NEURAL INFORMATION PROCESSING SYSTEMS

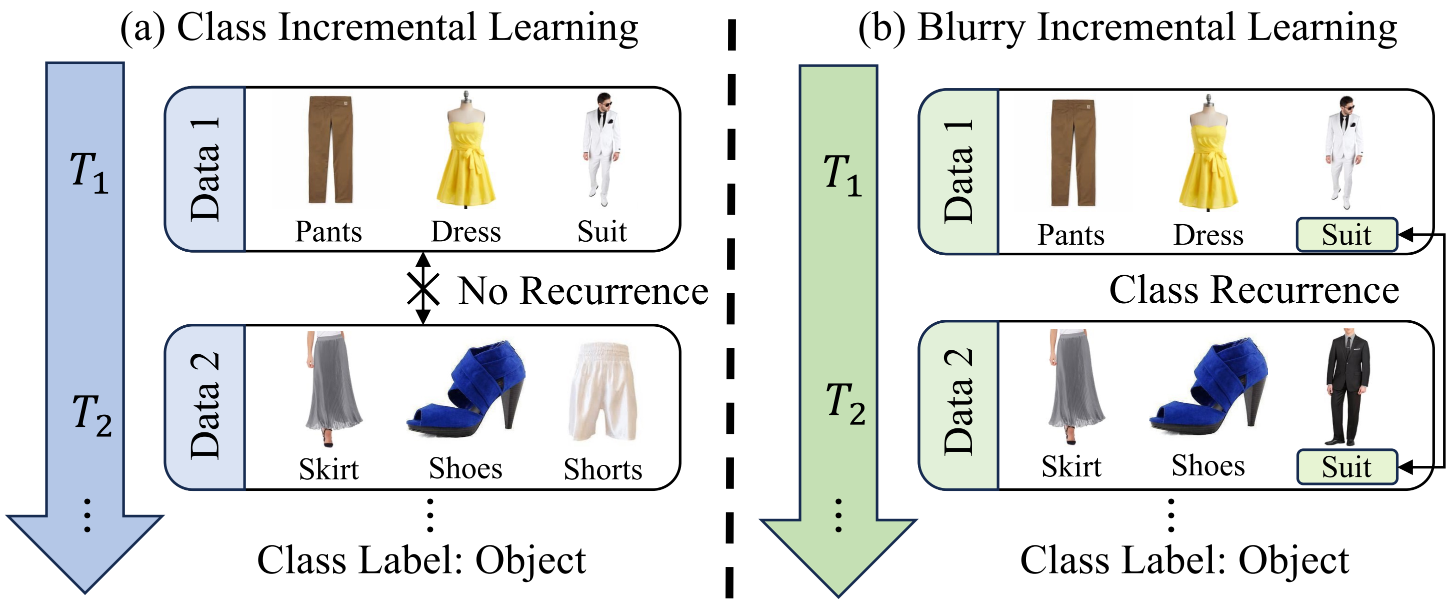
Not Just Object, But State: Compositional Incremental Learning without Forgetting

Yanyi Zhang[1], Binglin Qiu [1], Qi Jia [1], Yu Liu [1*], Ran He [2]

