EvoCL: Continual Learning over Evolving Domains

Vishnuprasadh Kumaravelu[1,2] id22resch11017@iith.ac.in
P. K. Srijith[1] srijith@cse.iith.ac.in
Sunil Gupta[2] sunil.gupta@deakin.edu.au
[1]Indian Institute of Technology Hyderabad [2]Deakin University

Abstract

Continual Learning aspires to build models capable of learning new tasks, without forgetting previously learnt tasks. In real-world settings, the distributions underlying the tasks are prone to shift. This necessitates a model capable of observing how the task distributions drift with time and adapt proactively. We present a novel framework of continual learning under evolving domains. Our approach employs a hypernetwork with separate embeddings conditioned on both domain and task to address this problem. The hypernetwork generates customised classifier weights corresponding to any domain-task pair. We employ a separate network that is trained end to end along with the hypernetwork to predict the next domain embedding, which in turn helps to generate classifier parameters corresponding to the next future domain in the evolution. We conduct extensive experiments on various datasets with a wide variety of distribution shifts to demonstrate the efficacy of our model in generalizing to future domains across all the tasks.

1. Introduction

One of the hallmarks of intelligence lies in its ability to adapt. This adaptive intelligence stems from the real-world requirement to tackle continuously evolving challenges [27]. This contrasts starkly with our assumption that the distributions of data when the model is trained and when it is put to use are similar. This is simply not true as the data arising from the real world is prone to drift. A majority of ML models in deployment suffer from progressive performance decay with time as the gap between the distribution that the model was originally trained on and the current distribution grows. To drive home the idea, consider a facial recognition model. Aligning with practical needs, such a model must be capable of continually learning new faces with time. Aside from such a continual learning capability, the model must also be able to address changes in already learnt faces as they age with time and learn to generalize and extrapolate to future settings, failing in which, the model's performance will degrade with time [25]. Furthermore, dynamic systems such as Traffic Modelling systems could benefit from having the ability to take on new tasks in the form of new neighbourhoods and model the evolution of existing neighbourhoods as they change with time. IoT systems have penetrated the home and society at large. Such systems benefit from continually learning new tasks and coping with domain changes such as that posed by sensor degradation. Resilience to changing conditions is key as machine learning permeates into every aspect of our lives. [22].

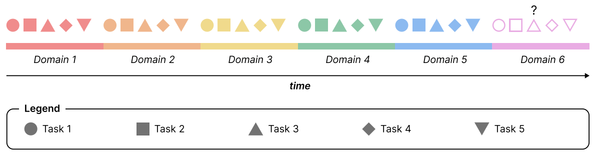

Figure 1. A scenario comprising of tasks (*shapes*) that are learnt over periods long enough that the underlying domain (*color*) evolves. This illustrates the need for a solution such as EvoCL where the focus is not only on learning new tasks but ensuring that this learning generalizes to future domains. (Diagram best viewed in colour)

Over the past few years, Continual Learning (CL) [17, 18, 24] has matured significantly as a field, heavily driven by practical requirements. Practitioners have widely taken into cognizance that it is highly impractical in terms of cost and time to retrain a model every time new data is encountered. Although CL models are rapidly closing the gap between them and models trained in offline settings, an oft-ignored fact is that learnt tasks are prone to age. CL is a paradigm where a model is trained over time. Longer periods might result in significant temporal drift in the underlying ground truths pertaining to the learnt tasks. This caving divergence between the learnt distribution and the ground truth distribution will lead to degradation in model performance. Addressing performance degradation that stems from such scenarios is studied under the umbrella of Domain Generalization (DG). DG [15, 16, 19] focuses on minimizing the test error on a dataset drawn from an unseen domain after being trained on a collection of arbitrarily seen domains. However, we need to consider a more specific situation where the distribution underlying the tasks evolves in a smooth manner and the tasks are required to generalize well to an unseen future domain arising as part of this evolution. The mitigation of such continuous drift in domains is recently considered under the framework of evolving domain generalization (EDG) [1, 21], where the model has to adapt to temporally evolving domains and minimize the generalization error over some unseen future domain. In the EDG framework, there is no provision for distributional shifts incurring new classes. The shift in distribution is only covariate and can therefore not accommodate tasks as in a Continual Learning setting.

We propose an Evolving Continual Learning framework to bring the disjoint paradigms of Continual Learning and Domain Evolution together. CL and EDG both reckon with the problem of shifting domains but their goals vary, with CL focusing on catastrophic forgetting mitigation and EDG dealing with unseen domain generalization. A related work that aims to bring together DG and CL, Cross Domain Continual Learning (CDCL) [26], requires that all training domains are seen at once within each task, which is not a practical assumption in most situations. Evolving Continual Learning on the other hand, offers the flexibility of being fully continual in terms of both tasks and domains. In the pursuit of flexible models that can continually learn from the environment and generalize to changing conditions, we propose a model that effectively addresses the evolving continual learning problem. We illustrate the idea of an Evolving Continual Learning scenario in Figure 1.

