DIRA: **D**ynamic **I**ncremental **R**egularised **A**daptation

Abanoub Ghobrial[1,*], Xuan Zheng[1], Darryl Hond[2], Hamid Asgari[2], and Kerstin Eder[1]
[1] University of Bristol, Bristol, UK
[2] RTI, Thales UK, Reading, UK
*corresponding author

***Abstract*—Autonomous systems (AS) often use Deep Neural Network (DNN) classifiers to allow them to operate in complex, high-dimensional, non-linear, and dynamically changing environments. Due to the complexity of these environments, DNN classifiers may output misclassifications during operation when they face domains not identified during development. Removing a system from operation for retraining becomes impractical as the number of such AS increases. To increase AS reliability and overcome this limitation, DNN classifiers need to have the ability to adapt during operation when faced with different operational domains using a few samples (e.g. 2 to 100 samples). However, retraining DNNs on a few samples is known to cause catastrophic forgetting and poor generalisation. In this paper, we introduce Dynamic Incremental Regularised Adaptation (DIRA), an approach for dynamic operational domain adaption of DNNs using regularisation techniques. We show that DIRA improves on the problem of forgetting and achieves strong gains in performance when retraining using a few samples from the target domain. Our approach shows improvements on different image classification benchmarks aimed at evaluating robustness to distribution shifts (e.g.CIFAR-10C/100C, ImageNet-C), and produces state-of-the-art performance in comparison with other methods from the literature.**

I. Introduction

Autonomous systems (AS) often are developed using deep neural network (DNN) classifiers to interact and adapt in dynamically changing real-world environments to achieve their intended goals. The benefit of using DNNs in autonomous systems is their ability to learn complicated patterns from complex environments, and thus produce highly non-linear decision boundaries to cope with the complexity of operational environments. However, it is a challenge to verify the behaviour of DNNs. A popular example of such ASs is self-driving cars. Current research shows that for each self-driving car, an impractical amount of testing is required to verify the system for deployment [1]. Innovative methods of increasing the efficiency of testing and validation are actively being developed to make the process more practical, e.g. [2, 3]. However, due to the vast operational environments and the enormous effort required in testing to achieve deployment, the community is additionally incorporating trustworthiness assessment of AS to allow for reliable deployment and progressive improvements during the operational lifetime of such systems [4]. This follows a two-stage approach presented by Koopman et.al. [5], which stipulates that, given an AS passes some minimum safety validation case, the system is deployed and is improved during operation to increase its reliability over time. Thus, the system is allowed to adapt to dynamically changing operational environments.

The process of continual learning or adaptation can be broken into three stages 1) detection of change in the operational domain, e.g. [6, 7, 8, 9], 2) supply of labels from an oracle or ground truth for new operational domain samples, e.g. [10, 11], 3) retraining. Alternatively, 2) and 3) can be substituted by one stage of self-supervised or unsupervised retraining. We aim to investigate this option in future work. In this paper, we focus on point 3), i.e. retraining.

DNN classifiers use gradient-based optimisation algorithms to learn. The gradient optimiser modifies the decision boundary based on the samples used in training. Retraining using few samples can result in a phenomenon known as *catastrophic forgetting*, where the model overfits to the few training samples used and does not generalise to the domain distribution [12]. Generally, to overcome catastrophic forgetting, new samples are added to the initial training dataset and the classifier is fully retrained. Full retraining, however, can be cumbersome to perform during operation. In this paper, we propose Dynamic Incremental Regularised Adaptation (DIRA), a framework to achieve operational domain adaption by retraining using only few samples from the target domain. We utilise the concept of regularisation in our framework to overcome the need for full retraining.

Practically, upon adaptation of AS, reassessment of the system's safety may be required. The safety compliance of evolving DNN classifiers during operation against a set of requirements or regulations is beyond the scope of this paper, but may be achieved through the use of runtime safety behavioural checkers as presented by Harper et.al. [13] or by using online methods for quantifying trustworthiness in predictions during operation as shown by Ghobrial et al [14].

In the next section, we discuss related work material. Section III introduces our method. Experimentation Setup and Results & Discussion are handled by sections IV and

This research is part of an iCASE PhD funded by EPSRC and Thales UK. Abanoub Ghobrial (e-mail: abanoub.ghobrial@bristol.ac.uk), Xuan Zheng (e-mail: dq18619@bristol.ac.uk), and Kerstin Eder (e-mail: kerstin.eder@bristol.ac.uk) are with the Trustworthy Systems Lab, Department of Computer Science, University of Bristol, Merchant Ventures Building, Woodland Road, Bristol, BS8 1UB, United Kingdom. Darryl Hond (e-mail: darryl.hond@uk.thalesgroup.com) and Hamid Asgari (e-mail: hamid.asgari@uk.thalesgroup.com) are with Technology and Innovation Research, Thales, Reading, United Kingdom.

