A. More Literature Review

Efficient ViTs. As previously mentioned in Sec. 2, efficient attention can be roughly categorized into two groups: local attention [2, 38, 56, 58] or kernel-based linear attention [1, 5, 6, 14, 29, 37, 39, 43, 66]. For local attention, Swin [38] restricts the window size of self-attention, so that only neighboring tokens will perform similarity measurements each other instead of all tokens; MaxViT [56] also adopts block attention within windows but additionally takes dilated global attention into account for learning both local and global information; QnA [2] shares the attention queries among all tokens; Linformer [58] approximates the queries and keys with low-rank factorization to reduce their vector length. For kernel-based linear attention, XCiT [1] proposes a "transposed" version of self-attention that operates across feature channels rather than tokens, resulting in linear complexity with the number of tokens; Linformer [58] explores a low-rank matrix to approximate the self-attention; Reformer [32] replaces self-attention by one that uses locality-sensitive hashing, changing its complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\log(n))$ where n denotes the number of tokens; Longformer [3] combines a windowed local-context self-attention and a task-motivated global attention that encodes inductive bias about that task; Nystromformer [66] adapts the Nystrom method to approximate standard self-attention with $\mathcal{O}(n)$ complexity; Scatterbrain [8] unifies both low-rank approximation and sparse attention to improve accuracy-efficiency tradeoffs. Different from all above works, we explore from a new perspective by taking spectral angles into consideration when measuring the similarity among tokens, resulting in linear-angular attention with sparse training techniques that can achieve comparable or even better performance than softmax-based attention.

ViTs for Downstream Tasks. Apart from image classification tasks, ViTs have also been leveraged to serve as backbones for downstream tasks, such as object detection [7,34,60,74,77] and semantic segmentation [11,12,42]. For example, DETR [7] directly detects and predicts objects by combining a common CNN with a transformer architecture; Maskformer [12] proposes to use a simple mask classification model to predict a set of binary masks, each associated with a single global class label prediction. One big difference is that ViTs can beat CNNs on classification tasks that have a lower image resolution while are still less efficient than lightweight CNNs on downstream tasks that heavily rely on multi-scale resolution features. Therefore, there have been various debates on designing powerful ViT backbones: (1) *plain ViTs (e.g., ViTDet [33]) or hierarchical ViTs (e.g., MViTv2 [34], or Swin [38])?* Plain ViTs win in terms of simplicity but could be hard to scale down to lower resolution and computation regimes; Hierarchical ViTs seamlessly match with feature pyramid networks (FPNs) for extracting multi-scale feature maps but have more design factors to be considered or searched over. (2) *pure ViTs or hybrid CNN-ViTs?* Pure ViTs are compatible with self-supervised masked autoencoder (MAE) pretraining [25]; Hybrid CNN-ViTs can suffer from the information leakage problem [22] when adopting MAE pretraining, while being more efficient especially for feature extractions in early layers. Our proposed method does not fall into the aforementioned debates. Instead, it is compatible with all ViT variants relying on the softmax-based attention.

B. More Results and Clarification

Improvement from Our Training Recipe. Recall that in Sec. 4, we conduct experiments on three classical computer vision tasks. For object detection and semantic segmentation, we follow the baseline's training recipe for a fair and direct comparison. For the image classification, our training recipe has a minor difference due to the increased batch size and training epochs with more GPU nodes. As such, we further provide the detailed improvement breakdown here. Specifically, our adopted training recipe leads to $\uparrow 0.2\% \sim \uparrow 1.6\%$ top-1 accuracy improvements and our Caslting-ViT further reduces up to $\downarrow 40\%$ MACs and increases $\uparrow 0.1\% \sim \uparrow 1.2\%$ top-1 accuracy simultaneously.

Ablation Studies on Image Classification. Our ablation studies are mostly done on the detection task as shown in Sec. 4.4 because of its less training time as compared to training ImageNet. Note that for these ablation studies, we do not adopt pretraining on ImageNet as specified in Sec. 4.4. After finishing the trial-and-error and when it comes to comparing with SOTA works, we then pretrain final models with the training recipe the same as LeViT [24], resulting in final results in Tab. 3. According to our experiments, training ImageNet takes nearly one week, while training COCO without pretraining on ImageNet takes only one day. In fact, ablation results on the classification task are consistent. To deliver more comprehensive ablation studies, we train Castling-LeViT-256 on ImageNet afterwards and find that: (1) + Lin.: 81.5%; (2) + Lin. & DWConv: 82.4%; (3) + Lin. & DWConv & SparseAttn: 82.6%, those results are consistent with our observation on detection experiments.

Conjecture of Why Linear-Angular Attention Sometimes Beats the Original Self-Attention. To better understand why the result of our Castling-ViT is even better than softmax-based ViTs. We summarize three differences between our method and previous linear attentions: (1) In addition to linear attention, we also take DWConv and sparse softmax-based attention into the training process; (2) We use a SGD optimizer instead of Adam, which is not common for training ViTs. Although Adam optimizer leads to faster convergence, we find that SGD optimizer helps to deliver better results if being trained sufficiently converged, e.g., we train 1000 epochs on ImageNet; (3) After revisit-

Table 8. Throughputs/memory measurements on a V100 for image classification, under various input resolutions denoted as $r \times r$.

Model	Throughputs (Images/s)			GPU Peak Memory (MB)		
	r = 512	r = 1024	r = 1536	r = 512	r = 1024	r = 1536
DeiT-Base	40	6	OOM	1220	12369	OOM
Castling-DeiT-Base	48 (↑20%)	8 (↑33%)	6	998 (↓18%)	4863 (↓61%)	15478
MViTv2-Base	43	5	OOM	1762	14686	OOM
Castling-MViTv2-Base	50 (↑16%)	10 (↑100%)	4	1483 (↓16%)	7028 (↓52%)	16028

Table 9. Ablation study on the patch size (measured on a V100).