We propose a model that adapts to the changes in the tasks and domains by capturing them through the parameters of the deep learning architecture used to solve the task, e.g. a classifier network in case of classification tasks. The proposed model is centred around a hypernetwork [29] that generates the classifier network parameters bespoke to any domain and task. The hypernetwork achieves this through the use of distinct task and domain embeddings as input to generate the parameters. We learn the task embeddings from the corresponding task data. We use domain embeddings to capture the temporal evolution in domains. The domain embeddings are generated by a fully connected network that we term the *Domain Predictor* which captures dependencies across the domains. We train our model, which comprises the classifier network, the hypernetwork, and the domain predictor through a joint loss that maintains continual learning capability. We conduct extensive experiments with datasets such as MNIST, CIFAR10, CIFAR100, Eurosat, FER, and Fashion MNIST, that are transformed in a myriad of ways to simulate the evolution. Our proposed approach achieves state-of-the-art performance on a multitude of different benchmarks that we have proposed to evaluate for the evolving continual learning setup.

The principal contributions of our work are as follows:

- We propose a more general treatment of continual learning and evolving domain generalization wherein we simultaneously tackle the problems of catastrophic forgetting over tasks and temporal generalization over domains.
- We propose a novel architecture comprising a hypernetwork with a domain predictor module capable of continually learning tasks arising from temporally evolving domains.
- We introduce a new set of benchmarks to pave the way for future work in this direction.
- We conduct extensive experiments with real-world data sets and demonstrate that our method can successfully handle continual learning under evolving domain distributions.

2. Related Works

Our work lies at the crossroads of domain generalization (DG) [30] and continual learning (CL) [31]. As far as we are aware, there is a singular work that addresses continual learning with tasks that span a myriad of domains. Cross domain continual learning (CDCL) [26] considers a problem setting wherein each task contains samples drawn from different domains. The model is trained on a subset of the domains and tested on a held-out test domain at the end of learning each task. There are two fundamental differences between CDCL's setting and ours. All the domains are available simultaneously during training each task whereas we consider temporally changing domains that are made available with time. Secondly in EvoCL (i.e, the proposed work), the nature of how domains differ significantly differs from that of CDCL which tackles a vanilla DG setting where the domains are independent. In contrast, we approach a problem where the domains temporally evolve. Such a setting is known in existing literature as evolving domain generalization or temporal domain generalization.

Approaches addressing evolving domain generalization are predominantly recent endeavors and thus are few in number. Particularly, the problem focuses on achieving out-of-distribution generalization when the domains are evolving, in the sense that there is a temporal pattern to how each domain varies from the preceding and the succeeding domains. The problem was addressed using techniques incorporating ideas from meta-learning [15], gradient interpolation [20], latent space exploration [21], and recurrent graph generation [1].

The field of Continual Learning has enjoyed widespread success and adoption in recent years and has branched out to a variety of different strategies to combat catastrophic forgetting. Broadly speaking, they can be classified into regularization-based approaches [10, 17, 29], replay-based techniques [23, 24, 28] and dynamic architecture-based approaches [33]. One of the recent advances in the field of Continual Learning is hypernetwork. Hypernetwork [5] was first proposed as a means of weight compression. It was not long before this idea of bespoke network generation was found to be an effective approach for continual learning [2, 29]. Hypernetworks possess some qualities that are greatly desired in CL. Hypernetworks have a large capacity to learn extended sequences of tasks. Furthermore, hypernetworks demonstrate strong knowledge transfer capabilities in between tasks. Their versatility makes them a suitable candidate for our CL strategy. We build atop this framework to incorporate the concept of generalizing those continually learnt tasks to future domains in a temporally evolving setting. CL techniques such as Hypernetworks are founded with a focus on mitigating catastrophic forgetting. They are not, however, built to anticipate temporal domain changes and generalize to future unseen domains.

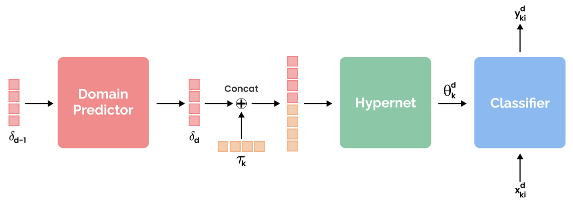

Figure 2. Architecture of the model with its components Domain Predictor, Hypernetwork, and Classifier; along with their interactions. The *Domain Predictor* takes in the previous domain embedding as its input and outputs the current *domain embedding* ξ_d. This *domain embedding* is concatenated with the *task embedding* τ_k, which is initialised randomly and learnt via backpropagation. The concatenated embedding is input to the *hypernetwork* which in turn generates the parameters θ_k^d of the *classifier* for the k^{th} task and d^{th} domain. The *classifier* finally acts on the input x_k^d and generates prediction y_k^d. The *Hypernetwork*, the *Domain Predictor* and the *Task Embeddings* are jointly optimised via backpropagation.

