Bi-Directional Self-Attention for Vision Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Self-Attention (SA) maps a set of key-value pairs to an output by aggregating information from each pair according to its compatibility with a query. This allows SA to aggregate surrounding context (represented by key-value pairs) around a specific source (e.g. a query). Critically however, this process cannot also *refine* a source (e.g. a query) based on the surrounding context (e.g. key-value pairs). We address this limitation by inverting the way key-value pairs and queries are processed. We propose Inverse Self-Attention (ISA), which instead maps a query (source) to an output based on its compatibility with a set of key-value pairs (scene). Leveraging the inherent complementary nature of ISA and SA, we further propose Bi-directional Self-Attention (BiSA), an attention layer that couples SA and ISA by convexly combining their outputs. BiSA can be easily adapted into any existing transformer architecture to improve the expressibility at any (or all) attention layers. We showcase this flexibility by extensively studying the effects of BiSA on CIFAR100 [1], ImageNet1K [2], and ADE20K [3], and extend the Swin Transformer [4] and LeViT [5] with BiSA, and observe substantial improvements.

1 Introduction

Self-Attention has become one of the most prevalent mechanisms in machine learning. [6] proposed using it to alleviate the autoregressive limitations burdening preceding frameworks [7, 8, 9]. Self-attention based architectures have since become dominant for solving natural language processing problems (NLP) [10, 11, 12, 13, 14]. In the same breath, self-attention architectures have also achieved significant success within the computer vision domain, exhibiting comparable if not better performance than rival convolutional neural networks (CNNs) across many benchmarks [15, 16, 5, 17, 18, 19, 20, 21, 22, 23, 24].

Given the success of self-attention across a multitude of benchmarks spanning different modalities, it is no surprise that a significant amount of research has been devoted to improving the attention mechanism. These efforts can be roughly separated into two categories: works that augment the type of information input to self-attention, and works that alter its architecture. The works which augment the information have studied the effects of different positional encodings on a diverse set of vision and language problems [25, 6, 26, 25], ways to augment the types of tokens input to self-attention [27, 23, 15, 20, 24, 21], and ways to filter the set of tokens input to the attention module [4, 17, 28, 29, 30]. On the other hand, methods that modify the architecture of the attention mechanism seek to increase representational capacity and improve efficiency and/or robustness [19, 31, 32, 22, 32].

Self-attention — and by extension each of these methods — maps a set of key-value pairs to an output based on their compatibility to a query representation. This makes such approaches particularly well suited for selectively aggregating contextual information (e.g. represented by values) from an input around a particular source of interest (e.g., a query) based on importance (e.g., individual key-query alignments). For instance, in the popular task of image classification, methods [16, 20, 19] have

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

shown how self-attention consistently is able to selectively aggregate over regions of interest (e.g. patches) in an image while filtering out spurious background information. However, these methods are not suited for modeling the converse: understanding the importance of source features (e.g., queries) according to surrounding context (e.g., keys-values). That is, transforming a query to a single output according to its compatibility to a key-value set. This would allow transformers to determine what kind of information to extract from individual tokens (queries) based on surrounding context (key-values), where distinct tokens can be processed differently (e.g., emphasizing boundaries between element types) based on their spatial locations. For instance, elements within differing semantic segmentation classes may be emphasized based on the type of class they belong to. Moreover, as such transformations principally involve refining local features (e.g., individual queries), such a framework would be ideally placed early within a transformer architecture.

To this end, we propose a novel attention mechanism, **I**nverse **S**elf-**A**ttention (ISA), which explicitly transforms a query to an output vector based on the key-value pair set. This enables ISA to reason over single instances based on their perceived importance to a broader context. Notably, ISA is by design complementary to self-attention, as it inverses how its attention components interact with one another when computing outputs. Leveraging this compatibility, we further propose a novel attention mechanism that couples (e.g. through a convex combination of their outputs) self-attention and ISA into a single cohesive unit. We term this framework **Bi**-directional **S**elf-**A**ttention (BiSA), and empirically demonstrate its applicability by substantially improving baseline model performance on CIFAR100 [1], ImageNet1K [2] and ADE20K [3].

2 Related work

Computer vision transformers. Inspired by the impact of self-attention in NLP, researchers have sought to leverage the power of these modules in computer vision. The seminal Vision Transformer (ViT) [16] applies a Bert-like [10] architecture on discrete non-overlapping image patches for classification tasks. [18] explores improving the training efficiency of ViTs through knowledge distillation. There has been substantial efforts to bridge the gap between the inductive biases obtained through convolutional operations present in Convolution Neural Networks (CNNs), and the information pooling in self-attention networks. [24] enabling multi-scale feature processing by incorporating a pyramidal structure into the transformer, resembling the effects of CNN backbones [33]. In a complementary direction, [4] mimics convolutional operations through Sliding Window Self-Attention (SWSA). Following this work [20] alters the resolution within windows through coarse and fine-grained transformations over regions deviating from the window center. These methods [24, 4, 20] additionally lower the quadratic time complexity in feature space.

Self-Attention variants. Given the success and broad applicability of the self-attention framework, it is no surprise that substantial efforts has been devoted to adapting it for downstream tasks. One avenue of research investigates modifying inputs to the module. [25] explored using relative positional encodings rather than frequency-based measures originally proposed by [6]. [26] furthered this idea with conditional positional encodings. [27] progressively refines visual patch tokens through patch token-to-token pooling. [23, 15, 20] present input tokens at varying resolutions to the attention heads. [30] adaptively chooses which image patches to feed into a self-attention module based on relevance to an area of interest. A second branch of research has explored introducing convolutional operations to the self-attention framework. [29] proposes a sliding window framework within the text sequence domain, while [4] presents a variant for image processing. [28] integrates self-attention within convolutional sliding windows by only applying it on the center of each receptive field. Still more work has been dedicated to exploring efficiency within strict window sizes [21, 24, 23, 17]. Another equally critical direction of work has explored modifying the self-attention layer itself. [19] enriches feature channel capacity through the addition of a static "ghost" attention head preserved through attention steps. [31] presents a unifying framework for both convolutional filters and self-attention, and proposes a specialized module that incorporates both. [22] replaces the pairwise dot-product self-attention mechanism with dense and random projections. [32] increases the representational capacity of self-attention by projecting components to higher dimensional spaces.

