Bi-Directional Self-Attention for Vision Transformers

George Stoica*
Georgia Tech
gstoica3@gatech.edu

Taylor Hearn
Georgia Tech
thearn6@gatech.edu

Bhavika Devnani
Georgia Tech
bdevnani3@gatech.edu

Judy Hoffman
Georgia Tech
judy@gatech.edu

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 of 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 ways to apply 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].

*Corresponding Author

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 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, Inverse Self-Attention (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 Self-Attention (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 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.

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

\[
SA((K, V), Q) = \text{Col-Softmax} \left( \frac{(QW_Q)(KW_K)^T}{\sqrt{d_k}} \right) (VW_V)
\]  

where Col-Softmax denotes column-wise softmax, and \(\text{Col-Softmax} \left( \frac{(QW_Q)(KW_K)^T}{\sqrt{d_k}} \right)\) 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 \(h d_v\) and (optionally) fed through a linear layer that compresses it to \(d_o\). 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**

1: **Input:** \(K, Q, V\)
2: \(W_Q^L = \text{nn.Linear}(d_z, d_k)\); \(W_K = \text{nn.Linear}(d_z, d_k)\);
3: \(L = \text{nn.Softmax}((QW_Q^L)(KW_K^L)/\sqrt{d_k}, \text{dim}=0)\) \(\triangleright \text{Left branch ISA Fig. 1}\)
4: \(W_Q^R = \text{nn.Linear}(d_z, d_s)\); \(W_V = \text{nn.Linear}(d_z, d_s)\);
5: \(G = \text{nn.Parameter}(d_s, d_s)\)
6: \(\hat{V} = \text{nn.GeLU}(V \odot W_V)\)
7: \(\hat{Q} = \text{nn.GeLU}(Q \odot W_Q^R)\)
8: \(W_{Q,V} = \text{torch.tensordot}({\hat{V}, G, \text{dims} = 1})\)
9: \(R = \text{torch.tensordot}(\hat{Q}, W_{Q,V}, \text{dims} = 1)\) \(\triangleright \text{Right branch ISA Fig. 1}\)
10: **Output:** \(L \odot R\)

### 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 \(\text{Row-Softmax} \left( \frac{(QW_Q)(KW_K)^T}{\sqrt{d_k}} \right)\). 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, \( W_Q \). Instead, we propose to learn a function, \( g(V) \), that produces a unique projection matrix for the query conditioned on the value, \( g(V) \rightarrow W_{Q|V} \). 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, \( A \in \mathbb{R}^{d_z \times d_x \times d_v} \), such that a projection matrix, \( W_{Q|V} \) is defined as: \( W_{Q|V} = V \times_1 A \), where \( \times_n \) indicates tensor contraction along the \( n \)th mode. Then, this projection matrix could be applied to the query as \( Q W_{Q|V} \) in place of the fixed query projection matrix \( W_Q \) 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:

\[
\hat{Q} = \text{GeLU}(QW_Q^2) \quad \text{and} \quad \hat{V} = \text{GeLU}(VW_V), \quad \text{where} \quad W_Q^2, W_V \in \mathbb{R}^{d_z \times d_x} \quad \text{are learned parameters. This differs from self-attention in that we learn a second query projection matrix,} \quad W_Q^2 \quad \text{(i.e., one not used to compute the compatibility matrix) and add non-linear activations to the projected outputs.}
\]

Efficiency. We note that directly learning the hypernetwork, \( g(V) \), which is parameterized by \( A \in \mathbb{R}^{d_z \times d_x \times d_v} \), can result in a large number of learnable parameters. Even when \( d_z \) is small (e.g., 256 as in [16]), \( G \) can be exorbitantly large (e.g., 16M parameters)! Thus, we instead learn a much smaller tensor \( G \in \mathbb{R}^{d_z \times d_x \times d_v} \), and make \( W_Q^2, W_V \in \mathbb{R}^{d_z \times d_x} \) such that \( \hat{V} \) and \( \hat{Q} \) are of dimension \( d_z \), where \( d_x << d_Z \). 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: \( W_{Q|V} = \hat{V} \times_1 G \), which is then applied to the non-linear projection of the query as: \( \hat{Q} W_{Q|V} \).