3. Problem Formulation

We consider a problem setting consisting of sequential tasks that evolve over time. Each task T_k follows a probability distribution P_k. In several practical scenarios, tasks are subject to distribution shift with time, leading to an evolution over their domains. Let us assume this temporal evolution results in D domains, with each domain consisting of K sequential tasks. We represent a task T_k^d to follow a probability distribution P_k^d for a task k over a domain d. Let us consider an ordered collection of data from the sequence of tasks over domains as

$$\mathbf{S} = \{S_1^d, S_2^d, \ldots, S_K^d\}_{d=1}^{D} \tag{1}$$

with S_k^d representing the data pertaining to each task T_k^d. The data within each task is represented as:

$$S_k^d = \{(x_{ki}^d, y_{ki}^d) \in \mathcal{X}_k^d \times \mathcal{Y}_k\}_{i=1}^{N_k^d} \tag{2}$$

Each task is a collection of N_k^d labeled samples and $\mathcal{X}_k^d$ and $\mathcal{Y}_k$ are the corresponding feature and label sets specific to a particular task and domain. Our objective here is to continually learn from the sequence of tasks $\mathbf{S}$ and to generalize to tasks associated with a future unseen domain. We aim to develop a model that exhibits strong generalization performance on data arising from tasks of a future unseen domain $\{S_1^{D+1}, S_2^{D+1}, \ldots, S_K^{D+1}\}$.

4. Methodology

Due to continual learning restrictions, it is impractical to maintain separate classifier parameters trained on each task, and each domain that it evolves through. Moreover, we want to capture the shift in domains through our parameterization of the classifier. To achieve this, we propose a model that can generate unique weights and biases to parameterize the classifier on the fly as per the task and domain evolution. The proposed model incorporates a hypernetwork, which is well suited to the role of generating the parameters of a classifier network. The hypernetwork can generate model parameters unique to every task by conditioning on a task embedding. Secondly, the covariate space $Pr(\mathcal{X}_k^d)$ of each task evolves temporally across the domains but in a similar manner across the tasks. This necessitates a model capable of capturing this evolutionary pattern in the parameter space. We let our hypernetwork capture this by conditioning it with respect to a domain embedding that evolves over time. We model the evolution in domain embedding by utilizing a network we term the *domain predictor* which is trained to predict the next domain embedding given the current domain embedding in a Markovian fashion. The Markovian formulation is necessitated by the absence of data from previous domains, mandated by the Continual Learning setting. Thus, our proposed model consists of three components: a classifier network, a hypernetwork, and a domain predictor network. Figure 2 provides an overview of our proposed architecture. We describe in detail different components in our model and learning of these model parameters to perform continual learning over evolving domains.

4.1. Classifier

We assume there is a classifier network $\mathcal{F}(x_k^d; \theta_k^d)$ with parameters θ_k^d associated with task k and domain d. The classifier network $\mathcal{F} : \mathcal{X}_k^d \rightarrow \mathcal{Y}_k$ is a function that maps from an input space that is both domain and task-specific to a task-specific label space. The classifier $\mathcal{F}(x_k^d; \theta_k^d)$ can be any network such as a fully connected neural network

(FNN) or any type of convolutional neural network (CNN) [13] such as LeNets [14] or ResNets [6]. A significant departure with our approach is that the classifier network parameters are not updated through backpropagation. Instead, it is generated through a hypernetwork whose parameters are learnt through backpropagation.

4.2. Hypernetwork

The hypernetwork is a meta-model that serves as a weight generator that can dynamically parameterize a target model based on the input that it receives. The hypernetwork $\mathcal{H}(\xi, \tau; \phi)$ is a function that is modeled as a neural network with parameters ϕ. The hypernetwork takes in two inputs: a domain embedding ξ and a task embedding τ. As the setting being studied consists of a variety of tasks over different domains, we require that the hypernetwork generates model weights that best befit the data conforming to the domain-task pair under consideration.

Each task T_k has its own corresponding unique task embedding τ_k. Note that the task embedding is domain agnostic, meaning that there is a singular task embedding per task irrespective of whatever domain it's drawn from. The task embeddings and hypernetwork parameters are learnt by optimizing with respect to the cross-entropy loss computed on the data associated with each task as we see them. We optimize the following task-specific loss to learn the parameters while we are at task k in domain d,

$$\mathcal{L}_{task} = \sum_{i=1}^{N_k^d} \mathcal{L}_{ce}(\mathcal{F}(x_{ki}^d, \mathcal{H}(\xi_d, \tau_k; \phi)), y_{ki}^d) \quad (3)$$

where $\mathcal{L}_{ce}$ is the cross entropy loss. The task-specific loss is optimized through backpropagation to learn the task embeddings and hypernetwork parameters. During backpropagation, the gradients flow through the classifier, then through the hypernetwork's parameters, and finally to the task embeddings. The hypernetwork parameters and the task embeddings are updated during backpropagation. As for the domain embeddings, we introduce the mechanism underneath their generation in the following section.