[1]Dalian University of Technology [2]Chinese Academy of Sciences

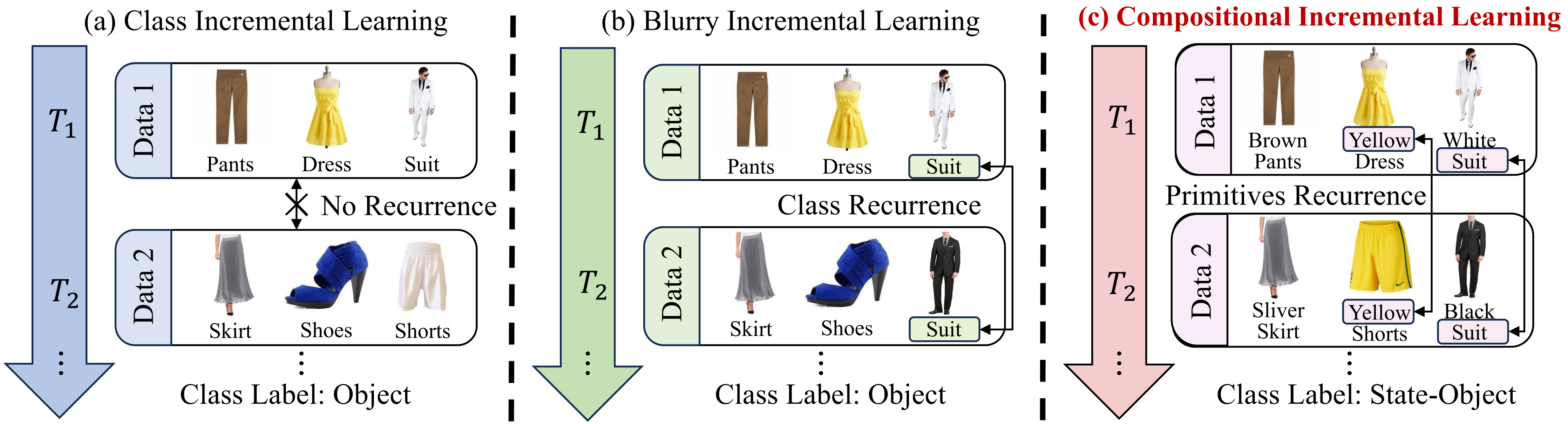
Background

- **Class incremental learning (class-IL)** sets a strict limit on the old classes such that they should not recur in newly incoming tasks.
- **Blurry incremental learning (blur-IL)** allows the recurrence of previous classes in incremental sessions.
- However, both class-IL and blur-IL aims to improve object classification only, **overlooking fine-grained states attached to the objects**.



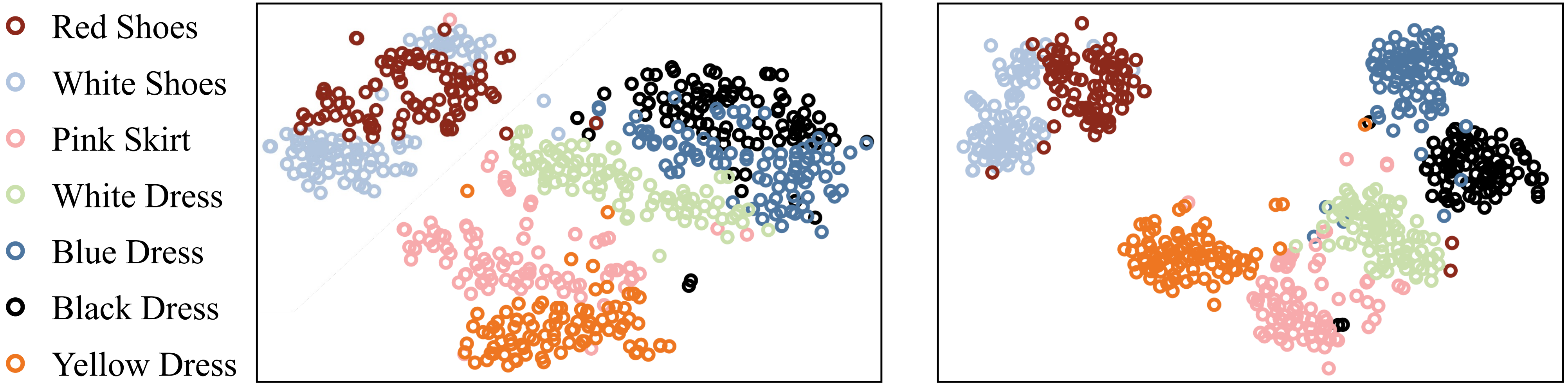
Compositional Incremental Learning

- We conceive a novel task named Compositional Incremental Learning (composition-IL), enabling the model to continually learn new state-object compositions in an incremental fashion.
- The **composition classes are disjoint** across incremental tasks.
- The **primitive classes** encountered in old tasks **are allowed to reappear** in new tasks randomly.