Models	Throughputs (Images/s) under p patch sizes		
	p = 8	p = 4	p = 2
DeiT-Tiny	398	40	3
Castling-DeiT-Tiny	410 (↑1.0×)	103 (↑2.6×)	20 (↑6.7×)
DeiT-Base	60	8	OOM
Castling-DeiT-Base	64 (↑1.1×)	15 (↑1.9×)	4

Table 10. Latency measurements on a V100 for object detection.

Models	Params (M)	MACs (G)	mAP	Latency (ms)
YOLOv5-S	7.3	17.1	36.7	8.7 [23]
RetinaNet+PVT-Tiny [59]	23.0	221	36.7	-
Castling-ViT-L-416	13.1	5.3	37.3	3.9 (↓55.2%)

ing the attention design, we remove token/feature pooling and adopt post-$\mathbf{Q}$ pooling and residual connections [34] in our attention blocks. All above three differences contribute to the the final accuracy apart from the improvement of using linear-angular attention. We also show the breakdown analysis for each of these three points, see Sec. 4.4, Sec. B, and Sec. 3.1 for detailed analysis, respectively.

Actual Latency, Throughputs, and Memory Measurements. Our final models are dense and thus well compatible with GPUs. We measure and report the latency (↓55%), throughputs (↑16 $\sim$ ↑100%), and GPU memory (↓16 $\sim$ ↓61%) for both classification and detection tasks, as shown in Tab. 8/10. For throughputs, we measure both our Castling-ViT and baselines under their maximum allowed batch sizes (bs), i.e., bs=16/2/1, for different input resolutions r=512/1024/1536 in a fair and consistent V100 environment. Note that when the input resolution r=224, our models cannot beat the baseline in terms of throughputs because of (1) the newly added DWConv; (2) the removal of token/feature pooling. However, in terms of accuracy-efficiency tradeoffs, our Castling-ViT consistently beats all baselines as shown in Sec. 4. For memory, we record the peak memory per image. For latency, we benchmark with SOTA CNN-based detectors. Our model achieves 37.3mAP at 3.9ms latency on a V100, while YOLOv5-S only achieves 36.7mAP at 8.7ms latency). Moreover, Castling-ViT wins more throughputs (up to ↑6.7×) for smaller patch sizes and/or larger input resolutions, as shown in Tab. 9 and 8. Note that we record CUDA latency following the literature [38, 54]. All reported results are averaged among three runs.

Compare with ViT-based Baselines on Detection. We benchmark with SOTA CNN-based detectors under 6G MACs in Sec. 4 because that ViT-based detectors are too expensive. For example, our Castling-ViT achieves 37.3mAP at 5.3G MACs, while RetinaNet+PVT-Tiny only achieves 36.7mAP at even 221G MACs [59], as shown in 10.

Advantages of Angular Kernels? Angular kernels take into account extra spectral characteristics and enjoy good properties, e.g., positive semi-definite function $\rightarrow$ inner product in a high-dimensional and rich feature space, as analyzed in Sec. 3.2. It also achieves comparable accuracy with vanilla attention as validated by Sec. 4.

Large-Scale Ablation Studies on Attention Design. We use small ViTs for idea validation in Tab. 5 and the conclusion generalizes to larger ones. Here we add another ablation study on a larger model LeViT-384 as shown in Tab. 11, from which we see that the attention design insights consistently generalize from small models to larger models, further validating our design insights.

Why More Parameters Than Others in Low MACs? ViTs tend to have more parameters than CNNs under small MACs, e.g., LeViT [24] and Efficient-ViT [6]. For the LeViT, it features more layers with gradually downsampled input resolutions. For example, LeViT-256 requires 18.9M parameters at only 1.1G MACs, LeViT-384 requires 39.1M parameters at 2.4G MACs. Since we adopt LeViT-like structure to construct our Castling-ViT on image classification tasks, the parameter looks higher than other else baselines. Also, as indicated in Sec. 3.1, Castling-LeViT uses merely post-$\mathbf{Q}$ pooling, causing slightly higher hidden dimensions for $\mathbf{Q}$/$\mathbf{K}$ than LeViT. In this work, we focus more on the FLOPs/latency instead of parameters since storage is not a major concern in modern hardware [48].

Will Auxiliary Attention and DWConv Work for Existing Linear Attentions? Yes, we train a DeiT-Tiny (w/o distill.; Acc.: 72.2%) w/ linear attention [17] for 300 epochs and observe that: (1) + Lin.: 68.3%; (2) + Lin. & DWConv: 71.7%; (3) + Lin. & SparseAttn: 70.2%; (4) + Lin. & DWConv & SparseAttn: 72.4%.

Clarify ReLU-S vs. ReLU-E in Tab. 5. During approximation, i.e., $\text{Sim}(\mathbf{Q}, \mathbf{K}) \approx \phi(\mathbf{Q})\phi(\mathbf{K})^T$, both of them use ReLU as $\phi(\cdot)$, but ReLU-S takes the whole Softmax as $\text{Sim}(\cdot)$, while ReLU-E takes the $\text{Exp}(\cdot)$ as $\text{Sim}(\cdot)$, e.g.,

Table 11. Additional ablation studies on attention designs using LeViT-384 on ImageNet.

Pooling				Residual Q	MACs (G)	Top-1 Accuracy (%)
Token	Feat.	Pre-Q	Post-Q			
✓		✓			2.50	82.63
	✓	✓			2.36	82.55
		✓			2.61	82.65
✓			✓	✓	2.83	82.19
	✓		✓	✓	2.80	81.86
			✓	✓	2.96	83.65

Efficient-ViT, resulting in additional divisions.