

# Disrupting Positional Encoding for Effective Open Set Recognition

## Supplementary Material

### 002 1. Pseudo code for ViPER

---

**Algorithm 1** ViPER
 

---

```

x (input tensor) , shuffle (boolean) x (shuffled tensor)
if shuffle is True then Extract class token and patch embeddings
  cls_pos  $\leftarrow pos\_embed[:, :1, :]$ 
  patch_pos  $\leftarrow pos\_embed[:, 1 :, :]$ 
  Shuffle patch position embeddings
  idx  $\leftarrow torch.randperm(patch\_pos.size(1))$ 
  shuffled_patch_pos  $\leftarrow patch\_pos[:, idx, :]$ 
  Recombine embeddings
  pos_embed  $\leftarrow torch.cat([cls\_pos, shuffled\_patch\_pos], dim = 1)$  torch.cuda.synchronize()
else pos_embed  $\leftarrow$  original position embedding
  Add position embeddings
  x  $\leftarrow x + pos\_embed$ 
return x

```

---

consumption, contributing to greener AI objectives when deployed at scale.

027  
028

**Limitations** The current methodology primarily relies on permuting positional encodings to disrupt global spatial relationships. This singular approach may not fully address other critical near-OOD variations. Meanwhile, the framework inherently depends on the positional encoding mechanisms specific to Vision Transformers. Its effectiveness remains unverified for architectures employing learnable positional embeddings or alternative spatial encoding strategies, necessitating non-trivial architectural modifications for broader applicability.

029  
030  
031  
032  
033  
034  
035  
036  
037  
038

### 003 2. Reproducibility

#### 004 2.1. Code

005 Our code is available at [ViPER](#). The repository structure includes the dataset folder for dataset initialization, the models folder containing backbone model implementations, `split.py` for dataset partitioning, and `train.py` for handling all training procedures. With the required environment configured, ViPER offers exceptional reproducibility, and experimental results can be conveniently uploaded to the Weights & Biases platform for comprehensive data analysis and visualization.

#### 013 2.2. Split information

014 In Table 1, we present the dataset splits used for AUROC testing. For the CIFAR+N setup, the “CIFAR100” row indicates the selection of unknown classes, while all other rows represent known classes.

### 018 3. More discussion

019 **Broader impacts** ViPER’s lightweight self-supervised design eliminates the need for large-scale auxiliary datasets, enabling the application of open-set recognition technology in data-scarce domains (such as rare disease diagnosis or niche industrial scenarios). This significantly lowers the adoption barrier for researchers and practitioners in resource-constrained environments. Compared to data-intensive approaches, ViPER’s computational efficiency reduces energy

Table 1. Dataset split information.

Dataset	Split 1	Split 2	Split 3	Split 4	Split 5	
CIFAR10	[0,1,2,4,5,9]	[0,3,5,7,8,9]	[0,1,5,6,7,8]	[3,4,5,7,8,9]	[0,1,2,3,7,8]	
CIFAR+10	CIFAR10	[0,1,8,9]	[0,1,8,9]	[0,1,8,9]	[0,1,8,9]	
	CIFAR100	[26,31,34,44,45, 63,65,77,93,98]	[7,11,66,75,77, 93,95,97,98,99]	[2,11,15,24,32, 34,63,88,93,95]	[1,11,38,42,44, 45,63,64,66,67]	[3,15,19,21,42, 46,66,72,78,98]
CIFAR+50	CIFAR10	[0,1,8,9]	[0,1,8,9]	[0,1,8,9]	[0,1,8,9]	
	CIFAR100	[1,2,7,9,10,12, 15,18,21,23,26, 30,32,33,34,36, 37,39,40,42,44, 45,46,47,49,50, 51,52,55,56,59, 60,61,63,65,66, 70,72,73,74,76, 78,80,83,87,91, 92,96,98,99]	[0,2,4,5,9,12, 14,17,18,20,21, 23,24,25,31,32, 33,35,39,43,45, 49,50,51,52,54, 55,56,60,64,65, 66,68,70,71,73, 74,77,78,79,80, 82,83,86,91,93, 94,96,97,98]	[0,4,10,11,12, 14,15,17,18,21, 23,26,27,28,29, 31,32,33,36,39, 40,42,43,46,47, 51,53,56,57,59, 60,64,66,71,73, 74,75,76,78,79, 80,83,87,91,92, 93,94,95,96,99]	[0,2,5,6,9,10, 11,12,14,16,18, 19,21,22,23,26, 27,28,29,31,33, 35,36,37,38,39, 40,43,45,49,52, 52,53,54,55,56, 56,59,61,62,63, 64,65,71,74,75, 66,67,68,73,74, 77,82,83,86,87, 91,93,94,96]	[0,1,4,6,7,12, 15,16,17,19,20, 21,22,23,26,27, 28,32,39,40,42, 43,44,47,49,50, 52,53,54,55,56, 59,61,62,63,65, 66,67,68,73,74, 77,82,83,86,87, 93,94,97,98]
Tiny-ImageNet		[2,3,13,30, 44,45,64,66, 76,101,111, 121,128,130, 136,158,167, 170,187,193]	[4,11,32,42,51, 53,67,84,87, 104,116,140, 144,145,148, 149,155,168, 185,193]	[3,9,10,20,23, 28,29,45,54, 74,133,143, 146,147,156, 159,161,170, 184,195]	[1,15,17,31,36, 44,66,69,84, 89,102,137, 154,160,170, 177,182,185, 195,197]	[4,14,16,33,34, 39,59,69,77, 92,101,103, 130,133,147, 161,166,168, 172,173]