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

\[
\text{ISA}(Q, (K, V)) = \text{Row-Softmax} \left( \frac{(QW_Q^1)(KW_K)^T}{\sqrt{d_k}} \right) (\hat{Q}W_{Q|V})
\]  

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. \( V W_V \)). 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 \( G \) and learns \( h \) different non-linear functions on \( Q \) and \( V \). These are defined as \( \hat{Q}^{(i)} = \text{GeLU}(Q(W_Q^2)^{(i)}) \) and \( \hat{V}^{(i)} = \text{GeLU}(V(W_V)^{(i)}) \), where \( W_Q^2, W_V^{(i)} \in \mathbb{R}^{d_z \times d_x} \) and \( (i) \) denotes the matrix for the \( i \)th head. Thus, the total parameter count imposed by MISA is given by \( h \times d_z \times d_x \times 2 + (d_x \times d_v \times d_v) \). For our implementation purposes, \( d_x, d_v \), and \( d_z \) is \( C/h \), thus this module proportionally adds \( (C/h^3) \) 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 \( \lambda \) can also be learned.

\[
\text{BiSA}(Q, K, V) = \lambda \cdot MSA((K, V), Q) + (1 - \lambda) \cdot MISA(Q, (K, V))
\]  

Since it is a convex combination, BiSA reduces to MSA or MISA with an appropriate \( \lambda \) 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), and $aAcc$ (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 split across 4 "stages". The first, second and fourth stages have two layers, while the third has six. 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
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. \( \lambda \) 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Variant</th>
<th>Norm</th>
<th>( \lambda )</th>
<th>#Layers Replaced</th>
<th>#Params</th>
<th>#Flops</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Top-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIFAR100</td>
<td>Swin-Tiny</td>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>27.6M</td>
<td>4.5G</td>
<td>80.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>-</td>
<td>0.5</td>
<td>2</td>
<td>27.7M</td>
<td>5.3G</td>
<td>81.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>0.5</td>
<td>2</td>
<td>27.7M</td>
<td>5.3G</td>
<td>80.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>2</td>
<td>27.7M</td>
<td>5.3G</td>
<td>80.34</td>
</tr>
<tr>
<td>ImageNet1K</td>
<td>Swin-Tiny</td>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>7.8M</td>
<td>306M</td>
<td>73.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>-</td>
<td>0.5</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>77.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>0.5</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>75.29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>75.05</td>
</tr>
<tr>
<td>LeViT-128s</td>
<td>Swin-Tiny</td>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>28.3M</td>
<td>4.5G</td>
<td>81.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>-</td>
<td>0.5</td>
<td>2</td>
<td>28.4M</td>
<td>5.3G</td>
<td>77.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>2</td>
<td>7.8M</td>
<td>306M</td>
<td>76.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>77.52</td>
</tr>
</tbody>
</table>

BiSA by allowing the \( \lambda \) 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 \( \lambda \) 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 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. For our image classification experiments, we use the LeViT hyperparameter configuration from the Unified training Scheme for ImageNet (USI) [37] for LeViT (see A.3 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 as in [4]. 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.3. Additionally, we observe that whitening BiSA, and further allowing \( \lambda \) 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 \( \lambda \) 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. 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 degradations as BiSA is placed at higher stages — by as much as
We then experiment with replacing two layers of Swin-Tiny with BiSA, as in our previous Swin window (measured by maximum the compatibility). Bright colors (e.g., yellow) indicate high maximum value) or suppressed (e.g., blue) indicate low importance. These

<table>
<thead>
<tr>
<th>Calculation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAcc</td>
<td>3.06%</td>
</tr>
<tr>
<td>mIoU</td>
<td>0.37%</td>
</tr>
</tbody>
</table>

