EvoCL: Continual Learning over Evolving Domains - Appendix

A. Additional Analyses

In addition to the experiments performed in the main paper, we perform some additional studies pertaining to various aspects of EvoCL. Firstly, we conduct a series of experiments that showcase EvoCL's performance in a Class Incremental Learning setting, as opposed to the Task Incremental Learning setting presented in the main paper. This is presented in Section A.1. Secondly, we conduct a series of experiments varying the number of training domains. This is done to show that EvoCL is capable of ensuring good generalization performance even when presented with a limited set of training domains, as opposed to other baselines which seem to take a palable hit to performance. The results of this series of experiments is summarised in Section A.2. Finally, we evaluate the robustness of EvoCL to varying orders of tasks, the results of which are presented in Section A.3.

A.1. Class Incremental EvoCL

The primary experiments dealt with a Task Incremental Learning (TIL) setup where each domain enclosed tasks which appear in a sequence. EvoCL can also be applied to a Class Incremental setting. Class Incremental Learning (CIL) can be disintegrated into a two step process:

1. Infer the task identity
2. Perform classification within inferred task

Given that our setup performs the second step when provided with a task identity, a mechanism to predict the task identity is required to convert to a class incremental setting. In pursuit of that, we employ a strategy that is centered around entropy.

From an information theoretic standpoint, we define entropy **H** of a random variable as a measure of uncertainty alluded to its various possible outcomes. In other words, a state with a high entropy means greater uncertainty and vice versa. Formally, entropy is defined as follows:

$$\mathbf{H}(\mathbf{x}) = -\sum_{\mathbf{x}\in\mathcal{X}} p(x) log p(x) \tag{1}$$

In our case, when we calculate entropy of the logits, a state with a distribution peaked at one of the classes would signal greater confidence that translated to a lower entropy.

Algorithm 1 Class Incremental Learning: Inference Algorithm

Input: Sequence of tasks from domain $D+1$, $\mathbf{S^{D+1}}$
Models: Domain Predictor $\mathcal{R}(\xi;\lambda^*)$, Hypernetwork $\mathcal{H}(\xi,\tau;\phi)$ & Classifier $\mathcal{F}(x;\theta)$

$\xi_{D+1} \leftarrow \mathcal{R}(\xi_D;\lambda)$
for $k = 1, 2, ..., K$ **do**
  $\theta_k^{D+1} \leftarrow \mathcal{H}(\xi_{D+1}, \tau_k;\phi)$
  $\hat{z}_k^{D+1} \leftarrow \tilde{\mathcal{F}}(x_k^{D+1};\theta_k^{D+1})$
  $r_k^{D+1} \leftarrow \mathbf{H}(\hat{z}_k^{D+1})$
end for
$\hat{k} \leftarrow argmin_k(r_k^{D+1})$
$\hat{y}^{D+1} \leftarrow \mathcal{F}(x^{D+1};\theta_{\hat{k}})$

A flat distribution, on the other hand, signals lower confidence in the prediction and thus, greater entropy.

With this in mind, during inference, we predict the logits z_k^d for all the classifiers pertaining to each task identity. We then calculate their respective entropies r_k^d. We choose the classifier $\hat{\theta}$ that results in logits that possess the lowest entropy. We summarise our CIL inference methodology in Algorithm 1. Note that we define the classifier with its logit output in place of the class prediction as $\tilde{\mathcal{F}}$.

We compile the results of our CIL Experiments in Figure 1 that follows: It can be observed that while performance deteriorates a little, as is with CIL setups when compared to TIL setups in general, EvoCL still outperforms its baselines and maintains its lead across all the datasets.

A.2. On the effect of training domain numbers

All of our experiments entailed 5 training domains. Herein, we perform a study to investigate the effects of having a varying number of training domains on the generalization accuracy of the test domain. We employ the CIFAR 10 example with a domain delta of 0.2, the same as with the standard experimental setup. We conduct a range of experiments starting with just 2 training domains and working the way up to 10 training domains. We train for a single epoch, similar to the primary experiments. We compare our model's performance on the unseen test domain with the leading baselines DRAIN and Progressive Finetuning (Pro-

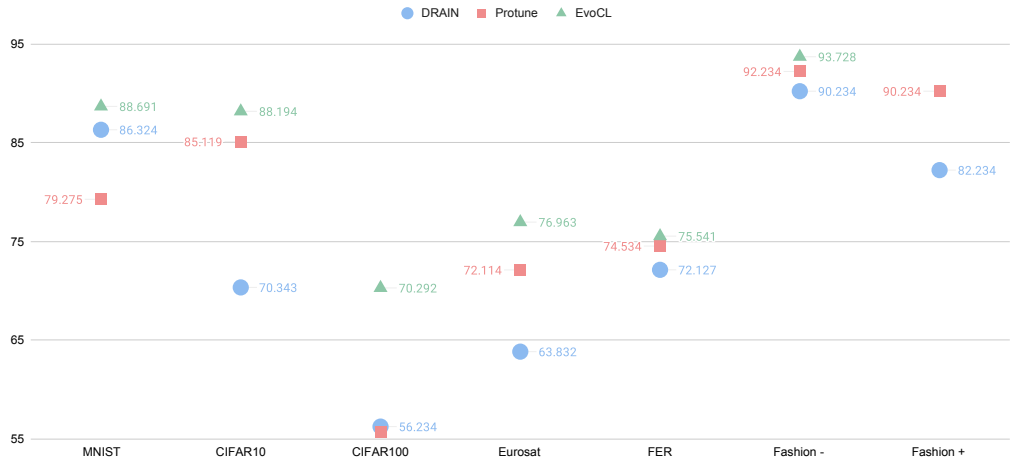

Figure 1. Average task accuracy on unseen test domain in a Class Incremental setting. EvoCL(ours) is compared with Progressive Finetuning (ProTune) and DRAIN.