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00090

V, respectively. We conclude and discuss future works in Section VI.

II. Related work

A. Types of Incremental Learning

Gradually assimilating new information from a continuously changing data stream, known as 'continual learning', poses a challenge for deep neural networks. Continual learning, however, is a fundamental aspect of evolving autonomous systems. In a continual learning setting the problem is broken down into several parts that need to be learned sequentially. In the continual learning literature, these parts are often called *tasks*. Thus, the term tends to have several meanings. These several connotations of the term task make it difficult to study the different challenges associated with continual learning. To overcome this problem, Ven et al.[15], proposed to brake down continual learning into three incremental learning scenarios: task-incremental, domain-incremental, and class-incremental learning (see Table I). Each scenario describes the context of the parts required to be learned sequentially, formerly the three scenarios contexts were referred to using the term task. Braking continual learning into different scenarios makes it more convenient to study the different challenges associated with each scenario, and subsequently develop appropriate techniques to overcome the associated challenges [16, 17, 18, 19].

The first scenario (task-incremental learning), describes the case where the algorithm is required to learn incrementally a set of distinct tasks. For example, if a neural network model was to classify numbers from 0 - 9 in English (like in the MNIST dataset [20]), then a new task for the model can be to learn to classify samples in Permuted-MNIST [12] or Fashion-MNIST [21] i.e. the same number of classes but the pattern has changed distinctively. For more examples see [22, 23, 24]. In the second scenario (domain-incremental learning), the model needs to learn the same problem but in different contexts, because the domain or input distribution has shifted. Using our previous example of classifying digits 0 - 9, in domain incremental learning, the model is required to learn to classify digits 0 - 9 but with Gaussian noise or contrast noise added to the input samples. See [25, 26] for more examples. The third scenario (class-incremental learning), describes when the model needs to learn a growing number of classes. In the example of classifying digits 0 - 9, class-incremental learning is the model learning to classify digit '10' as an additional class to the existing 0 - 9 classes. See examples [27, 28].

Since our focus is on dynamic distributional shifts during operation, we are interested in the second scenario of incremental learning i.e. domain-incremental learning. We focus on trying to achieve domain incremental adaptation using a limited number of samples.

B. Domain Adaptation Frameworks

There have been a number of introduced approaches in the literature that address the problem of domain-incremental adaptation. For a breakdown of categories for the different introduced approaches in the literature, we direct interested readers towards [29]. Here we will cover some state-of-the-art examples of these approaches relevant to our results discussed later in section V.

Scenario	Description
Task-Incremental Learning	Sequentially learn to solve a number of distinct tasks.
Domain-Incremental Learning	Learn to solve the same problem in different contexts.
Class-Incremental Learning	Differentiate between incrementally observed classes.

TABLE I: Overview of incremental learning scenarios [15]

One popular approach is using self-supervision to achieve domain adaptation. Test-time training (TTT) combine different self-supervised auxiliary contexts to achieve domain adaptation. They break down neural network parameters into three parts, such that pictorially the architecture has a *Y*-structure. The bottom section of the Y-structured architecture represents the input layer and the layers responsible for the shared feature extraction, whilst the other two sections contain layers for learning and outputting labels for the main and auxiliary tasks independently. An example of this auxiliary task is being able to tell the rotation of the input image. During training time the whole neural network is optimised using a combined loss function that aims to maximise performance on both the main and auxiliary tasks. During retraining to adapt to a new domain, only parameters of the shared feature extraction and the auxiliary task sections are allowed to change. By doing so the the shared feature extraction section of the network modifies to learn the new domain, so then the network may output correct predictions on the unchanged branch of the network responsible for the main task [30, 31].

Correcting domain statistics is another common approach to achieving domain adaption, e.g. [29, 32, 33, 34]. Some of these approaches rely on using a large number of samples to recalculate the running mean and standard deviation of batch normalisation layers for the target domain e.g. [33, 32]. Other approaches, like Dynamic Unsupervised Adaption (DUA) [29], combine the running mean and standard deviation for normalisation layers from the original domain and the target domain to achieve adaptation in an unsupervised fashion, whilst using significantly fewer samples (typically ≈ 100 samples).

Our proposed DIRA method aims at achieving adaptation through the regularisation of new and old information. We retrain the model on samples from the target domain. Therefore, we require labels to be provided with the retraining samples, which makes our approach a supervised instead of an unsupervised method. We use regularisation techniques to avoid catastrophic forgetting and achieve adaptation using very few samples. By doing so, we benefit from transfer learning of information from the initial domain to the target domain. Our philosophy is that if humans use transfer learning to learn and adapt to different environments, we believe that neural networks can also achieve domain adaptation in a similar

fashion. We see our approach can be combined with self-supervision methods, such as done in TTT, to overcome the need for providing labels. Exploring the use of self-supervision in our approach is left as future work. In this paper, we assume labels for samples from the target domain are available for retraining.

