Appendix for

“PLOT: Prompt Learning with Optimal Transport for Vision-Language Models”

Appendix organization:

A1 METHOD DETAILS

The Optimal Transport (Monge, 1781) is initially introduced to find a transportation plan to move simultaneously several items at a minimal cost, such as moving a pile of sand to fill all the holes. Recently, it is widely used for the comparison of distributions. Mathematically, given two probability density function U and V over space $\mathcal{X}$ and $\mathcal{Y}$, the OT (Wasserstein) distance (Thorpe, 2019) can be defined as

$$D_{\text{OT}}(U,V) = \inf_{\Gamma} \int_{\mathcal{X}\times\mathcal{Y}} \boldsymbol{C}(\boldsymbol{x},\boldsymbol{y}) d\gamma(\boldsymbol{x},\boldsymbol{y}), \tag{10}$$

where $\boldsymbol{C}(\boldsymbol{x},\boldsymbol{y})$ is the cost between two points in the space $\mathcal{X}\times\mathcal{Y}$, and Γ denotes the set of transport plans between support points $\boldsymbol{x}$ and $\boldsymbol{y}$ (e.g. $\gamma(\boldsymbol{x},\boldsymbol{y})$). We can regard two probability density functions U and V as piles and holes and $\boldsymbol{C}$ is the cost function of moving a unit of sand.

In our problem of multiple prompts learning, we formulate the sets of visual features and prompt features as two discrete distributions as

$$U = \sum_{m=1}^{M} u_m \delta_{\boldsymbol{f}_m} \qquad \text{and} \qquad V = \sum_{n=1}^{N} v_n \delta_{\boldsymbol{g}_n}, \tag{11}$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ are the discrete probability vectors that sum to 1, and $\delta_{\boldsymbol{f}}$ is a Dirac delta function placed at support point $\boldsymbol{f}$ in the embedding space. Given two support points $\boldsymbol{f}_m$ and $\boldsymbol{g}_n$, the cost function is written as $\boldsymbol{C}(\boldsymbol{f}_m,\boldsymbol{g}_n) = 1 - \text{sim}(\boldsymbol{f}_m,\boldsymbol{g}_n) = 1 - \frac{\boldsymbol{f}_m^\top \boldsymbol{g}_n}{||\boldsymbol{f}_m||\cdot||\boldsymbol{g}_n||}$. For simply, in this discrete situation, $\boldsymbol{C} \in \mathbb{R}^{M\times N}$ is a cost matrix in which each point denotes the cost between $\boldsymbol{f}_m$ and $\boldsymbol{g}_n$.

Algorithm A1: The training process of Prompt Learning with Optimal Transport

```
Input: Training few-shot image data: $\mathbf{X} = \{\boldsymbol{x}\}$, pretrained CLIP model $f$ and $g$, number of prompts $N$, entropy parameter $\lambda$, maximum number of iterations in inner and outer loops $T_{in}, T_{out}$.
Output: The parameters of prompts $\{\boldsymbol{\omega}_n|_{n=1}^N\}$
Initialize $\{\boldsymbol{\omega}_n|_{n=1}^N\}$
for $t_{out} = 1, 2, \dots, T_{out}$ in the outer loop do
    Obtain a visual feature set $\boldsymbol{F} \in \mathbb{R}^{M \times C}$ with the visual encoder $f(x)$;
    Generate prompt feature set $\boldsymbol{G}_k \in \mathbb{R}^{N \times C}$ of each class with the textual encoder $\{g(t_k^n)\}|_{n=1}^N$;
    Calculate the cost matrix $\boldsymbol{C}_k = \mathbf{1} - \boldsymbol{F}^\top \boldsymbol{G}_k \in \mathbb{R}^{M \times N}$ of each class
    Calculate the OT distance with an inner loop: Initialize the $\boldsymbol{v}^{(0)} = \mathbf{1}$, $\delta = 0.01$ and $\Delta_v = \infty$
    for $t_{in} = 1, 2, \dots, T_{in}$ do
        Update $\boldsymbol{u}^{(t_{in})} = \boldsymbol{u}/((\exp(-\boldsymbol{C}/\lambda)\boldsymbol{v}^{(t_{in}-1)})$
        Update $\boldsymbol{v}^{(t_{in})} = \boldsymbol{v}/((\exp(-\boldsymbol{C}/\lambda)^\top \boldsymbol{u}^{(t_{in})})$
        Update $\Delta_v = \sum |\boldsymbol{v}^{(t_{in})} - \boldsymbol{v}^{(t_{in}-1)}|/N$
        if $\Delta_v < \delta$ then
            break
        end if
    end for
    Obtain optimal transport plan as $\boldsymbol{T}_k^* = \text{diag}(\boldsymbol{u}^{(t)})\exp(-\boldsymbol{C}_k/\lambda)\text{diag}(\boldsymbol{v}^{(t)})$,
    Calculate the OT distance $d_{\text{OT}}(k) =< \boldsymbol{T}_k^*, \boldsymbol{C}_k >$
    Calculate the classification probability $p_{\text{OT}}(y = k|\boldsymbol{x})$ with the OT distance
    Update the parameters of prompts $\{\boldsymbol{\omega}_n|_{n=1}^N\}$ with cross-entropy loss $L_{\text{CE}}$
end for
return $\{\boldsymbol{\omega}_n|_{n=1}^N\}$
```

