time choice. Growing the central table helps at least as much as widening the MLP at every scale. This supports allocating adaptation capacity to individual offsets near the origin, consistent with the role of the correction in Section 3.2. Growing both is best on Tiny and Base and tied for best on Small, and we use it at all scales. Even fixed capacity recovers most of the gain, so the anchor fit rather than the extra capacity is the main mechanism.

Assignment. A side between two anchors can read the lower anchor, the upper anchor, the nearer of the two in the coordinate $x(g)$, or the blend of Eq. 6. Blending is the best rule in all twelve capacity–scale combinations of Table 18. On the full validation set (Table 19) it is the best rule in 39 of the 48 non-anchor shape–model pairs and tied for best in 8; the exception is 512×384 on DeiT-S, where the lower anchor is 0.1 point higher. The order of the two single-anchor rules changes with the target size and the model, so neither one-sided rule is consistently strongest in these measurements. Blending combines both responses without an additional fitted routing model. Because each axis is routed on its own, the rule applies unchanged to rectangles, whose two sides often fall between different anchor pairs. Replacing the normalized coordinate by a linear ruler over side lengths changes the probe mean by at most 0.01 point; the coordinate is kept as the export convention because it is the quantity the network consumes.

H The Learned Encoding

The following diagnostics are read from the trained 224^2 checkpoints of TRAPE-DeiT-Ti/S/B without data; controlled performance ablations are reported separately. The visualizations re-evaluate the stored weights in float64 with the fixed-reference coordinate of Eq. 2. Before plotting, the reconstructed biases were checked against the attention probabilities of the original implementation for every block on the 14×14 grid and for the first and last block on a 9×11 grid (maximum absolute deviation 1.2×10^{-7} in float32), and the centre-slice identity was verified for all blocks and heads with zero deviation. Offsets are key minus query, and blocks and heads are numbered from zero. Figures 6–8 show block 5 and, for each model, the head whose 224^2 bias range is closest to the median over the heads of that block (heads 0, 5 and 6 of DeiT-Ti/S/B); all raw heat maps share one symmetric colour scale, without per-head normalization.

Table 18: **Generalization after position-only fine-tuning: capacity crossed with routing.** Same TRAPE-DeiT anchors $\{144, 336, 480, 704, 1024\}^2$ and 10,000 images as Table 17; mean over its nine unseen square sizes. Fixed/grow MLP/grow table/grow both specify capacity. Lower/Upper/Nearest/Blend use the definitions in Table 17; No-FT is the unadapted checkpoint. Bold: best rule per row.

Scale	Capacity	Lower	Upper	Nearest	Blend
Tiny	No-FT		71.3		
	fixed	73.2	73.1	73.2	**73.4**
	grow MLP	73.2	73.2	73.3	**73.4**
	grow table	73.3	73.2	73.4	**73.6**
	grow both	73.4	73.3	73.5	**73.7**
Small	No-FT		79.2		
	fixed	80.6	80.6	80.7	**80.8**
	grow MLP	80.7	80.7	80.8	**80.9**
	grow table	80.7	80.7	80.8	**81.0**
	grow both	80.8	80.7	80.8	**81.0**
Base	No-FT		80.4		
	fixed	82.5	82.5	82.6	**82.8**
	grow MLP	82.6	82.5	82.6	**82.9**
	grow table	82.5	82.5	82.6	**82.9**
	grow both	82.6	82.6	82.7	**82.9**

Table 19: **Generalization after position-only fine-tuning: routing ablation.** TRAPE-DeiT, top-1 (%), all 50,000 validation images; PE-FT at $\{144, 336, 480, 704, 1024\}^2$, with both MLP width and central support grown. Rows are unseen non-anchor sizes. Lower/Upper: centre-slice one neighbouring anchor per axis; Blend: weighted combination of both. Bold: best rule; final rows give means and win/tie counts.

Input	DeiT-Ti			DeiT-S			DeiT-B		
	lower	upper	blend	lower	upper	blend	lower	upper	blend
176^2	69.6	69.5	**69.8**	**78.5**	78.3	**78.5**	80.2	80.5	**80.6**
192^2	70.9	70.8	**71.1**	79.2	79.4	**79.6**	80.7	**81.3**	**81.3**
256^2	72.8	72.8	**72.9**	80.9	80.7	**81.0**	82.2	81.8	**82.3**
288^2	73.7	73.8	**73.9**	81.3	81.3	**81.4**	82.6	82.5	**82.7**
384^2	74.9	74.9	**75.1**	**82.0**	81.9	**82.0**	83.1	83.0	**83.3**
400^2	75.1	75.0	**75.2**	82.0	82.0	**82.1**	83.1	83.0	**83.2**
560^2	74.9	74.7	**75.1**	81.8	81.5	**81.9**	83.0	82.8	**83.3**
608^2	74.6	74.5	**74.8**	81.4	81.4	**81.6**	82.7	82.7	**83.0**
800^2	**73.1**	72.0	**73.1**	79.8	79.1	**80.0**	82.0	81.5	**82.2**
896^2	72.0	71.7	**72.2**	78.5	78.4	**78.8**	81.2	81.5	**81.9**
960^2	70.9	71.1	**71.3**	77.3	77.8	**78.0**	80.5	81.3	**81.5**
384×512	75.0	74.7	**75.1**	82.0	81.8	**82.1**	83.2	83.0	**83.3**
512×384	**74.9**	74.4	**74.9**	**81.9**	81.4	81.8	83.0	82.7	**83.1**
448×672	74.5	74.6	**74.7**	81.3	81.6	**81.7**	82.6	83.0	**83.1**
672×448	73.9	74.2	**74.3**	81.1	81.1	**81.2**	82.3	**82.7**	**82.7**
320×960	69.4	**70.4**	**70.4**	77.3	77.7	**77.8**	80.2	**80.8**	**80.8**
mean, 16 shapes	73.1	73.1	**73.4**	80.4	80.3	**80.6**	82.0	82.1	**82.4**
blend best (tied)		13 (3) of 16			13 (2) of 16			13 (3) of 16	