C. Regularisation

The concept of regularisation allows a neural network to learn new information whilst retaining previously learned information. This allows a neural network to learn new information without experiencing catastrophic forgetting and without needing access to training data of previously learnt information. Regularisation achieves this by presenting a penalisation term in the loss function of the optimisation problem. Several works in the literature have introduced different penalisation terms, some popular examples are Synaptic Intelligence (SI) [35], Learning without forgetting (LWF) [36], and Elastic Weight Consolidation (EWC) [37]. In DIRA, different regularisation techniques may be utilisable, however, based on surveys such as the one provided by Kemker [38], the EWC penalisation term results in state-of-the-art performance within regularisation techniques. Therefore, we developed our method predominantly based on EWC.

III. Method

We first summarise EWC regularisation [37] as our approach revolves around it. Then we discuss the details of our DIRA method.

A. Elastic Weight Consolidation

Kirkpatrick et al. [37] introduced Elastic Weight Consolidation (EWC) to overcome forgetting in task-incremental learning. EWC overcomes forgetting by introducing a penalisation term in the loss function when retraining. This penalisation term provides a sense of the importance of each weight in the trained model on the original classification task. Therefore, when retraining on a new task, the algorithm is guided to avoid making significant changes to weights with high importance to the initial task. In this paper, we are interested in adapting to new domains, instead of adapting to new tasks. In the rest of this section, we will discuss the derivation of EWC and outline the assumptions that need to be taken into account when using EWC for domain adaptation.

During training of a DNN, the goal is to minimise the loss function $\mathcal{L}(\theta)$, represented as the Log-Likelihood function $-log(P(\theta|D))$ [39]. This aims at estimating θ, which is the set of weights and biases in a DNN, given D, the dataset representing the samples of the distribution of interest. D can be split into two independent datasets such that $D = \{D_A, D_B\}$, where D_A and D_B are datasets that are trained on sequentially and each of them may represent a different distribution. Using the chain rule in probability it can be shown that:

$$log(P(\theta|D_A, D_B)) = log(P(D_B|\theta, D_A)) + log(P(\theta|D_A)) - log(P(D_B|D_A)) \tag{1}$$

Considering the RHS of equation 1:

- First term, using conditional independence $log(P(D_B|\theta, D_A)) = log(P(D_B|\theta))$ and hence can be seen as the loss function, $\mathcal{L}_{\mathcal{B}}(\theta)$ that needs to be minimised for the new distribution or dataset D_B alone.
- Second term, $log(P(\theta|D_A))$ is the loss function for training the neural network on distribution D_A only. Thus can be denoted as $\mathcal{L}_{\mathcal{A}}(\theta)$.
- The Third term, is irrelevant as this term is constant with respect to θ and thus is lost when optimising using the stochastic gradient descent (SGD) i.e. does not need to be computed. We will neglect this term for the rest of the derivation.

Therefore, the overall loss function in equation 1 can be written as:

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathcal{B}}(\theta) + \mathcal{L}_{\mathcal{A}}(\theta) \tag{2}$$

In continual learning, distribution A would have been trained on initially, and later samples from distribution B arise and must be learnt by the DNN. In this case, the term $\mathcal{L}_{\mathcal{A}}(\theta)$ is considered to be intractable as it is assumed that access to training samples for distribution A is not available after initial training.

The underlying idea of EWC is to take a Bayesian approach to adapt the DNN model parameters, therefore learning additional distributions whilst avoiding catastrophic forgetting or minimising forgetting. However, due to intractable terms, it is not possible to maintain the full posterior $P(\theta|D))$. An inference technique is required to approximate these intractable terms. EWC can be seen as an online approximate inference algorithm [40]. An essential assumption for EWC to approximate $\mathcal{L}_{\mathcal{A}}(\theta)$ is that the DNN has been optimised for D_A such that θ has reached a local or a global minimum, θ^*_A, for distribution D_A. This allows for $-log(P(\theta|D_A))$ to be approximated as a Gaussian distribution function at its mode using Laplace's method [41]. Expanding $-log(P(\theta|D_A))$ using Taylor series around θ^*_A:

$$\begin{aligned} -log(P(\theta|D_A)) \approx & - log(P(\theta^*_A|D_A)) \\ & + (\frac{\partial(-log(P(\theta|D_A)}{\partial\theta}|_{\theta^*_A})(\theta - \theta^*_A) \\ & + \frac{1}{2}(\theta - \theta^*_A)^T H(\theta^*_A)(\theta - \theta^*_A) \\ & + \cdots \end{aligned} \tag{3}$$

Considering the RHS of equation 3:

- The First term, is a constant and similar to earlier it will get lost in the SGD optimiser.
- Second term, evaluates to gradient 0 as it is assumed that θ^*_A is at the mode of the distribution.
- Third term; $H(\theta^*_A)$ is the Hessian of $-log(P(\theta|D_A))$ with respect to θ evaluated at θ^*_A, which is $(\frac{\partial^2(-log(P(\theta|D_A)}{\partial\theta^2}|_{\theta^*_A})$.