Then, the total distance of these two distributions is written as:

$$< \boldsymbol{T}, \boldsymbol{C} >= \sum_{m=1}^{M} \sum_{n=1}^{N} \boldsymbol{T}_{m,n} \boldsymbol{C}_{m,n}, \tag{12}$$

where the $\boldsymbol{T} \in \mathbb{R}^{M \times N}$ is a matrix of transport plan, which is learned to minimize the total distance. Each point $\boldsymbol{T}_{m,n}$ in $\boldsymbol{T}$ is a weight of local cost $\boldsymbol{C}_{m,n}$.

The optimization problem of optimal transport is formulated as:

$$\begin{aligned} & d_{\text{OT}}(\boldsymbol{u}, \boldsymbol{v}|\boldsymbol{C}) = \underset{\boldsymbol{T}}{\text{minimize}} < \boldsymbol{T}, \boldsymbol{C} > \\ & \text{subject to} \quad \boldsymbol{T}\mathbf{1}_N = \boldsymbol{u},\ \boldsymbol{T}^\top \mathbf{1}_M = \boldsymbol{v},\ \boldsymbol{T} \in \mathbb{R}_+^{M \times N}. \end{aligned} \tag{13}$$

These constraints of $\boldsymbol{T}$ are used to match its marginal distributions and original discrete distributions in Eq. 11. In our framework, we treat visual features $\boldsymbol{f}_m$ and prompt features $\boldsymbol{g}_n$ equally and thus $\boldsymbol{u} = \mathbf{1}_{M \times 1}/M$ and $\boldsymbol{v} = \mathbf{1}_{N \times 1}/N$.

As directly optimizing the above objective is always time-consuming, we apply the Sinkhorn distance (Cuturi, 2013) to use an entropic constraint for fast optimization. The optimization problem with a Lagrange multiplier of the entropy constraint is:

$$\begin{aligned} & d_{\text{OT},\lambda}(\boldsymbol{u}, \boldsymbol{v}|\boldsymbol{C}) = \underset{\boldsymbol{T}}{\text{minimize}} < \boldsymbol{T}, \boldsymbol{C} > -\lambda h(\boldsymbol{T}) \\ & \text{subject to} \quad \boldsymbol{T}\mathbf{1}_N = \boldsymbol{u},\ \boldsymbol{T}^\top \mathbf{1}_M = \boldsymbol{v},\ \boldsymbol{T} \in \mathbb{R}_+^{M \times N}, \end{aligned} \tag{14}$$

where $h(\cdot)$ is entropy and $\lambda \geq 0$ is a hyper-parameter. Then we can have a fast optimization solution with a few iterations as:

$$\boldsymbol{T}^* = \text{diag}(\boldsymbol{u}^{(t)}) \exp(-\boldsymbol{C}/\lambda) \text{diag}(\boldsymbol{v}^{(t)}), \tag{15}$$

where t denotes iteration and in each iteration $\boldsymbol{u}^{(t)} = \boldsymbol{u}/\left((\exp(-\boldsymbol{C}/\lambda)\boldsymbol{v}^{(t-1)}\right)$ and $\boldsymbol{v}^{(t)} = \boldsymbol{v}/\left((\exp(-\boldsymbol{C}/\lambda)^\top \boldsymbol{u}^{(t)}\right)$, with the initiation $\boldsymbol{v}^{(0)} = \mathbf{1}$. The detailed algorithms of the training and testing processes are shown in Algorithms A1 and A2

Algorithm A2: The inference process of Prompt Learning with Optimal Transport

```
Input: Testing image data: $\mathbf{X} = \{\boldsymbol{x}\}$, number of prompts $N$, number of classes $K$, learned prompts
    $\{\boldsymbol{t}_k^n|_{k=1,n=1}^{K,N}\}$, a frozen pretrained CLIP model including image encoder $f$ and text encoder$g$
Output: The classification of each image
for $\boldsymbol{x}$ in $\mathbf{X}$ do
    Obtain a visual feature set $\boldsymbol{F} \in \mathbb{R}^{M \times C}$ with the visual encoder $f(x)$;
    Generate prompt feature set $\boldsymbol{G}_k \in \mathbb{R}^{N \times C}$ of each class with the textual encoder $\{g(t_k^n)\}|_{n=1}^{N}$;
    Calculate the cost matrix $\boldsymbol{C}_k = \mathbf{1} - \boldsymbol{F}^\top \boldsymbol{G}_k \in \mathbb{R}^{M \times N}$ of each class
    Calculate the OT distance with an inner loop: Initialize the $\boldsymbol{v}^{(0)} = \mathbf{1}$, $\delta = 0.01$ and $\Delta_v = \infty$
    for $t_{in} = 1, 2, \ldots, T_{in}$ do
        Update $\boldsymbol{u}^{(t_{in})} = \boldsymbol{u}/((\exp(-\boldsymbol{C}/\lambda)\boldsymbol{v}^{(t_{in}-1)})$
        Update $\boldsymbol{v}^{(t_{in})} = \boldsymbol{v}/((\exp(-\boldsymbol{C}/\lambda)^\top \boldsymbol{u}^{(t_{in})})$
        Update $\Delta_v = \sum |\boldsymbol{v}^{(t_{in})} - \boldsymbol{v}^{(t_{in}-1)}|/N$
        if $\Delta_v < \delta$ then
            break
        end if
    end for
    Obtain optimal transport plan as $\boldsymbol{T}_k^* = \text{diag}(\boldsymbol{u}^{(t)}) \exp(-\boldsymbol{C}_k/\lambda)\text{diag}(\boldsymbol{v}^{(t)})$,
    Calculate the OT distance $d_{\text{OT}}(k) =< \boldsymbol{T}_k^*, \boldsymbol{C}_k >$
    Calculate the classification probability $p_{\text{OT}}(y = k|\boldsymbol{x})$ with the OT distance
    return $k^* = \max_k p_{\text{OT}}(y = k|\boldsymbol{x})$
end for
```

A2 EXPERIMENTAL DETAILS

A2.1 DATASET DETAILS

The datasets we used in the experiments follow CoOp (Zhou et al., 2021b), which include 11 datasets for few-shot visual recognition and 4 ImageNet-based datasets for generalization (robustness) evaluation. The details of each dataset are shown in Table A1, including the number of classes, the sizes of training and testing sets, and the original tasks.