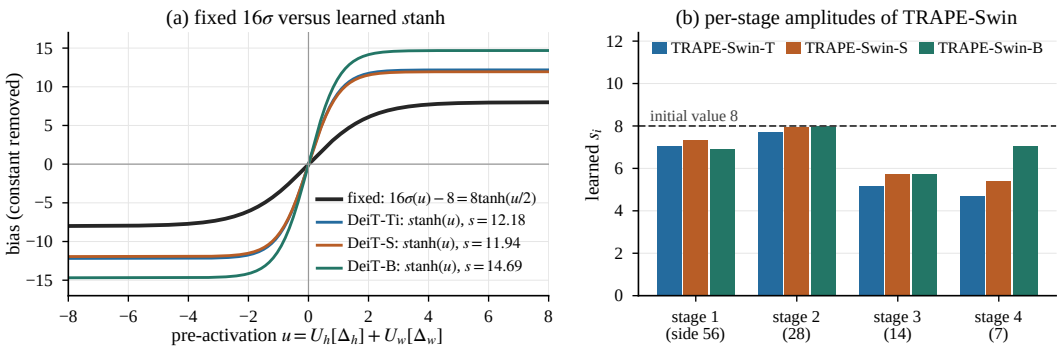


Figure 5: **Learned amplitudes. (a)** For a row in which the constant is added to every key, the fixed 16σ is a $\tanh$ with slope 4 at the origin; the single s learned by each TRAPE-DeiT model gives a 3.0–$3.7\times$ larger slope. **(b)** The per-stage amplitudes s_i of TRAPE-Swin stay below their initial value: a single-axis block applies $\tanh$ to one axis term instead of a sum of two.

Amplitude. Figure 5a plots the fixed squashing of row (b) in Table 1 next to the learned ones. By the identity $16\sigma(u) = 8 + 8\tanh(u/2)$, the fixed version is, up to a constant, a $\tanh$ with slope 4 at the origin, while the learned amplitudes give slopes of $12.18/11.94/14.69$, i.e. $3.05/2.99/3.67$ times larger. These are slopes of the scalar output maps, not of the full coordinate-to-bias functions, whose MLPs also differ. The learned s allows a bias range 49–84% wider than $[0, 16]$. On the 224^2 grid, the largest peak-to-peak bias of a head is 13.2, 14.3 and 19.2 for DeiT-Ti/S/B; the last exceeds the width 16 of the fixed range.

Axis responses and the central table. Figure 6 decomposes the raw axis responses into the continuous term and the central correction. The continuous term varies smoothly with the log coordinate, whereas E_a varies over its whole window $|\Delta| \leq 6$ rather than acting only at $\Delta{=}0$. Because E_a acts only inside this window, the composed axis table has a step at the window edge (up to 1.8 in DeiT-Ti and DeiT-S; one head of DeiT-B reaches 4.2). Row (d) of Table 1 includes this step; it leads the axial rows of that ablation. This discrete boundary is part of the model: continuity of f_a does not imply continuity of the corrected response when extended to real-valued offsets. Figure 7 shows the effect on the composed bias. Each correction enters the joint $\tanh$ together with the other axis, so it changes the bias along the bands $|\Delta_h| \leq 6$ or $|\Delta_w| \leq 6$ rather than on a $13{\times}13$ square. Setting $E_h{=}E_w{=}0$ in the trained weights changes the bias of the median head of blocks 0–10 by up to 1.8–2.0 logits, and that of the most affected head by up to 8.8–11.5 logits. This is a contrast within one trained model, not an accuracy ablation.

One function at every resolution. Figure 8 evaluates the same heads at 224^2, 512^2, 1024^2 and $320{\times}960$ from one parameter state. Every larger table contains the 224^2 table exactly at its centre (dashed squares), as stated by Eq. 5; the rectangle uses height and width tables of different lengths from the same functions. Outside the training offsets, the continuous functions extrapolate smoothly in the log coordinate. No anchor adaptation or tape is involved in this figure.

Rank and depth. For a single head, let u_h, u_w be column vectors of the two axis responses and $\mathbf{1}_h, \mathbf{1}_w$ the corresponding all-ones vectors. The offset surface $u_h\mathbf{1}_w^\top + \mathbf{1}_h u_w^\top$ has rank at most two before the nonlinearity; the elementwise $\tanh$ need not preserve this bound, so it does not make the expanded bias rank two. In 29 of the 231 heads of blocks 0–10, the raw range of one axis is below a quarter of the other, so these heads vary mainly along one direction. Figure 9 summarizes all heads over depth: median head ranges are largest in intermediate blocks and fall to 1.0–2.3 in blocks 9–10. Block 11 is a special case. Only its class-token output reaches the classifier, and the class-token query row carries no positional bias, so the positional parameters of block 11 receive no gradient: its central tables remain at their zero initialization, weight decay alone shrinks its axis functions, and $|b| < 10^{-9}$ for every head, which is negligible against float32 logits. Because class-token entries are zero, the mean level of the spatial bias is not softmax-invariant; a negative level, as in most blocks, shifts the attention of patch queries towards the class token.

Placement of the nonlinearity. Table 20 compares the joint squashing with an axis-wise one.

