Troika: Multi-Path Cross-Modal Traction for Compositional Zero-Shot Learning

Supplementary Material

A. Experimental Details

In this section, we give more details about the architecture, training and evaluation for reference.

A.1. Visual Adapter

Here we detail how we introduce Adapter [9] into the Transformer-based image encoder. Specifically, we insert small learnable modules (*i.e.*, adapters) after the multi-head self-attention layer and the feed-forward network inside each Transformer block. Given an input feature $\mathbf{x} \in \mathbb{R}^d$, the adapter module uses a down-projection with the parameter matrix $\mathbf{W}^{down} \in \mathbb{R}^{d\times r}$ to project the feature to a lower-dimensional space specified by the bottleneck dimension r ($r \ll d$), followed a nonlinear activation function $\sigma(\cdot)$, and an up-projection with the parameter matrix $\mathbf{W}^{up} \in \mathbb{R}^{r\times d}$. Adopting a residual connection design, the overall computation of the adapter module is defined as

$$\text{Adapter}(\mathbf{x}) = \mathbf{x} + \sigma(\mathbf{x}\mathbf{W}^{down})\mathbf{W}^{up},$$

where $\sigma(\cdot)$ is implemented as GELUs [8]. Keeping the original image encoder frozen, we only optimize the parameters of these inserted adapters during training.

A.2. Hyperparameters

Tab. 8 lists the hyperparameters that differ on each dataset and are determined with the validation performance. For other hyperparameters, the CLIP's pre-trained word embeddings of "a photo of" are used to initialize all three prefixes. For Adapter inserted into the image encoder, the bottleneck dimension r is set to 64, and the dropout rate is set to 0.1. In the cross-modal traction module, the feed-forward network first expands the dimension of the input features to 4× its original value, and then shrinks it back. The number of attention heads h is 12, and the dimension of the single-head attention d^{attn} is 64. The strength parameter vector $\boldsymbol{\lambda}$ is initialized with the scalar value 0.1. During training, we use the Adam [17] optimizer and decay the learning rate of all trainable parameters by 0.5 every 5 epochs.

A.3. Feasibility Calibration for Open-World Setting

Following [25, 29], we apply the post-training feasibility calibration to filter out infeasible compositions that might be present in the open-world evaluation. The calibration assumes that similar objects share similar states while dissimilar objects are unlikely to share states. Therefore, given a candidate pair $c = \langle s, o\rangle$, similarities between the objects

Hyperparameter	MIT-States	UT-Zappos	C-GQA
Learning rate	10^{-4}	2.5×10^{-4}	1.25×10^{-5}
Batch size	64	64	64
Number of epochs	10	15	15
Attribute dropout rate	0.3	0.3	0
CMT dropout rate	0.1	0	0
Weight decay	10^{-5}	10^{-5}	10^{-5}
Number of CMT layers N	3	2	2
Coefficients $\alpha^c, \alpha^s, \alpha^o$	1, 1, 1	1, 1, 1	1, 0.1, 0.1

Table 8. **Hyperparameters for different datasets.**

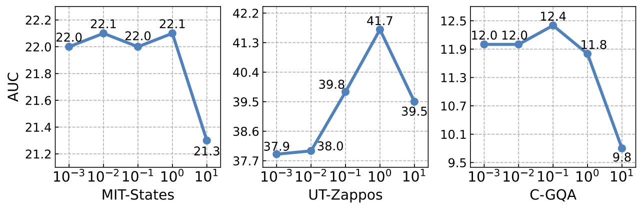

Figure 7. **Sensitivity analysis on loss weighting coefficients α^s and α^o.**

can be computed as

$$\rho_o(s,o) = \max_{\hat{o}\in\mathcal{O}^{se}} \frac{\phi(o)\cdot\phi(\hat{o})}{\|\phi(o)\|\|\phi(\hat{o})\|},$$

where $\mathcal{O}^{se}$ is the object set that contains those paired with the state s in seen compositions. And $\phi(\cdot)$ is an embedding function that maps the primitive to a pre-trained embedding, which is implemented with GloVe embeddings [31]. We also compute similarities between the states as $\rho_s(s,o)$ in the same way. Next, the feasibility score for the composition (s,o) can be computed by combining the two similarities with a mean pooling function μ:

$$\rho(s,o) = \mu(\rho_o(s,o), \rho_s(s,o)).$$

Finally, by only considering compositions above a threshold T, infeasible compositions can be filtered out. And the inference of *Troika* now becomes

$$\hat{c} = \mathop{\arg\max}_{c_{i,j}\in\mathcal{C}^{tgt},\rho(s,o)>T} \left(\widetilde{p}(c_{i,j}|x)\right),$$

where the threshold T is calibrated based on the performance on the validation set.