Bi-directional Self-Attention. These attention mechanisms improve on aspects of self-attention while retaining the framework's original interactions between input components (e.g. queries, keys and values). Specifically, like self-attention, these models map a set of key-value pairs to an output

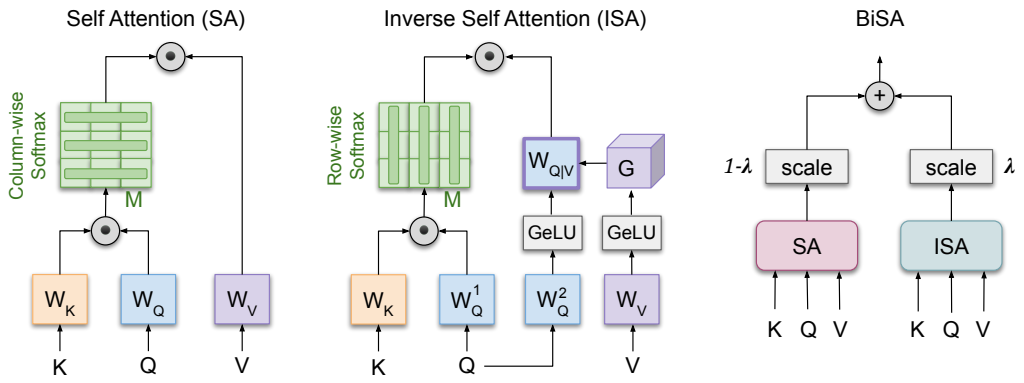

Figure 1: Illustration of the bi-directional self-attention (BiSA) framework. In the self-attention segment, keys (K) and queries (Q) are projected via query($\mathbf{W_Q}$) and key($\mathbf{W_K}$) projection matrices to form the canonical score matrix(M). We then take the dot product of the column-wise softmax of this score matrix and the global value projections to compute the output. In the ISA branch, the row-wise softmax of M is computed and multiplied with the projected queries along $\mathbf{W_{Q|V}}$. $\mathbf{W_{Q|V}}$ is obtained via a hypernetwork [34] on the values (V). The outputs of the two branches are then convexly combined according to a learnable λ to form the final output of the layer.

based on a query. To the best of our knowledge however, these works have not explored the effects of inverting this relationship: transforming a query into an output based on a set of key-value pairs. In contrast to these methods, we propose ISA, an attention framework that explicitly models this relationship. Notably, this design decision makes ISA inherently reciprocal to Self-Attention, enabling the two mechanisms to be efficiently coupled together into a single cohesive attention layer. Thus, we further propose BiSA, a framework that applies Self-Attention and ISA in parallel, and convexly combines their outputs.

3 Method

Let Q be a set of query vectors and (K, V) be a set of key-value vector pairs. For each query vector from Q, Self-Attention (SA) maps the set of key-value vector pairs to an output vector by computing a weighted sum over the projected value vectors. The weight for each value vector is determined by the compatibility of its corresponding key vector to the query vector, relative to all other keys (achieved using column-wise softmax). While this formulation enables SA to aggregate information stored in (K, V) given Q, it imposes a *direction* on the information flow between (K, V) and Q. This is because each output vector is *solely* composed of elements from (K, V), while the query *only* determines how to combine K and V when generating the final output. While this makes SA naturally amenable to fusing information from the context (e.g. a key-value set) around a particular source (e.g. a query), it does not lend well to refining a single source based on the surrounding context. Inverse Self-Attention (ISA) addresses this restriction by inverting the information flow between Q and (K, V). Specifically, ISA first uses each value vector to extract a unique representation from the query (e.g. via a projection). Then, a weight is computed for each representation based on the compatibility of the query to the key(associated with the value) relative to all other queries. Finally, these representations are linearly combined using their weights to obtain a single output vector. Thus, in ISA each output vector is *solely* composed of transformations of the query, and the elements of (K, V) determine how to transform the query.

By the above definitions, it is easy to see that ISA and SA are complementary to one another — SA computes its output by fusing the elements of (K, V) based on a query from Q, while ISA computes its output by transforming the same query Q based on the elements of (K, V). Bi-directional Self-Attention (BiSA) leverages the complementary natures of SA and ISA by using both the attention mechanisms in a cohesive way. Specifically, BiSA applies SA and ISA in parallel and computes a weighted sum over the normalized outputs. Going forward, we first summarize SA and describe ISA

in the context of the SA framework. We then arrive at BiSA by merging the two mechanisms, and finish by describing how BiSA can be easily integrated into existing models.

Self-Attention. Let $\mathbf{W_Q}, \mathbf{W_K} \in \mathbb{R}^{d_z \times d_k}$ and $\mathbf{W_V} \in \mathbb{R}^{d_z \times d_v}$ be projection matrices that transform elements from Q, K and V respectively. Self-Attention (SA) is then computed as:

$$SA((K,V),Q) = \texttt{Col-Softmax}\left(\frac{(Q\mathbf{W_Q})(K\mathbf{W_K})^T}{\sqrt{d_k}}\right)(V\mathbf{W_V}) \tag{1}$$

where `Col-Softmax` denotes column-wise softmax, and $\texttt{Col-Softmax}\left(\frac{(Q\mathbf{W_Q})(K\mathbf{W_K})^T}{\sqrt{d_k}}\right)$ measures the relative compatibility of every key to a query. Thus, SA combines value vectors according to the compatibility of their corresponding keys to each query.

Multi-Head Self-Attention. Multi-headed self-attention (MSA) is a commonly used extension of self-attention. Instead of a computing SA a single time, MSA computes it h times by linearly projecting queries, keys and values h different ways (i.e., transforming each via h different linear projections) and compressing each input vector to size d_h where $d_h \leq d_v$. SA is then applied on each set of these projected queries, keys, and values to obtain h d_h dimensional output vectors. The results are then concatenated to form a vector of size hd_v and (optionally) fed through a linear layer that compresses it to d_v. Note that MSA reduces to SA when $h = 1$. Thus, we use MSA and SA interchangeably for the remainder of the paper.

Algorithm 1 Pytorch Pseudocode of Inverse Self-Attention

Input: K, Q, V
$\mathbf{W_Q^1} = \texttt{nn.Linear}(d_z, d_k)$; $\mathbf{W_K} = \texttt{nn.Linear}(d_z, d_k)$;
$L = \texttt{nn.Softmax}((Q\mathbf{W_Q^1})(K\mathbf{W_K})/\sqrt{d_k}, \texttt{dim=0})$ ▷ Left branch ISA Fig. 1
$\mathbf{W_Q^2} = \texttt{nn.Linear}(d_z, d_s)$; $\mathbf{W_V} = \texttt{nn.Linear}(d_z, d_s)$;
$\mathbf{G} = \texttt{nn.Parameter}(d_s, d_s, d_s)$
$\hat{V} = \texttt{nn.GeLU}(V \texttt{ @ } \mathbf{W_V})$
$\hat{Q} = \texttt{nn.GeLU}(Q \texttt{ @ } \mathbf{W_Q^2})$
$\mathbf{W_{Q|V}} = \texttt{torch.tensordot}(\hat{V}, \mathbf{G}, \texttt{dims} = 1)$
$R = \texttt{torch.tensordot}(\hat{Q}, \mathbf{W_{Q|V}}, \texttt{dims} = 1)$ ▷ Right branch ISA Fig. 1
Output: L @ R

3.1 Inverse Self-Attention

On the other hand, Inverse Self-Attention (ISA) transforms query vectors based on the set of key-value vector pairs. It does so by projecting a given query vector differently for each value vector. The projected queries are then linearly combined, with the weight for each v-projected query determined by the corresponding k, where $(k, v) \in (K, V)$. This procedure can be broken down into three steps: (i) measuring compatibility between a query and the keys, (ii) transforming the query based on each key-value pair, and (iii) obtaining a weighted sum of the transformed queries based on the corresponding compatibilities.