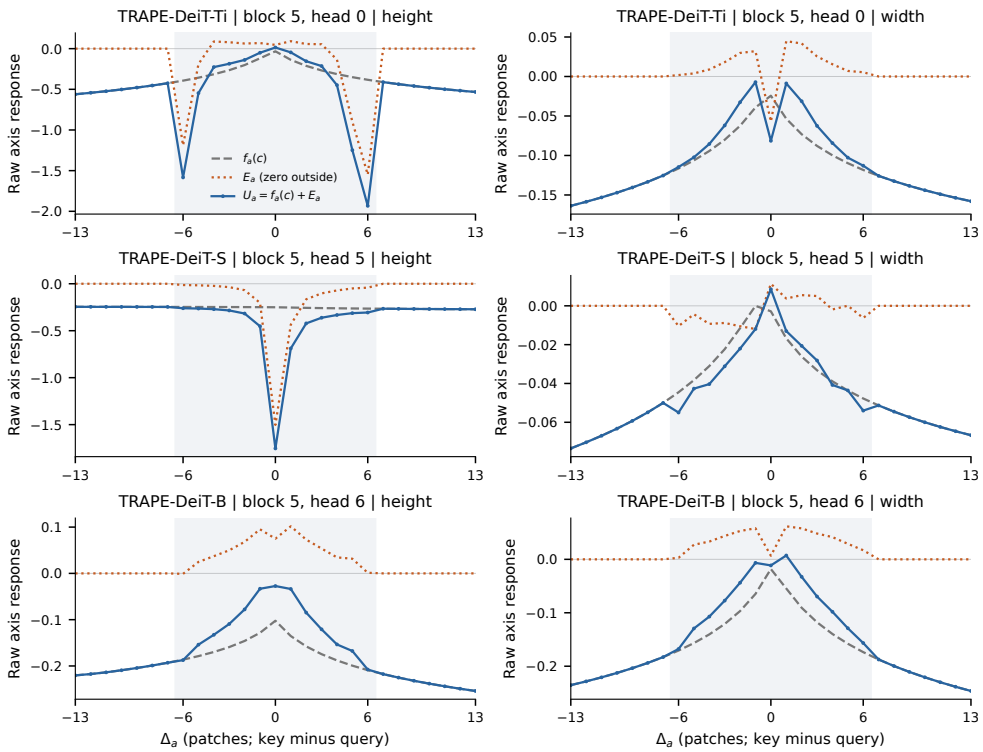


Figure 6: **Learned axis responses** of block 5 of the 224^2 TRAPE-DeiT checkpoints (no adaptation), for height (left) and width (right). Dashed: continuous term $f_a\big(c(\Delta; G_0)\big)$; dotted: zero-extended central table $e_a(\Delta)$; solid: raw response $U_a = f_a + e_a$ of Eq. 3, the quantity stored in the tape. Shading: the window $|\Delta| \leq 6$ of E_a.

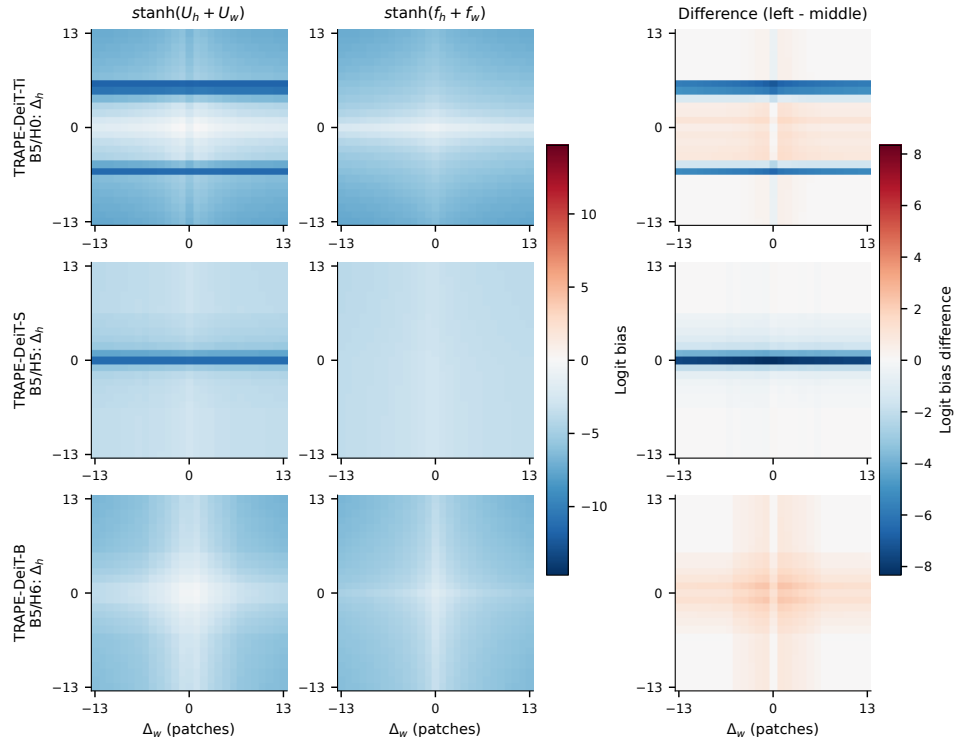


Figure 7: **Effect of the central tables on the composed bias** for the heads of Figure 6: $s\tanh(U_h{+}U_w)$ of the trained model (left), the same weights with $E_h{=}E_w{=}0$ (middle), and their difference (right, separate colour scale); offsets Δ_w (horizontal) and Δ_h (vertical) of the 224^2 grid.

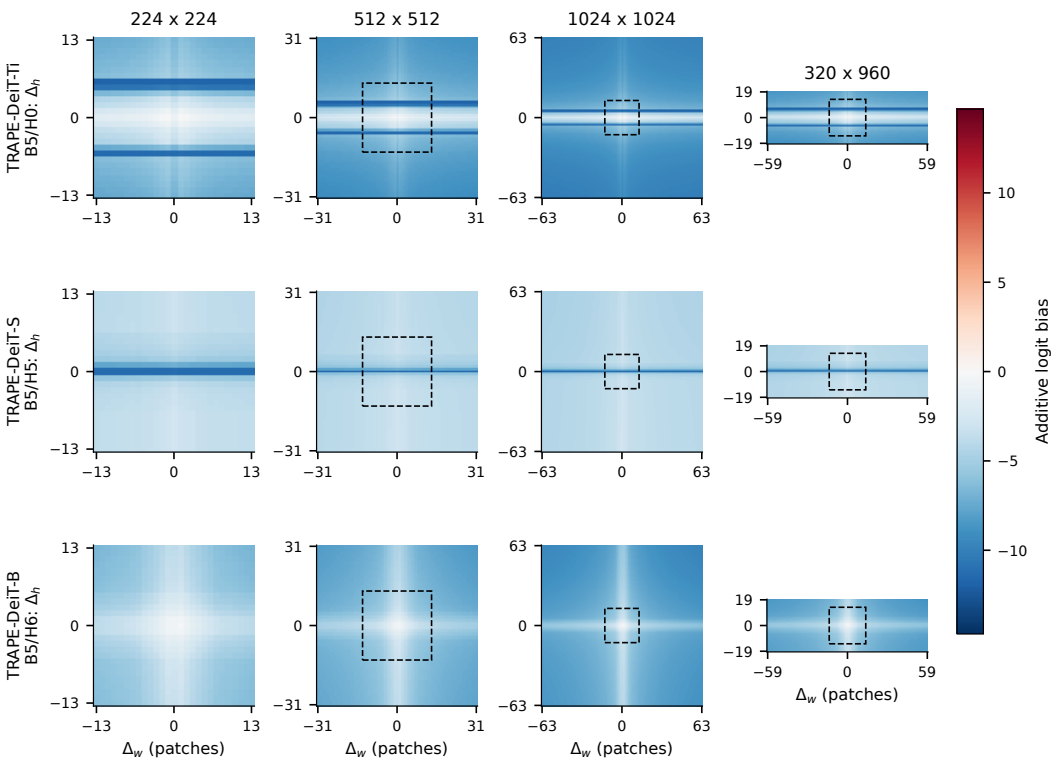


Figure 8: **One positional function at four input shapes.** Biases of the heads of Figure 6, generated without fine-tuning for 224^2, 512^2, 1024^2 and 320×960 inputs (token grids 14^2, 32^2, 64^2 and 20×60). Dashed squares: the offsets $|\Delta_h|, |\Delta_w| \leq 13$ of the 224^2 grid, whose table reappears exactly at the centre of every larger table.