Main obstacle: ambiguous composition boundary

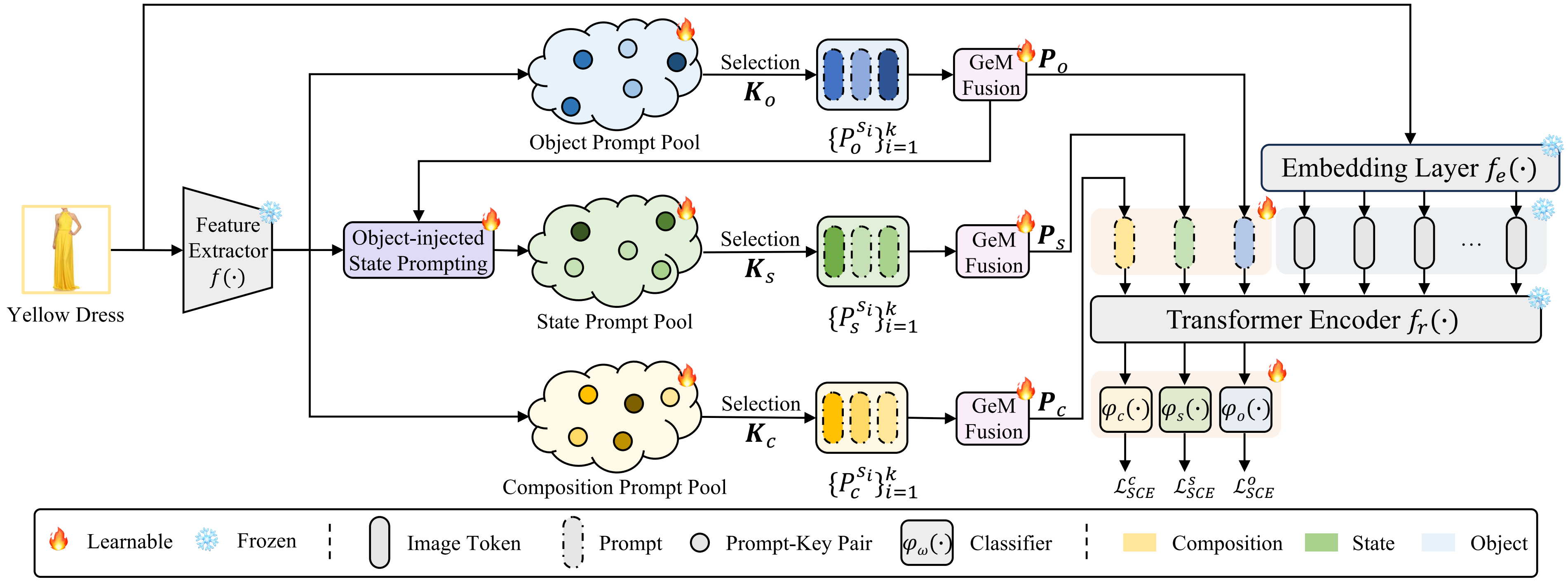
- The existing SOTA methods prioritize the object primitive while neglecting the state primitive.
- Consequently, the **compositions with the same object but with different states** become **ambiguous and indistinguishable**.
- To address it, we propose a new model namely **CompILer** with dedicated loss functions.



(a) L2P   (b) CompILer

CompILer: Compositional Incremental Learner

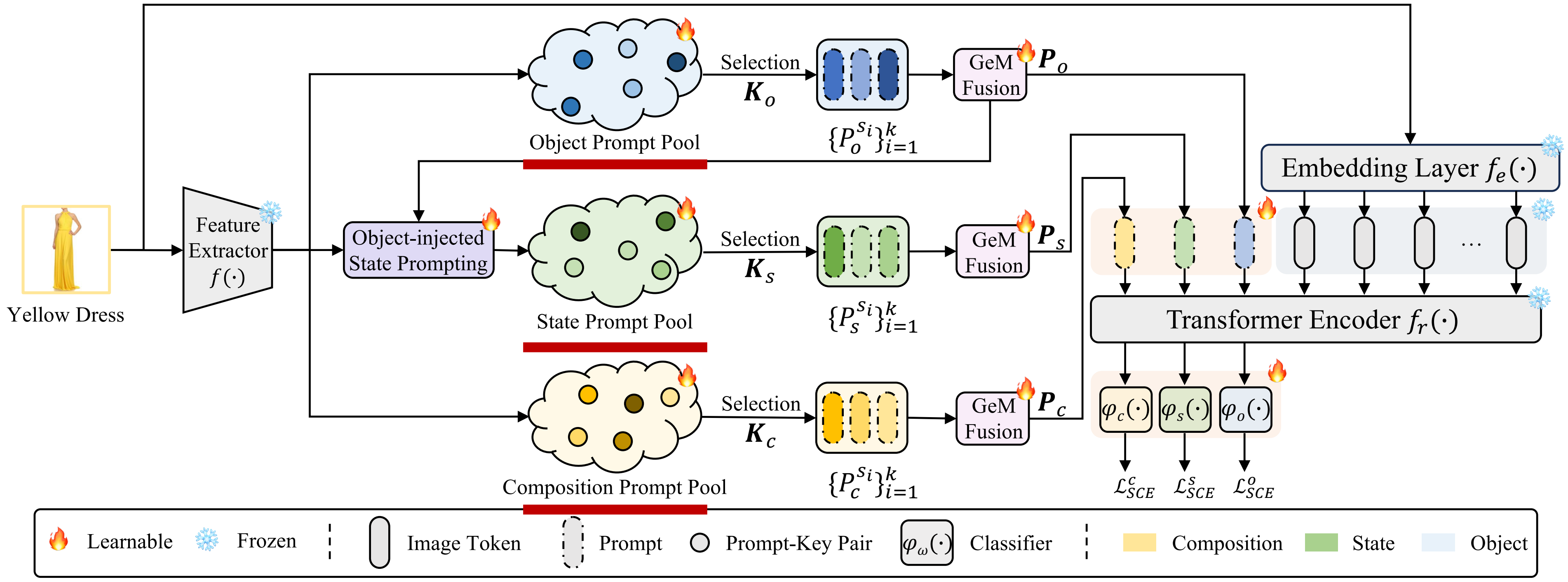
- **Multi-pool Prompt Learning**: construct three prompt pools to learn visual information related to states, objects and their compositions.
- **Object-injected State Prompting**: facilitate more judicious prompt selection within the state prompt pool, alleviating the hurdles posed by state learning.
- **Generalized-mean Prompt Fusion**: learns to achieve an optimal fusion, mitigating the influence of irrelevant information present in the prompts.



