Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning

Dipam Goswami[1,2] Albin Soutif–Cormerais[1,2] Yuyang Liu[3,4,†] Sandesh Kamath[1,2]
Bartłomiej Twardowski[1,2,5] Joost van de Weijer[1,2]

Resurrecting Old Classes with New Data for Exemplar-Free Continual Learning

Dipam Goswami[1,2] Albin Soutif–Cormerais[1,2] Yuyang Liu[3,4,†] Sandesh Kamath[1,2]
Bartłomiej Twardowski[1,2,5] Joost van de Weijer[1,2]

- When the feature extractor is continually trained on new tasks, old class prototypes drift in the embedding space. In the absence of data from old classes, we cannot estimate the new position of old prototypes in the embedding space.

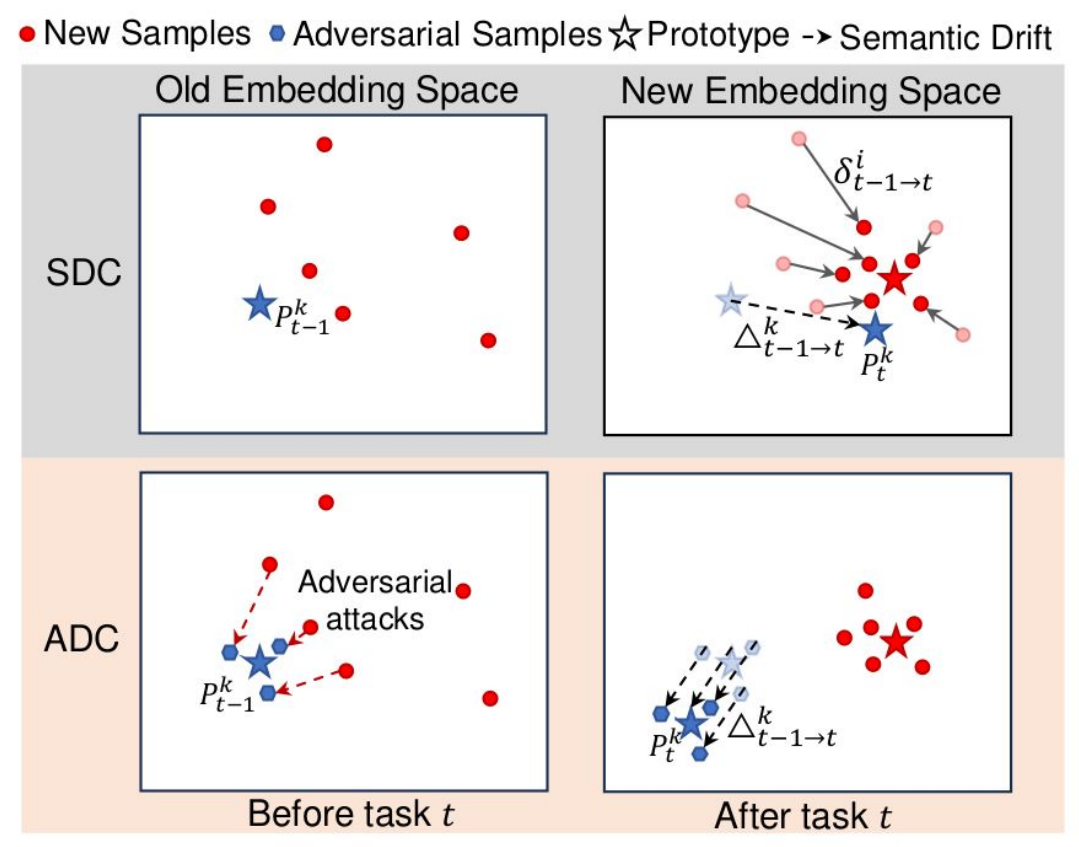
- We address the problem of feature drift compensation for exemplar-free methods. We propose to generate adversarial samples using the new task samples and then use these adversarial samples to track the movement of old class prototypes in the feature space.