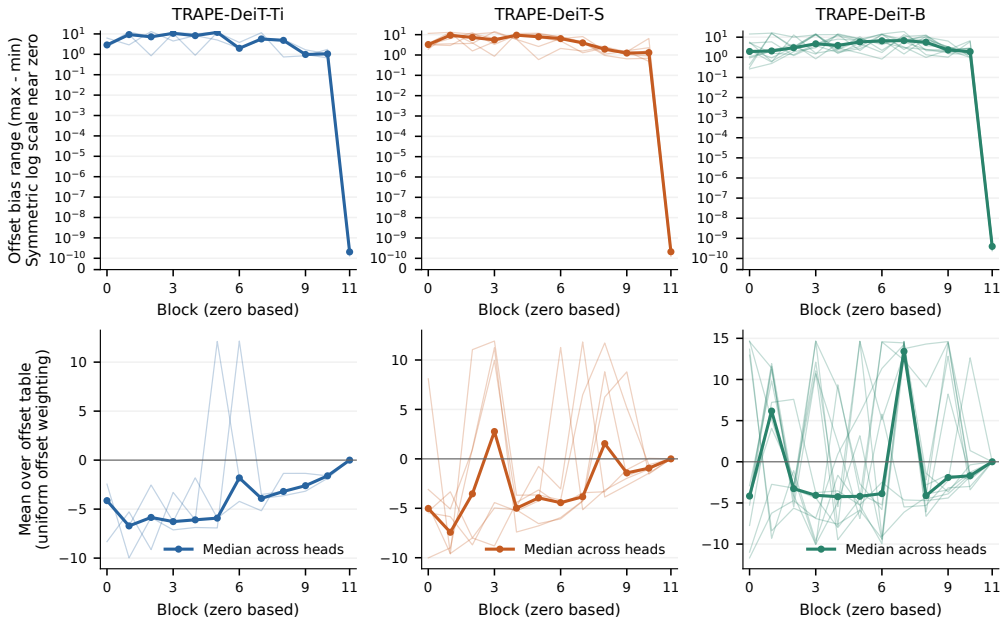


Figure 9: **The bias over depth**, all heads of the 224^2 TRAPE-DeiT checkpoints (thin lines) and their median (thick). Top: peak-to-peak range of each head's 27×27 offset table on a symmetric logarithmic axis. Bottom: mean over the 27×27 offsets, each offset weighted equally.

Table 20: **Cross-resolution generalization without fine-tuning: nonlinearity placement.** ImageNet-1k top-1 (%); TRAPE models trained only at 224^2. Axis-wise: $s\tanh U_h + s\tanh U_w$; joint: $s\tanh(U_h + U_w)$. Columns give test image sizes.

Model	Form	224^2	384^2	448^2	512^2	560^2
DeiT-Ti	$s\tanh(U_h + U_w)$ (default)	72.8	74.7	74.4	74.0	73.6
DeiT-Ti	$s\tanh(U_h) + s\tanh(U_w)$	72.8	74.5	74.5	74.1	73.7
DeiT-S	$s\tanh(U_h + U_w)$ (default)	80.6	81.8	81.7	81.1	80.7
DeiT-S	$s\tanh(U_h) + s\tanh(U_w)$	80.6	81.7	81.4	80.7	80.1

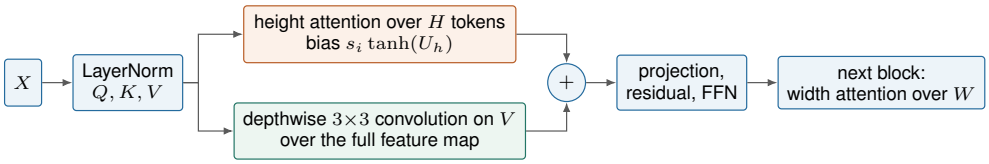


Figure 10: **The single-axis block of TRAPE-Swin.** A height block is shown; the next block exchanges the roles of H and W, so two blocks connect every pair of tokens at any resolution. There is no window partition, shift or mask; the four stages output features at strides $4, 8, 16, 32$, and each stage has its own amplitude s_i.

Table 21: **Single-component ablations on ADE20K** (UPerNet, 512×512 crops, 160k iterations, ImageNet-1k pre-training). Each row changes one component of Swin-T or of TRAPE-Swin-T, whose default uses $s_i\tanh$, $K{=}(55, 27, 13, 5)$ and a 3×3 depthwise branch on V; small numbers give the change from the unmodified model of the same group. 2D RoPE-Mixed: the rotary encoding of Heo et al. (2024) on queries and keys, replacing the bias. IN-1k: ImageNet-1k top-1 at 224^2; mIoU SS: single-scale mean intersection-over-union (%).

Backbone	Change	IN-1k (%)	mIoU SS
Swin-T (Liu et al., 2021)	none (published)	81.3	44.5
Swin-T	+ depthwise 3×3 branch on V	81.9 +0.6	45.8 +1.3
TRAPE-Swin-T	none (default)	81.5	**47.6**
TRAPE-Swin-T	− depthwise 3×3 branch on V	80.9 −0.6	47.1 −0.5
TRAPE-Swin-T	fixed $16\sigma(f_a)$, no E_a (Swin V2 style)	81.4 −0.1	46.6 −1.0
TRAPE-Swin-T	2D RoPE-Mixed instead of the TRAPE bias	81.6 +0.1	47.0 −0.6
TRAPE-Swin-T	depthwise 5×5	81.6 +0.1	47.1 −0.5
TRAPE-Swin-T	$K{=}(13, 13, 13, 5)$	81.4 −0.1	47.5 −0.1

I THE AXIAL BACKBONES

Architecture. With 224^2 pre-training the stage sides are $\{56, 28, 14, 7\}$, so $K = \{55, 27, 13, 5\}$ and $d_c = \{111, 55, 27, 13\}$. TRAPE-Swin-T/S use stage widths $96, 192, 384, 768$ and heads $3, 6, 12, 24$; TRAPE-Swin-B uses widths $128, 256, 512, 1024$ and heads $4, 8, 16, 32$. Depths are $(2, 2, 6, 2)$ for Tiny and $(2, 2, 18, 2)$ for Small and Base. The attention direction restarts with block parity in each stage, and the value convolution acts on the complete stage feature map (Figure 10). TRAPE-CSWin-S uses widths $64, 128, 256, 512$, depths $(2, 4, 32, 2)$ and heads $(2, 4, 8, 16)$, keeps the stem and merging layers of CSWin, and applies its local value convolution to the whole map. Both backbones keep every other architecture setting of the official Swin and CSWin models and are trained with their official classification and segmentation configurations (Appendix B).