The Hessian can be computed by approximating it to the empirical Fisher information matrix. Using Bayesian rule:

$$\begin{aligned} H(\theta_A^*) = &- \frac{\partial^2(log(P(D_A|\theta)))}{\partial\theta^2}\bigg|_{\theta_A^*} \\ &- \frac{\partial^2(log(P(\theta)))}{\partial\theta^2}\bigg|_{\theta_A^*} \\ &+ \frac{\partial^2(log(P(D_A)))}{\partial\theta^2}\bigg|_{\theta_A^*} \end{aligned} \tag{4}$$

Considering the RHS of equation 4:

- First term can be approximated as the negative of the empirical Fisher information matrix, F, [42, 43, 44]. The Fisher matrix can be defined as a way of measuring the amount of information that a random observation $D_A[n]$ carries about a set of unknown parameters θ of a distribution that models $log(P(D_A|\theta)$, where n is an index falling within the size, N, of the observable random samples D_A. Formally, it is the negative of the expected value of the observed information, hence it can be shown that it approximates to the first term of equation 4:

$$\begin{aligned} F(\theta) &= -NE\left[\frac{\partial^2(log(P(D_A[n]|\theta)))}{\partial\theta^2}\right] \\ &\approx -N\frac{1}{N}\sum_{n=1}^{N}\frac{\partial^2(log(P(D_A[n]|\theta)))}{\partial\theta^2} \\ &= -\sum_{n=1}^{N}\frac{\partial^2(log(P(D_A[n]|\theta)))}{\partial\theta^2} \\ &= -\frac{\partial^2(log(P(D_A|\theta)))}{\partial\theta^2} \end{aligned} \tag{5}$$

  The approximation made to the expectation in equation 5 becomes exact as the number of observations or samples becomes infinite. Therefore, the data size N of the previous information is crucial to the applicability of using the EWC approximation.
- Second term, is the *prior probability*.That is the probability distribution the DNN represents before being trained on any observations i.e. datasets. Given that often θ in DNNs are initialised using a random uniform distribution, then this term evaluates to zero and hence is ignored by the EWC algorithm.
- Third term, evaluates to zero as non-dependent on θ.

Putting terms together from the previous steps makes equation 2 reach the EWC loss function presented by [37]:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_j \frac{\lambda}{2}F_{A,j}(\theta_j - \theta_{A,j}^*)^2 \tag{6}$$

where λ is a hyper-parameter presented by Kirkpatrick et al. to allow for fine-tuning to minimise forgetting, and j labels each parameter. We summarise the list of assumptions for which equation 6 holds:

Assumption 1: The DNN was trained very well on the previous distribution represented by D_A that θ has reached a local or a global minimum i.e. $\theta_A^* = argmin_\theta\{-log(P(\theta|D_A))\}$.
Assumption 2: "Enough" observations are available in D_A to allow for the approximation from the Hessian to the empirical Fisher information matrix.

B. Dynamic Incremental Regularised Adaptation (DIRA)

This section describes the algorithmic details of our method. In order to achieve successful domain-adaptation we have taken into consideration the two assumptions outlined in section III-A when developing DIRA. Let M_0 be the model trained on the original domain dataset X_0. The standard optimisation problem in training a neural network on the original domain with a loss function $\mathcal{L}_0$ solves:

$$\min_\theta \mathcal{L}_0(\theta) \tag{7}$$

The aim of our approach is to adapt the trained model to out-of-distribution target data X_T using a few number of samples S_T from the target domain. To achieve this goal we utilise the concept of transfer learning, aiming at reserving beneficial information learnt from the original domain to allow for successful adaptation to the target domain. Our hypothesis is that by using regularisation techniques one should be able to utilise this notion of transfer learning to achieve adaption with a limited number of samples from the target domain. Therefore, the problem we try to optimise for during adaptation becomes a combination of the loss function for the original domain $\mathcal{L}_0$ and the target domain $\mathcal{L}_T$:

$$\min_\theta \mathcal{L}_T(\theta) + \mathcal{L}_0(\theta) \tag{8}$$

The $\mathcal{L}_0$ is intractable during adaptation since we have no access to the original domain training data. Therefore, an approximation of the original domain is done using EWC which yields the optimisation problem:

$$\min_\theta \mathcal{L}_T(\theta) + \sum_j \lambda F_{0,j}(\theta_j - \theta_{0,j}^*)^2 \tag{9}$$

To satisfy assumption 1, whenever we retrain we always start from the original model M_0. Practically this is achievable as a copy of M_0 can always be kept onboard of a system. Assumption 2 can be satisfied by calculating the Fisher matrix using the original training dataset during initial training and a copy of this calculated Fisher matrix would be saved on board of the system, omitting the need to keep a copy of the initial training data on board.

