

Supplementary Materials

SI-BiViT: Binarizing Vision Transformers with Spatial Interaction

Anonymous Authors

1. Plugged with existing binarized ViT methods

In the main paper, we evaluated the performance of our Spatial Interaction Module (SI Module) plugged with existing binarized ViT methods on ViT-Small backbones. In this section, we extend our evaluation to more ViT backbones, including DeiT-Tiny and Nest-Tiny. As shown in Table 1, our SI Module exhibits performance improvements on all these backbones. Notably, there is a significant improvement of 10.40% on the DeiT backbone when combined with BiBERT[3]. Overall, SI-MLP improves performance by an average of 5.07%. These results clearly demonstrate the flexibility and effectiveness of SI Module.

Table 1: Impact of spatial interaction module on existing binarized ViT methods on Tiny-ImageNet. SI module can be directly plugged into prevailing model binarization methods to further improve performance.

Model	Method	SI Module	Size _(MB)	Ops _(G)	Top-1(%)
DeiT-T	BiBERT [3]	✗	0.96	0.09	26.48
		✓	0.96	0.09	36.88(+10.40)
	BiViT [1]	✗	0.96	0.09	37.51
		✓	0.96	0.09	40.22(+2.71)
	Baseline	✗	1.05	0.09	44.11
		✓	1.08	0.09	49.80(+5.69)
ViT-S	BiBERT [3]	✗	1.62	0.15	36.65
		✓	1.68	0.15	43.15(+6.50)
	BiViT [1]	✗	1.62	0.15	42.91
		✓	1.68	0.16	51.04(+8.13)
	Baseline	✗	1.79	0.21	53.31
		✓	1.84	0.21	58.45(+5.14)
NesT-T	BiBERT [3]	✗	3.77	1.18	48.96
		✓	3.79	1.19	53.40(+4.44)
	BiViT [1]	✗	3.77	1.18	55.63
		✓	3.79	1.19	56.11(+0.48)
	Baseline	✗	3.77	1.18	57.90
		✓	3.79	1.19	60.00(+2.10)

2. Effect of expansion ratio in MLP module

Our dual branch consists of an MLP module and an SI module. To address efficiency concerns, we reduce the size of the MLP module. The Multi-Layer Perceptron (MLP) is composed of two linear layers: one expands the dimension, and another reduces it back to the original size. By adjusting the expansion ratio, we can control the size of the MLP. Initially, the expansion ratio is set to 4 by default. In stage 1, we apply knowledge distillation to the MLP module, enabling a smaller MLP without significant performance degradation. In stage 2, we introduce our SI module to enhance spatial interaction. As illustrated in Table 2, when the expansion ratio is set to 1 on the ViT-S backbone, the model struggles to

retain knowledge from full precision counterparts, resulting in a performance of only 51.80% at stage 1. Since the parameters in stage 2 are initialized based on stage 1, it inevitably leads to a low performance of 54.70%. Conversely, maintaining the expansion ratio at 4 yields the highest performance at stage 1 (58.50%), but due to limited spatial interaction, it degrades significantly in stage 2 (53.53%). Similar trends can be observed with the Swin-T backbone. Overall, setting the expansion ratio to 3 strikes a balance between preserving knowledge in stage 1 and enhancing spatial interaction in stage 2.

Table 2: Performance comparison with different expansion ratios of MLP module. ‘SIC’ denotes Spatial Interaction Capability.

Models	Expand-ratio	SIC(times)	Top-1(%)	
			Stage-1	Stage-2
ViT-S	1	7	51.80	54.70
	2	5	56.22	57.65
	3	3	56.12	58.45
	4	1	58.50	53.31
Swin-T	1	7	60.05	61.67
	2	5	63.06	63.19
	3	3	64.07	64.29
	4	1	64.32	64.22

3. Implementation of Dual-Branch

Our proposed dual-branch approach is straightforward to implement and can be seamlessly integrated into existing training pipelines. We directly replace the vanilla MLP module with it. Below is a simplified implementation using PyTorch [2]. The full implementation code will be made publicly available.

```
1 class SI_Module(nn.Module):
2     def __init__(self, dim_mlp, ratio_mlp, dim_spatial,
3         ratio_spatial):
4         super(SI_Module, self).__init__()
5         # MLP module
6         self.norm1 = nn.BatchNorm1d(dim_spatial)
7         self.mlp = Mlp1w1a(
8             in_features=dim_mlp,
9             hidden_features=int(dim_mlp * ratio_mlp),
10         )
11         # SI module
12         self.norm2 = nn.BatchNorm1d(dim_mlp)
13         self.spatial = Mlp1w1a(
14             in_features=dim_spatial,
15             hidden_features=int(dim_spatial *
16                 ratio_spatial),
17         )
18         # channel-wise balancing factor
19         self.learnablescale = nn.Parameter(torch.zeros
20             (1,1,dim_mlp), requires_grad=True)
```

117	18	
118	19	def forward(self, x):
119	20	x1 = self.mlp(self.norm1(x))
120	21	x2 = self.spatial(self.norm2(x.transpose(1, 2))).
121	22	transpose(1, 2)
122	23	x = x1 + self.learnablescale * x2
123		return x
124		
125		
126		
127		
128		
129		
130		
131		
132		
133		
134		
135		
136		
137		
138		
139		
140		
141		
142		
143		
144		
145		
146		
147		
148		
149		
150		
151		
152		
153		
154		
155		
156		
157		
158		
159		
160		
161		
162		
163		
164		
165		
166		
167		
168		
169		
170		
171		
172		
173		
174		

REFERENCES

[1] Yefei He, Zhenyu Lou, Luoming Zhang, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. BiViT: Extremely Compressed Binary Vision Transformers. In *IEEE International Conference on Computer Vision*. 5651–5663.

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[3] Haotong Qin, Yifu Ding, Mingyuan Zhang, Qinghua Yan, Aishan Liu, Qingqing Dang, Ziwei Liu, and Xianglong Liu. 2022. Bibert: Accurate fully binarized bert. *arXiv preprint arXiv:2203.06390* (2022).

175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232