B. Hyperparameter Sensitivity Analysis

In this section, we vary some key hyperparameters to examine how sensitive the proposed *Troika* is to them.

Loss weighting coefficients α^s and α^o. In Fig. 7, after fixing the weighting coefficient of the composition branch α^c as 1, we vary the loss coefficients on the state and object branches, *i.e.*, α^s and α^o. While setting α^s and α^o as 1

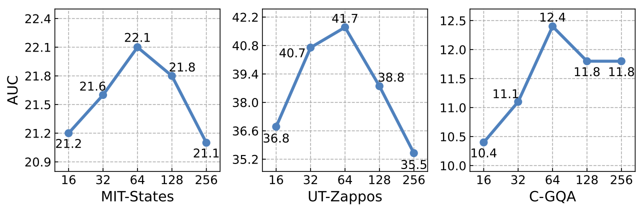

Figure 8. **Sensitivity analysis on Adapter bottleneck dimension r.**

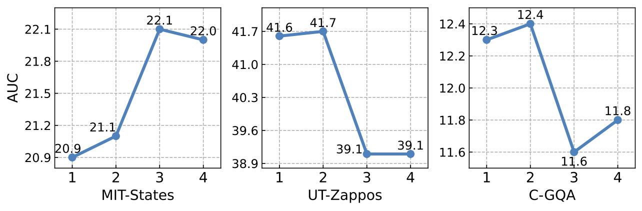

Figure 9. **Sensitivity analysis on the number of *Cross-Modal Traction* module layers N.**

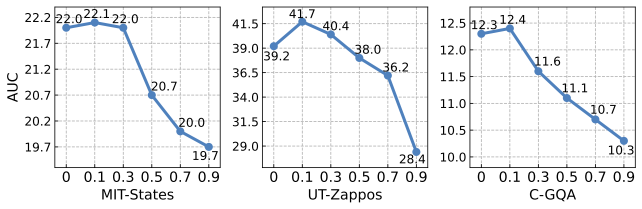

Figure 10. **Sensitivity analysis on initialization value of $\boldsymbol{\lambda}$.**

can achieve the best result on MIT-States and UT-Zappos, a smaller value as 0.1 is better on C-GQA, where learning in a composable way deserves a higher priority in more complex scenarios. Another observation is that setting α^s and α^o as 10 leads to a significant drop, which shows that it is detrimental to give the primitive branches a higher status than the composition branch during training.

Adapter bottleneck dimension r. In Fig. 8, we vary the bottleneck dimension r of the introduced adapters. For all three datasets, 64 is an optimal choice for r. And a higher r may cause a performance crash on smaller datasets like UT-Zappos due to over-fitting.

Number of CMT module layers N. In Fig. 9, a 2-layer *Cross-Modal Traction* module is optimal for C-GQA, while setting N as 3 is better for the other two datasets. Although not affecting its leadership over baseline methods, it is observed that further deepening the module may result in a loss of performance for *Troika*.

Initialization of $\boldsymbol{\lambda}$. In Fig. 10, we first vary the initialization value of the trainable parameter vector $\boldsymbol{\lambda}$, which controls the strength of the cross-modal traction. On all three datasets, initializing $\boldsymbol{\lambda}$ with 0.1 achieves the highest AUC, and steadily increasing the initialization value leads to a continuous decline in performance. We attribute this phenomenon to the fact that aggressive traction may destroy the cross-modal alignment already established by pre-training. Therefore, a larger initialization value for $\boldsymbol{\lambda}$ increases the difficulty of optimization. We also display the statistics of the trained $\boldsymbol{\lambda}$ values in Tab. 9, which remain near the initial 0.1. As a conclusion, the current initialization settings for $\boldsymbol{\lambda}$ are appropriate.