In each training step t, the model parameters are updated according to Equation 10, where η is the learning rate.

$$\theta_{t+1} = \theta_t - \eta\left(\frac{\mathcal{L}_\mathrm{T}(\theta)}{d\theta} - \frac{\sum_j \lambda F_0(\theta_{t,j} - \theta_{0,j}^*)^2}{d\theta}\right) \tag{10}$$

The two hyperparameters critical for the success of our optimisation problem are η and λ. Different numerical search methods can be used for finding values for these hyperparameters, e.g. grid search, Bayesian optimisation etc. From

empirical testing, we found that a combination of $\eta = 1\text{e-}5$ and $\lambda = 1$ yields near optimum adaptation for datasets we used in our experimentation.

IV. Experimentation Setup

We used the problem of image classification to showcase our method. All of our experimentation was based in PyTorch library [45]. In the rest of this section, we discuss the details of our experimentation setup. Code is available at this repository: https://github.com/Abanoub-G/DIRA

A. Benchmarks

We utilise CIFAR-10C, CIFAR-100C, and ImageNet-C datasets in our experimentation. These are image classification benchmarks to evaluate a model's robustness against common corruptions [46]. The benchmarks add different corruptions to the tests sets of CIFAR-10/CIFAR-100[47] and ImageNet [48] There are 20 corruptions in total with five different levels of severity, however, most SOTA domain-incremental retraining frameworks utilise 15 corruptions out of the 20 in their comparisons, e.g. [29, 30]. These are deemed the more common corruptions. We use the same 15 common corruptions used by other methods in the literature.

B. Baselines

We list below the different baselines we assess against our DIRA approach:

1) **Source**: Refers to results of the corresponding baseline model trained on the incorrupt data (i.e. X_0), without adaption to the target domain.
2) **SGD**: Denotes retraining on samples of the corrupt data using only Stochastic Gradient Decent optimisation [39], i.e. without using any complimentary incremental learning frameworks, similar to how initial training on the incorrupt data is done.
3) **TTT** [30]: Test-Time Training (TTT) adapts parameters in the initial layers of the network by using auxiliary tasks to achieve self-supervised domain adaption.
4) **NORM** [32, 33]: Ignores the initial training statistics and recalculates the batch normalization statistics using samples from the target domain only (requiring a large number of samples).
5) **DUA** [29]: Dynamic Unsupervised Adaptation (DUA), takes into account initial training statistics and updates batch normalization statistics using samples from the target domain (requiring few samples).

C. Models and Hardware

We used ResNets [49] in our experiments, utilising two versions of ResNet: ResNet-18 (18-layer) and ResNet-26 (26-layer). For CIFAR-10/CIFAR-100, we used ResNet-26. Initial training for the models was done locally. For ImageNet, we used a pre-trained off-the-shelf ResNet-18 model from PyTorch [45]. Experiments for CIFAR10 and CIFAR100 were done on an MSI GF65 THIN 3060 Laptop with 64GB RAM and a Linux Ubuntu 20.04.2 LTS (64-bit) operating system, whilst for ImageNet we used a Dell Alienware Desktop PC with 64GB RAM and a Linux Ubuntu 18.04.4 LTS (64-bit) operating system.

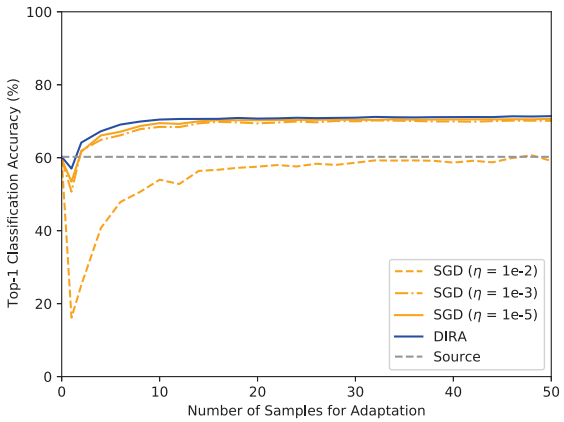

Fig. 1: ResNet-26 mean classification accuracy over 15 different corruption types on CIFAR-10C at the highest severity (Level 5).

To achieve reliable comparisons against baselines the starting model parameters from which retraining is done must be the same. Otherwise, the accuracy improvements cannot be reliably attributed to the effectiveness of the retraining method and can be argued that it is due to varying starting model accuracies or parameters. Therefore, in our results for CIFAR10/100 on ResNet-26 we only compare against Source, as we do not have the initial models used in retraining by other SOTA methods. For ImageNet on ResNet-18 we compare against SOTA methods because the starting trained model used by other SOTA retraining methods is the same off-the-shelf ResNet-18 model from PyTorch.