Introduction

- Why exemplar-free continual Learning ?
  - Multiple challenges in practical settings with exemplar-based methods such as legal concerns with new regulations (e.g. European GDPR where users can request to delete personal data), and privacy issues when dealing with sensitive data like in medical imaging

- Small-start vs Big-start settings -
  - To reduce potential drift in the feature extractor, existing exemplar-free methods are typically evaluated in settings where the first task is significantly larger than subsequent tasks (half of the dataset in first task) and the model learns high-quality feature representations in the first task.
  - Usually, this drift is minimized with heavy functional regularization, which consequently restricts the plasticity of the network.
  - Small-start settings (equally splitting the dataset into tasks) starts from a smaller initial task and thus is a more challenging setting.

Introduction



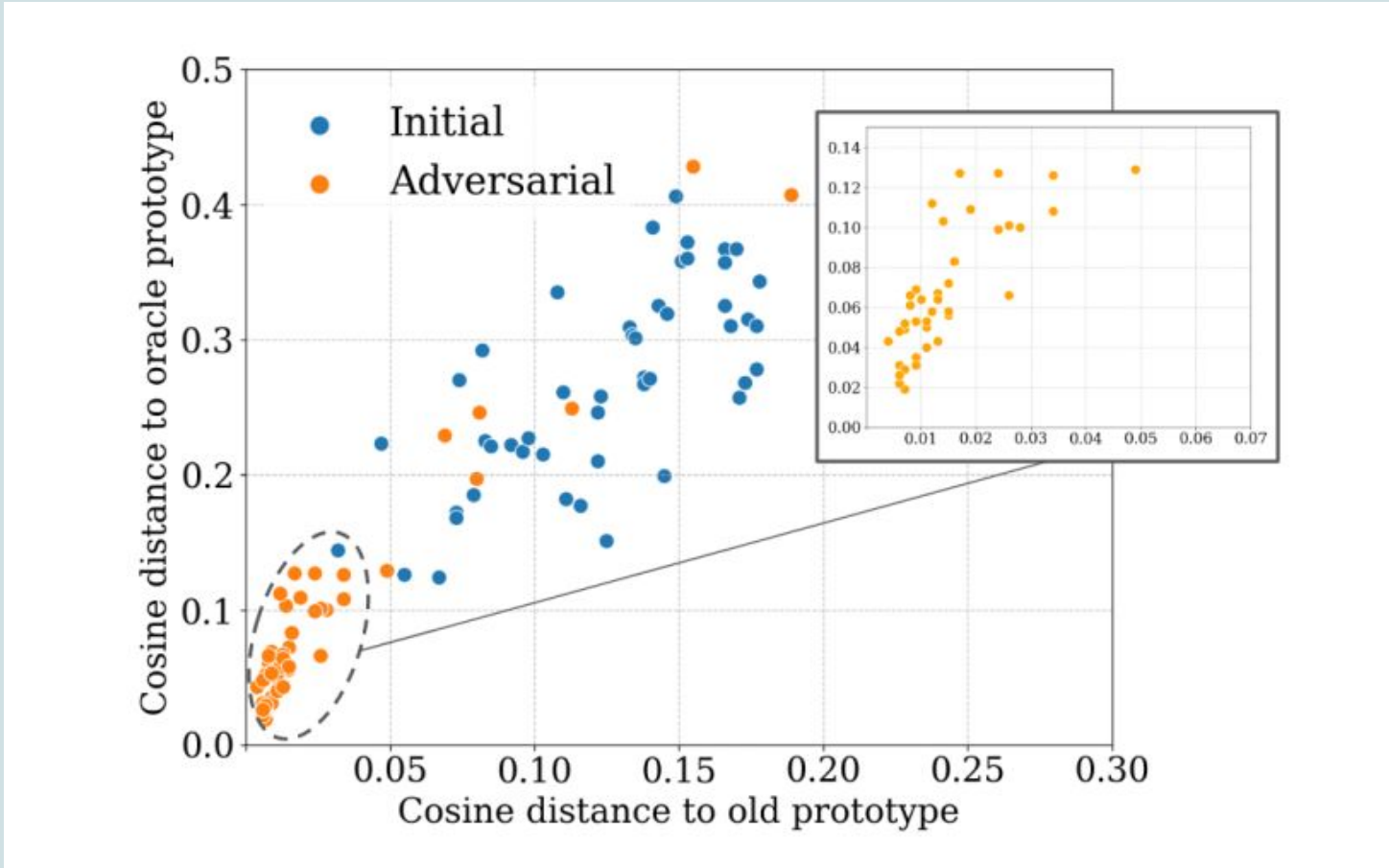
Introduction

Traditional Buffer (Exemplars)