Multi-pool Prompt Learning

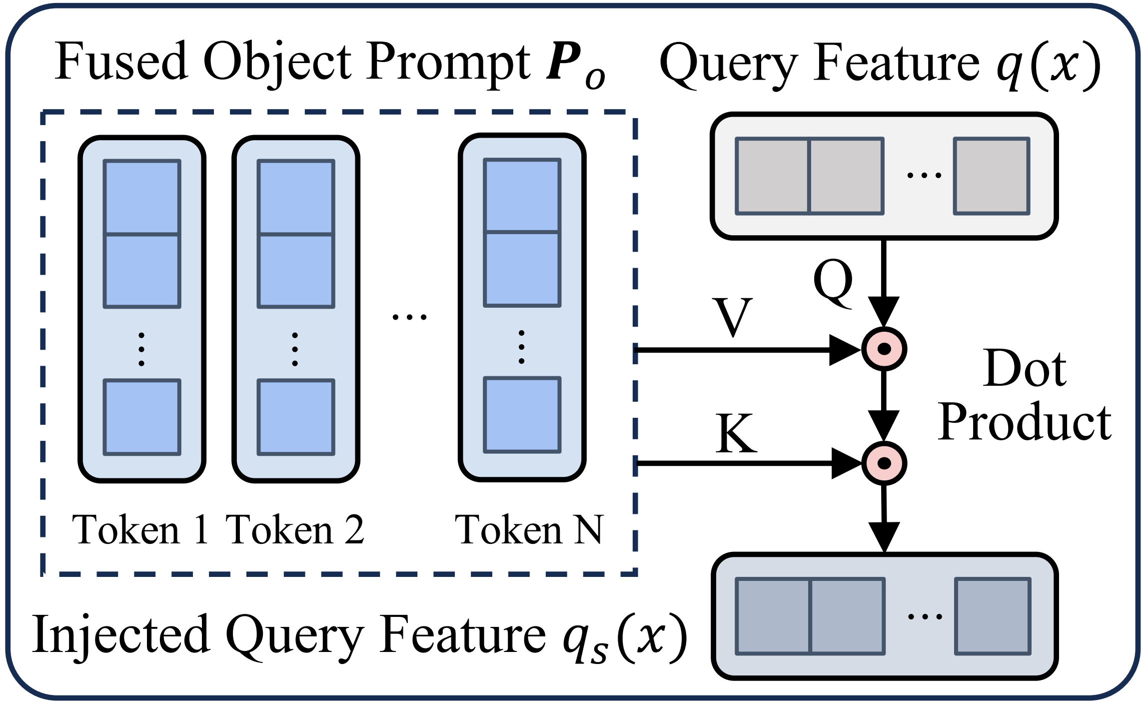
- We construct **three prompt pools** for learning the states, objects and compositions individually.
- To ensure inter-pool prompt **discrepancy** and intra-pool prompt **diversity**, we use directional decoupled loss between any two pools.

$$\mathcal{L}_{dd}^{(i,j)} = \frac{2}{M(M-1)} \sum_{n=1}^{M} \sum_{m=1}^{M} \max\left(0, \theta_{\text{thre}} - \theta_{nm}\right) \qquad \theta_{nm} = \cos^{-1}\left(\frac{(P_i^n)^{\mathrm{T}} P_j^m}{\max(\|P_i^n\|_2, \epsilon) \cdot \max(\|P_j^m\|_2, \epsilon)}\right),$$



Object-injected State Prompting

- Pre-trained backbones are typically trained for object classification, thus under-performing for state representation learning.
- We strategically **inject object prompts** to **guide the selection of state prompts** by cross attention mechanism.
- Query feature serves as Q, while fused object prompt serves as both K and V.