A2.2 IMPLEMENTATION DETAILS

The original CoOp method has different versions with different class token positions and parameter initialization strategies. As the performance gap among different versions is limited, we directly chose one of them as our baseline, where the token position is “end”, the parameter initialization strategy is “random”, and the length of learnable context tokens is set as 16. Following the widely used setting in (Zhou et al., 2021b; 2022; Gao et al., 2021; Zhang et al., 2021a), we also chose RN50 (He et al., 2016) as the backbone network of the visual branch. All the code of our method is based on CoOp, which adopted the SGD optimizer with 0.002 initial learning rate, CosineAnnealingLR schedule, and a warmup trick with 1e-5 learning rate. We also followed the epoch strategy to train more epochs for more shots. For small datasets such as FGVCAircraft, OxfordFlowers, and StanfordCars, the batch size is set as 32, while for the larger dataset such as Imagenet and SUN397, the batch size is set as 128.

We apply $N = 4$ prompts for each category and use $M = 7 \times 7$ due to the feature map size. We set the hyper-parameters in the Sinkhorn distances algorithm (Cuturi, 2013) as $\lambda = 0.1$ for all the datasets. We set the maximum iteration number of the inner loop as 100 and will early stop the iteration when the average absolute update value $\Lambda < 0.01$. We initialize all values in the vector v and μ as $1/N$ and $1/M$ respectively. All models are conducted on the Pytorch (Paszke et al., 2019) 1.7.1 and trained on 4 NVIDIA A100 GPUs. We repeated the experiments three times with different seeds and reported the average.

Table A1: The detailed statistics of datasets used in experiments.

Dataset	Classes	Training size	Testing size	Task
Caltech101 (Fei-Fei et al., 2004)	100	4,128	2,465	Object recognition
DTD (Cimpoi et al., 2014)	47	2,820	1,692	Texture recognition
EuroSAT (Helber et al., 2019)	10	13,500	8,100	Satellite image recognition
FGVCAircraft (Maji et al., 2013)	100	3,334	3,333	Fine-grained aircraft recognition
Flowers102 (Nilsback & Zisserman, 2008)	102	4,093	2,463	Fine-grained flowers recognition
Food101 (Bossard et al., 2014)	101	50,500	30,300	Fine-grained food recognition
ImageNet (Deng et al., 2009)	1,000	1.28M	50,000	Object recognition
OxfordPets (Parkhi et al., 2012)	37	2,944	3,669	Fine-grained pets recognition
StanfordCars (Krause et al., 2013)	196	6,509	8,041	Fine-grained car recognition
SUN397 (Xiao et al., 2010)	397	15,880	19,850	Scene recognition
UCF101 (Soomro et al., 2012)	101	7,639	3,783	Action recognition
ImageNetV2 (Recht et al., 2019)	1,000	-	10,000	Robustness of collocation
ImageNet-Sketch (Wang et al., 2019)	1000	-	50,889	Robustness of sketch domain
ImageNet-A (Hendrycks et al., 2019)	200	-	7,500	Robustness of adversarial attack
ImageNet-R (Hendrycks et al., 2020)	200	-	30,000	Robustness of multi-domains

Table A2: The few-shot visual recognition accuracy on 11 datasets.