An issue with the aforementioned optimization is that the hyper-network parameters get adapted to the latest task, and this leads to catastrophic forgetting of the earlier tasks in the sequence. As the hyper-network gets adapted to generate the latest task parameters, it fails to generate correctly the parameters of the initial task classifier. We overcome this by performing a regularization over the hypernetwork outputs as we learn them on the new task data. The regularization [29] takes the following form:

$$\mathcal{L}_{reg} = \frac{1}{k-1} \sum_{j=1}^{k-1} ||\mathcal{H}(\xi_d^*, \tau_j; \phi^*) - \mathcal{H}(\xi_d, \tau_j; \phi)||^2 \quad (4)$$

where ϕ^* and ξ_d^* are the frozen hypernetwork weights and frozen domain embedding weights respectively before learning the current task k in the domain d. The regularization ensures that while learning on the data from task k, the current hypernetwork parameters (ϕ) will be able to generate the parameters associated with previous tasks similar to the one generated using frozen hyper-network weights. This regularization term over all the previous tasks ensures that the hypernetwork will be able to generate correctly the parameters for the previous tasks given the corresponding task embedding. Consequently, the parameters of the hypernetwork are learnt by considering the following total loss function:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \beta \cdot \mathcal{L}_{reg} \quad (5)$$

where β is a hypernetwork regularization hyperparameter that controls the magnitude of regularization.

4.3. Domain Predictor

The Domain Predictor $\mathcal{R}(\xi; \lambda)$ is a fully connected network that maps the drift between two consecutive domains. The domain predictor is conditioned on the prior domain embedding and predicts the current domain embedding. The domain predictor is trained end to end along with the hypernetwork, across all tasks. As the problem setting consists of a sequence of tasks within each domain, the domain predictor learns through one task after another, leading to a drift in its parameters towards the later tasks. This puts the domain predictor at risk of predicting a drifted domain embedding that no longer generates ideal classifier weights in conjunction with the task embeddings for the earlier tasks, resulting in catastrophic forgetting of the earlier tasks. To mitigate this, we regularise the output of the domain predictor along with the hypernetwork parameters. This regularisation is over all previous tasks. We regularise the outputs of the hypernetwork with its parameters ϕ and the domain embedding ξ_d in their current states with the outputs of the frozen hypernetwork with its parameters ϕ^* and domain embedding ξ_d^*, both frozen prior to learning the current task. This leads to the following regularisation term formulation:

$$\mathcal{L}_{reg} = \frac{1}{k-1} \sum_{j=1}^{k-1} ||\mathcal{H}(\xi_d^*, \tau_j; \phi^*) - \mathcal{H}(\mathcal{R}(\xi_{d-1}; \lambda), \tau_j; \phi)||^2 \quad (6)$$

Summarily, the overall optimisation objective is as follows:

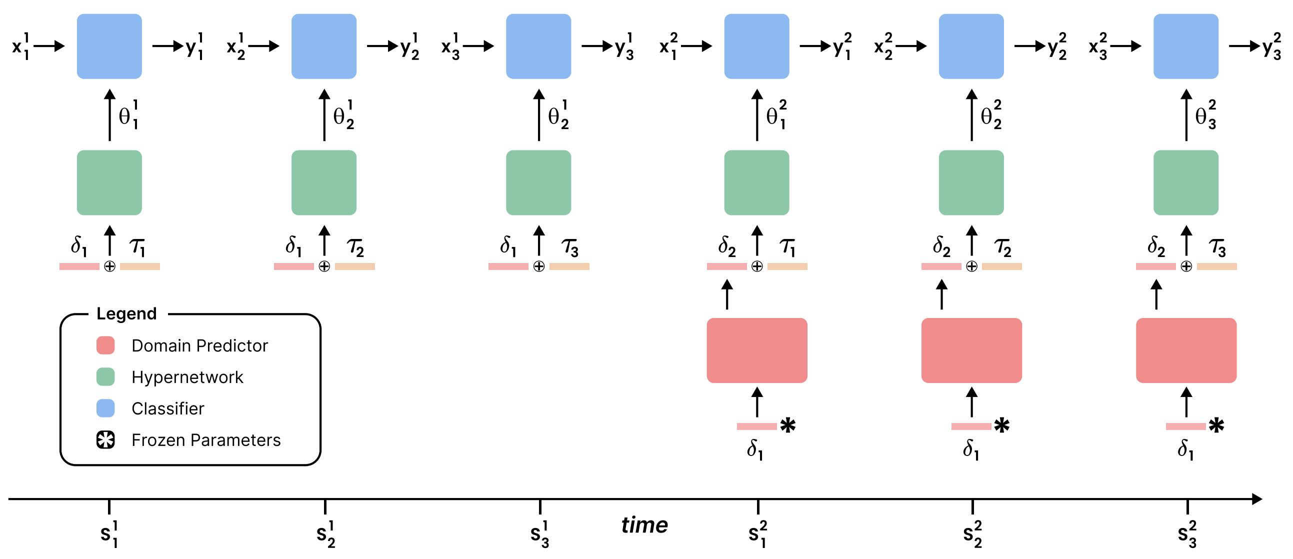

Figure 3. Unfolded schematic of the architecture showcasing embedding combinations used across domains. The figure contains three tasks over two domains. (Diagram best viewed in color)