$\boldsymbol{\lambda}$ value	MIT-States	UT-Zappos	C-GQA
mean	0.092	0.092	0.101
max	0.113	0.106	0.107
min	0.079	0.079	0.099

Table 9. **Statistics of the trained $\boldsymbol{\lambda}$.**

CMT in *Troika*	UT-Zappos				C-GQA			
	S	U	HM	AUC	S	U	HM	AUC
$\mathbf{t} \leftarrow \mathbf{t} + \boldsymbol{\lambda} \cdot \widetilde{\mathbf{t}}$	**66.8**	**73.8**	**54.6**	**41.7**	**41.0**	**35.7**	**29.4**	**12.4**
$\mathbf{t} \leftarrow \widetilde{\mathbf{t}}$	64.8	71.9	48.9	36.1	40.4	31.6	28.2	11.1

Table 10. **Ablation on the implementation of the CMT module.**

$\boldsymbol{\lambda}$		UT-Zappos				C-GQA			
Vectorized	Trainable	S	U	HM	AUC	S	U	HM	AUC
✓	✓	**66.8**	**73.8**	**54.6**	**41.7**	**41.0**	**35.7**	**29.4**	**12.4**
✓		66.2	73.5	54.2	41.3	39.8	33.2	29.1	11.5
	✓	65.2	73.1	52.9	40.1	40.8	35.0	28.5	12.0

Table 11. **Ablation on the strength parameter $\boldsymbol{\lambda}$.**

Prefix initialization. In Tab. 12, we report the results of trying several combinations of initialization for prompt prefixes from different branches, which confirm that *Troika* might be marginally sensitive to the prefix initialization. In our experiments, we have selected a simplest combination as the default initialization for convenience.

C. Additional Ablation Study

In this section, we add more ablation experiments to analyze the effects of each design in *Troika*.

Ablation on strength parameter $\boldsymbol{\lambda}$. Since an aggressive traction may destroy the cross-modal alignment already established by pre-training, the *Cross-Modal Traction* module summarizes the features with a small weight for $\widetilde{\mathbf{t}}$, avoiding its dominance. We first prove the necessity of $\boldsymbol{\lambda}$ and the residual structure in Tab. 10, where directly replacing $\mathbf{t}$ with $\widetilde{\mathbf{t}}$ leads to a decrease compared to the current implementation. In Tab. 11, we ablate $\boldsymbol{\lambda}$ with two adjustments: (1) freeze $\boldsymbol{\lambda}$ after the initialization, and (2) change the parameter vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ to a trainable scalar. We observe that each of both adjustments leads to a drop in performance, which reveals the importance of adaptively scaling the strength of the cross-modal traction performed on each dimension.

Ablation on backbone. Tab. 13 compares *Troika* with other methods when using different ViT-based CLIP backbones. Note that only CLIP [33] and CSP [29] are included in the comparison, as other CLIP-based methods have not reported the results with different backbones. We can observe that our *Troika* consistently outperforms the compared methods, and a larger backbone leads to better performance.

D. Additional Comparison Results

In this section, we present a more comprehensive comparison of our *Troika* to demonstrate its superiority.

branch			MIT-States		UT-Zappos		C-GQA	
c	s	o	HM	AUC	HM	AUC	HM	AUC
"a photo of"	"a photo of"	"a photo of"	**39.3**	22.1	54.6	41.7	29.4	**12.4**
"a photo of"	"the object is"	"the object is"	39.2	**22.4**	**55.6**	40.9	**29.7**	12.0
"the object is"	"the object is"	"the object is"	38.7	21.7	52.8	41.0	28.5	11.5
"a photo of"	"the state of the object is"	"the class of the object is"	39.1	22.2	54.5	**42.7**	28.7	11.6
"a photo of"	"the state is"	"the class is"	38.8	21.4	53.2	38.5	29.6	12.0

Table 12. **Sensitivity analysis on initialization of prefixes.**

		MIT-States				UT-Zappos				C-GQA			
Method	Backbone	S	U	HM	AUC	S	U	HM	AUC	S	U	HM	AUC
CLIP [33]	ViT-B/32	25.1	39.1	21.4	7.5	9.6	42.4	10.0	2.4	7.3	22.1	7.4	1.2
CSP [29]	ViT-B/32	36.4	42.5	28.6	12.4	57.1	57.3	39.3	24.2	30.1	23.4	19.4	5.7
***Troika* (Ours)**	ViT-B/32	**39.5**	**42.8**	**30.5**	**13.9**	**60.5**	**67.4**	**47.3**	**32.3**	**36.3**	**27.2**	**24.4**	**8.4**
CLIP [33]	ViT-L/14	30.2	46.0	26.1	11.0	15.8	49.1	15.6	5.0	7.5	25.0	8.6	1.4
CSP [29]	ViT-L/14	46.6	49.9	36.3	19.4	64.2	66.2	46.6	33.0	28.8	26.8	20.5	6.2
***Troika* (Ours)**	ViT-L/14	**49.0**	**53.0**	**39.3**	**22.1**	**66.8**	**73.8**	**54.6**	**41.7**	**41.0**	**35.7**	**29.4**	**12.4**