Current Data

Adversarial Samples
(Pseudo-exemplars)

Motivation


Initial
Adversarial
Cosine distance to oracle prototype
Cosine distance to old prototype
0.0
0.1
0.2
0.3
0.4
0.5
0.05
0.10
0.15
0.20
0.25
0.30
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.01
0.02
0.03
0.04
0.05
0.06
0.07

Adversarial Drift Compensation

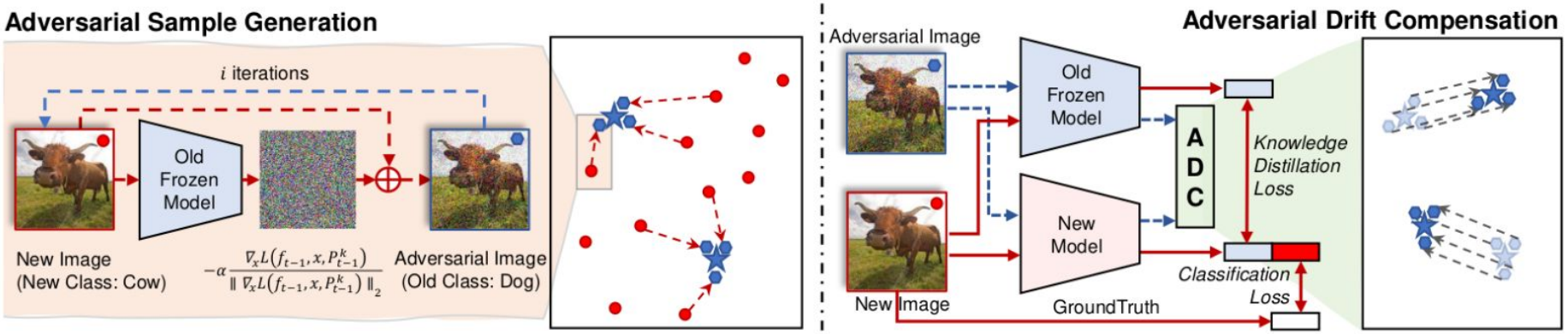


Figure 3. (a) Adversarial Sample Generation: On the old model feature space, the new samples closest to the old prototype are selected and iteratively perturbed in the direction of the target old prototype to generate adversarial samples which are now misclassified as the target old class resulting in embeddings closer to the old prototype. We perform this for every old class (we show 2 classes here for demonstration). (b) Model Training with Drift Compensation: The new model is trained using the classification loss for learning new classes and knowledge distillation loss to prevent forgetting of old classes. After the new model is trained, the adversarial samples generated using the old model are passed through both the models and the drift from old to new feature space is estimated. This is then used to update the old prototypes.

Adversarial Sample Generation

We propose the following optimization objective by computing the mean squared error between the features $f_{t-1}(x)$ and the prototype P^k_{t-1} as:

$$L(f_{t-1}, \mathcal{X}^k, P^k_{t-1}) = \frac{1}{|\mathcal{X}^k|} \sum_{x \in \mathcal{X}^k} \|f_{t-1}(x) - P^k_{t-1}\|^2_2. \tag{2}$$

In order to move the feature embedding in the direction of the target prototype P^k_{t-1}, we obtain the gradient of the loss with respect to the data $x \in \mathcal{X}^k$, normalize it to get the unit attack vector and scale it by α as follows:

$$x_{adv} \leftarrow x - \alpha \frac{\nabla_x L(f_{t-1}, x, P^k_{t-1})}{\|\nabla_x L(f_{t-1}, x, P^k_{t-1})\|_2} \; \forall x \in \mathcal{X}^k \tag{3}$$

where $\nabla_x L(f_{t-1}, x, P^k_{t-1})$ is the gradient of the objective function with respect to the data x and α refers to the step size. We perform the attack for i iterations.