Compatibility. Compatibility in ISA is nearly equivalent to SA, except that a `Row-Softmax` (i.e. row-wise softmax) is used instead of a `Col-Softmax`, taking the form $\texttt{Row-Softmax}\left(\frac{(Q\mathbf{W_Q})(K\mathbf{W_K})^T}{\sqrt{d_k}}\right)$. This has two implications: (i) the computed alignment is now of the query vector to a key vector (instead of the other way around as in SA), and (ii) the compatibilities of the query to each key may no longer sum to 1.

Query projection. We aim to extract different information from a query, $q \in Q$ based off of a value, $v \in V$. In standard self-attention, the query is projected according to a fixed learned projection matrix, $\mathbf{W_Q}$. Instead, we propose to learn a function, $g(V)$, that produces a unique projection matrix for the query conditioned on the value, $g(V) \rightarrow \mathbf{W_{Q|V}}$. This function, $g(V)$, can be thought of as a hypernetwork [34] that specifies a distinct transformation of the query for each value.

The simplest parameterization of $g(V)$ would be to define a 3D tensor, $\mathbf{A} \in \mathbb{R}^{d_z \times d_z \times d_v}$, such that a projection matrix, $\mathbf{W_{Q|V}}$ is defined as: $\mathbf{W_{Q|V}} = V \times_1 \mathbf{A}$, where $\times_n$ indicates tensor contraction along the nth mode. Then, this projection matrix could be applied to the query as $Q\mathbf{W_{Q|V}}$ in place of the fixed query projection matrix $\mathbf{W_Q}$ from self-attention. However, this operates directly on the query and key vectors in a linear fashion, which may be limiting.

Instead, we pre-process both the query and value vectors using non-linear functions: $\hat{Q} = \text{GeLU}(Q\mathbf{W_Q^2})$ and $\hat{V} = \text{GeLU}(V\mathbf{W_V})$, where $\mathbf{W_Q^2}, \mathbf{W_V} \in \mathbb{R}^{d_z \times d_z}$ are learned parameters. This differs from self-attention in that we learn a second query projection matrix, $\mathbf{W_Q^2}$, (*i.e.*, one not used to compute the compatibility matrix) and add non-linear activations to the projected outputs.

Efficiency. We note that directly learning the hypernetwork, $g(V)$, which is parameterized by $\mathbf{A} \in \mathbb{R}^{d_z \times d_z \times d_v}$, can result in a large number of learnable parameters. Even when d_z is small (*e.g.,* 256 as in [16]), $\mathbf{G}$ can be exorbitantly large (*e.g.,* 16M parameters)! Thus, we instead learn a much smaller tensor $\mathbf{G} \in \mathbb{R}^{d_s \times d_s \times d_v}$ and make $\mathbf{W_Q^2}, \mathbf{W_V} \in \mathbb{R}^{d_z \times d_s}$ such that $\hat{V}$ and $\hat{Q}$ are of dimension d_s, where $d_s << d_z$. This can dramatically reduce the number of parameters needed to learn while still producing a unique projection matrix conditioned on V.

Finally, this results in a value conditioned projection matrix: $\mathbf{W_{Q|V}} = \hat{V} \times_1 \mathbf{G}$, which is then applied to the non-linear projection of the query as: $\hat{Q}\mathbf{W_{Q|V}}$.

Weighted sum. After computing the compatibilities and query projections, the output is:

$$ISA(Q, (K, V)) = \texttt{Row-Softmax}\left(\frac{(Q\mathbf{W_Q^1})(K\mathbf{W_K})^T}{\sqrt{d_k}}\right)(\hat{Q}\mathbf{W_{Q|V}}) \tag{2}$$

Note that this is similar to SA, with the differences being that we impose a `Row-Softmax` instead of a `Col-Softmax`, and we sum over query projections rather than value transformations (e.g. $V\mathbf{W_V}$). Thus, ISA computes an output vector by transforming the query based on the set of key-value vector pairs. Algorithm 1 summarizes ISA.

Multi-Headed Inverse Self-Attention. Similarly to SA, ISA can also be adapted for multi-headed attention (MISA). In order to keep parameter counts low, MISA retains $\mathbf{A}$ and learns h different non-linear functions on Q and V. These are defined as $\hat{Q}^{(i)} = \text{GeLU}(Q[\mathbf{W_Q^2}]^{(\mathbf{i})})$ and $\hat{V}^{(i)} = \text{GeLU}(V[\mathbf{W_V}]^{(\mathbf{i})})$, where $[\mathbf{W_Q^2}]^{(\mathbf{i})}, [\mathbf{W_V}]^{(\mathbf{i})} \in \mathbb{R}^{d_z \times d_s}$ and (i) denotes the matrix for the i$^{\text{th}}$ head. Thus, the total parameter count imposed by MISA is given by $(h \times d_z \times d_s \times 2) + (d_s \times d_s \times d_v)$. For our implementation purposes, d_s, d_v and d_z is C/h, thus this module proportionally adds approximately (C/h^3) extra parameters relative to self attention. As observed, this addition is quite small for $h^3 < C$. Note that MISA reduces to ISA when $h = 1$. Thus, we use MISA and ISA interchangeably for the remainder of the paper.

3.2 Bi-Directional Self-Attention

As highlighted in Section 3, MSA and MISA attend over a set of inputs in a complementary manner. MSA maps a set of key-value pairs to an output vector based on a query, while MISA transforms the query to an output vector based on the key-value pairs. We propose to leverage their complementary features by coupling MSA and MISA into a single attention mechanism, termed bi-directional self-attention (BiSA). BiSA simply consists of the outputs from MISA and MSA, then convexly combining the two outputs using a hyperparameter $0 \leq \lambda \leq 1$. Note that λ can also be learned.

$$\text{BiSA}(Q, K, V) = \lambda \cdot MSA((K, V), Q) + (1 - \lambda) \cdot MISA(Q, (K, V)) \tag{3}$$

Since it is a convex combination, BiSA reduces to MSA or MISA with an appropriate λ selection ($\lambda = 1$ for MSA or $\lambda = 0$ for MISA). Figure 1 illustrates BiSA and MISA (when $h = 1$).