Ablations. Table 21 changes one component at a time. Removing the depthwise value branch, with everything else unchanged, costs TRAPE-Swin-T 0.6 top-1 and 0.5 mIoU: at 47.1 mIoU it still exceeds Swin-T by 2.6 although its ImageNet accuracy is 0.4 lower, so the segmentation gain is not inherited from a stronger classifier. The same branch adds 1.3 mIoU to Swin-T but only 0.5 to TRAPE-Swin-T; these architecture-dependent effects are not additive, so the controls do not split the gain into separate contributions of axial attention and of the bias. The Swin V2-style variant generates each one-dimensional bias as $16\sigma(f_a(c))$ with the same coordinates and MLP widths, without central table and without learnable amplitude; attention, depthwise branch and training are unchanged. Within a single-axis block the identity $16\sigma(u) = 8 + 8\tanh(u/2)$ makes this a $\tanh$ of amplitude 8 and slope 4, whereas the default TRAPE-Swin-T learns $s_i = 7.03/7.70/5.19/4.69$ (slopes 1.2–1.9×

larger) and uses the central tables. The variant matches the default on ImageNet (81.4 against 81.5) and loses 1.0 mIoU, consistent with the encoding mattering most when the segmentation grids exceed the pre-training grids. Replacing the bias by 2D RoPE-Mixed (Heo et al., 2024) on queries and keys, with attention, depthwise branch and training unchanged, likewise matches the default on ImageNet (81.6 against 81.5) but loses 0.6 mIoU; it still exceeds Swin-T by 2.5 mIoU, so the axial backbone and the bias contribute jointly. Widening the depthwise kernel to 5×5 gains 0.1 top-1 and loses 0.5 mIoU, and shrinking the central tables of the first three stages to $K{=}13$ costs 0.1 on both metrics.

Receptive field and dense prediction. Two axial blocks connect all token pairs at any resolution, whereas two stage-1 Swin blocks cover at most 14 positions along an axis: 25% at 224^2 and about 11% at 512^2. Each TRAPE-Swin stage is globally contextualized after its first block pair, so the pyramid levels fused by UPerNet differ in granularity rather than in receptive field, with the depthwise branch adding local detail; classification pools the last map into one vector. This is a possible explanation for the larger segmentation gains, rather than a causal conclusion established by the present ablations.

Per-stage amplitudes. All twelve s_i of TRAPE-Swin-T/S/B end below their initial value of 8 (Figure 5b), unlike the single s of TRAPE-DeiT. This is consistent with, but does not establish, an effect of the different input of the nonlinearity: an axial block squashes one axis term, a plain ViT the sum of two.

Depthwise branch. The learned 3×3 value kernels are cross-shaped: the centre holds 0.17–0.36 of the absolute weight and each of the four neighbours 0.10–0.14, while the four diagonal taps together hold 0.14–0.33, less than the $4/9$ of a uniform kernel. These weight magnitudes indicate an emphasis on axial neighbors, although the nonzero diagonal taps still provide two-dimensional local mixing.

CSWin implementation variant. The evaluated CSWin-derived model is axial but does not implement Eq. 3 verbatim. For the current axis length g, define $c_{\mathrm{CS}}(\Delta; g) = \operatorname{sign}(\Delta)\log_2(1 + 8|\Delta|/\max(g-1,1))/\log_2 8$. Its reference therefore changes with g, and with $r_i = (K_i - 1)/2$ for the central table of stage i its raw response is

$$U_a^{\mathrm{CS}}(\Delta; g) = \begin{cases} E_a[\Delta + r_i, :], & |\Delta| \le r_i, \\ f_a(c_{\mathrm{CS}}(\Delta; g)), & |\Delta| > r_i, \end{cases} \tag{7}$$

so the table *replaces*, rather than adds to, the continuous output inside its window; its small-grid implementation centre-slices E_a when necessary. Because the coordinate depends on g, its tables do not satisfy Eq. 5: the adaptation code regenerates responses at the requested length from stored parameter states, blends parameters rather than tables, and disallows the fixed-reference table-only export. We report this variant as evaluated rather than reinterpreting its results as those of Eq. 3. It trades 0.5 ImageNet top-1 for $+0.5$ single-scale and $+0.6$ multi-scale mIoU, at 51 fewer GFLOPs and $1.09\times$ the measured throughput. Its baseline already factorizes attention into stripes, and attention, bias and the location of the value convolution change together, so the gain cannot be attributed to the bias alone. For attention products alone, a cross-shaped stripe block of stripe width w_s costs w_s times a single-axis block at the same dimensions; over CSWin-S this is 0.41 against 0.11 G at 224^2 and 48.9 against 13.5 G at 512×2048 (operation counts).

J Position-Only Fine-Tuning for Semantic Segmentation

Models and data. We start from the four UPerNet checkpoints of Table 4, trained for 160k iterations on 512^2 crops, so the 512^2 training scale plays the role of the pre-training size. The code splits the 20,210 ADE20K training images deterministically into 19,750 fitting, 300 selection and 160 gate images, disjoint for this adaptation (the original segmentation training may already have used all of them); the 2,000 validation images are used only for the final evaluation.

Adaptation. The anchors are square crops of 256, 352, 704, 992 and 1344 pixels, 0.5–2.625 times the training scale and approximately evenly spaced in the stage-1 coordinate; the 512^2 training scale keeps its original state. Each anchor is trained for 240 steps with global batch 16 on eight GPUs, using the model's own segmentation loss including the auxiliary head, AdamW without weight

decay, warm-up followed by a cosine decay, fp16 with gradient checkpointing, and the network in evaluation mode, which fixes dropout, drop path and batch normalization while gradients stay enabled for the trainable parameters. Only the output layers of the axis functions and the central tables are updated; first-layer hinge features, the backbone content, normalization, both decode heads and the stage amplitudes s_i are frozen, and capacity grows with the anchor as in Appendix B.3, with new coefficients starting at zero. Candidate states are compared on the 300 selection images, and the best one is kept only if it improves on the starting state. For TRAPE-Swin-T/B and the CSWin variant, a deployment gate on the 160 gate images then decides per anchor between the fitted state, at strength $1/2$ or 1, and the original state, maximizing the gate multi-scale mIoU subject to no loss of gate single-scale mIoU; this constraint is local to the gate images and does not imply a non-decreasing validation accuracy. It kept the anchors listed in Table 24. A side between two anchors is served by Eq. 6 for each axis and stage; amplitudes stay fixed.