Drift Compensation

The adversarial samples when passed through the new feature extractor f_t are expected to lie close to the drifted prototype and hence are used to compute the drift. After generating the adversarial samples for each target class k , we measure the prototype drift as:

$$\Delta^k_{t-1 \to t} = \frac{1}{|\mathcal{X}^k_{adv}|} \sum_{x_{adv} \in \mathcal{X}^k_{adv}} (f_t(x_{adv}) - f_{t-1}(x_{adv})) \tag{4}$$

where $x_{adv} \in \mathcal{X}^k_{adv}$ is the set of only those adversarial samples which are classified as the target class k using the NCM classifier. We resurrect the old prototypes by compensating the drift as follows:

$$P^k_t = P^k_{t-1} + \Delta^k_{t-1 \to t} \tag{5}$$

Algorithm 1 Adversarial Drift Compensation

input: Images X_t, feature extractors f_t and f_{t-1}, old prototypes $P_{t-1}^{Y_{1:t-1}}$, step size α, number of generated samples m, number of iterations for perturbation i.

output: Resurrected prototypes $P_t^{Y_{1:t-1}}$.

for $k \in Y_{1:t-1}$ **do**

  Sample a set $\mathcal{X}^k$ of m new samples from X_t which are closest to P_{t-1}^k based on L2 distance.

  for i iterations **do**

    $L(f_{t-1}, \mathcal{X}^k, P_{t-1}^k) = \frac{1}{|\mathcal{X}^k|} \sum_{x \in \mathcal{X}^k} \|f_{t-1}(x) - P_{t-1}^k\|_2^2.$

    $x_{adv} = x - \alpha \frac{\nabla_x L(f_{t-1}, x, P_{t-1}^k)}{\|\nabla_x L(f_{t-1}, x, P_{t-1}^k)\|_2} \forall x \in \mathcal{X}^k$

    $x \leftarrow x_{adv}$

  end for

  add x in $\mathcal{X}_{adv}^k$ if $\underset{y \in Y_{1:t-1}}{\operatorname{argmin}} \|f_{t-1}(x) - P_{t-1}^y\|_2 = k.$

  ▷ Store only those samples in $\mathcal{X}_{adv}^k$ which are successfully misclassified as k

  $\Delta_{t-1 \to t}^k = \frac{1}{|\mathcal{X}_{adv}^k|} \sum_{x_{adv} \in \mathcal{X}_{adv}^k} f_t(x_{adv}) - f_{t-1}(x_{adv})$