Vision models. Although MISA and BiSA can be applied to any framework where MSA has been studied, we restrict our experiments to visual understanding. In these models, it is assumed that $Q = K = V = X$, where X is the input set.

4 Experiments

We study the effect of BiSA and MISA on three benchmark image classification and semantic segmentation benchmarks: CIFAR100 [1], ImageNet-1K [2] and ADE20K [3]. We measure performance on image classification according to Top-k accuracy on a validation set, where Top-k corresponds to percentage of times a ground truth class is within the top-k classes predicted by a model. Specifically, we apply Top-1 and Top-5. We report our semantic segmentation results on *mIoU* (Mean intersection over union), *mAcc* (Mean accuracy of each class), *aAcc* (All pixel accuracy). We run all our experiments on nodes with 4 Nvidia Quadro RTX 6000 GPUs.

4.1 Baselines and BiSA integration

Our objective is to understand the behavior of BiSA when integrated into different attention layers within vision transformers. We examine this for the LeViT [5] model and the Swin Transformer [4].

Baseline: LeViT-128s. The LeViT [5] family of transformers was optimized for high throughput by, among other tricks, placing bottleneck CNN layers before the attention stages. The CNN layers significantly reduce the input size of the attention layers, while minimizing information loss as each output element from the CNN pools information from a broad receptive field on the image. We choose LeViT-128s (7.8M parameters) for our experiments due to its desirable speed to accuracy tradeoff. Notably, LeViT-128s is able to achieve competitive accuracy to an EfficientNet-B0 [35] on ImageNet [2] while nearly tripling its image throughput. Following the CNN layers, LeViT-128s employs 9 self-attention layers. We refer the reader to [5] for additional information.

Baseline: Swin-Tiny. Swin-Tiny [4] is the "tiny" version of the Swin-Transformer model (27.7M parameters) suite and offers the best performance per parameter trade-off. The model employs a hierarchical Self-Attention scheme that operates over shifting windows applied on inputs. Specifically, given a set of image patches, Swin groups the patches into non-overlapping window region sets and applies MSA within each window. The windows are then shifted by moving the resultant patch representations into new window sets and MSA is once again applied on each window. Notably, Swin-Tiny has 12 such attention layers. We refer readers to [4] for more information.

BiSA integration. Integrating BiSA with LeViT-128s and Swin-Tiny is straightforward, and only requires replacing the chosen self-attention layer(s) with the BiSA module. We explore the effects of replacing MSA layers with BiSA within the first two layers of the network. While in principle BiSA can replace any combination of MSA layers, we choose to apply it as close as possible to image-patches themselves. This is primarily due to the implications from the design of the MISA mechanism. Specifically, MISA transforms *single* elements (*e.g.,* patches) from an input based on the surrounding context (*e.g.,* all other patches). Thus, MISA reasons over and extracts *local features* within a network, and is thus most amenable to operating at lower layers of a network — where layers tend to focus on these kinds of features.

Architectural case studies. We study the behavior of BiSA on LeViT and Swin-Tiny by modifying two complementary components of its architecture. Specifically, we initially specify $\lambda = 0.5$ to equally weight the contributions from MSA and MISA within the BiSA framework. Interestingly however, we consistently observed that the average output magnitudes from MISA were substantially greater than MSA. This indicated that BiSA was learning to weight information from MISA much more heavily than information from MSA. To further study the importance of this magnitude effect, we opted to apply independent instance norms [36] on the outputs of MSA and MISA. This serves to whiten the outputs from MSA and MISA, thereby placing them on equivalent scales and equalizing their influence throughout the BiSA layer, while also serving to regularize each attention module. It may be the case that weighting the outputs of MSA and MISA is not desirable, but an effective regularization mechanism (*e.g.,* instance norm) could still be useful. Thus, we further augmented BiSA by allowing the λ parameter to be learned during training. This enables the network to choose how much influence to assign to the MISA mechanism vs MSA throughout training, while keeping the regularizing effects of instance norm in place — thereby acting as a bridge between BiSA with and without the norm. Note however that learning λ on top of instance norm is *strictly* less expressive than the original BiSA framework (with $\lambda = 0.5$ and no norm). This is because this variation *forces*

Table 1: CIFAR100 [1] and ImageNet1K [2] results for MISA and BiSA integrated with Swin-Tiny [4] and LeViT-128s [5]. Norm indicates whether the output of an attention layer (either MSA, MISA, or BiSA) is instance-normalized. λ refers to the convex weight used for BiSA; $\lambda = 1$ only utilizes MSA, while $\lambda = 0$ only utilizes MISA. #Layers Replaced denotes how many of the first two attention layers have been replaced with MISA or BiSA. #Params and #Flops are reported, along with top-1 and top-5 accuracies.

Dataset	Model	Variant	Norm	λ	#Layers Replaced	#Params	#Flops	Metrics	
								Top-1	Top-5
CIFAR100	Swin-Tiny	Baseline	-	-	0	27.6M	4.5G	80.09	94.62
		BiSA	-	0.5	2	27.7M	5.3G	**81.68**	**95.53**
		BiSA	x	0.5	2	27.7M	5.3G	80.75	95.14
		BiSA	x	Learned	2	27.7M	5.3G	80.34	94.91
	LeViT-128s	Baseline	-	-	0	7.8M	306M	73.58	92.98
		BiSA	-	0.5	2	8.0M	395M	**77.63**	**94.79**
		BiSA	x	0.5	2	8.0M	395M	75.29	93.76
		BiSA	x	Learned	2	8.0M	395M	75.05	93.42
ImageNet1K	Swin-Tiny	Baseline	-	-	0	28.3M	4.5G	81.19	95.52
		BiSA	-	0.5	2	28.4M	5.3G	**81.45**	**95.66**
	LeViT-128s	Baseline	-	-	0	7.8M	306M	76.45	92.97
		BiSA	-	0.5	2	8.0M	395M	**77.52**	**93.50**

BiSA to apply equal weight on *every* input instance of the mechanism — due to the normalization scaling effects — whereas the original BiSA framework enables *different scaling for each input*.

Training details. We use the LeViT hyperparameter configuration from the Unified training Scheme for ImageNet (USI) [37] for LeViT (see A.2 for details), and train all variants (both baseline and BiSA) in the same environment. Similarly to LeViT, we train all of our Swin-Tiny variants using the same training configuration. We use the same hyperparameters reported in [4], but find that doubling the number of epochs trained (600 vs 300) significantly improves Top-1 performance (*e.g.,* 80.1% vs. 77.4% for the baseline Swin-Tiny). For our experiments on ADE20K, we use the exact same configuration as described in [4].