D. Optimisation Details

We used Stochastic Gradient Decent (SGD) for optimisation during training and retraining in our work. For retraining DIRA, we found from empirical testing that $\eta = 1\text{e-}5$ and $\lambda = 1$ yields near optimum adaptation. The retraining is relatively quick as only a small number of samples are used and the retraining is done over 10 epochs. We use top-1 classification accuracy as our assessment metric in all experiments [50].

V. Results & Discussion

A. Regularisation effect on adaptation

To investigate overall adaptation improvement using regularisation we plot Figure 1. Figure 1 shows how top-1 classification accuracy changes as the number of samples available from the target domain increases. The naive approach would be to retrain relying only on the Stochastic Gradient Decent (SGD) optimiser using a fixed learning rate (η). When

	gaus	shot	impul	defcs	gls	mtn	zm	snw	frst	fg	brt	cnt	els	px	jpg	mean
Source	58.5	61.3	37.3	51.9	59.6	58.6	58.1	73.3	67.8	50.0	80.7	19.2	71.8	66.1	79.8	59.6
DIRA	73.6	75.6	61.9	79.7	65.8	77.9	80.0	77.4	77.0	72.6	84.2	60.2	74.9	76.9	79.5	74.5

TABLE II: Top-1 Classification Accuracy (%) for each corruption in CIFAR-10C at the highest severity (Level 5). Source shows the results from the same model trained on the clean train set (CIFAR-10) and tested on the corrupted test set (CIFAR-10C). ResNet-26 is used with 100 retraining samples.

	gaus	shot	impul	defcs	gls	mtn	zm	snw	frst	fg	brt	cnt	els	px	jpg	mean
Source	24.2	27.0	9.7	30.0	30.9	33.6	35.5	38.8	34.6	19.6	44.8	8.4	43.4	39.8	50.0	31.4
DIRA	44.7	45.1	33.6	50.9	40.4	49.6	52.3	47.3	46.6	37.9	55.2	33.3	47.0	51.5	51.7	45.8

TABLE III: Top-1 Classification Accuracy (%) for each corruption in CIFAR-100C at the highest severity (Level 5). Source shows the results from the same model trained on the clean train set (CIFAR-100) and tested on the corrupted test set (CIFAR-100C). ResNet-26 is used with 100 retraining samples.

	gaus	shot	impul	defcs	gls	mtn	zm	snw	frst	fg	brt	cnt	els	px	jpg	mean
Source	1.6	2.3	1.6	9.4	6.6	10.2	18.2	10.5	15.0	13.7	48.9	2.8	14.7	23.1	28.3	13.8
TTT	3.1	4.5	3.5	10.1	6.8	13.5	18.5	17.1	17.9	20.0	47.0	**14.4**	20.9	22.8	25.3	16.4
NORM	**12.9**	10.4	9.5	**12.4**	10.6	**20.0**	28.1	**29.4**	18.5	33.1	52.2	10.2	26.5	35.8	31.5	22.7
DUA	10.6	12.4	**11.9**	12.0	11.4	15.3	25.7	22.2	21.6	31.4	54.4	4.1	27.8	33.5	**32.6**	21.8
DIRA	12.0	**13.5**	11.6	10.2	**11.5**	18.7	**31.2**	26.6	**27.2**	**36.3**	**56.3**	9.2	**35.7**	**38.1**	32.0	**24.7**

TABLE IV: Top-1 Classification Accuracy (%) for each corruption in ImageNet-C at the highest severity (Level 5). Source shows the results from the same model trained on the clean train set (ImageNet) and tested on the corrupted test set (ImageNet-C). For a fair comparison with TTT, NORM, and DUA, we use the same initially trained ResNet-18 model. 100 retraining samples are used. Highest accuracy is highlighted in bold.

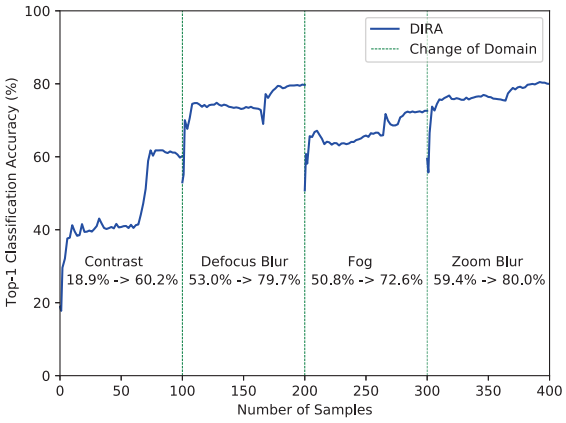

Fig. 2: Dynamic adaptation scenario example for DIRA to different domains from CIFAR-10C. Pre-trained ResNet-26 on CIFAR-10 adapts to different corruption examples from CIFAR-10C dataset at the highest severity (Level 5), to show how well DIRA can dynamically adapt to operational domains.