$$\arg\min_{\tau_k,\phi,\lambda} \sum_{i=1}^{N_k^d} \mathcal{L}_{ce}(\mathcal{F}(x_{ki}^d, \mathcal{H}(\mathcal{R}(\xi_{d-1};\lambda), \tau_k;\phi)), y_{ki}^d) +$$
$$\beta \cdot \frac{1}{k-1} \sum_{j=1}^{k-1} ||\mathcal{H}(\xi_d^*, \tau_j;\phi^*) - \mathcal{H}(\mathcal{R}(\xi_{d-1};\lambda), \tau_j;\phi)||^2 \quad (7)$$

5. Experiments

For the primary set of experiments, we train the model on 5 evolving domains continually. Each domain contains a number of tasks that vary between experiments, the details of which are presented in the Dataset section that follows. The results are measured on tasks from a test domain that is not seen during training. The results presented in 1 are averaged across all tasks.

5.1. Architecture details

The hypernetwork we employ is a fully connected feed-forward neural network comprising three hidden layers. Each hidden layer of the hypernetwork consists of 64 or 100 nodes depending on the experiment (100 for CIFAR10/100 and 64 for the rest). We utilize domain and task embeddings of dimensions 16 and 32 respectively.

The domain predictor is a fully connected network with three hidden layers of dimension 64 each. The input to the domain predictor is the previous domain embedding of dimension 16, and the output is the current domain embedding of the same dimension.

Algorithm 1 Training Algorithm

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

```
1:  Initialise domain embedding $\xi_d$
2:  Initialize list of task embeddings $\mathcal{T}$
3:  for $d = 1, 2, ..., D$ do
4:      for $k = 1, 2, ..., K$ do
5:          if $d \neq 1$ then
6:              $\xi_d \leftarrow \mathcal{R}(\xi_{d-1};\lambda)$
7:          end if
8:          $\theta_k^d \leftarrow \mathcal{H}(\xi_d, \tau_k;\phi)$
9:          $\hat{y}_k^d \leftarrow \mathcal{F}(x_k^d;\theta_k^d)$
10:         if $k = 1$ then
11:             $loss \leftarrow \mathcal{L}_{task}$
12:         else
13:             $loss \leftarrow \mathcal{L}_{task} + \beta \cdot \mathcal{L}_{reg}$
14:             Optimize $loss$ w.r.t $\tau_k, \phi, \lambda$
15:             $\phi^* \leftarrow \phi$
16:             $\xi_d^* \leftarrow \xi_d$
17:         end if
18:     end for
19: end for
```

The main network or the classifier is a CNN [13], specifically a ResNet [6]. The Resnet consists of 4 residual blocks with 8, 8, 32, and 64 feature maps respectively. Each residual block comprises 2 convolutional layers. The string of residual blocks is followed by 2 fully connected layers.

The loss function considered is a cross entropy loss. We use Adam [9] as the optimizer. Adam is configured with exponential decay rates β_1 and β_2 of 0.90 and 0.99, respectively, and a learning rate of 0.001.

5.2. Datasets

To illustrate the problem of continuously evolving domains, we employ the following classification datasets each demonstrating distribution shift through different transformations. The specific values of each domain are highlighted in the figure that follows.

(1) **MNIST**: The MNIST [3] dataset comprises of grayscale images of handwritten digits. The digits are sequentially paired in this setup to form 5 binary classification tasks. To simulate a continuous distribution shift, the digits are rotated by increments of 30 degrees to create 5 training domains and one test domain.

(2) **CIFAR-10**: CIFAR-10 [12] contains ten classes of images of various real-world objects which we organize sequentially into five tasks. To simulate drift, we continuously reduce the image exposure by changing gamma values across domains by increments of 0.2. The evolution portrayed here amounts to a lossy transformation of information as with a reduction in exposure, the information retained in each subsequent domain is less than the previous one.

(3) **CIFAR-100**: To further demonstrate that our approach can scale to more domains and tasks, we employ the CIFAR-100 [11] dataset and organize it in a similar fashion to CIFAR-10 resulting in 10 tasks across 15 training domains ranging from $\gamma = 1.0$ to $\gamma = 3.8$ and 1 test domain at $\gamma = 4.0$.

(4) **Eurosat**: Eurosat [8] is a collection of Sentinel-2 satellite images of different land cover types. We use the variant with only the RGB spectral bands. Similar in fashion to the above two setups, the ten classes of Eurosat are organized into 5 binary classification tasks. We incrementally add cloud cover following the method proposed in [7] to showcase distribution shift through progressive occlusion of data. This too is a lossy evolution as increasing parts of the images are occluded.