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 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

**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.

**Table 1.** Summary of results on ImageNet1K validation set. 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Swin-Tiny</th>
<th>BiSA</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAcc</td>
<td>91.5%</td>
<td>91.6%</td>
<td>+0.1%</td>
</tr>
<tr>
<td>mIoU</td>
<td>79.4%</td>
<td>80.4%</td>
<td>+1.0%</td>
</tr>
</tbody>
</table>

**Table 2.** Summary of results on ADE20K validation set. 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Swin-Tiny</th>
<th>BiSA</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAcc</td>
<td>91.5%</td>
<td>91.8%</td>
<td>+0.3%</td>
</tr>
<tr>
<td>mIoU</td>
<td>79.4%</td>
<td>80.4%</td>
<td>+1.0%</td>
</tr>
</tbody>
</table>
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.

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

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

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 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


[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.


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.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Stage Replaced</th>
<th>Query Projection</th>
<th>Module Order</th>
<th>Metrics Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>1</td>
<td>Hypernetwork</td>
<td>Parallel</td>
<td>81.68</td>
</tr>
<tr>
<td>Placement Ablations</td>
<td>2</td>
<td>Hypernetwork</td>
<td>Parallel</td>
<td>80.66</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Hypernetwork</td>
<td>Parallel</td>
<td>79.89</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Hypernetwork</td>
<td>Parallel</td>
<td>78.62</td>
</tr>
<tr>
<td>Hypernetwork Ablation</td>
<td>1</td>
<td>MLP</td>
<td>Parallel</td>
<td>81.17</td>
</tr>
<tr>
<td>Module Order Ablations</td>
<td>1</td>
<td>Hypernetwork</td>
<td>MISA → MSA</td>
<td>78.74</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Hypernetwork</td>
<td>MSA → MISA</td>
<td>80.80</td>
</tr>
</tbody>
</table>

A Appendix

A.1 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 [57], 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), is 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.

A.2 Ablations

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

A.3 Knowledge Distillation

Since LeViT [5] was originally trained using knowledge distillation [55], 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.
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. \( \lambda \) 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.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Norm</th>
<th>( \lambda )</th>
<th>#Layers Replaced</th>
<th>#Params</th>
<th>#Flops</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Top-1</td>
</tr>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>7.8M</td>
<td>306M</td>
<td>76.91</td>
</tr>
<tr>
<td>MISA</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>7.9M</td>
<td>346M</td>
<td>77.00</td>
</tr>
<tr>
<td>BiSA</td>
<td>-</td>
<td>0.5</td>
<td>1</td>
<td>7.9M</td>
<td>351M</td>
<td>\textbf{79.15}</td>
</tr>
<tr>
<td>BiSA</td>
<td>x</td>
<td>0.5</td>
<td>1</td>
<td>7.9M</td>
<td>351M</td>
<td>77.98</td>
</tr>
<tr>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>1</td>
<td>7.9M</td>
<td>351M</td>
<td>78.36</td>
</tr>
<tr>
<td>MISA</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>8.0M</td>
<td>385M</td>
<td>79.98</td>
</tr>
<tr>
<td>BiSA</td>
<td>-</td>
<td>0.5</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>\textbf{81.27}</td>
</tr>
<tr>
<td>BiSA</td>
<td>x</td>
<td>0.5</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>80.04</td>
</tr>
<tr>
<td>BiSA</td>
<td>x</td>
<td>Learned</td>
<td>2</td>
<td>8.0M</td>
<td>395M</td>
<td>79.51</td>
</tr>
</tbody>
</table>

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.4 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, 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: #Epochs BiSA

\[
\frac{\text{# Epochs BiSA}}{\text{# Epochs Baseline}}
\]

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,
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 handkerchief, 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.

by adding just 0.37% more parameters to Swin-Tiny, BiSA can speed up training by at a factor of at least 1.09×.

A.5 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 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.