Supplemental Materials for
Fine-Grained Object Classification via Self-Supervised Pose Alignment

1. Graph Matching Algorithm

The framework of the proposed graph matching algorithm for parts alignment is presented in Alg. 1. We aims to align the top N discriminative parts to an unified order.

Algorithm 1: Graph Matching for Parts Alignment

Input: Top N parts representations: $[\boldsymbol{r}_{p_1}, \boldsymbol{r}_{p_2}, ..., \boldsymbol{r}_{p_N}]$

Output: Sorted parts representations: *e.g.*, $[\boldsymbol{r}_{p_i}, \boldsymbol{r}_{p_j}, ..., \boldsymbol{r}_{p_k}]$.

```
*Initialize parts centers as empty: c;
*Initialize reference matrix as empty: M ∈ R^{N×N};
for image in samples do
    Initialize the matching degree: D = 0;
    Initialize all possible permutations of [r_{p_1}, r_{p_2}, ..., r_{p_N}];
    for permutation in possible permutations do
        if c is empty then
            Let c = [r_{p_1}, r_{p_2}, ..., r_{p_N}];
            Update M;
        else
            Arrange [r_{p_1}, r_{p_2}, ..., r_{p_N}] given current permutation;
            Compute the correlation matrix: M';
            if sum(M ∘ M') > D then
                D = sum(M ∘ M');
                Set current permutation as the best;
            end
        end
    end
    Resort [r_{p_1}, r_{p_2}, ..., r_{p_N}] according to the best permutation;
    Update c using exponential moving average: c_t = βc_{t-1} + (1 − β)θ_t;
    Update M;
end
```

$\{\alpha^{(s)}\}$	$\hat{\boldsymbol{y}}^{(1)}$	$\hat{\boldsymbol{y}}^{(2)}$	$\hat{\boldsymbol{y}}^{(3)}$	$\hat{\boldsymbol{y}}^{(4)}$	$\hat{\boldsymbol{y}}^{(final)}$
$\{0.7, 0.8, 0.9, 1.0\}$	81.7	**87.3**	**85.8**	**87.8**	**88.4**
$\{1.0, 1.0, 1.0, 1.0\}$	**82.1**	87.0	85.6	87.3	88.2

Table 1. Accuracy (%) of different predictions on CUB with different label smoothing setting.

2. Ablation Studies

Smoothing Factor $\{\alpha^{(s)}\}$. Remember that $\{\alpha^{(s)}\}$ is involved for label smoothing during curriculum learning. To prove its positive role, we further conduct a contrastive experiment without employing label smoothing (*i.e.*, $\{\alpha^{(s)}\} = \{1, 1, 1, 1\}$). Results on CUB of each stage's prediction are reported in Table 1. The method using increasing $\{\alpha^{(s)}\}$ consistently outperforms the method without using label smoothing except on the first output $\hat{\boldsymbol{y}}^{(1)}$.

Unsupervised Part Alignment (UPA). Since UPA is directly applied on image representations at each stage when using curriculum supervision, we plot every intermediate prediction's accuracy in the Fig. 1. As the bar graph shows, most of the results follow a consistent order on compared datasets: baseline+CS < baseline+CS+FR(w/o UPA) < baseline+CS+FR(w/ UPA), which demonstrate that UPA plays an effective role in learning better representation. In addition, the ensemble prediction $\hat{\boldsymbol{y}}^{(final)}$ exhibits slightly better performance than the other predictions $\hat{\boldsymbol{y}}^{(s)}$, which verifies our hypothesis that aggregating predictions from different stages yields a stronger classifier.

UPA's Influence on Pose Changes. Since it is hard to obtain the images of the same instance in other poses, we alternatively flip or rotate the original images to attain this end. Therefore, we choose the methods Baseline+FR(w/o UPA) and Baseline+FR(w/o UPA) to see the amplitude of representation changes when the input images is being rotated or flipped. Results is reported in Table 2. We can see that, when the input image is changed (*i.e.*, object pose is changed), the method with UPA shows smaller changes in object representation compared to the method without UPA in nearly all rotation and flipping cases. This result verifies that UPA has an effective role against object pose changes.