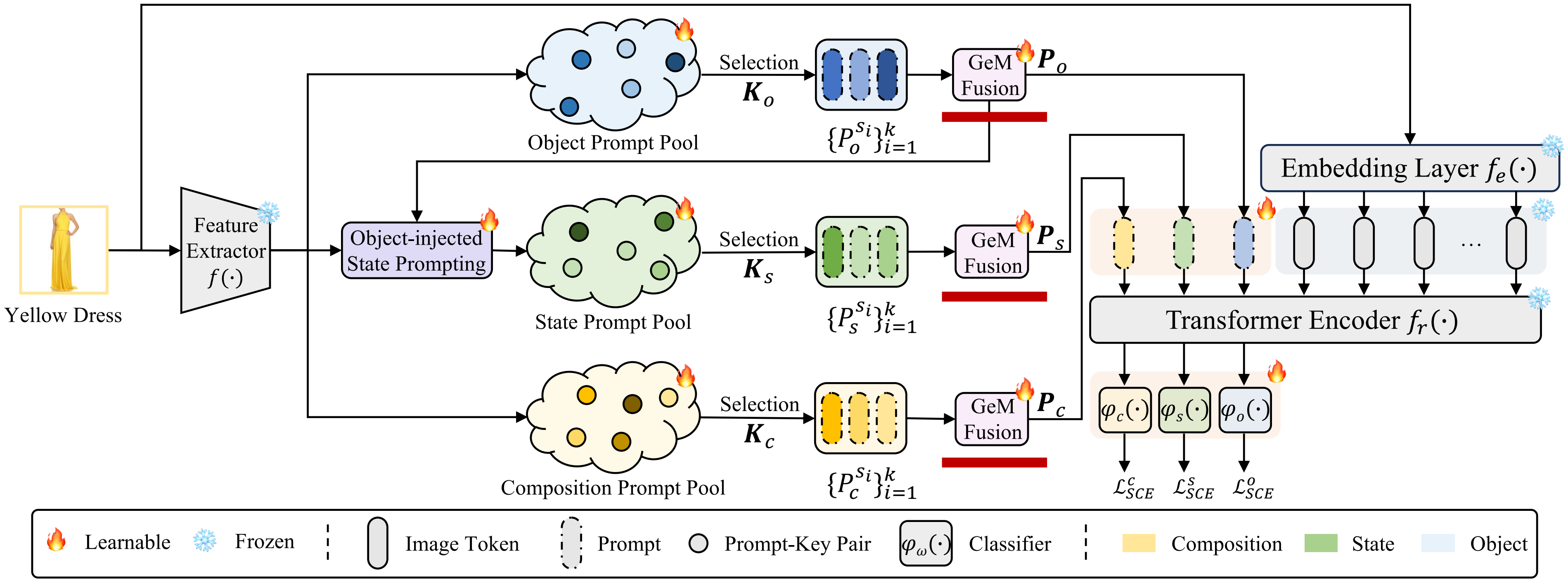


Generalized-mean Prompt Fusion

- Mean pooling overlooks the relative importance of each selected prompt.
- In order to **strengthen useful prompts and eliminate irrelevant ones**, we exploit generalized-mean (GeM) prompt fusion which is given by:

$$\boldsymbol{P}_\omega = \mathrm{GeM}_\omega(P_\omega^{s_1}, P_\omega^{s_2}, \cdots, P_\omega^{s_k}) = \left(\frac{1}{k}\sum_{i=1}^{k} P_\omega^{s_i\,\eta}\right)^{\frac{1}{\eta}}, \omega \in \{s, o, c\}$$

- η is a learnable parameter.



Classification Objectives

- We advocate using a symmetric cross entropy loss, which incorporates reverse cross entropy with vanilla cross entropy, to **mitigate the impact of noisy data**.

$$\mathcal{L}^{\omega}_{SCE} = \mathcal{L}^{\omega}_{CE} + \alpha\mathcal{L}^{\omega}_{RCE}$$

$$\mathcal{L}^{\omega}_{CE} = -\sum_{\omega=1}^{\Omega} q(\omega \mid x) \log p(\omega \mid x), \Omega \in [|\mathcal{S}|, |\mathcal{O}|, |\mathcal{C}|] \quad \mathcal{L}^{\omega}_{RCE} = -\sum_{\omega=1}^{\Omega} p(\omega \mid x) \log q(\omega \mid x), \omega \in \{s, o, c\}$$

- To **establish alignment between the query and the selected prompts**, we optimize a surrogate loss for state, object and composition prompting jointly.

$$\mathcal{L}_{sur} = \sum_{\omega}\sum_{q_\omega} \cos(f_\omega(x), K^{s_i}_{\omega}),\ \omega \in \{s, o, c\}$$

- The total loss for training the whole CompILer model is:

$$\mathcal{L}_{total} = \lambda_1\mathcal{L}_{inter} + \lambda_2\mathcal{L}_{intra} + \lambda_3\mathcal{L}_{sur} + \mathcal{L}_{SCE},$$

Experiments

- New benchmarks:
  - Split-Clothing: a fine-grained clothing dataset.
  - Split-UT-Zappos: a fine-grained shoes dataset.

Dataset	Compositions	States	Objects	Images
Split-Clothing	35	9	8	15.9k
Split-UT-Zappos	80	15	12	28.5k

- Number of incremental tasks:
  - T=5 in Split-Clothing.
  - T=5 or T=10 in Split-UT-Zappos.
- Evaluation metrics:
  - Avg Acc: average accuracy on compositions. Higher is better.
  - FTT: forgetting rate on compositions. Lower is better.
  - State: average accuracy on states. Higher is better.
  - Object: average accuracy objects. Higher is better.
  - HM: harmonic mean between State and Object. Higher is better.