The separately run TRAPE-Swin-S. The TRAPE-Swin-S adaptation (marked *) was run with a revision of the adaptation code, logged as mixed-mode training without the gate, with the same anchors, steps and batch; it keeps all five anchors and updates at most 72,012 parameters (0.0885%). That revision is not part of the archived code of the other three runs, so we report it separately and do not assert identical selection semantics; its evaluation pipelines are identical.

Budget. The runs process $5 \cdot 240 \cdot 16 = 19{,}200$ training crops, or $19{,}200/20{,}210 = 0.95$ passes over the ADE20K training set by image count, and 0.75% of the 2.56M crops of the 160k-iteration schedule. Updated parameters reach 36,012 (0.060%) for TRAPE-Swin-T, 96,016 (0.079%) for TRAPE-Swin-B and 20,456 (0.032%) for the CSWin variant (Table 24). These step and sample budgets exclude the images processed for candidate selection, the deployment gate and the final multi-protocol evaluation; the times of Table 24 add the per-anchor training and selection runs and exclude the gate and the final evaluation. Crop sizes and hence forward and backward costs vary strongly across anchors.

Evaluation protocols. All numbers use MMSegmentation on the 2,000 validation images with whole-image inference. The standard single-scale protocol (SS) resizes with `img_scale=(2048,512)` and kept aspect ratio, without flipping; the multi-scale protocol (MS) uses the ratios $\{0.5, 0.75, 1, 1.25, 1.5, 1.75\}$ with horizontal flips, resizing both the short-side target and the long-side cap, and averages the class probabilities at the original label resolution before the argmax. The same harness reproduces the SS values of all four checkpoints and the MS values of the three TRAPE-Swin checkpoints in Table 4, and it provides the MS value of the CSWin variant (52.05, rounded half up to 52.1; after adaptation 51.93). Wide MS adds the ratios 2 and 2.5; native evaluation only downsizes images whose long side exceeds 2048; Long-2048 resizes the long side to 2048 pixels, upsampling included, a stress test beyond the largest anchor. The scale sweep uses one ratio ρ at a time without flipping. Natural aspect ratios give different feature-grid sides along the two axes, so a short-side target that equals an anchor does not make a test image a fitted square. Adapted segmentation evaluations regenerate responses from stored anchor parameters. Within a common exported offset range, the fixed-reference Swin responses can equivalently be read from tables; sizes beyond that range require another export. The Long-2048 protocol also probes beyond the largest fitted crop. These experiments measure positional adaptation of the segmentation models; they do not use the classification tapes.

Results. Tables 22 and 23 list every measured scale and protocol with two decimals; no per-image statistics or repeated seeds are available, so no intervals are given. Below the training scale the changes lie between 0.00 and +0.70 mIoU, at the training scale between −0.21 and −0.07, and at 1.25× between −0.16 and +0.09. From 1.5× on, every backbone gains, and the gain grows with the scale, to +2.95 to +6.14 at 2.5× and +4.40 to +8.79 for Long-2048, as for the classification tapes of Table 2. The standard MS protocol changes by −0.28 to +0.22, and wide MS, which includes larger scales, by +0.05 to +0.91.

K Inference Cost of Table-Only Deployment

Setup. Every measurement runs on a single NVIDIA A800-SXM4-80GB GPU with PyTorch 2.6 (CUDA 12.4) and Triton 3.2, in bf16 with TF32 disabled, on synthetic inputs. The models are

Table 22: **Position-only fine-tuning for ADE20K: complete single-scale sweep.** mIoU (%) on all 2,000 validation images, whole-image inference without flipping; each cell is No-FT → PE-FT. The short side is resized to 512ρ pixels (long side at most 2048ρ, aspect ratio preserved), so test images keep their natural rectangles; ρ=1 is the 512^2 training scale. * Adapted with a revision of the code that has no deployment gate; ‡ an implementation variant whose bias is normalized by the current axis length (Appendix I), adapted by blending parameters rather than tables.

ρ	Short side	TRAPE-Swin-T	TRAPE-Swin-S*	TRAPE-Swin-B	TRAPE-CSWin-S‡
0.5	256	41.45 → 42.15	44.57 → 44.68	45.52 → 45.84	45.44 → 45.44
0.75	384	46.12 → 46.12	49.40 → 49.46	50.04 → 50.08	49.59 → 49.59
1	512	47.59 → 47.50	50.67 → 50.46	51.14 → 51.07	50.94 → 50.80
1.25	640	47.90 → 47.99	50.56 → 50.57	51.20 → 51.15	51.51 → 51.35
1.5	768	46.58 → 47.38	49.65 → 49.95	50.40 → 50.66	51.57 → 51.70
1.75	896	46.14 → 47.94	49.85 → 50.42	50.47 → 51.33	50.59 → 51.44
2	1024	44.53 → 47.30	48.11 → 49.87	49.22 → 50.61	49.17 → 50.66
2.5	1280	40.91 → 47.05	45.10 → 49.03	46.49 → 49.72	46.02 → 48.97

Table 23: **Position-only fine-tuning for ADE20K: every evaluation protocol.** mIoU (%) on all 2,000 validation images, whole-image inference; each cell is No-FT → PE-FT. SS/MS: the standard protocols of Table 4 (MS: ratios $\{0.5, 0.75, 1, 1.25, 1.5, 1.75\}$ with horizontal flips); wide MS adds ratios 2 and 2.5; native only downsizes images whose long side exceeds 2048; Long-2048 resizes the long side to 2048, upsampling included. * Adapted with a revision of the code that has no deployment gate; ‡ an implementation variant whose bias is normalized by the current axis length (Appendix I), adapted by blending parameters rather than tables.

Protocol	TRAPE-Swin-T	TRAPE-Swin-S*	TRAPE-Swin-B	TRAPE-CSWin-S‡
SS	47.59 → 47.50	50.67 → 50.46	51.14 → 51.07	50.94 → 50.80
MS	48.52 → 48.74	51.66 → 51.38	52.08 → 52.12	52.05 → 51.93
Wide MS	47.97 → 48.88	51.22 → 51.27	51.88 → 52.12	51.94 → 52.09
Native	46.30 → 46.32	49.71 → 49.62	50.75 → 50.66	50.14 → 50.07
Long-2048	36.49 → 45.28	41.34 → 47.54	43.20 → 48.20	41.78 → 46.18

Table 24: **Budget of position-only fine-tuning for ADE20K** (5 anchors × 240 steps at batch 16, eight GPUs). Trained: parameters at the largest anchor and their share of the model; Time: summed per-anchor training and selection, excluding the deployment gate and the final evaluation; Kept: anchors retained by the gate.