(5) **FER**: The Facial Expression Recognition dataset [4] entails images of human faces displaying 7 different types of emotions. We structure the 7 classes into two tasks of 4 and 3 classes respectively. We progressively perturb the image by adding increasing levels of Gaussian noise by keeping the mean at 0 and varying the standard deviation across domains in increments of 10 to create 5 training domains and 1 test domain.

(6) **FMNIST-**: Fashion- [32] utilizes the Fashion MNIST dataset that contains 28 x 28 grayscale images of 10 types of apparel. We organize this into tasks in a manner similar to MNIST. We exhibit distribution shifts through different viewing distances. To simulate this, we progressively reduce the image resolution in steps of 4 pixels on both dimensions and pad with black pixels to keep the image dimensions constant at 28 x 28.

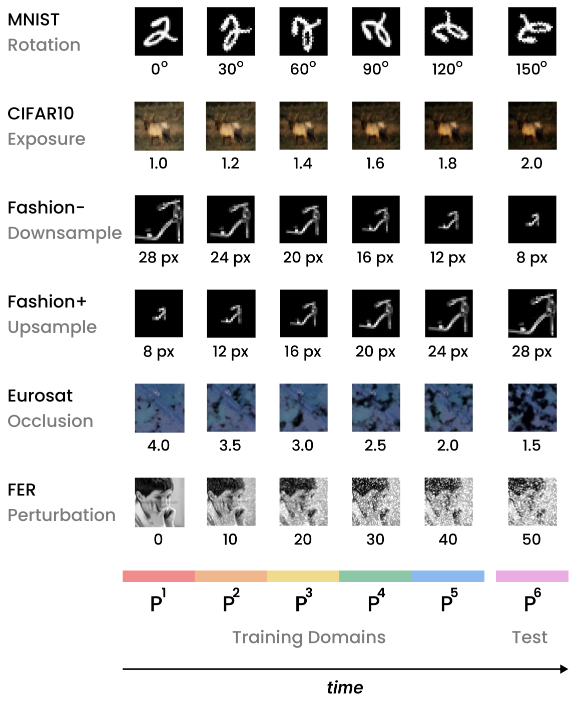

Figure 4. Figure displaying how each dataset temporally evolves. We choose a wide array of transformations ranging from lossless transformations such as rotation in the MNIST dataset to lossy transformations such as noise perturbation in FER. We train EvoCL on 5 training domains and test it on the 6th domain which is unseen during training.

(7) **FMNIST+**: Fashion+ proceeds the same way as the previous experiment but the order is reversed. So we start with a small image and progressively grow its size in similar steps as in the above experiment.

(8) **TinyImageNet**: We utilize the TinyImageNet dataset comprising 200 classes that are split into 20 tasks of 10 classes each. Then, we progressively blur the images to simulate a domain evolution over tasks. Specifically, we apply Gaussian Blur over the image with increasing kernel sizes ranging from 1 to 9 for the training domains and 11 for the test domain.

5.3. Baselines

We consider the following six baselines:

(1) **Last Domain**. Last Domain is a continual learning baseline wherein we train a hypernetwork only on the last training domain. In other words, we train on the domain closest to the test domain.

Table 1. Classification accuracy in a Task Incremental Setting on test domains comparing EvoCL with baselines

	Last	**Last+**	**ADTI**	**ADTI+**	**DRAIN**	**ProTune**	**EvoCL**
MNIST	90.982 ± 0.54	93.756 ± 0.82	82.697 ± 0.23	93.233 ± 0.61	85.961 ± 0.88	94.002 ± 0.79	**96.752** ± 0.95
CIFAR10	69.934 ± 1.09	62.231 ± 1.22	69.561 ± 1.45	78.992 ± 1.30	78.340 ± 1.27	85.432 ± 1.22	**90.944** ± 1.15
CIFAR100	55.682 ± 1.13	56.232 ± 1.34	71.841 ± 1.09	82.121 ± 1.21	55.735 ± 1.24	82.834 ± 1.43	**83.512** ± 1.11
Eurosat	61.612 ± 0.68	83.772 ± 0.92	71.014 ± 1.07	85.723 ± 0.69	68.836 ± 0.81	89.046 ± 1.24	**93.312** ± 0.76
FER	70.563 ± 1.61	73.434 ± 0.96	85.214 ± 0.64	85.972 ± 0.85	74.012 ± 0.73	88.212 ± 0.68	**90.002** ± 0.71
FMNIST+	87.642 ± 0.73	87.301 ± 0.91	86.935 ± 0.70	96.823 ± 0.78	93.421 ± 0.84	93.435 ± 1.16	**98.563** ± 0.77
FMNIST-	82.432 ± 0.81	88.991 ± 1.29	74.254 ± 0.78	93.752 ± 0.82	90.226 ± 0.84	91.203 ± 0.80	**99.111** ± 0.85
TinyImgNet	60.432 ± 1.70	61.672 ± 1.63	50.992 ± 1.33	51.631 ± 1.63	51.402 ± 1.27	54.026 ± 1.35	**63.913** ± 1.38

(2) **Last Domain +**: This setup is an extension of the above wherein we follow the same training procedure but fine-tune the model on a small subset of the test set. We then evaluate the performance of the rest of the test set.