using a learning rate of 1e-2, which is a common value used when training a model on a dataset, we can notice that the retrained model incurs catastrophic forgetting and does not improve on the target domain beyond the Source model, even when retraining samples are increased. Lowering the learning rate overcomes the issue of forgetting and allows the model to adapt gradually to the target domain eventually as the number of samples increases. Using DIRA improves the issue of forgetting further for low number of samples (i.e. 1 to 2 samples) and allows the model to reach higher accuracies when retraining using less than 10 samples. Eventually, the performance of retraining using SGD (with low η) converges with DIRA as the number of retraining samples increase.

Tables II and III shows the improvement DIRA achieves upon retraining on 100 samples for each type of corruption in CIFAR-10C and CIFAR-100C benchmarking datasets, respectively, compared to the Source accuracy.

B. Dynamic adaptation scenario for DIRA

In real-life scenarios, varying domains may occur during operation. To visualise how DIRA tackles such a scenario, we plot in Figure 2 the shift to four different domains consecutively from CIFAR-10C. The results depicted show that as soon as samples from the target domain are presented an abrupt improvement occurs in the accuracy of the model. This accuracy continues to grow as more samples from the target domain become present.

C. Comparison with SOTA

To assess how well our approach performs compared with SOTA domain adaptation frameworks we compare results with three domain adaptation frameworks from the literature: TTT [30], NORM [32, 33], and DUA [29], on ImageNet-C benchmarking dataset. Table IV show top-1 classification accuracy for the highest severity level on dataset ImageNet-C. Our DIRA framework performs competitively with SOTA domain adaptation approaches. As can be seen from the table, we achieve SOTA overall performance averaged between the

different corruptions the dataset. This is while using a limited number of samples from the target domain (100 samples).

VI. Conclusions and Future Works

We have introduced a novel domain incremental learning framework, named DIRA (Dynamic Incremental Regularised Adaptation). DIRA allows for dynamic adaptation to changing operation environments using a limited number of samples. Our approach achieves this using the notion of transfer learning. Whereby relevant knowledge from the original domain is retained using regularisation techniques to allow the model to adapt to the target domain making use of transfer learning. Our DIRA approach proves to be competitive to available domain adaptation approaches in the literature, and achieves SOTA results compared to these approaches.

DIRA is currently categorised as a supervised retraining approach, as it relies on ground truth labels to be provided with samples from the target domain for adaptation. This is acceptable but may limit its applications where a source to provide ground truth labels is unavailable. Our future work is to explore the combination of DIRA with self-supervised approaches to remove the need for ground truth labels during adaptation.

References

[1] N. Kalra and S. M. Paddock. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* Santa Monica, CA: RAND Corporation, 2016.

[2] G. Chance et al. "An agency-directed approach to test generation for simulation-based autonomous vehicle verification". In: *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE. 2020, pp. 31–38.

[3] K. I. Eder, W.-l. Huang, and J. Peleska. "Complete Agent-driven Model-based System Testing for Autonomous Systems". In: *arXiv preprint arXiv:2110.12586* (2021).

[4] G. Chance et al. *Assessing Trustworthiness of Autonomous Systems*. 2023. arXiv: 2305.03411 [cs.AI].

[5] P. Koopman and M. Wagner. *Positive Trust Balance for Self-driving Car Deployment*. Vol. 2. Springer International Publishing, 2020, pp. 351–357.

[6] D. Hond, H. Asgari, and D. Jeffery. "Verifying Artificial Neural Network Classifier Performance Using Dataset Dissimilarity Measures". In: *Proceedings - 19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020* (2020), pp. 115–121.

[7] J. D. Schaffer and W. H. Land. "Predicting with Confidence: Classifiers that Know What They Don't Know". In: *Procedia Computer Science* 114 (2017), pp. 200–207.

[8] A. Mandelbaum and D. Weinshall. "Distance-based Confidence Score for Neural Network Classifiers". In: (2017). arXiv: 1709.09844.

[9] C. Xing et al. "Distance-Based Learning from Errors for Confidence Calibration". In: (2019), pp. 1–12. arXiv: 1912.01730.

[10] E. T. Barr et al. "The oracle problem in software testing: A survey". In: *IEEE Transactions on Software Engineering* 41.5 (2015), pp. 507–525.

[11] J. M. Zhang et al. "Machine Learning Testing: Survey, Landscapes and Horizons". In: *IEEE Transactions on Software Engineering* (2020), pp. 1–1. arXiv: 1906.10742.

[12] I. J. Goodfellow et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings* (2014). arXiv: 1312.6211.