4.2 Results & discussion

CIFAR100. Table 1 shows our results on CIFAR100 [1]. We find that the BiSA framework significantly improves performance by up to 4.05% on the LeViT architecture with the first two layers swapped. Note that using knowledge distillation, as LeViT was originally trained, results in an even greater performance improvement. Details can be found in A.2. Additionally, we observe that whitening BiSA, and further allowing λ to be learned both severely degrade performance, echoing the results and implications from our knowledge distillation experiments.

We also observe substantial improvements when applying BiSA to the Swin-Tiny model, yielding a 1.5% performance increase while raising the parameter count by just .37%! Moreover, we observe similar performance degradation when opting to normalize BiSA's MISA and MSA modules, and when allowing λ to be learned. This not only illustrates the efficacy for BiSA to improve performance across *differing* vision transformer architectures, but also its consistency — the best architecture is *always* the original BiSA framework across *all* experimental settings.

CIFAR100 ablations. We analyze the efficacy of BiSA through several ablation experiments, each studying distinct design choices. For ease of experimentation and compute, we conduct our ablations on the CIFAR100 dataset using the Swin-Tiny [4] transformer. We first study the effects of incorporating BiSA at varying levels of the Swin-Transformer, iteratively replacing all MSA attention layers in successive stages with BiSA. As expected, we observe performance degredations as BiSA is placed at higher-layer — by as much as 3.06% — over placing it in the first stage. Second, we validate our choice of query projection method in MISA by instead concatenating each query and value, and learning an Multi-Layer Perceptron of equal size. Importantly, this decreases performance by $.51\%$, highlighting the expressibility of the query projection hypernetwork. We believe this is principally due to the hypernetwork enabling *multiplicative interactions* between queries and values,

Table 2: ADE20K [3] results for BiSA integrated with Swin-Tiny [4] and Upernet [38] as evaluated on the test split. We report results on two evaluation types. `All` refers to standard semantic segmentation over the entire image. `Boundary` restricts evaluation to *only boundary pixels*. The bottom table row shows relative improvements of BiSA on the `Boundary` evaluation compared to its improvements over the baseline on `All`. We report results on *mIoU*, *mAcc*, *aAcc*.

Type	Backbone	Method	Variant	Norm	λ	#Layers Replaced	#params	Metrics		
								mIoU	mAcc	aAcc
`All`	`Swin-Tiny`	`Upernet`	Baseline	-	-	0	60M	44.51	55.61	81.09
	`Swin-Tiny`	`Upernet`	`BiSA`	-	0.5	2	60.1M	**45.11 (+1.3)**	**57.15 (+2.8)**	**81.21 (+0.1)**
`Boundary`	`Swin-Tiny`	`Upernet`	Baseline	-	-	0	60M	24.37	35.75	53.86
	`Swin-Tiny`	`Upernet`	`BiSA`	-	0.5	2	60.1M	**24.80 (+1.7)**	**37.11 (+3.8)**	**53.87 (+0.0)**
`Boundary vs. All`								+0.4	+1.0	-0.1

rather than limiting their interactions to being *additive*. Third, we experiment with integrating MISA and MSA sequentially (i.e. first applying MSA and then MISA within each layer, or vice-versa). Both configurations decrease performance by as much as 2.94% over BiSA's parallel architecture, strongly supporting our design decision. Table 3 in Section A.1 summarizes these results.

ImageNet1K. We further study BiSA with the Swin transformer [4] and LeViT [5] on the ImageNet [2] dataset and show results in Table 1. Similarly to CIFAR100 [1] experiments, we replace the first stage of the Swin-Tiny and LeViT-128s models with BiSA, do not add instance normalization, and define $\lambda = 0.5$. Table 1 shows the results of the BiSA models compared to the Swin-Tiny and LeViT-128s baselines. We observe that BiSA improves upon Swin-Tiny by 0.26% whilst *only adding* 0.37% additional parameters, showcasing its ability to boost baseline models with very little parameter increase. BiSA adds relatively more parameters to LeViT-128s compared to Swin-Tiny (2.56% vs 0.37% added parameters), but the additional added parameters result in a 1.07% improvement.

ADE20K. We also evaluate BiSA's applicability to semantic segmentation using the benchmark ADE20K [3] dataset (shown in Table 2). It consists of 150 semantic categories, and has 25K images in total. These are split into 20K training images, 2K for validation and 3K for testing. We use UperNet [38] from mmsegmentation [39] as our base framework and Swin-Tiny [4] as our backbone. We then experiment with replacing two layers of Swin-Tiny with BiSA, as in our previous Swin experiments. For our baseline, we use the Swin-Tiny+Upernet architecture, finetuning a Swin-Tiny pretrained on ImageNet1K. Similarly, our BiSA variant finetunes our best performing ImageNet1K BiSA model. Both experiments are performed using the same augmentations and hyperparameters as in [4]. Interestingly, our BiSA variant outperforms the baseline Swin-Tiny model across all three evaluation metrics when including all pixels in the metrics, while only increasing the total parameter count by 0.1M parameters (+0.17%). Specifically, we observe a +1.3% relative improvement in mIoU and a +2.8% relative improvement in mAcc. This significant improvement in mAcc highlights how BiSA is able to improve an underlying model's *precision* within semantic segmentation, while also uplifting its mIoU results. This showcases the capability of BiSA to improve feature extraction to localize different critical aspects of images, improving metrics while negligibly adding parameters.

In order to further gauge the ability of BiSA to process individual regions of an input differently based on surrounding context (e.g., emphasize boundaries between region types), we further conduct an experiment that checks a model's ability to accurately detect class boundaries. We do this by computing an edge mask for each image wherein all non-boundary pixels are removed from evaluation, and apply this mask to all evaluation images. Table 2 summarizes our results under `Boundary`. Interestingly, we observe substantially higher relative improvements over the baseline across both mAcc (+3.8%) and mIoU (+1.7%), especially when compared to performance over all image pixels (top half of the table). Notably, BiSA detects achieves a 1.0% boundary mAcc improvement over the baseline relative to its improvements across all pixels, highlighting the framework's substantially higher precision towards boundary detections. Similarly, this translates to an uplift of 0.4% on the boundary mIoU improvement relative to BiSA's improvements when considering all pixels.

