Warper: Efficiently Adapting Learned Cardinality Estimators to Data and Workload Drifts - Extended Report

PRSA dataset		w1-5, data drft c1		w12/345, wkld drft c3	
Method		FT	Warper	FT	Warper
Annotation cost*.		1.2 @ 0.01s/input			
Model building cost*		-	57.2s	-	49.8s
Avg CPU Usage	5 min @ same	0.03%	1.6%	0.03%	1.4%
	10 min @ #queries to	0.02%	0.8%	0.02%	0.7%
	30 min @ annotate	0.01%	0.27%	0.01%	0.24%

HIGGS dataset		w1-5, data drft c1		w12/345, wkld drft c3	
Method		FT	Warper	FT	Warper
Annotation cost*.		140s @ 0.39s/input			
Model building cost*		-	57.0s	-	50.0s
Avg CPU Usage	5 min @ same	3.9%	5.3%	3.9%	5.3%
	10 min @ #queries to	1.9%	2.7%	1.9%	2.7%
	30 min @ annotate	0.06%	0.9%	0.06%	0.9%

Table 13: We show cost comparison with FT which randomly picks the same number of query predicates to annotate in each adaptation step. For data drift `c1` and workload drift `c3`, Warper updates and uses only the picker. *: Costs in a single CPU thread.

Algorithm 1: Warper procedures in an invocation (multi-thread).

Input : {q, gt} from $\mathbb{I}_{\texttt{test}}$, $\mathbb{M}, \mathbb{G}, \mathbb{D}, \mathbb{E}$ from the previous invocation, `mode` flag from `det_drft`.

1 push({(q, gt, *l*=test)});
2 **if** mode contains c1, c2 or c3:
3 //*async: invoke asynchronously in a new thread.*
4 async { gt ← anno(q|gt=-1) **for** record **in** Pool; }
5 Clear all z in Pool;
6 async {z ← embed(q, gt|$z = \emptyset$) **for** record in Pool; }
7 Compute $\mathcal{L}_{\text{AE}}$ using q ∈ Pool;
8 **if** mode contains c2 :
9 async { l', c' ← dsc($z|l$ = s) **for** record in Pool; }
10 **while** n_i--: //*Update GAN.*
11 q_{gen} ← {gen($z|l$ = test}, n_s)} //*Generate by* $\mathbb{G}$;
12 Compute $\mathcal{L}_{\text{GAN}}$ using q_{gen}, q ∈ Pool;
13 $\mathbb{G}, \mathbb{E}, \mathbb{D}$ ← update_MultiTask($\mathbb{G}, \mathbb{E}, \mathbb{D}, \mathcal{L}_{\text{AE}} + \mathcal{L}_{\text{GAN}}$);
14 push({q_{gen}, *l*=gen});
15 $\mathbb{G}, \mathbb{E}$ ← update_AutoEncoder($\mathbb{G}, \mathbb{E}, \mathcal{L}_{\text{AE}}$);
16 pick(n_p); //*Pick by* $\mathbb{P}$ *which* q *to annotate.*
17 sync(); //*Wait for async modules, e.g., anno() (Line#4), to finish.*
18 $\mathbb{M}$ ← update($\mathbb{M}$, {q,gt} ∈ Pool); //*Update* $\mathbb{M}$ *in all cases.*

Output : Updated $\mathbb{M}, \mathbb{G}, \mathbb{D}, \mathbb{E}$.

1 APPENDIX

1.1 Multi-thread Warper update.

Our main paper describes a single-thread Warper update algorithm in Section 3.1. We briefly summarize a multi-thread version of the same algorithm here. In a nutshell, various Warper components can be called in synchronous threads. We use a simplification to keep the pseudocode short: the `async{}` keyword denotes parallel actors that pull from the pool and perform the stated action and `sync()` waits for all actors to finish. Line#2 discerns the kind of drift if any as before. Line#4 initiates an asynchronous thread that pulls from the pool any unlabeled (`gt=-1`) predicates and labels using $\mathbb{A}$. Line#6 and Line#11 compute z and l' for predicates in a simliar asynchronous way; Lines#10–#13 update the generator and discriminator components iteratively as before. Queries to use for training are picked in line#16. Finally, in line#18, Warper updates the underlying model using annotated predicates from the pool.

Dataset	Case	Wkld	Model	δ_m	δ_{js}	$\Delta_{.5}$	$\Delta_{.8}$	Δ_1
Higgs	c2	w12/345	LM-mlp	12	0.60	3.8	3.7	3.5
Forest	c2	w12/345	LM-mlp	0.71	0.33	1.0	1.07	1.13
Weather	c2	w12/345	LM-mlp	0.98	0.4	1.0	1.0	1.0
Power	c2	w12/345	LM-mlp	1.91	0.41	1.38	2.37	1.67

Table 12: Adaptation efficiency for the datasets used in [10].

1.2 Supplementary Results

Adaptation efficiency on more datasets. Beyond the results shown in Section 4.1, we further demonstrate in Table 12 the adaptation efficiency on all datasets used in [10]. We use the same experimental settings and metrics as in Section 4.1 to adapt `LM-mlp` for workload drift `c2`. Results indicate that Warper behaves similarly as shown in our paper submission. Note that Warper did not help significantly on the Weather dataset; on such a dataset we found that the original CE model converges quickly with FT at only 10-50 arriving queries. However, Warper has no accuracy regression in such a worst case and accelerates the model adaptation on the rest three datasets.

Cost analyses for data drift `c1` and workload drift `c3`. Table 6 in Section 4.1.1 demonstrates the cost decomposition of Warper and FT in a workload drift `c2` in which Warper utilizes all components (to generate, pick and update). Here we show a similar cost decomposition for data drift `c1` and workload drift `c3`. Recall from Section 3 that in both of these drift cases, synthesizing new queries is not needed and Warper as well as the baseline (FT) only requires picking among available queries which ones to annotate. Table 13 illustrate results on two datasets with different annotation costs per query predicate. Warper incurs a small overhead in these different drifts which is insignificant beyond FT.