Model	**Trained**	**Share**	**Time**	**Kept anchors (pixels)**
TRAPE-Swin-T	36,012	0.06%	13.4 min	256, 704, 1344
TRAPE-Swin-S	72,012	0.09%	15.4 min	all five (no gate)
TRAPE-Swin-B	96,016	0.08%	16.4 min	256, 352, 704, 1344
TRAPE-CSWin-S	20,456	0.03%	18.2 min	704, 992, 1344

the TRAPE-DeiT-Ti/S/B model code used for training, imported unchanged; costs do not depend on the weight values, so the weights are those of the initialization. The tape is exported from the model's own positional parameters at the anchors $\{9, 14, 21, 30, 44, 64\}$, each stored up to the next, which gives the 131.6/263.3/526.5 KiB tapes of Section 3.3; a position-only fine-tuned tape has the same shapes and serving cost. Before a job starts, the GPU is checked to be idle, and every timed configuration first reproduces the logits of the training implementation on the same input.

Execution paths. *MLP/dense* is the training implementation: every forward pass rebuilds the log coordinates and the $HW \times HW$ index, evaluates the axis functions of every block, adds the central tables, composes $s\tanh(U_h \oplus U_w)$, gathers the $N \times N$ bias and runs an explicit softmax. *Tape/dense* serves the 2D tables of all blocks once per shape (Figure 3) and keeps the same attention operators. *MLP/fused* evaluates the axis functions in every forward pass but reads the composed table inside the fused kernel of Appendix A.4. *Tape/fused* is table-only deployment: nothing positional runs in the forward pass. The four paths share weights, inputs and all non-positional operators.

Protocol and repetitions. Throughput uses a batch that keeps the number of tokens roughly constant, $\lfloor b \cdot 224^2/(R_h R_w) \rfloor$ with $b = 128/64/32$ for Ti/S/B, 10 warm-up and 30 timed forward passes; latency uses batch 1, 20 warm-up and 100 timed passes; each value is the median of the timed

Table 25: **Throughput and batch-1 latency of TRAPE-DeiT, every measured shape** (one A800 GPU, bf16; mean ± s.d. over three independent runs, each the median of 30 timed forward passes for throughput and 100 for latency, after 10 and 20 warm-up passes). B: throughput batch, $\lfloor b \cdot 224^2/(R_h R_w) \rfloor$ with $b = 128/64/32$ for Ti/S/B, so every batch holds about the same number of tokens. MLP: axis functions evaluated in every forward; tape: tables served once per shape; dense: explicit attention over the gathered $N \times N$ bias; fused: Triton kernel reading the compact 2D table. Every configuration first reproduced the logits of MLP/dense on the same input.

			Throughput (img/s)				**Latency**, batch 1 (ms)			
Model	**Input**	B	MLP dense	tape dense	MLP fused	tape fused	MLP dense	tape dense	MLP fused	tape fused
DeiT-Ti	224^2	128	9466 ±6	9905 ±19	13453 ±321	17597 ±10	8.76 ±0.25	5.80 ±0.12	9.27 ±0.16	5.49 ±0.10
	384^2	43	2270 ±1	2349 ±1	4451 ±118	5520 ±3	8.87 ±0.16	5.73 ±0.12	9.40 ±0.19	5.46 ±0.09
	512^2	24	580 ±0	589 ±0	2474 ±35	2783 ±1	8.84 ±0.13	5.70 ±0.09	9.32 ±0.16	5.41 ±0.09
	704^2	12	214 ±0	216 ±0	1094 ±0	1181 ±0	9.71 ±0.01	8.85 ±0.00	9.38 ±0.15	5.44 ±0.09
	1024^2	6	56.2 ±0.0	56.6 ±0.0	388 ±0	409 ±0	31.89 ±0.00	31.17 ±0.01	9.42 ±0.18	5.45 ±0.11
	320×960	20	452 ±0	459 ±0	2031 ±30	2277 ±3	9.03 ±0.09	5.74 ±0.08	9.50 ±0.12	5.45 ±0.09
	448×672	21	465 ±0	473 ±0	2131 ±40	2361 ±1	9.05 ±0.13	5.76 ±0.10	9.50 ±0.12	5.42 ±0.08
DeiT-S	224^2	64	4679 ±7	4881 ±10	6773 ±64	8681 ±7	8.86 ±0.15	5.81 ±0.08	9.34 ±0.08	5.48 ±0.06
	384^2	21	1104 ±0	1143 ±0	2189 ±11	2699 ±5	8.99 ±0.08	5.78 ±0.04	9.50 ±0.11	5.49 ±0.06
	512^2	12	280 ±0	285 ±0	1231 ±24	1370 ±0	8.92 ±0.03	6.14 ±0.00	9.42 ±0.03	5.39 ±0.10
	704^2	6	99.6 ±0.0	101 ±0	538 ±0	582 ±0	17.52 ±0.00	16.74 ±0.00	9.29 ±0.15	5.36 ±0.10
	1024^2	3	24.0 ±0.0	24.2 ±0.0	191 ±0	202 ±0	63.68 ±0.00	62.93 ±0.00	9.41 ±0.15	5.86 ±0.00
	320×960	10	217 ±0	221 ±0	1022 ±9	1126 ±1	9.14 ±0.11	7.79 ±0.01	9.40 ±0.12	5.33 ±0.07
	448×672	10	222 ±0	226 ±0	1022 ±10	1093 ±1	9.03 ±0.04	7.65 ±0.01	9.32 ±0.05	5.30 ±0.07
DeiT-B	224^2	32	2046 ±3	2127 ±2	3188 ±8	3448 ±5	8.66 ±0.06	5.74 ±0.04	9.17 ±0.05	5.39 ±0.05
	384^2	10	482 ±1	497 ±0	972 ±3	1056 ±1	8.84 ±0.04	5.70 ±0.02	9.30 ±0.06	5.37 ±0.02
	512^2	6	129 ±0	131 ±0	526 ±1	565 ±1	12.58 ±0.00	11.77 ±0.00	9.24 ±0.08	5.33 ±0.05
	704^2	3	41.3 ±0.0	41.7 ±0.0	231 ±0	247 ±0	36.68 ±0.02	35.99 ±0.01	9.27 ±0.08	5.36 ±0.06
	1024^2	1	7.1 ±0.0	7.1 ±0.0	83.6 ±0.1	89.3 ±0.1	141.65 ±0.07	140.87 ±0.01	11.98 ±0.00	11.19 ±0.01
	320×960	5	97.5 ±0.0	98.7 ±0.1	433 ±1	464 ±1	16.91 ±0.01	16.01 ±0.00	9.43 ±0.08	5.38 ±0.06
	448×672	5	98.9 ±0.1	100 ±0	435 ±1	469 ±3	16.97 ±0.02	16.06 ±0.00	9.39 ±0.06	5.35 ±0.01