Inspection of attention head outputs. We seek to understand how MISA chooses to attend over queries by visualizing the outputs from its various attention softmax row-wise computations. We do this by computing a column-wise max over these softmax values, and plotting the results. This gives

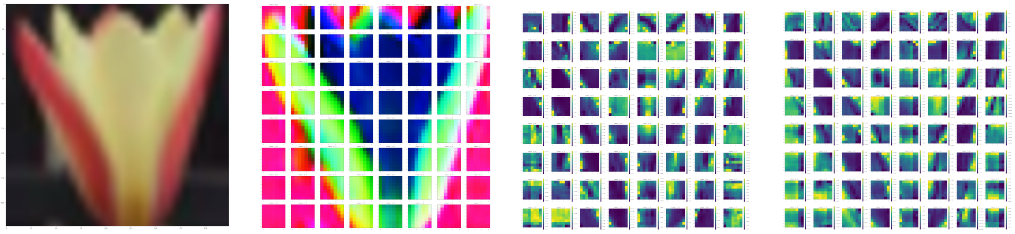

Figure 2: Visualization MISA's attention head outputs for a CIFAR100 tulip image on Swin-Tiny [4].

us a gauge of the *maximum* compatibility of a single query element to the key set as a whole, and thus whether information from the respective query is amplified (*e.g.,* high maximum value) or suppressed (*e.g.,* low maximum value). Fig 2 illustrates the results from this procedure on a CIFAR100 input image of a tulip. Each tile in the latter three figures corresponds to a specific window composed of patches, and the colors on the right two images depict the importance of the patch to the surrounding window (measured by maximum the compatibility). Bright colors (*e.g.,* yellow) indicate that a patch has high perceived importance, whereas dark colors (*e.g.,* blue) indicate low importance. These colors are scaled independently within each window. Interestingly, we observe that the attention learned within each head is different, and further not-equivalent across windows either, suggesting that MISA learns to focus over distinct window regions. Moreover, we observe that MISA places emphasis on discernible aspects with each window, attenuating boundary patches, while focusing on patches within object boundaries.

5 Limitations

MISA suffers from a similar tendency of many transformer based models that learn from scratch on a classical supervised learning objective. Transformers tend to be data hungry, and thus they need the aid of significantly larger datasets, tools such as contrastive losses, some form of parametric instance discrimination [40], or distillation (as we observed). MISA leads to a definite parameter increase, and in cases when $h^3 > C$ (h is number of heads and C is the channel count, described in the *Multi-Headed Inverse Self-Attention* section), relatively more than the self-attention module. However, these counts are significantly reduced given the efficiency modification elaborated on in the *Query Projection* discussion. We also see an increase in number of FLOPS by nearly 16% when incorporating our module, even when the parameter increase for the entire architecture is small.

Ethical considerations The proposed method doesn't focus on a new applied domain but instead proposes a variation on how Self-Attention is performed. It captures a new flavor of features: those that describe how each token/patch fits in the entire image. The module is plug and play, and thus can be used in a variety of architectures for different applications. Thus the main ethical considerations lie in potential biases that the datasets may have, and the domain where the final model will be used.

6 Conclusion

We propose Inverse Self-Attention (ISA), a novel attention framework that explicitly inverts how queries, keys, and values interact with each other relative to the Self-Attention (SA) mechanism. Specifically, rather than using a set of key-value pairs to compute an output according to their compatibility with a query, ISA transforms a query based on its compatibility to the set of key-value pairs to generate an output. This design choice makes ISA naturally complementary to SA, so we further introduce Bi-Directional Self-Attention (BiSA) to integrate both ISA and SA under a single unified framework using a convex λ weight. Notably, BiSA can reduce to either ISA or SA depending on the choice of λ, making it strictly more expressive than either. We demonstrate the applicability of BiSA by extending two vision transformer architectures: Swin [4] and LeViT [5] and substantially improve their performances on CIFAR100 [1], ImageNet1K [2] and ADE20K [3].

References

[1] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[3] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017.

[4] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[5] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12259–12269, 2021.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[8] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[9] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367, 2015.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020.

[12] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[13] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529, 2019.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[15] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *CoRR*, abs/2103.14899, 2021.

[16] 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. *CoRR*, abs/2010.11929, 2020.

[17] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *CoRR*, abs/2103.11816, 2021.

[18] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *CoRR*, abs/2012.12877, 2020.

[19] Jingkai Zhou, Pichao Wang, Fan Wang, Qiong Liu, Hao Li, and Rong Jin. ELSA: enhanced local self-attention for vision transformer. *CoRR*, abs/2112.12786, 2021.

[20] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *CoRR*, abs/2107.00641, 2021.

[21] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. *CoRR*, abs/1910.10781, 2019.

[22] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *CoRR*, abs/2005.00743, 2020.

[23] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *CoRR*, abs/2103.00112, 2021.

[24] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *CoRR*, abs/2102.12122, 2021.

[25] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *CoRR*, abs/1803.02155, 2018.

[26] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *CoRR*, abs/2102.10882, 2021.

[27] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis E. H. Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *CoRR*, abs/2101.11986, 2021.

[28] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. VOLO: vision outlooker for visual recognition. *CoRR*, abs/2106.13112, 2021.

[29] Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. Convolutional self-attention networks. *CoRR*, abs/1904.03107, 2019.

[30] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. *CoRR*, abs/2201.00520, 2022.

[31] Peng Gao, Jiasen Lu, Hongsheng Li, Roozbeh Mottaghi, and Aniruddha Kembhavi. Container: Context aggregation network. *CoRR*, abs/2106.01401, 2021.

[32] Daquan Zhou, Yujun Shi, Bingyi Kang, Weihao Yu, Zihang Jiang, Yuan Li, Xiaojie Jin, Qibin Hou, and Jiashi Feng. Refiner: Refining self-attention for vision transformers. *CoRR*, abs/2106.03714, 2021.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[34] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[36] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

[37] Tal Ridnik, Hussam Lawen, Emanuel Ben-Baruch, and Asaf Noy. Solving imagenet: a unified scheme for training any backbone to top results. *arXiv preprint arXiv:2204.03475*, 2022.

[38] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. *CoRR*, abs/1807.10221, 2018.

[39] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[40] Yun-Hao Cao, Hao Yu, and Jianxin Wu. Training vision transformers with only 2040 images. *arXiv preprint arXiv:2201.10728*, 2022.

[41] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. *CoRR*, abs/2012.11582, 2020.

[42] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *CoRR*, abs/2104.02057, 2021.

[43] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[44] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *CoRR*, abs/2011.09315, 2020.

[45] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. UP-DETR: unsupervised pre-training for object detection with transformers. *CoRR*, abs/2011.09094, 2020.

[46] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *CoRR*, abs/2103.15808, 2021.

[47] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. *CoRR*, abs/1911.05507, 2019.

[48] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *CoRR*, abs/2007.14062, 2020.

[49] Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. ETC: encoding long and structured data in transformers. *CoRR*, abs/2004.08483, 2020.

[50] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake A. Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *CoRR*, abs/2103.12731, 2021.