Dataset	Methods	1 shot	2 shots	4 shots	8 shots	16 shots
Caltech101	**PLOT**	89.83 ± 0.33	90.67 ± 0.21	90.80 ± 0.20	91.54 ± 0.33	92.24 ± 0.38
	CoOp	87.51 ± 1.02	87.84 ± 1.10	89.52 ± 0.80	90.28 ± 0.42	91.99 ± 0.31
DTD	**PLOT**	46.55 ± 2.62	51.24 ± 1.95	56.03 ± 0.43	61.70 ± 0.35	65.60 ± 0.82
	CoOp	43.62 ± 1.96	45.35 ± 0.31	53.94 ± 1.37	59.69 ± 0.13	62.51 ± 0.25
EuroSAT	**PLOT**	54.05 ± 5.95	64.21 ± 1.90	72.36 ± 2.29	78.15 ± 2.65	82.23 ± 0.91
	CoOp	52.12 ± 5.46	59.00 ± 3.48	68.61 ± 3.54	77.08 ± 2.42	83.69 ± 0.47
FGVCAircraft	**PLOT**	17.90 ± 0.09	18.94 ± 0.44	22.36 ± 0.42	26.17 ± 0.29	31.49 ± 0.89
	CoOp	8.59 ± 5.79	16.52 ± 2.38	20.63 ± 2.46	26.63 ± 0.86	31.43 ± 0.96
Flowers102	**PLOT**	71.72 ± 0.97	81.19 ± 0.79	87.82 ± 0.20	92.43 ± 0.25	94.76 ± 0.34
	CoOp	67.98 ± 1.98	77.58 ± 1.46	86.10 ± 1.05	91.27 ± 0.83	94.49 ± 0.40
FOOD101	**PLOT**	77.74 ± 0.47	77.70 ± 0.02	77.21 ± 0.43	75.31 ± 0.30	77.09 ± 0.18
	CoOp	74.25 ± 1.52	72.61 ± 1.33	73.49 ± 2.03	71.58 ± 0.79	74.48 ± 0.15
ImageNet	**PLOT**	59.54 ± 0.16	60.64 ± 0.06	61.49 ± 0.23	61.92 ± 0.09	63.01 ± 0.13
	CoOp	56.99 ± 1.03	56.40 ± 0.87	58.48 ± 0.47	60.39 ± 0.57	61.91 ± 0.17
OxfordPets	**PLOT**	87.49 ± 0.57	86.64 ± 0.63	88.63 ± 0.26	87.39 ± 0.74	87.21 ± 0.40
	CoOp	85.99 ± 0.28	82.22 ± 2.15	86.65 ± 0.97	85.36 ± 1.00	87.02 ± 0.89
StanfordCars	**PLOT**	56.60 ± 0.36	57.52 ± 0.71	63.41 ± 0.29	67.03 ± 0.50	72.80 ± 0.75
	CoOp	55.81 ± 1.67	58.41 ± 0.43	62.74 ± 0.16	67.64 ± 0.06	73.60 ± 0.19
SUN397	**PLOT**	62.47 ± 0.43	61.71 ± 0.65	65.09 ± 0.43	67.48 ± 0.04	69.96 ± 0.24
	CoOp	60.12 ± 0.82	59.60 ± 0.76	63.24 ± 0.63	65.77 ± 0.02	68.36 ± 0.66
UCF101	**PLOT**	64.53 ± 0.70	66.83 ± 0.43	69.60 ± 0.67	74.45 ± 0.50	77.26 ± 0.64
	CoOp	62.13 ± 1.14	64.05 ± 0.99	67.79 ± 0.71	72.71 ± 0.50	76.90 ± 0.50
Average	**PLOT**	62.59 ± 1.13	65.23 ± 0.72	68.60 ± 0.52	71.23 ± 0.51	73.94 ± 0.54
	CoOp	59.56 ± 2.06	61.78 ± 1.39	66.47 ± 1.29	69.85 ± 0.69	73.33 ± 0.42

A2.3 FEW-SHOT RECOGNITION ACCURACY

In Section 4.3.1, we provide a line chart to show and compare the performance of **PLOT** and CoOp. Here, we provide detailed performance results on all 11 few-shot recognition datasets in Table A2, where we use gray for our method and white for CoOp. To highlight, we respectively use dark cyan and light cyan to represent the performance of **PLOT** and CoOp on the average of all 11 datasets. We repeat all experiments 3 times and report the mean and standard deviation in the table.