  $P_t^k = P_{t-1}^k + \Delta_{t-1 \to t}^k$ ▷ Prototype resurrection

end for

Experiments

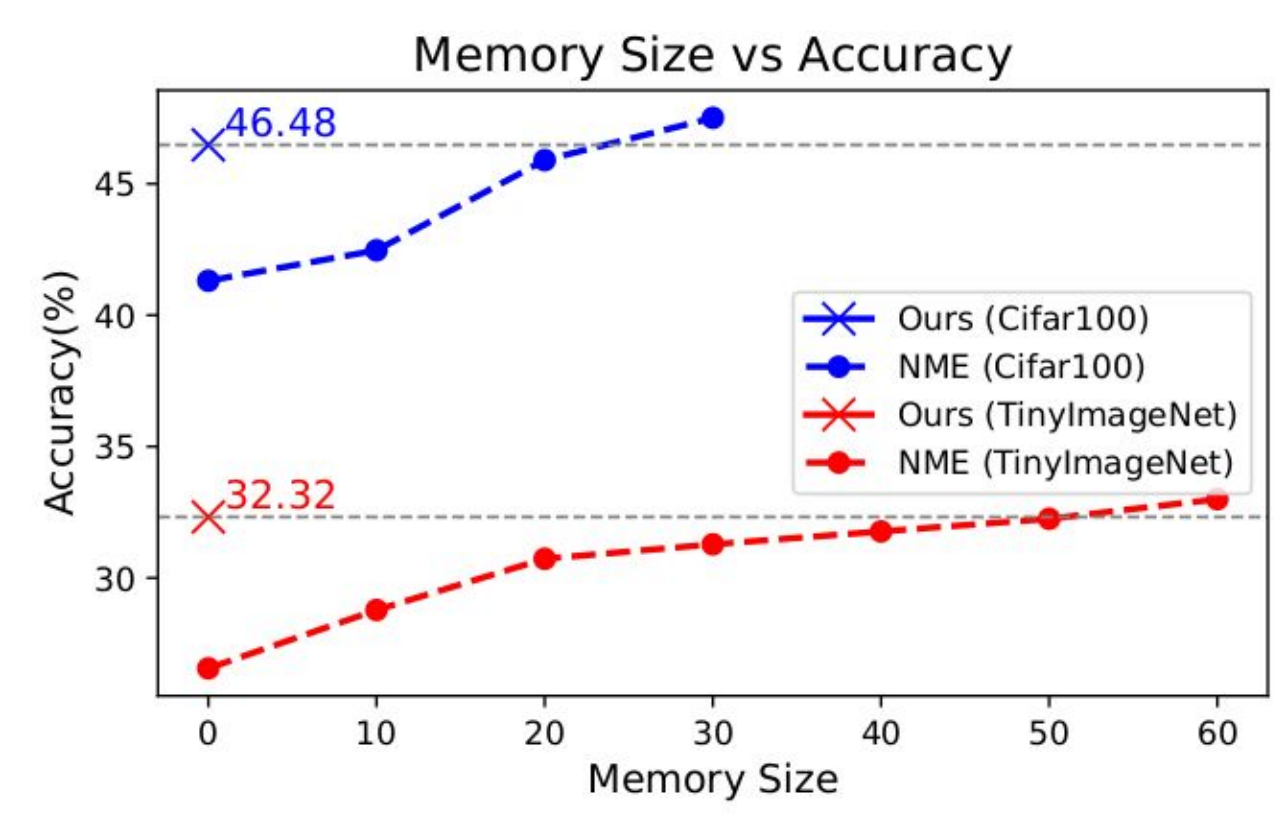
Method	CIFAR-100				TinyImageNet				ImageNet-Subset			
	T = 5		T=10		T = 5		T =10		T = 5		T = 10	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [24]	45.35	61.94	26.14	46.14	38.81	49.70	<u>27.42</u>	38.77	50.88	69.11	37.90	61.60
NCM	53.53	<u>66.35</u>	41.31	57.85	38.69	50.45	26.56	<u>41.04</u>	57.74	71.99	<u>45.86</u>	65.04
SDC [52]	<u>54.94</u>	64.82	<u>41.36</u>	<u>58.02</u>	<u>40.05</u>	<u>50.82</u>	27.15	40.46	<u>59.82</u>	<u>74.10</u>	43.72	<u>65.83</u>
PASS [55]	49.75	63.39	37.78	52.18	36.44	48.64	26.58	38.65	50.96	66.15	38.90	54.74
SSRE [56]	42.39	56.57	29.44	44.38	30.13	43.20	22.48	34.93	40.30	57.57	28.12	45.87
FeTrIL [35]	45.11	60.42	36.69	52.11	29.91	43.99	23.88	36.35	49.18	63.83	40.26	55.12
FeCAM [11]	47.28	61.37	33.82	48.58	25.62	39.85	23.21	35.32	54.18	67.21	42.68	57.45
ADC (Ours)	**59.14**	**69.62**	**46.48**	**61.35**	**41.0**	**50.94**	**32.32**	**43.04**	**62.40**	**74.84**	**47.58**	**67.07**

Table 1. Evaluation of EFCIL methods on small-start settings. Best results in **bold** and second best results are <u>underlined</u>.