(3) **All Domain Task Incremental**: ADTI is a standard continual learning setup wherein each task consists of samples from all domains. Citing the MNIST example, we first train on the first binary classification task between 0 and 1 with samples at all the training domain angles. We then move on to training on the subsequent tasks in a continual fashion and finally test on test domain tasks. The model architecture is again similar to the two baselines stated previously.

(4) **ADTI+**: This setup is an extension of the above wherein we follow the same training procedure but fine-tune the model on a small subset of the test set. We then evaluate the performance on the rest of the test set.

(5) **Progressive Finetuning**: In progressive fine tuning (ProTune), we train the model solely on the first training domain and fine-tune it over and over with the successive training domains. We then test it on the test domain.

(6) **DRAIN**: Unlike the above baselines, DRAIN [1] is not a continual learning baseline. In this setup, there is no notion of tasks. Instead, all the classes of a domain are treated as a single multi-class classification problem. The domains, however, do arrive in sequence.

5.4. Results

The classification accuracy as averaged over the tasks on the test domains is presented in Table 1. This is in a Task Incremental Learning setting. Class Incremental Learning results are delegated to the Appendix. Empirically, we exhibit that EvoCL outperforms all the baselines across experiments involving all the datasets. EvoCL easily outclasses rudimentary baselines such as Last Domain and Last Domain +. Baselines utilizing the entire domain spectrum in an offline setting (ADTI and ADTI+) also pale in comparison to EvoCL's numbers. Conversely, we also consider DRAIN as a baseline where the domains occur in sequence, with each domain possessing all tasks in an offline arrangement. DRAIN's performance is also eclipsed by EvoCL. Progressive Finetuning that iteratively fine tunes on newer domains, as they arrive, performs best among all baselines but is still outclassed by EvoCL.

5.4.1 Forward Transfer

As evident from Figure 5, we can observe a strong forward transfer of knowledge across domains as the model learns. The plot shows progressively increasing accuracy with each subsequent domain. In fact, with training on just a single epoch on each task, the model's parameters can converge which can be attributed to good forward transfer characteristics.

5.4.2 Hypernetwork Regularisation

Experiments on varying hypernetwork regularization demonstrate that the best value of regularization varies from dataset to dataset. However, a regularization value of $\beta = 0.1$ demonstrates superior test domain accuracy numbers on the majority of datasets. The results of the comparison study are compiled in Table 3.

5.4.3 Ablation Studies

To validate our claims that the Domain Predictor plays a key role in modeling the temporal evolution of the domains, we undertake an ablation study wherein we compare the standard EvoCL architecture with a version bereft of the domain predictor. The Domain Predictor-less version is composed solely of a hypernetwork with task embeddings that generate classifier parameters. This modified version is trained in the same manner as the standard EvoCL model with the same order of tasks and domains. The difference is that the modified version is oblivious to task evolution and the scenario plays out similarly to a model that is fine-tuned

Table 2. Results of the ablation demonstrating the architecture bereft of (i) task embeddings as displayed in the first row, (ii) Domain Predictor and domain embeddings as represented in the middle row; compared with the complete EvoCL architecture with both task and domain embeddings as showcased in the last row.

	MNIST	**CIFAR 10**	**CIFAR100**	**Eurosat**	**FER**	**FMNIST-**	**FMNIST+**	**TinyImageNet**
w/o $\mathcal{T}$	85.445	78.504	54.856	64.624	73.245	92.296	90.824	49.231
w/o $\mathcal{R}$	92.495	82.383	82.728	85.736	87.758	92.737	91.029	52.573
EvoCL	96.882	91.324	83.235	93.891	90.472	98.982	99.213	64.338

Table 3. A comparison of results from varying the hypernetwork regularisation hyperparameter β

β	**MNIST**	**CIFAR 10**	**CIFAR100**	**Eurosat**	**FER**	**FMNIST-**	**FMNIST+**	**TinyImageNet**
1e-1	96.882	91.324	83.235	93.891	90.472	98.982	99.213	64.338
1e-2	96.756	91.287	83.724	93.720	90.954	99.877	98.802	62.234
1e-3	96.192	90.235	93.245	93.831	88.472	94.213	96.932	61.845

progressively as new domains are presented one after another. Similarly, we perform yet another ablation retaining the Domain Predictor but removing task embeddings. The experimental setup again is identical to the above two. The results of the ablation are presented in the Table 2.