Table 1. Classification accuracy for various task orderings on tasks from the test domain.

Task Order	Task 1	Task 2	Task 3	Task 4	Task 5	Average Accuracy
1, 2, 3, 4, 5	94.864	93.939	91.251	90.943	85.877	91.3748
5, 4, 3, 2, 1	90.197	92.814	86.255	93.172	91.699	90.8274
4, 2, 1, 3, 5	92.466	94.974	91.411	86.295	92.565	91.5422
2, 4, 5, 1, 3	92.864	93.74	88.943	91.042	85.051	90.328
5, 3, 1, 4, 2	91.082	88.624	92.366	93.72	95.71	92.3004

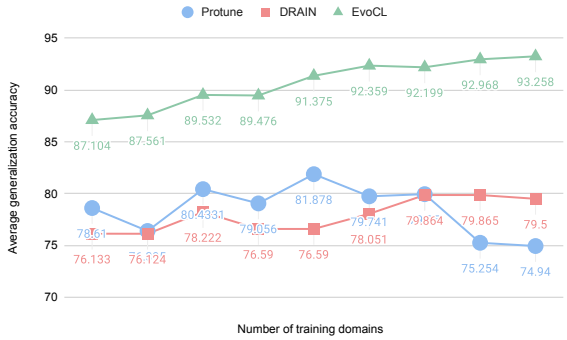

Figure 2. Chart plotting number of training domains against the average accuracy across all tasks on an unseen test domain. Results displayed are on the CIFAR 10 dataset.

Tune). We present our results in Figure 3.

Our model displays a palpable performance lead on unseen domains even when trained on as little as two domains. EvoCL maintains a solid performance lead over both DRAIN and Progressive Finetuning on the CIFAR 10 dataset.

We perform the same experiment with the Rotating MNIST dataset to more concretely display EvoCL's superior performance in generalizing to future unseen domains even when the number of training domains is low. Progressive Finetuning is only able to catch up with EvoCL's performance only when trained on a lot of domains. DRAIN's performance too improves with more training domains but its performance falls short when compared to EvoCL and Progressive Finetuning. The results of this experiment are compiled in Figure 3.

A.3. On the impact of varying task ordering

In Continual Learning, it is sometimes observed that changing the ordering of tasks can lead to different outcomes. To investigate if such a behaviour exists in EvoCL, we conduct experiments on the CIFAR 10 dataset with varying task orders. We summarise our results in Table 1. It can be observed that changing up the ordering in which the tasks appear does not seem to lead to significant differences between the various configurations.

A.4. Effects of Evolution Rate Perturbation

In our primary experiments, we proceed with the assumption that the rate at which the domain evolves is con-

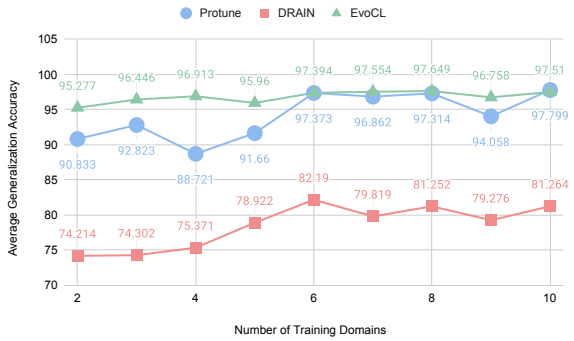

Figure 3. Chart plotting number of training domains against the average accuracy across all tasks on an unseen test domain. Results displayed are on the MNIST dataset.

stant. In real-world settings, this might not always be the case and even when it is, the data sampling intervals might not be uniform. This requires that any approach to modelling the evolution of domains is robust to such inconsistencies. To study and evaluate the robustness of EvoCL, we conduct a series of experiments where the rate of evolution is perturbed. We experiment with varying levels of perturbation and present our findings in Table 2. If Δ is the difference between each domain, we perturb the domains to the extent of $\Delta \pm k \cdot \Delta$ where $k = 0.25, 0.5.0.75.1.0$.

Table 2. Comparison of various perturbations to evolution rate.

k	MNIST	CIFAR10	FMNIST+
0	96.75	90.94	98.56
0.25	96.55	90.83	98.43
0.5	96.12	90.42	98.16
0.75	95.34	89.67	97.33
1.0	94.26	88.94	96.45

A.5. Non-uniform task subsets across domains

We consider non uniform task subsets by dropping one task for each domain. We find that knowledge transfer between tasks enables strong generalisation and upholds performance even in the absence of some tasks in between. We present our findings in Table 3.

Table 3. Comparison of standard and non uniform task settings.

k	MNIST	CIFAR10	FMNIST+
Standard	96.75	90.94	98.56
Non Uniform	91.55	86.83	95.43