[13] C. Harper et al. "Safety Validation of Autonomous Vehicles using Assertion-based Oracles". In: *arXiv preprint arXiv:2111.04611* (2021).

[14] A. Ghobrial et al. "A Trustworthiness Score to Evaluate DNN Predictions". In: *2023 IEEE International Conference On Artificial Intelligence Testing (AITest)*. 2023, pp. 9–16.

[15] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias. "Three types of incremental learning". In: *Nature Machine Intelligence* 4.12 (2022), pp. 1185–1197.

[16] S. Li et al. "Energy-based models for continual learning". In: *Conference on Lifelong Learning Agents*. PMLR. 2022, pp. 1–22.

[17] T. Lesort, M. Caccia, and I. Rish. "Understanding continual learning settings with data distribution drift analysis". In: *arXiv preprint arXiv:2104.01678* (2021).

[18] C. Zeno et al. "Task agnostic continual learning using online variational bayes". In: *arXiv preprint arXiv:1803.10123* (2018).

[19] A. Gepperth and B. Hammer. "Incremental learning algorithms and applications". In: *European symposium on artificial neural networks (ESANN)*. 2016.

[20] L. Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[21] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: (2017), pp. 1–6. arXiv: 1708.07747.

[22] R. Ramesh and P. Chaudhari. "Model Zoo: A Growing" Brain" That Learns Continually". In: *arXiv preprint arXiv:2106.03027* (2021).

[23] N. Y. Masse, G. D. Grant, and D. J. Freedman. "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization". In: *Proceedings of the National Academy of Sciences* 115.44 (2018), E10467–E10475.

[24] P. Ruvolo and E. Eaton. "ELLA: An efficient lifelong learning algorithm". In: *International conference on machine learning*. PMLR. 2013, pp. 507–515.

[25] M. Jehanzeb Mirza et al. “An Efficient Domain-Incremental Learning Approach to Drive in All Weather Conditions”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2022-June (2022), pp. 3000–3010. arXiv: 2204.08817.

[26] Z. Ke et al. “CLASSIC: Continual and contrastive learning of aspect sentiment classification tasks”. In: *arXiv preprint arXiv:2112.02714* (2021).

[27] X. Tao et al. “Few-Shot Class-Incremental Learning”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), pp. 12180–12189. arXiv: 2004.10956.

[28] S.-A. Rebuffi et al. “icarl: Incremental classifier and representation learning”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017, pp. 2001–2010.

[29] M. J. Mirza et al. “The norm must go on: dynamic unsupervised domain adaptation by normalization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14765–14775.

[30] Y. Sun et al. “Test-time training with self-supervision for generalization under distribution shifts”. In: *International conference on machine learning*. PMLR. 2020, pp. 9229–9248.

[31] Y. Sun et al. “Unsupervised domain adaptation through self-supervision”. In: *arXiv preprint arXiv:1909.11825* (2019).

[32] S. Schneider et al. “Improving robustness against common corruptions by covariate shift adaptation”. In: *Advances in neural information processing systems* 33 (2020), pp. 11539–11551.

[33] Z. Nado et al. “Evaluating prediction-time batch normalization for robustness under covariate shift”. In: *arXiv preprint arXiv:2006.10963* (2020).

[34] F. Maria Carlucci et al. “Autodial: Automatic domain alignment layers”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5067–5075.

[35] F. Zenke, B. Poole, and S. Ganguli. “Continual learning through synaptic intelligence”. In: *34th International Conference on Machine Learning, ICML 2017* 8 (2017), pp. 6072–6082. arXiv: 1703.04200.

[36] Z. Li and D. Hoiem. “Learning without Forgetting”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12 (2018), pp. 2935–2947. arXiv: 1606.09282.

[37] J. Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[38] R. Kemker et al. “Measuring catastrophic forgetting in neural networks”. In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 3390–3398. arXiv: 1708.02072.

[39] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[40] F. Huszár. “Note on the quadratic penalties in elastic weight consolidation”. In: *Proceedings of the National Academy of Sciences of the United States of America* 115.11 (2018), E2496–E2497. arXiv: arXiv:1712.03847v1.

[41] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002.

[42] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. 1993, p. 180.

[43] A. Ly et al. “A Tutorial on Fisher information”. In: *Journal of Mathematical Psychology* 80 (2017), pp. 40–55. arXiv: 1705.01064.

[44] J. Martens. “New insights and perspectives on the natural gradient method”. In: *Journal of Machine Learning Research* 21 (2020), pp. 1–76. arXiv: 1412.1193.

[45] A. Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[46] D. Hendrycks and T. Dietterich. “Benchmarking neural network robustness to common corruptions and perturbations”. In: *ICLR* (2019).

[47] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images”. In: (2009).

[48] J. Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[49] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[50] A. Ghobrial et al. “Evaluation Metrics for DNNs Compression”. In: *arXiv preprint arXiv:2305.10616* (2023).