Code

Prime Once, then Reprogram Locally: An Efficient Alternative to Black-Box Service Model Adaptation

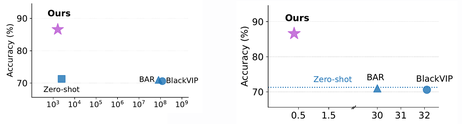
Yunbei Zhang[1], Chengyi Cai[2], Feng Liu[2], Jihun Hamm[1]
[1]Tulane University [2]University of Melbourne

CVPR JUNE 3-7, 2026 DENVER COLORADO

Motivation

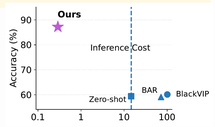
Black-box reprogramming spends the budget in the wrong place.

- BAR and BlackVIP optimize visual prompts through Zeroth-Order Optimization (ZOO).
- ZOO needs repeated API calls per image, noisy gradient estimates, and service access again at inference time.
- Modern robust APIs can be insensitive to small input perturbations, so costly search may not move the prediction.



Question: can we use a service API once, then adapt and deploy locally?

$$\nabla \ell(\mathbf{P}) \approx \frac{d}{q} \sum_{i=1}^{q} \frac{\ell(\mathbf{P} + \mu u_i) - \ell(\mathbf{P})}{\mu} u_i$$

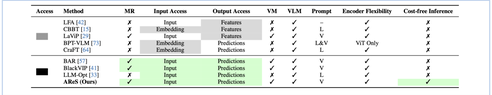


Each loss evaluation routes back through the service model.

Design Goal

1x	0	Local
service pass for priming	service calls at inference	gradients after priming

Black Box	input / prediction only
Grey Box	features or embeddings
White Box	gradients or parameters



Access	Method	MR	Input Access	Output Access	VM	VLM	Prompt	Encoder Flexibility	Cost-free Inference
	LFA [42]	✗	Input	Features	✗	✓	-	✓	✗
	CIBT [17]	✗	Embedding	Features	✗	✓	L	✓	✗
	LaVIP [28]	✓	Input	Features	✗	✓	V	✓	✗
	BPT-VLM [73]	✗	Embedding	Predictions	✗	✓	L&V	VIT Only	✗
	CraFT [64]	✗	Embedding	Predictions	✗	✓	L	✓	✗
	BAR [17]	✓	Input	Predictions	✓	✓	V	✓	✗
	BlackVIP [61]	✓	Input	Predictions	✓	✓	V	✓	✗
	LLM-Opt [13]	✓	Input	Predictions	✓	✓	V	✓	✗
	AReS (Ours)	✓	Input	Predictions	✓	✓	V	✓	✓

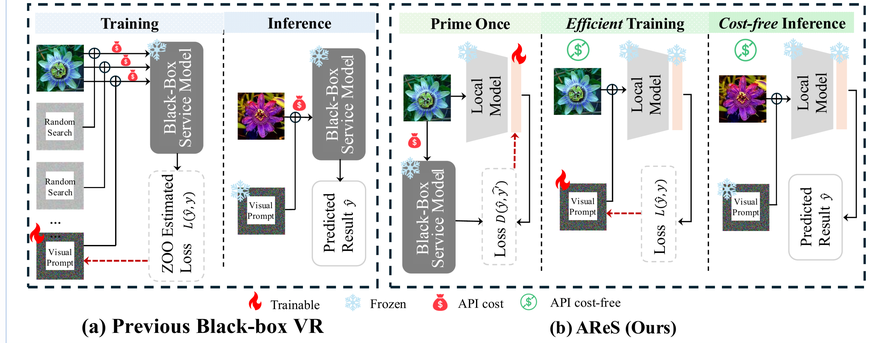
- Stay in strict Black Box access for the service API.
- Prime once, then learn and deploy the visual prompt locally.
- Use the same primitive for VMs and VLMs.

AReS: Prime Once, Then Reprogram Locally

1. Query Service
Collect service probabilities once per target-domain training image.

2. Prime Local
Freeze the local encoder and train a lightweight linear layer.

3. Local VR
Learn the visual prompt with exact White Box gradients on the primed local model.



AReS moves adaptation and inference to the local proxy after one-time priming.

Priming loss

$$\theta^* = arg\ min_{\theta} \sum_{i=1}^{N} \mathcal{L}_{\mathcal{P}}\big(\mathcal{F}_{\mathcal{L}}(x_i; \theta), p_{\mathcal{S}}(x_i)\big)$$

Local VR loss

$$\mathbf{P}^* = arg\ min_{\mathbf{P}}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathbb{D}^{\mathcal{T}}}\big[\ell\big(g_{out}\big(\mathcal{F}_{\mathcal{L}}(g_{in}(x, \mathbf{P}); \theta^*)\big), y\big)\big]$$

Method details are the two optimization objectives: one service pass for priming, then all prompt learning with local gradients.