Experiments

Method	CUB-200				Stanford Cars			
	T = 5		T=10		T = 7		T =14	
	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}	A_{last}	A_{inc}
LwF [24]	58.68	71.31	41.96	60.15	45.18	61.14	30.33	49.93
NCM	52.74	67.13	38.47	57.83	42.22	59.06	31.60	51.34
SDC [52]	55.20	68.64	41.63	60.43	45.03	61.75	32.15	53.18
PASS [55]	34.04	49.00	26.37	41.08	20.71	37.13	12.30	25.46
FeTrIL [35]	54.66	67.45	49.09	62.42	36.92	54.09	34.29	50.41
FeCAM [11]	53.47	66.39	51.78	64.97	40.64	56.24	37.50	52.78
ADC (Ours)	**64.46**	**73.49**	**57.97**	**68.91**	**54.86**	**67.07**	**45.07**	**61.39**

Experiments



Drift Estimation Quality

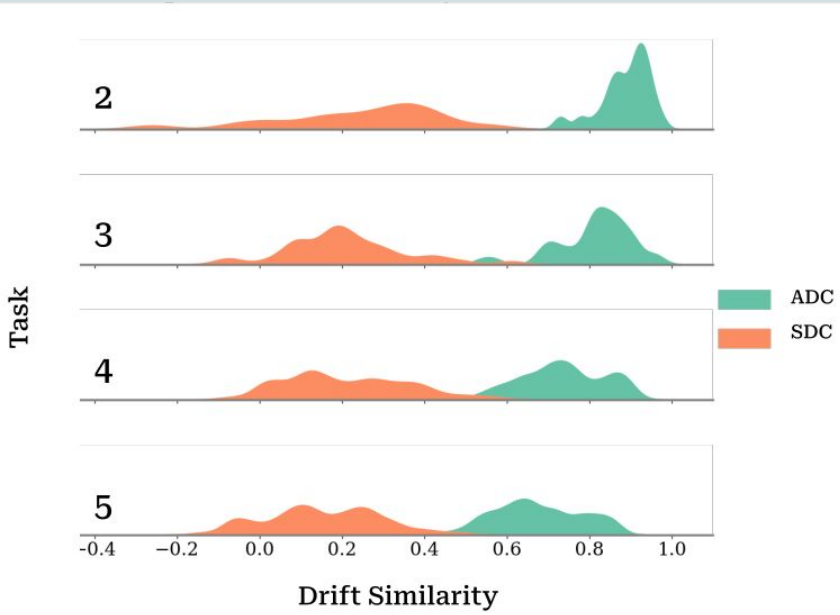
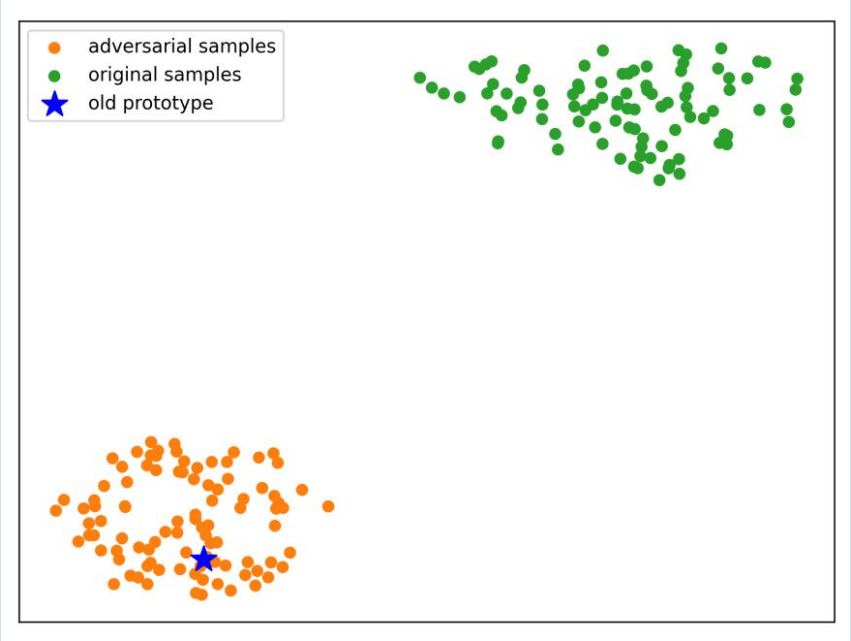


Figure 6. Comparison between the two drift estimation methods SDC [52], and the proposed ADC, on CIFAR-100 (5 tasks). We compute the drift for each class with the two methods and report the distribution of drift estimation quality, measured by computing the cosine similarity between the estimated drift vector and the true drift (obtained using old data), for all previous class prototypes.

t-SNE Visualization



Thanks