Experiments

- CompILer consistently outperforms all competitors on Avg Acc by a significant margin.
- For FTT scores, CompILer excels previous methods slightly on 5-task Split-Clothing and 5-task Split-UT-Zappos, while falling behind Dual-Prompt and LGCL for the 10-task Split-UT-Zappos.

Datasets	Split-Clothing (5 tasks)		Split-UT-Zappos (5 tasks)		Split-UT-Zappos (10 tasks)	
Metrics	Avg Acc(↑)	FTT(↓)	Avg Acc(↑)	FTT(↓)	Avg Acc(↑)	FTT(↓)
Upper Bound	97.02±0.10	-	68.71±0.41	-	68.71±0.41	-
EWC [10]	47.89±0.87	52.75±0.44	37.59±2.06	55.70±2.76	24.63±0.94	61.31±2.29
LwF [16]	49.96±0.68	44.22±0.53	40.15±0.43	49.61±0.68	30.38±1.41	58.15±0.20
iCaRL [32]	68.65±0.41	31.74±1.89	37.78±2.14	55.06±3.50	31.40±1.96	59.65±2.40
L2P [43]	80.22±0.41	14.23±0.44	42.20±2.18	20.41±2.76	31.65±0.16	31.02±1.62
Deep L2P++[43, 33]	80.55±0.45	12.60±1.90	42.37±0.65	30.10±1.56	30.68±0.35	32.20±1.96
Dual-Prompt [42]	87.87±0.63	7.71±0.25	43.30±0.19	19.41±2.80	33.01±1.65	24.61±1.11
CODA-Prompt [33]	86.35±0.20	8.99±0.71	43.35±0.29	21.76±2.45	31.40±0.36	30.54±2.63
LGCL [7]	87.32±0.10	7.58±0.06	-	-	33.56±0.31	**24.37**±0.56
Sim-CompILer	88.38±0.08	8.01±0.42	45.70±0.68	20.06±0.62	33.30±0.10	30.31±0.03
CompILer	**89.21**±0.24	**7.26**±0.60	**46.48**±0.26	**19.27**±0.75	**34.43**±0.07	28.69±0.82

Experiments

- CompILer consistently outperforms all competitors on state accuracy and HM simultaneously.
- The prompt-free methods achieve higher accuracy in state prediction than object prediction for Split-Clothing. This contrast is because the states in Split-Clothing are color-related descriptions, which are easier to capture with the help of parameter fine-tuning.

Datasets	Split-Clothing (5 tasks)			Split-UT-Zappos (5 tasks)			Split-UT-Zappos (10 tasks)		
Metrics	State	Object	HM	State	Object	HM	State	Object	HM
Upper Bound	97.44±0.08	97.09±0.10	97.26±0.08	75.10±0.10	88.13±0.03	81.90±0.06	75.10±0.10	88.13±0.03	81.90±0.06
EWC [10]	86.49±0.97	52.72±1.30	67.50±0.97	47.95±1.26	76.53±0.91	58.90±0.53	39.29±2.69	67.64±1.97	49.69±2.30
LwF [16]	87.11±0.66	54.57±0.69	67.10±0.33	53.13±1.08	75.48±0.82	62.35±0.31	38.70±2.33	68.90±1.97	49.54±1.30
iCaRL [32]	91.21±1.05	71.70±0.99	80.28±0.74	51.71±0.95	75.03±0.49	61.22±0.78	38.94±2.01	67.10±1.05	49.27±1.58
L2P [43]	83.03±0.42	95.56±0.57	88.85±0.16	52.20±2.92	79.05±0.01	62.87±1.61	42.66±0.87	76.60±0.03	54.80±0.55
Dual-Prompt [42]	90.77±0.25	94.18±0.31	92.45±0.20	52.25±0.77	77.46±0.05	62.40±0.34	44.34±1.61	77.92±0.37	56.51±1.11
LGCL [7]	91.45±0.20	94.87±0.33	93.13±0.10	-	-	-	43.44±0.79	**78.64**±0.64	55.96±0.43
Sim-CompILer	91.15±0.10	96.32±0.02	93.66±0.02	55.93±1.23	**79.69**±0.06	65.72±0.53	45.88±0.38	75.72±0.67	57.14±0.06
CompILer	**91.81**±0.23	**96.67**±0.01	**94.18**±0.06	**56.85**±0.34	79.56±0.04	**66.31**±0.15	**46.27**±1.56	76.65±1.19	**57.69**±0.42

Analyzing multi-pool prompt learning

- The inclusion of primitive prompt pool yields consistent gains over the baseline.
- The best results are achieved when the model integrates all three pools simultaneously.