Table 13. **Ablation on backbone architecture of CLIP.**

D.1. Efficiency Comparison with SOTA Method

One concern may arise that, compared to the existing single-branch methods, methods following our *Multi-Path* paradigm need to extract and align the multi-modal features for each branch individually, thus requiring more number of parameters and computation. However, we point out that this can be avoided by carefully designing the efficient solutions. For both text and vision feature extraction, the design of *Troika* maintains the idea of parameter efficiency to minimize the number of training parameters. Moreover, for inference, *Troika* only needs to extract the text features for all three branches once for the whole dataset, and the decoupling of the image features occurs after the output projection, which does not require multiple forward computations by the visual encoder. Therefore, when calculating the average inference time for a single sample, the increase due to the *Multi-Path* paradigm is actually much lower than expected. To demonstrate this, we conduct an efficiency comparison with the state-of-the-art DFSP [23] method on the UT-Zappos dataset, and the results are listed in Tab. 14. It is observed that DFSP actually requires more trainable parameters and inference time than *Troika*, due to its reliance on heavy cross-attention and self-attention blocks outside of the CLIP backbone for cross-modal information interaction. As a conclusion, the proposed efficient implementations on the *Multi-Path* paradigm allow our approach to outperform the state-of-the-art methods in terms of recognition accuracy without sacrificing storage and operational efficiency.

Methods	#Params (M) ↓	Inference Time (ms) ↓
DFSP [23]	31.81	18.56
***Troika* (Ours)**	**21.70**	**17.31**

Table 14. **Efficiency comparison between DFSP and *Troika*.** We report the number of trainable parameters and the average inference time. *Troika* is superior in terms of storage and operational efficiency.

D.2. Comparison with Existing CZSL Methods

Baselines. We compare *Troika* with both CLIP-based methods [23, 29, 33, 37, 43] and existing CZSL methods [1, 14, 15, 19, 20, 24–28, 32] with a pre-trained ResNet-18 [7] backbone. For CompCos [24] and Co-CGE [25], we report the results of which version of the models according to the experimental setup, *i.e.*, the closed-world version for the closed-world setting, and the open-world version for the open-world setting.

Results. Tab. 15 reports the closed-world results and Tab. 16 reports the open-world results. Credit to the transferred pre-trained knowledge, we find that CLIP-based methods significantly outperform other CZSL methods in both settings. And our proposed *Troika* achieves the state-of-the-art performance in all cases.

Method	**MIT-States**				**UT-Zappos**				**C-GQA**			
	S	U	HM	AUC	S	U	HM	AUC	S	U	HM	AUC
AoP [28]	14.3	17.4	9.9	1.6	59.8	54.2	40.8	25.9	17.0	5.6	5.9	0.7
LE+ [26]	15.0	20.1	10.7	2.0	53.0	61.9	41.0	25.7	18.1	5.6	6.1	0.8
TMN [32]	20.2	20.1	13.0	2.9	58.7	60.0	45.0	29.3	23.1	6.5	7.5	1.1
SymNet [20]	24.2	25.2	16.1	3.0	49.8	57.4	40.4	23.4	26.8	10.3	11.0	2.1
CompCos [24]	25.3	24.6	16.4	4.5	59.8	62.5	43.1	28.1	28.1	11.2	12.4	2.6
CGE [27]	28.7	25.3	17.2	5.1	56.8	63.6	41.2	26.4	28.1	10.1	11.4	2.3
Co-CGE [25]	27.8	25.2	17.5	5.1	58.2	63.3	44.1	29.1	29.3	11.9	12.7	2.8
SCEN [19]	29.9	25.2	18.4	5.3	63.5	63.1	47.8	32.0	28.9	12.1	12.4	2.9
CVGAE [1]	28.5	25.5	18.2	5.3	65.0	62.4	49.8	34.6	28.2	11.9	13.9	2.8
CANet [36]	29.0	26.2	17.9	5.4	61.0	66.3	47.3	33.1	30.0	13.2	14.5	3.3
CAPE [15]	30.5	26.2	19.1	5.8	60.4	67.4	45.5	31.3	32.9	15.6	16.3	4.2
CLIP [33]	30.2	46.0	26.1	11.0	15.8	49.1	15.6	5.0	7.5	25.0	8.6	1.4
CoOp [43]	34.4	47.6	29.8	13.5	52.1	49.3	34.6	18.8	20.5	26.8	17.1	4.4
CSP [29]	46.6	49.9	36.3	19.4	64.2	66.2	46.6	33.0	28.8	26.8	20.5	6.2
PromptCompVL [37]	48.5	47.2	35.3	18.3	64.4	64.0	46.1	32.2	-	-	-	-
DFSP(i2t) [23]	47.4	52.4	37.2	20.7	64.2	66.4	45.1	32.1	35.6	29.3	24.3	8.7
DFSP(BiF) [23]	47.1	52.8	37.7	20.8	63.3	69.2	47.1	33.5	36.5	32.0	26.2	9.9
DFSP(t2i) [23]	46.9	52.0	37.3	20.6	66.7	71.7	47.2	36.0	38.2	32.0	27.1	10.5
***Troika* (Ours)**	**49.0**±0.4	**53.0**±0.2	**39.3**±0.2	**22.1**±0.1	**66.8**±1.1	**73.8**±0.6	**54.6**±0.5	**41.7**±0.7	**41.0**±0.2	**35.7**±0.3	**29.4**±0.2	**12.4**±0.1