The ablation studies demonstrate the superior performance of having both Domain Predictor and Hypernetwork incorporated in the EvoCL architecture when compared with just a hypernetwork or without task embeddings. This is due to the fact that EvoCL can accurately capture the shift in domains through its Domain Predictor component and model the different tasks via its hypernetwork. On the contrary, losing out on task embeddings hampers the model's ability to distinguish between a new task and an already learnt task appearing from a new domain. Similarly, having the Hypernetwork alone is akin to progressive finetuning of the model which inhibits learning of any pattern in the domain evolution.

The experiments were performed on an x64 machine with an Intel Xeon Gold 6130 CPU, with 32 cores, clocked at 2.10GHz. The system is equipped with 128 gigabytes of RAM, an Nvidia A6000 GPU with 48 GB VRAM, and 10752 CUDA cores.

6. Conclusion

We introduce a novel problem setting comprising both continual learning and domain evolution. To tackle the challenge, we propose a hybrid hypernetwork architecture conditioned on learnable task embeddings and domain embeddings generated by a domain predictor. We perform extensive experiments on a wide variety of datasets, each showcasing domain evolution through different styles of progressive transformations. We compare our model with competitive baselines in a variety of experiments and demonstrate state-of-the-art performance on the aforementioned datasets. One limitation of the model is that it expects all the tasks to evolve at the same rate which might not necessarily be the case in a practical scenario. We can encounter tasks that evolve at different rates which the current approach cannot capture. With EvoCL, we open a new problem direction and set the stage for future works in this direction.

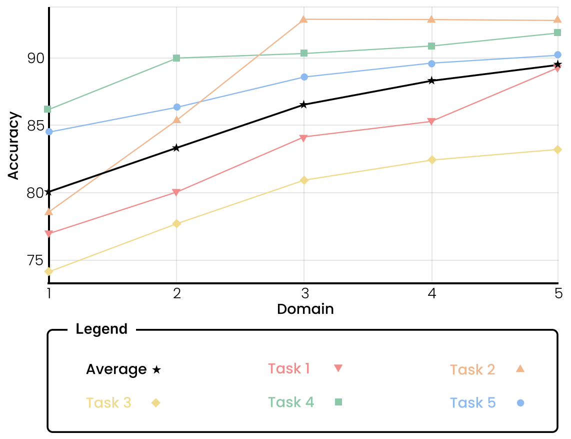

Figure 5. Demonstration of forward transfer characteristics across domains during training. Individual task accuracy and average accuracy on the CIFAR10 dataset are presented.

7. Acknowledgment

We acknowledge the funding support from the Science and Engineering Research Board, India(SERB - CRG/2021/006436) and the infrastructure support from the Japan International Cooperation Agency (JICA).

References

[1] Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks, 2023. 2, 7

[2] D.S. Chandra, S Varshney, P.K. Srijith, and S Gupta. Continual learning with dependency preserving hypernetworks. *WACV 2022*, 2022. 3

[3] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 6

[4] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests, 2013. 6

[5] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016. 3

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4, 5

[7] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018. 6

[8] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019. 6

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

[10] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 2

[11] 2009 Krizhevsky. Cifar100. 6

[12] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 6

[13] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4, 5

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[15] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Sequential learning for domain generalization, 2020. 1, 2

[16] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization, 2017. 1

[17] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017. 1, 2

[18] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. 1

[19] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching, 2021. 1

[20] Anshul Nasery, Soumyadeep Thakur, Vihari Piratla, Abir De, and Sunita Sarawagi. Training for the future: A simple gradient interpolation loss to generalize along time, 2021. 2

[21] Tiexin Qin, Shiqi Wang, and Haoliang Li. Generalizing to evolving domains with latent structure-aware sequential autoencoder, 2022. 2

[22] J. Quinonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer. *Dataset shift in machine learning*. 2009. 1

[23] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference, 2019. 3

[24] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. Experience replay for continual learning, 2019. 1, 3

[25] Muhammad Sajid, Nouman Ali, Naeem Iqbal Ratyal, Muhammad Usman, Faisal Mehmood Butt, Imran Riaz, Usman Musaddiq, Mirza Jabbar Aziz Baig, Shahbaz Baig, and Umair Ahmad Salaria. Deep Learning in Age-invariant Face Recognition: A Comparative Study. *The Computer Journal*, 65(4):940–972, 12 2020. 1

[26] Christian Simon, Masoud Faraki, Yi-Hsuan Tsai, Xiang Yu, Samuel Schulter, Yumin Suh, Mehrtash Harandi, and Manmohan Chandraker. On generalizing beyond domains in cross-domain continual learning, 2022. 2

[27] Robert J. Sternberg. Adaptive intelligence: Its nature and implications for education. *Education Sciences*, 11(12), 2021. 1

[28] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, mar 1985. 3

[29] J von Oswald, C Henning, B.F. Grewe, and J Sacramento. Continual learning with hypernetworks, 2022. 2, 3, 4

[30] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip S. Yu. Generalizing to unseen domains: A survey on domain generalization, 2022. 2

[31] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. 2

[32] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 6

[33] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018. 3