Prompt Pool			Split-Clothing (5 tasks)		
C	S	O	Avg Acc	FTT(↓)	HM
✓			80.22±0.41	14.23±0.44	88.85±0.16
✓		✓	88.10±0.11	7.79±0.04	93.55±0.04
✓	✓		88.09±0.50	**7.26**±0.54	93.52±0.13
✓	✓	✓	**88.38**±0.08	8.01±0.42	**93.66**±0.02

Analyzing object-injected prompting & GeM

- S→O exhibits a decrease in all metrics, implying that state prompts may interfere with the selection of object prompts.
- O → S outperforms the None model as we expect.

- GeM performs better than both max and mean pooling across various metrics.
- It validates the benefit of GeM on mitigating irrelevant information in the selected prompts.

(a) Object-injected state prompting.

Dataset	Split-Clothing (5 tasks)		
Metrics	Avg Acc	FTT(↓)	HM
None	88.45±0.10	7.93±0.11	93.70±0.03
S→O	88.27±0.02	7.99±0.05	93.67±0.01
O→S	**89.21**±0.24	**7.26**±0.60	**94.18**±0.06

(b) Prompt fusion method.

Dataset	Split-Clothing (5 tasks)		
Metrics	Avg Acc	FTT(↓)	HM
Max	84.70±0.64	12.24±2.25	91.54±0.30
Mean	87.80±0.12	7.82±0.01	93.38±0.03
GeM	**89.21**±0.24	**7.26**±0.60	**94.18**±0.06

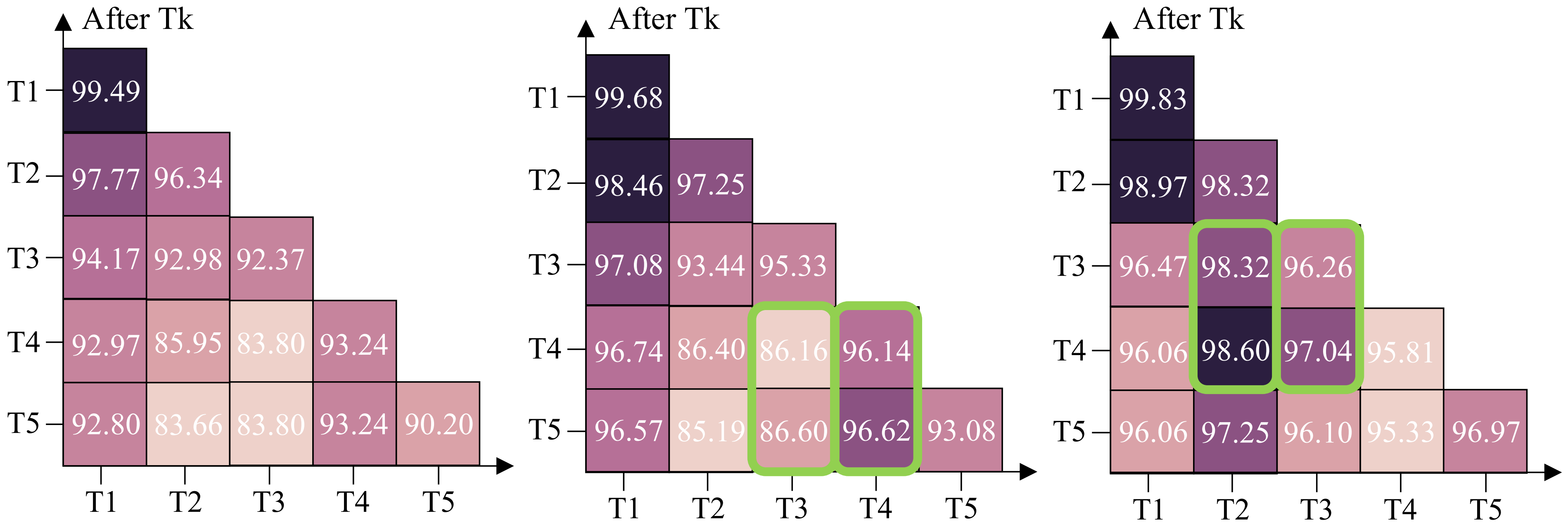
Analyzing loss function

- Baseline model (first row) includes all modules but is trained by cross entropy loss only.
- CompILer achieves the best results when combing all the loss terms during training.