Table 15. **Closed-world results.** For our *Troika*, we report the average performance on 5 random seeds with standard error.

Method	**MIT-States**				**UT-Zappos**				**C-GQA**			
	S	U	HM	AUC	S	U	HM	AUC	S	U	HM	AUC
AoP [28]	16.6	5.7	4.7	0.7	50.9	34.2	29.4	13.7	-	-	-	-
LE+ [26]	14.2	2.5	2.7	0.3	60.4	36.5	30.5	16.3	19.2	0.7	1.0	0.08
TMN [32]	12.6	0.9	1.2	0.1	55.9	18.1	21.7	8.4	-	-	-	-
SymNet [20]	21.4	7.0	5.8	0.8	53.3	44.6	34.5	18.5	26.7	2.2	3.3	0.43
CompCos [24]	25.4	10.0	8.9	1.6	59.3	46.8	36.9	21.3	28.4	1.8	2.8	0.39
CGE [27]	29.6	4.0	4.9	0.7	58.8	46.5	38.0	21.5	28.3	1.3	2.2	0.30
Co-CGE [25]	26.4	10.4	10.1	2.0	60.1	44.3	38.1	21.3	28.7	1.6	2.6	0.37
KG-SP [14]	28.4	7.5	7.4	1.3	61.8	52.1	42.3	26.5	31.5	2.9	4.7	0.78
CVGAE [1]	27.3	9.9	10.0	1.8	58.6	48.4	41.7	22.2	26.6	2.9	6.4	0.7
CLIP [33]	30.1	14.3	12.8	3.0	15.7	20.6	11.2	2.2	7.5	4.6	4.0	0.27
CoOp [43]	34.6	9.3	12.3	2.8	52.1	31.5	28.9	13.2	21.0	4.6	5.5	0.70
CSP [29]	46.3	15.7	17.4	5.7	64.1	44.1	38.9	22.7	28.7	5.2	6.9	1.20
PromptCompVL [37]	48.5	16.0	17.7	6.1	64.6	44.0	37.1	21.6	-	-	-	-
DFSP(i2t) [23]	47.2	18.2	19.1	6.7	64.3	53.8	41.2	26.4	35.6	6.5	9.0	1.95
DFSP(BiF) [23]	47.1	18.1	19.2	6.7	63.5	57.2	42.7	27.6	36.4	7.6	10.6	2.39
DFSP(t2i) [23]	47.5	18.5	19.3	6.8	**66.8**	60.0	44.0	30.3	38.3	7.2	10.4	2.40
***Troika* (Ours)**	**48.8**±0.4	**18.7**±0.1	**20.1**±0.1	**7.2**±0.1	66.4±1.0	**61.2**±1.0	**47.8**±1.3	**33.0**±1.0	**40.8**±0.2	**7.9**±0.2	**10.9**±0.3	**2.70**±0.1

Table 16. **Open-world results.** For our *Troika*, we report the average performance on 5 random seeds with standard error.