Table 26: **Positional work, preparation and memory of TRAPE-DeiT** (one A800 GPU, bf16; means over three runs of the median of 100 timed calls; s.d. at most 0.09 ms). Per forward, all 12 blocks: positional work repeated in every forward pass by each path (MLP/dense: coordinates, axis functions, composition and the $N \times N$ gather; tape/dense: the gather only; MLP/fused: coordinates, axis functions and composition; tape/fused: none). Once per shape: serving the 2D tables of all blocks from the tape, and the complete preparation for the fused kernel. Memory: kept 2D tables of all blocks (fused) against the bf16 $N \times N$ bias of one block (dense). Tape: fp32, six anchors.

		Per forward (ms)			**Once per shape** (ms)		**Memory** (MiB)		**Tape**
Model	**Input**	MLP dense	tape dense	MLP fused	serve tables	prepare fused	2D tables all blocks	$N \times N$ bias one block	(KiB)
DeiT-Ti	224^2	3.61	0.58	3.01	0.06	0.25	0.2	0.2	131.6
	384^2	3.82	0.58	3.11	0.13	0.31	0.6	1.9	
	512^2	3.86	0.90	3.12	0.13	0.31	1.1	6.0	
	704^2	4.15	3.11	3.12	0.06	0.24	2.1	21.5	
	1024^2	15.15	14.10	3.11	0.06	0.25	4.5	96.0	
	320×960	3.99	1.17	3.21	0.13	0.31	1.3	8.3	
	448×672	3.98	1.11	3.21	0.13	0.31	1.3	7.9	
DeiT-S	224^2	3.58	0.57	2.99	0.06	0.24	0.4	0.4	263.3
	384^2	3.83	0.67	3.13	0.13	0.31	1.2	3.8	
	512^2	3.91	1.73	3.13	0.13	0.31	2.2	12.0	
	704^2	7.99	6.98	3.13	0.06	0.25	4.2	42.9	
	1024^2	32.92	31.88	3.13	0.06	0.25	8.9	192.1	
	320×960	4.07	2.28	3.21	0.13	0.31	2.6	16.5	
	448×672	4.04	2.22	3.20	0.13	0.31	2.5	15.9	
DeiT-B	224^2	3.58	0.58	3.00	0.06	0.25	0.8	0.9	526.5
	384^2	3.87	1.09	3.13	0.13	0.31	2.4	7.6	
	512^2	4.68	3.68	3.13	0.13	0.31	4.4	24.0	
	704^2	18.63	17.59	3.12	0.06	0.24	8.3	85.9	
	1024^2	79.70	78.66	3.12	0.06	0.25	17.8	384.2	
	320×960	6.96	5.82	3.22	0.13	0.31	5.1	33.0	
	448×672	7.13	5.97	3.22	0.13	0.31	5.0	31.7	

Table 27: **Largest batch on one** 80 **GB A800 GPU** (bf16), found by doubling and bisection to within 3%, and the throughput at that batch (mean ± s.d. over three runs; the largest batch is identical in all runs).

		MLP, dense (training implementation)		**tape, fused** (deployment)	
Model	**Input**	max batch	img/s	max batch	img/s
DeiT-S	704^2	800	119.3 ±0.0	4,352	689.9 ±0.8
DeiT-S	1024^2	192	32.4 ±0.0	2,048	229.9 ±1.2
DeiT-B	704^2	408	57.5 ±0.0	2,304	292.2 ±0.7
DeiT-B	1024^2	96	15.8 ±0.0	1,088	102.2 ±0.1

passes, measured with CUDA events. Positional work and per-shape preparation are medians of 100 calls. The largest batch is found by doubling and bisection to within 3% and timed with 5 passes. The whole benchmark was run three times independently, and Tables 5, 25, 26 and 27 report the mean and sample standard deviation over the three runs. Over all 168 timed cells the median ratio of standard deviation to mean is 0.15% and the largest 2.8% (batch-1 latency of DeiT-Ti at 224^2, which is dominated by kernel launches); peak memory and the largest batches agree exactly across runs.

Correctness. For all three models, at 224^2, 320×960 and 704^2, in fp32 and bf16 and with both kernel variants, the served tables match direct evaluation of the axis functions to 7.5×10^{-9} in fp32, and every path reproduces the logits of the training implementation to 4.5×10^{-6} in fp32 and to 2.7×10^{-2} in bf16, within the bf16 tolerance of the test.

Results. Reading the tape removes the per-forward positional work of the training implementation (Table 26): 3.0–3.2 ms for the axis functions of all 12 blocks at every input size, plus, with dense attention, a gather of the $N\times N$ bias that grows to 14–79 ms at 1024^2. Serving the tables of a new shape costs 0.06–0.13 ms, and the complete preparation for the fused kernel, including the per-token codes, 0.24–0.31 ms. At batch 1 the forward pass is dominated by kernel launches, so these milliseconds lower latency by 1.61–1.76× with the fused kernel for every model and shape except DeiT-B at 1024^2 (1.07×), where computation dominates; with dense attention the reduction is 1.51–1.56× up to 384^2 and vanishes as the gather dominates. At the throughput batch, skipping the axis functions raises throughput by 5–31% with the fused kernel and by at most 5% with dense attention (Table 25). The fused kernel changes how the cost scales with the input: relative to the training implementation, the deployed path raises throughput by 1.7–12.6× and lowers batch-1 latency by 1.6–12.7×; its peak memory at the throughput batch stays at 0.16–0.29 GiB for every shape, whereas explicit attention needs up to 1.84 GiB, and the largest batch on one GPU grows 5.4–11.3× (Table 27).

Scope. These measurements compare execution paths of the same TRAPE model, so the differences isolate the positional computation and the attention execution. They do not compare TRAPE with encodings that need no attention bias and can therefore use bias-free kernels such as FlashAttention-2, and the FlexAttention sketches of Appendix A.4 were checked for correctness but not timed.