Loss function				Split-Clothing (5 tasks)		Split-UT-Zappos (5 tasks)	
$\mathcal{L}_{CE}$	$\mathcal{L}_{RCE}$	$\mathcal{L}_{inter}$	$\mathcal{L}_{intra}$	Avg Acc	FTT(↓)	Avg Acc	FTT(↓)
✓				88.17±0.08	8.08±0.27	44.83±0.15	19.49±2.93
✓	✓			88.36±0.37	8.33±0.11	45.47±0.07	20.14±0.43
✓		✓		88.32±0.56	7.82±0.64	45.58±0.04	19.64±0.37
✓			✓	88.42±0.30	8.23±0.06	45.62±0.13	20.13±0.14
✓		✓	✓	88.61±0.61	7.72±0.87	46.01±0.69	19.50±0.86
✓	✓	✓	✓	**89.21**±0.24	**7.26**±0.60	**46.48**±0.26	**19.27**±0.75

Qualitative results

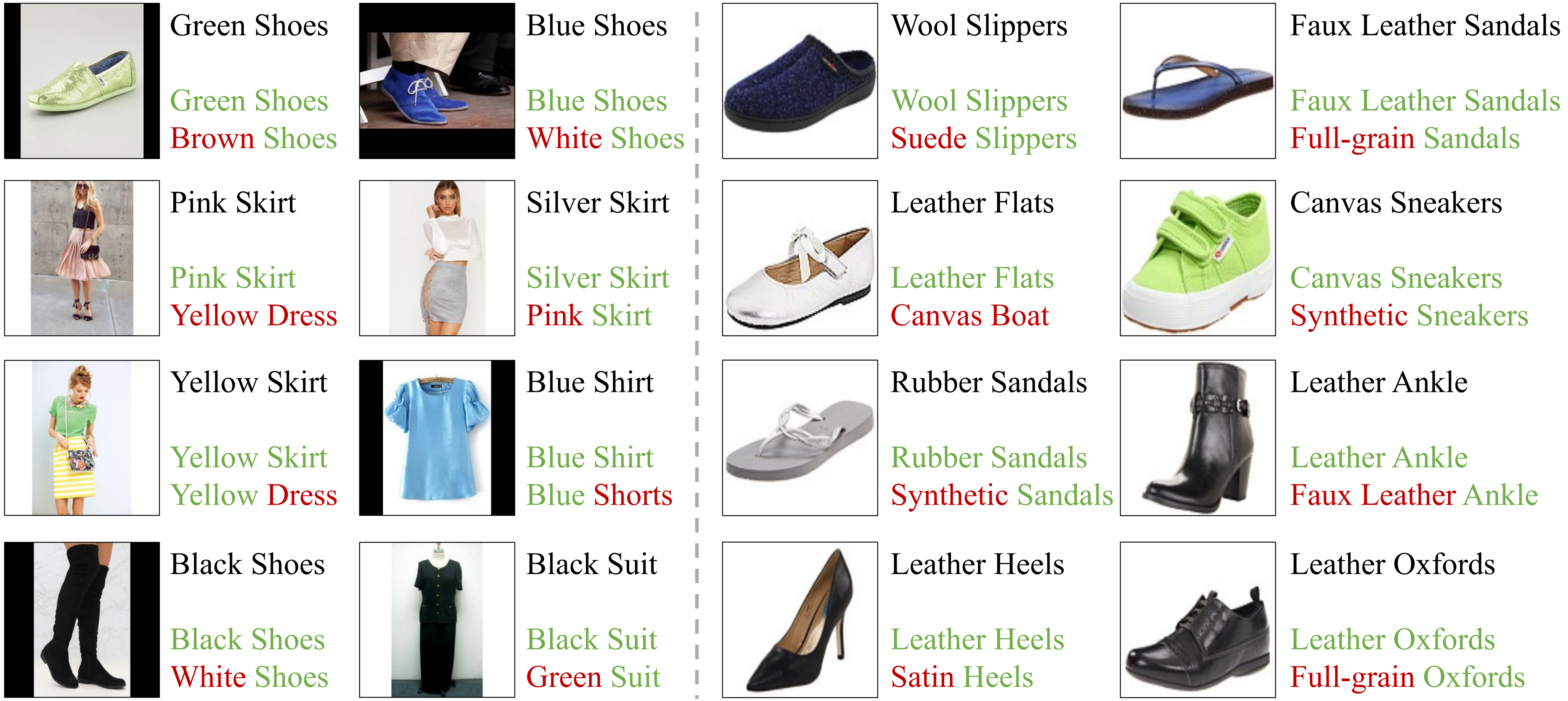
- (a) shows a decreasing trend in composition accuracy along with the introduction of new tasks.
- (b) and (c) showcase that the primitive accuracy occasionally increases as more tasks are learned.
- We conjecture the reason is mostly attributed to the re-occurrence of primitive concepts.



(a) Composition Accuracy (b) State Accuracy (c) Object Accuracy

Qualitative results

- Comparison on composition predictions between CompILer and L2P.



(a) Split-Clothing (5 tasks) (b) Split-UT-Zappos (5 tasks)

Summary

New Tasks: Compositional Incremental Learning

New model: Compositional Incremental Learner

New Benchmarks: Split-Clothing & Split-UT-Zappos

New Metrics: State, Object and HM

Code at https://github.com/Yanyi-Zhang/CompILer