Fusion Scheme of Parts Features. Given N parts fea-

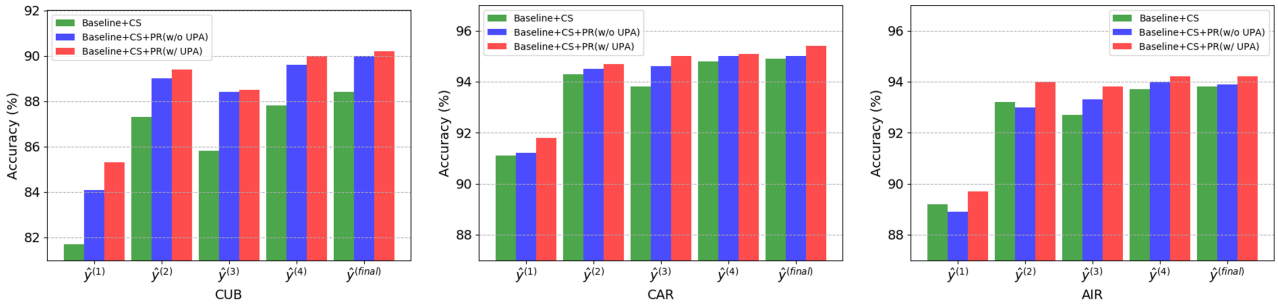

Figure 1. Stage-wise prediction accuracy of three methods on CUB, CAR and AIR datasets.

Processing	UPA	Euclidean distance		
		CUB	CAR	AIR
flip-x	w/o	32.01	26.52	57.35
	w	**31.69**	**20.12**	**55.11**
flip-y	w/o	207.11	154.36	269.73
	w/	**192.89**	**106.34**	**234.05**
r90	w/o	132.57	170.67	**349.46**
	w/	**123.00**	**135.25**	356.37
r180	w/o	213.84	155.23	274.50
	w/	**200.68**	**106.76**	**235.06**
r270	w/o	131.51	172.66	**355.31**
	w/	**124.68**	**138.57**	367.83

Table 2. The average Euclidean distance between the representations of original images and rotated/flipped images. The words flip-x and flip-y represent flipping the input image along the x-axis and y-axis respectively. The words r90, r180 and r270 denote rotating the input image anticlockwise by 90°, 180° and 270° respectively.

Method	Accuracy (%)		
	CUB	CAR	AIR
Average	90.1	95.1	94.0
Transformer	**90.2**	95.2	**94.2**
Ours (UPA+MLP)	**90.2**	**95.4**	**94.2**

Table 3. Comparison of different fusion schemes.

Parts number (N)		2	3	4	5	6
Accuracy (%)	CUB	88.1	89.9	**90.2**	**90.2**	90.0
	CAR	93.8	95.0	**95.4**	95.1	95.2
	AIR	91.2	93.6	94.2	**94.4**	94.3

Table 4. Influence of the number of discriminative parts. Setting of methods follow the same as baseline+CS+FR(w/ UPA) except the parts number. The backbone is ResNet50.

tures, there are other simple and feasible order-invariant ways to fuse them to obtain a pose-invariant representation. We consequently construct two methods with different fusion schemes. The first one use average of parts features as $\hat{\boldsymbol{r}}_{im}^{(s)}$, while the second one use transformer to output $\hat{\boldsymbol{r}}_{im}^{(s)}$. As we know, the transformer without using position embedding is invariant to the order of input tokens. Experiment results are shown in Table 3. The employed transformer consists of two encoder units and its original output is of the same shape as the input. We use the average output tokens as the final output. Remember that our method uses UPA to resort the parts in a consistent order before applying MLP to fuse features. It can be seen that our method shows improvements over the simple average fusion scheme and is slightly better than the transformer. But note that the proposed UPA+MLP has advantages in calculation and memory occupation that the transformer.

Parts Number N. The number of discriminative parts N is an important hyper-parameter of our method. Small N may forces the backbone to focus on a few regions and results in not discriminative enough representation. On the contrary, a large N may bring redundant information to affect the classification accuracy. A single part is not able to reveal object pose, we thereby conducted experiments with N ranges from 2 to 5 to see parts number's effects. Comparison results are reported in the Table 4. Accuracy increases significantly when N increases from 2 to 4. However, when N getting larger than 4, the accuracy's average increase on three datasets is less than 0. This result can be explained by the reason that object image has a limit number of salient parts and if we choose more than a specific number of discriminative parts, background regions or parts of common patterns will be inevitably selected as discriminative parts. These non-discriminative parts may have negative effects on the optimization of network parameters. In general, for simple objects like the used datasets, it is suitable to set the parts number as 4 or 5. For more complicated objects, larger N should be chose by experience.

3. Computation Time

Given the same settings, we list a comparison of training cost and inference latency in Tab 5. Although P2P-Net has

more parameters and FLOPs during training, but achieves comparable computational efficiency during testing, owing to the self-supervised part alignment module as feature regularization only activated in training.

Model	Params (M)	FLOPs (G)	Time (sec)
ResNet50 [11]	23.92	16.44	0.064/0.034
NTS-Net [41]	29.03	41.91	0.126/0.069
PMG [8]	45.13	37.47	0.270/0.043
API-Net [48]	46.06/42.91	31.53/31.52	0.104/0.054
P2P-Net (ours)	64.09/44.63	75.43/37.47	0.136/0.041

Table 5. Comparison of methods with official codes. Beyond the API-Net using the ResNet101 backbone, the others use the ResNet50. Values listed as train/test if they are different.