A2.4 ABLATION STUDIES DETAILS

In this section, we provide more details about the different variants in Table 2. We compare **PLOT** with the other 6 baseline methods briefly described below:

- CoOp: CoOp is the baseline method that only learns a single prompt and matches this single prompt and the global visual feature. We apply the officially released code to reproduce this method.
- “G”: In this paper, we propose to explore whether we can learn multiple prompts for more comprehensive textual representation and fine-grained visual-textual alignment. “G” denotes that we build multiple prompts (similar to our **PLOT**) and learn them by matching them with the single global visual feature.
- “G+V”: Matching all local prompts to a single visual feature will reduce the diversity of the learned prompts. To improve the variety of learned prompts, “G+V” further adds an objective function to increase the distances between every two prompts.
- “G+E”: “G+E” is also a method to increase the variety of prompts by separated initializations. It applies predefined different initializations to replace the random initialization, such as "a photo of a", "this is a photo", "this is a", and "one picture of a".
- “M”: One key difference between **PLOT** and CoOp is to utilize the feature map for more fine-grained information. To evaluate whether our improvement mainly comes from using a feature map, we design a method “M”, which removes the OT distance of **PLOT** and matches local visual features and multiple textual prompts by the average distance of each visual-textual pair.
- “M+V”: Similar to “G+V”, we add an objective function to increase the distances between every two prompts to the method “M” to increase the variety of prompts.

A2.5 BASE-TO-NEW RESULTS

To investigate the generalization of our method for other baseline prompt-learning-based methods, we apply our **PLOT** to CoCoOp Zhou et al. (2022), by learning multiple textual prompts (e.g. N=4) instead of the single prompt in CoCoOp. We name it **CoPLOT**. Specially, we learn multiple prompts and use the same meta-network for all local prompts. Then we apply the Optimal Transport to calculate the distance between multiple local prompts and local visual features. We evaluate both CoCoOp and CoPLOT in the setting of "base-to-new" and implement them using the same RN50 backbone. The results on the 11 datasets with 16 shots are provided in Table A3. We observe that PLOT achieves improvement on most datasets and on average, which demonstrates that it can be applied to different prompt-learning-based methods. For example, on average,**PLOT** achieves almost 3% improvement on the "new" side without the reduction of "base" performance. It suggests that these two methods are complementary: CoCoOp proposes a conditional formulation that uses each image feature as the context condition to refine the single prompt, while PLOT aims to learn multiple prompts.

A2.6 ZERO-SHOT SETTING ANALYSIS

PLOT can not benefit in the setting of zero-shot learning. Below we provide some experimental details and corresponding analysis. CLIP shows that manually designing the prompts can still achieve good performance. We obtain 7 prompts by prompt engineering on the ImageNet dataset and can further ensemble them to obtain $\mathbf{60.38}\%$ top 1 accuracy. In this section, we replace the cosine distance between the global visual feature and prompt ensemble with the OT distance between the feature map and all 7 prompts. However, without any learning, the OT distance only obtains $\mathbf{58.78}\%$ accuracy. It is a limitation of the **PLOT** to still need few-shot data for optimization, which cannot be directly applied in the zero-shot setting. We argue there are two reasons why the OT distance does not work without learning: 1) prompt engineering selects prompts based on the global feature and cosine distance, instead of OT distance with feature map; 2) all these selected prompts are close to the global feature and lack the complementarity.

Table A3: **Comparison of CoCoOp Zhou et al. (2022) and CoPLOT(ours) in the base-to-new generalization setting**. All methods are implemented with RN50 backbone and evaluated with 16 shots. We report the performance of the base classes, new classes, and the mean of them. We show that PLOT can be applied to CoCoOp Zhou et al. (2022) and achieve improvement.

(a) **Average** .

	Base	New	H
CoCoOp	75.7	64.6	70.2
CoPLOT	75.9	67.6	71.8

(b) ImageNet.

	Base	New	H
CoCoOp	68.3	63.1	65.7
CoPLOT	68.2	63.1	65.7

(c) Caltech101.

	Base	New	H
CoCoOp	95.0	90.0	92.5
CoPLOT	95.4	90.9	93.2

(d) OxfordPets.

	Base	New	H
CoCoOp	92.3	94.6	93.5
CoPLOT	92.1	95.9	94

(e) StanfordCars.

	Base	New	H
CoCoOp	61.8	65.3	63.6
CoPLOT	63.2	66.5	64.9

(f) Flowers102.