[51] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *CoRR*, abs/2103.15358, 2021.

[52] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[53] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[54] Ledyard R Tucker et al. The extension of factor analysis to three-dimensional matrices. *Contributions to mathematical psychology*, 110119, 1964.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[56] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[57] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[58] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[59] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.

- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Table 3: **CIFAR100 Ablations.** CIFAR100 [1] results for Swin-Tiny under several different ablations. Best refers to the best Swin-Tiny configuration from table 1. The placement ablations replace stages other than the first one with BiSA. The hypernetwork ablation replaces the hypernetwork on the right branch of ISA in figure 1 with an equivalently sized MLP. The module order ablations change MISA and MSA from being applied in parallel to being applied sequentially.

Variant	Stage Replaced	Query Projection	Module Order	Metrics
				Top-1
Best	1	Hypernetwork	Parallel	**81.68**
Placement Ablations	2	Hypernetwork	Parallel	80.66
	3	Hypernetwork	Parallel	79.89
	4	Hypernetwork	Parallel	78.62
Hypernetwork Ablation	1	MLP	Parallel	81.17
Module Order Ablations	1	Hypernetwork	MISA $\rightarrow$ MSA	78.74
	1	Hypernetwork	MSA $\rightarrow$ MISA	80.80

A Appendix

A.1 Ablations

Table 3 summarizes our ablation experiments, with discussion in Section 4.

A.2 Knowledge Distillation

Since LeViT [5] was originally trained using knowledge distillation [56], we decided to also run experiments where we added BiSA and trained the network using distillation. LeViT was originally trained via distillation for 1000 epochs on ImageNet. While originally opting to train our LeViT models within the same framework, we instead decided to leverage the recently released Unified training Scheme for ImageNet (USI) [37] for our distillation experiments. The USI framework has demonstrated improved performances on a range of architectures (including LeViT) on ImageNet, while training for only 310 epochs, and using the *same* hyperparameter configuration across all architectures. For consistency and fairness, we port USI's ImageNet hyperparameter setup to CIFAR100 and run both the LeViT-128s baseline and all of our experiments using the same setup. Due to the computational overhead induced from training Swin-Tiny on the USI framework, we restrict these experiments to CIFAR100 and LeViT-128s.

Our LeViT results with knowledge distillation can be found in Table 4. For this experimental setup, we study replacing either one or two of the first two attention layers of the LeViT model with BiSA. Additionally, we investigate each BiSA variation described in Section 4.1, including MISA only (*i.e.,* BiSA with $\lambda = 0.0$). MISA performs competitively with the baseline when replacing a single layer (77% vs 76.91% Top-1), but shows significant improvement of 2.07% Top-1 when both the first and second attention layers are swapped with BiSA. Moreover, we observe that the original BiSA framework achieves by far the best performance among all other BiSA variations when either a single or two MSA layers are replaced, improving the baseline model performance by as much as 4.34% Top-1 accuracy at the cost of increasing LeViT-128s's parameter count by just 2.56%. This illustrates the effective power of the BiSA framework, and the importance of enabling the outputs from MISA and MSA to have *independent scaling* with regards to each other for each instance. Additionally, we observe that introducing an instance norm with equal weighting (*e.g.,* $\lambda = 0.5$) decreases performance compared to the original BiSA, further illustrating that the two attention mechanisms MISA and MSA do not share equal importance within the vision transformer.

A.3 Training Speeds

We also examine the effect of BiSA on *training steps*. We perform this analysis using our best performing BiSA variants of the LeViT and Swin transformers on both CIFAR100 and ImageNet-1K,

Table 4: **CIFAR100 LeViT-128s distillation.** CIFAR100 [1] results for MISA and BiSA integrated with LeViT-128s [5] trained using knowledge distillation. Norm indicates whether the output of an attention layer (either MSA, MISA, or BiSA) is instance-normalized. λ refers to the convex weight used for BiSA; $\lambda = 1$ only utilizes MSA, while $\lambda = 0$ only utilizes MISA. #Layers Replaced denotes how many of the first two attention layers have been replaced with MISA or BiSA. #Params and #Flops are reported, along with top-1 and top-5 accuracies.

Variant	Norm	λ	#Layers Replaced	#Params	#Flops	Metrics	
						Top-1	Top-5
Baseline	-	-	0	7.8M	306M	76.91	94.05
MISA	-	-	1	7.9M	346M	77.00	94.52
BiSA	-	0.5	1	7.9M	351M	**79.15**	**95.25**
BiSA	x	0.5	1	7.9M	351M	77.98	94.82
BiSA	x	Learned	1	7.9M	351M	78.36	94.78
MISA	-	-	2	8.0M	385M	79.98	95.64
BiSA	-	0.5	2	8.0M	395M	**81.27**	**95.65**
BiSA	x	0.5	2	8.0M	395M	80.04	95.50
BiSA	x	Learned	2	8.0M	395M	79.51	95.43

measuring Top-1 accuracy. To do so, we compare the number of epochs it takes each baseline (*e.g.,* LeViT-128s) to reach its best performance on a dataset (*e.g.,* CIFAR100) with the number of epochs the BiSA integrated version takes to reach the same accuracy. Then, we calculate the ratio: $\frac{\text{\# Epochs BiSA}}{\text{\# Epochs Baseline}}$. For instance, if a baseline model requires 100 epochs to attain best performance while its BiSA integrated version achieves the same within 92 epochs, then our metric would be $\frac{92}{100} = 0.92$. Analogously, this would correspond to a $\frac{1}{0.92} = 1.09$ factor of training epoch gain. Figure 3 illustrates our results. Notably, we observe that BiSA applied to LeViT-128s matches LeViT-128s' best performance in up to 30% fewer epochs, while adding only 2.56% additional parameters (numbers in white from Figure 3). Similarly, we find that BiSA applied to Swin-Tiny requires at worst 92.0% of the training epochs of Swin-Tiny. While this 8% epoch improvement may seem minor, it is important to place it in the context of parameter increase. That is, by adding just 0.37% more parameters to Swin-Tiny, BiSA can speed up training by at a factor of at least 1.09×.

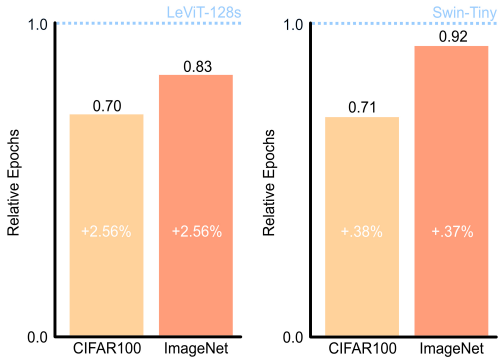

Figure 3: Epoch time required for each baseline to obtain its best performance on each dataset, as a fraction of the time it takes its respective BiSA variant to achieve equivalent performance. Lower is better. Percentage of added parameters by BiSA to each baseline are shown in white.