Main Results: Accuracy and Efficiency

AReS improves accuracy while replacing repeated service-model optimization with one-time priming.

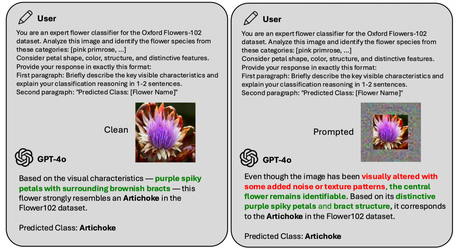
Table 2. VLM / VM accuracy, API calls, and training time

Method	Flowers	DTD	UCF	Food	GTSRB	EuroSAT	Pets	Cars	SUN	SVHN	Avg.	#API (M)	Time (h)
VR (glass-box)	86.8	62.0	74.0	81.6	65.1	90.9	90.7	66.2	66.7	60.2	74.4	16.82	9.8
Zero-shot	71.3	43.9	66.9	**85.9**	21.0	47.9	**89.1**	65.2	62.6	17.9	57.2	0.12	0
BAR	71.0	46.8	64.2	84.4	21.5	77.3	88.4	63.0	62.4	34.6	61.4	612.84	185.6
BlackVIP	70.6	45.3	**68.7**	**85.9**	21.3	73.3	**89.1**	**65.4**	**64.5**	44.4	62.9	754.20	197.5
LLM-Opt	67.6	45.0	59.9	78.5	21.2	48.0	87.7	56.2	60.3	20.2	54.5	5011.32	5.1
AReS	**86.6**	**48.2**	67.1	68.8	**39.4**	**85.7**	88.9	43.2	62.8	**63.2**	65.4	**0.02**	**3.7**
AReS-MS	**86.6**	**48.2**	67.1	**85.9**	**39.4**	**85.7**	88.9	65.2	62.8	**63.2**	**69.3**	**0.06**	**3.7**

Table 4. Modern API case study

Method	LLaVA				GPT-4o				Clarifai			
	Acc. ↑	Train	Infer.	Total (#) ↓	Acc. ↑	Train	Infer.	Total ($) ↓	Acc. ↑	Train	Infer.	Total ($) ↓
glass-box	91.3	–	–	–	–	–	–	–	–	–	–	–
Zero-shot	40.1	–	8100	8100	59.4	–	14.6	14.6	–	–	–	–
BAR	34.1	$\sim 10^6$	8100	$\sim 10^6$	59.1	57.6	14.6	72.2	68.1	38.4	9.7	48.1
BlackVIP	39.4	$\sim 10^7$	8100	$\sim 10^7$	60.1	86.4	14.6	101.0	72.1	57.6	9.7	67.3
AReS	73.1	160	0	**160**	87.2	0.3	0	**0.3**	**83.2**	0.2	0	**0.2**

Efficiency reading: the accuracy gains come with cost-free inference and no repeated API perturbation loop.



AReS is less dependent on service-model perturbation sensitivity because optimization happens on the primed local model.

Theoretical Results

- Priming is useful when it makes the local proxy faithful to the service model.
- Assume (ϵ) –faithful priming on target-domain inputs.
- With cross-entropy Lipschitzness, logit closeness transfers to risk.

$$\mathbb{E}_{\mathbb{D}^{\mathcal{T}}} |z_{\mathcal{S}}^*(x, \mathbf{Q}^*) - z_{\mathcal{L}}^*(x, \mathbf{P}^*)|_1 \leq \epsilon$$

$$\mathcal{R}_{\mathcal{L}}\big(\mathcal{D}^{\mathcal{T}}, \mathbf{P}^*\big) - \epsilon \leq \mathcal{R}_{\mathcal{S}}\big(\mathcal{D}^{\mathcal{T}}, \mathbf{Q}^*\big) \leq \mathcal{R}_{\mathcal{L}}\big(\mathcal{D}^{\mathcal{T}}, \mathbf{P}^*\big)$$

- The epsilon-faithful assumption is not free; it is exactly the goal of the priming stage.
- The mechanism is constructive: one service pass trains a lightweight head on a frozen local encoder.

- The downstream label space need not match the service label space, so priming is preparatory rather than ordinary final-task distillation.

Insight: faithful priming lets a stable White Box local route approximate the reprogrammed service model.

The theorem explains why API efficiency and stability improve together.

Conclusion and Discussion

Take-home: AReS turns black-box service adaptation into one-time transfer plus local, cost-free deployment.

Key conclusions

- Modern APIs are fragile under perturbation-based ZOO.
- AReS spends API calls once, then optimizes locally.
- Future work: make priming more faithful under larger domain gaps.