	Base	New	H
CoCoOp	91.2	67.5	79.4
CoPLOT	89.6	69.2	79.4

(g) Food101.

	Base	New	H
CoCoOp	85.0	86	85.5
CoPLOT	85.0	85.2	85.1

(h) FGVCAircraft.

	Base	New	H
CoCoOp	25.5	25.7	25.6
CoPLOT	25.6	26.6	26.1

(i) SUN397.

	Base	New	H
CoCoOp	75.1	73.6	74.4
CoPLOT	75.2	73.2	-

(j) DTD.

	Base	New	H
CoCoOp	73.1	50.0	61.6
CoPLOT	72.6	51.4	62.0

(k) EuroSAT.

	Base	New	H
CoCoOp	88.9	33.5	61.2
CoPLOT	91.0	55.3	73.2

(l) UCF101.

	Base	New	H
CoCoOp	76.5	61.6	69.1
CoPLOT	77.4	66.2	71.8

Table A4: The training and inference time comparison.

Settings	CoOp	**PLOT** (N=1)	**PLOT** (N=2)	**PLOT** (N=4)	**PLOT** (N=8)
Training Time (s)	1.127	1.135	1.148	1.182	1.267
Inference Time (images/s)	719.1	714.4	690.7	653.0	519.8

A2.7 COMPUTATION COST EVALUATION

As shown in Table A4, we provide the comparison of the training time and inference seed of the baseline method CoOp (Zhou et al., 2021b) and our **PLOT** with the different number of prompts. We report the one-epoch time training on the 1-shot setting of the Food101 (Bossard et al., 2014) dataset and the number of images processed by the model in 1 second. Taking $N = 4$ as an example, **PLOT** only reduces the 9.2% inference speed and requires an extra 4.9% training time, which is acceptable given the performance improvement.

A3 VISUALIZATION

A3.1 MORE ANALYSIS ON VISUALIZATION

In this section, we provide some visualization examples of the transport plans $\boldsymbol{T}$ related to different prompts (N=4). We translate each transport plan into colorful heatmaps and resize them to their original size and combine them with the raw image. As shown in Figure 4, we provide the heatmaps of 4 categories in ImageNet. We observe that different transport plans highlight different regions of the image, which demonstrates that the learned multiple prompts are complementary. For the class “Brambling”, the prompts respectively focus on the head, tail, wing, and environment. For “Dog Sled”, the prompts are related to dogs, the sled, some ties, and the snow environment.

Table A5: The nearest words for 16 context vectors of all $N = 4$ prompts learned by `PLOT` . N/A means non-Latin characters.

Number	Prompt 1	Prompt 2	Prompt 3	Prompt 4
1	ag	pa	trying	gaz
2	flint	as	field	**white**
3	leaving	wit	N/A	t
4	sot	l	icons	ario
5	tint	N/A	eclub	safe
6	tar	yl	indiffe	class
7	attn	N/A	ts	represented
8	2	job	cold	attend
9	rollingstones	built	yeah	vie
10	N/A	brought	band	recognized
11	N/A	or	love	old
12	bel	j	late	stel
13	**head**	ag	industry	awhile
14	artifact	bad	N/A	ded
15	an	chie	across	these
16	5	in	actual	visiting

A3.2 VISUALIZATION OF FAILURE CASES

To better understand the method and further discover the reason for the failure cases. we visualize the attention maps of some failure cases. As shown in Figure A1, we showed two failure examples with class "2000 AM General Hummer" in the StanfordCars dataset. During the training, we set the number of prompts as 4, but in these visualization results, we found that some of the learned prompts remarkably coincide with each other. These prompts can be roughly divided into two classes: Foreground and Background. For example, in both images, prompts 2 (right top) and 3 (left down) focus on the foreground car, while the others focus on the background. It demonstrates that not all classes have multiple complementary attributes, which motivates us to go further to learn the dynamic local prompts numbers to reduce the computational load in the future.