A.4 Sharpness Visualizations

Figures 4-7 offer a glimpse on the benefits of BiSA to MSA by investigating the structural similarities [59] between feature maps from matching layers of a baseline Swin-Tiny architecture and a corresponding BiSA variant. Specifically, all visualizations consist of the output feature maps from ImageNet1K images of the first attention layer from a baseline and BiSA model trained on ImageNet1K.

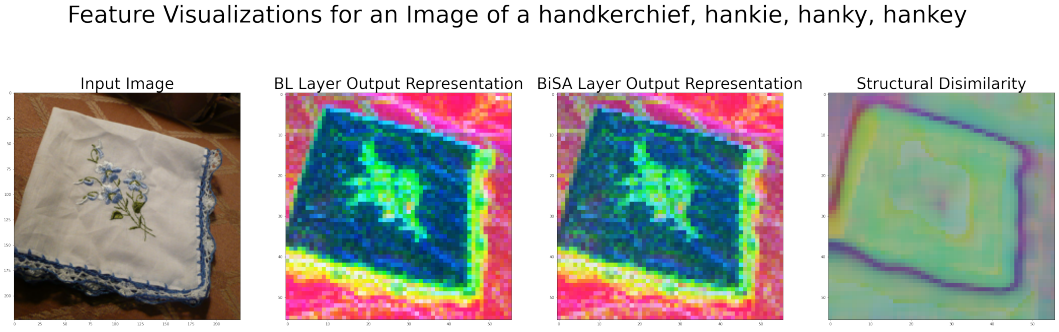

Figure 4: This is an example of where BiSA gets an output correct but the baseline (BL) does not. In this figure, we see that the BiSA Layer output captures finer granularity than the BL Layer output representation. Notice that the flower pattern is better captured by the BiSA layer output. We also notice that the structural dissimilarity[59] between the two outputs focus on identifying the shape of the hanky, which implies that BiSA is structurally dissimilar from the baseline around the handkerchief contour, thereby yielding a better object of interest localization for downstream image classification.

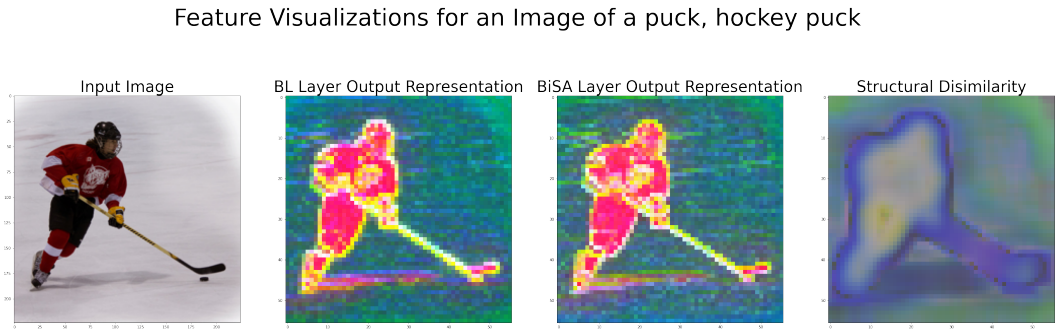

Figure 5: The structural dissimilarity[59] map illustrates how BiSA better captures the outline of the hockey player, his stick and puck — all relevant regions of interest. Moreover, we observe higher intensity in pixel values on the hockey puck itself in our BiSA variant over the baseline.

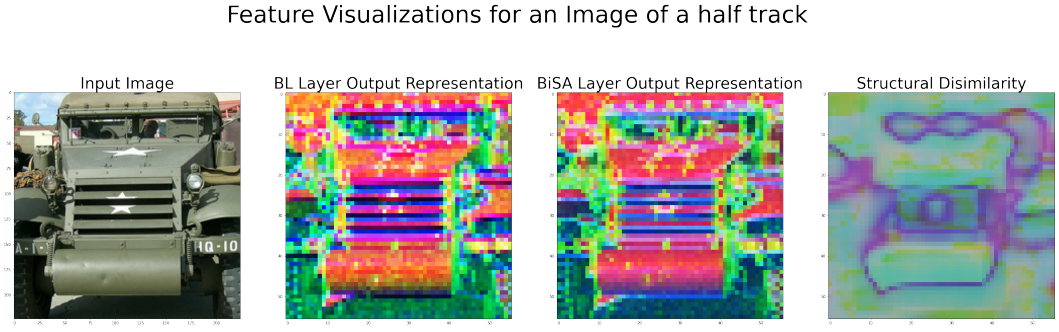

Figure 6: This example drives the point of sharpness/granularity home. We notice several details in the BiSA layer output that the baseline misses: the star in front of the truck, the detail on the roller and the grill in front of the truck. In addition, worth noticing is that the most important parts of the truck have a higher intensity of red, i.e. higher attention is placed there. We notice an extension of this observation in the structural dissimilarity[59] map where we can see that BiSA has captured identifying features like the grill, the sunshades and the roller that the baseline hasn't focused on as much.

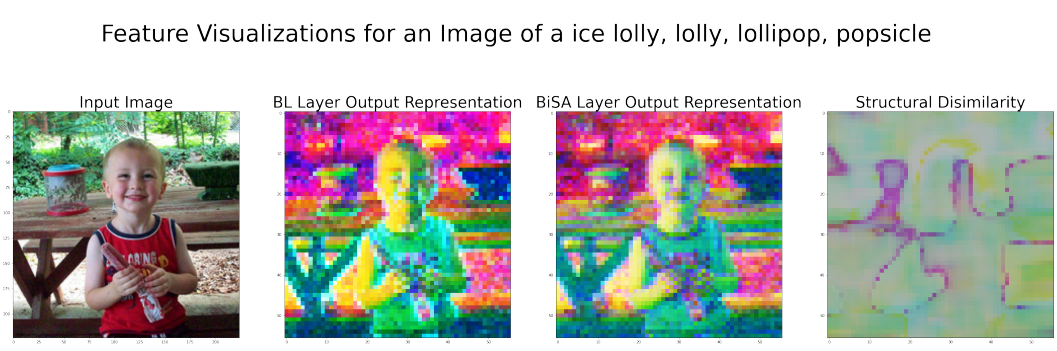

Figure 7: This is an example of where the baseline gets an output correct but BiSA does not. While the BiSA layer output is higher resolution, BiSA has captured features that may not be as valuable for the downstream task (identifying the ice lolly). The edges it has captured would most likely be more valuable in identifying larger components of the background.