A3.3 INTERPRETATION OF TEXT PROMPTS

The learned prompts are difficult to be understood by humans since the parameters are optimized in the continuous space (Zhou et al., 2021b). CoOp proposes to use the word which is nearest to learned prompts in the embedding space to visualize the prompts. Following this manner, we show the nearest words of our learned prompts in Table A5. Similar to CoOp, most words can not be directly understood by human logic. However, we still find the relations between the learned prompts and the corresponding optimal transport plan. As shown in Figure 4 in the main paper, we can observe that the optimal transport plan for Prompt 1 always focuses on the “head”, such as the head of “brambling”, the head of “rooster”, and even the head of “aircraft carrier”. It is because the word “head” is in Prompt 1. Similarly, we can find that Prompt 4 prefers the white part of images, such as the white environment in the image of “brambling” and the snow in the image of “dog sled”. It demonstrates that the learned multiple prompts focus on different characteristics of categories.

A3.4 T-SNE OF PROMPTS

To better understand the learned prompts, we provide a visualization with T-SNE Van der Maaten & Hinton (2008) for the learned textual prompts. Specifically, we randomly select 10 classes from ImageNet and generate the textual embedding with our learned prompts. Then, we obtain 4×10 embeddings with dimension $d = 1024$. Then we apply the T-SNE to reduce the dimension and visualize the embeddings. As shown in Figure A2, the textual embeddings of the same class with different prompts are clustered well. Besides, despite being well clustered, we found that the textual embeddings also have intra-diversities.

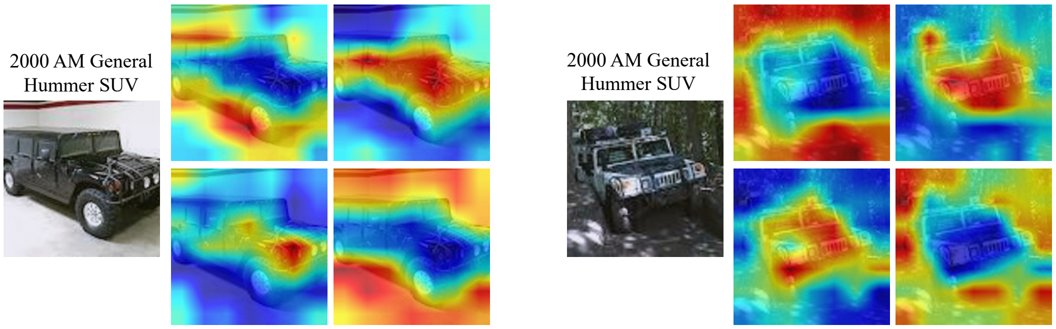

Figure A1: Failure Visualization. We provide the heatmaps of transport plan T related to each prompt on 2 failure examples in the StanfordCars dataset.

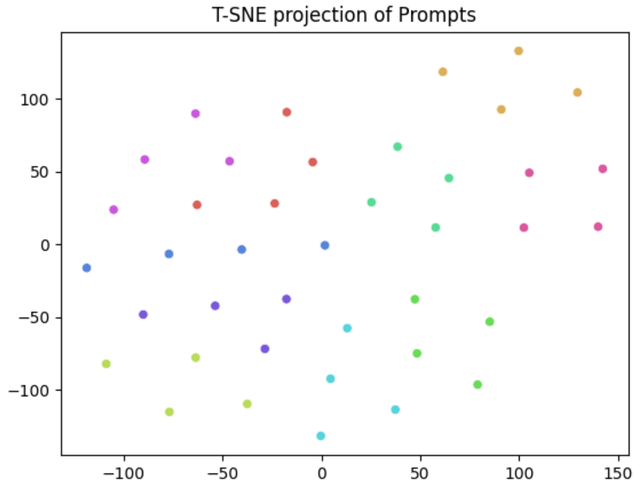

Figure A2: T-SNE Visualization of 10 classes with different prompts. We apply the T-SNE for the embeddings of 10 randomly selected classes in ImageNet with different prompts. Different color denotes different classes. We observe that the textual embeddings cluster well.