Adaptive Online Domain Incremental Continual Learning

Nuwan Gunasekara[1], Heitor Gomes[1], Albert Bifet[1], and Bernhard Pfahringer[1]

AI Institute, University of Waikato, Hamilton, New Zealand

Abstract. Continual Learning (CL) problems pose significant challenges for Neural Network (NN)s. Online Domain Incremental Continual Learning (ODI-CL) refers to situations where the data distribution may change from one task to another. These changes can severely affect the learned model, focusing too much on previous data and failing to properly learn and represent new concepts. Conversely, if a model constantly forgets previously learned knowledge, it may be deemed too unstable and unsuitable. This work proposes Online Domain Incremental Pool (ODIP), a novel method to cope with catastrophic forgetting. ODIP also employs automatic concept drift detection and does not require task ids during training. ODIP maintains a pool of learners, freezing and storing the best one after training on each task. An additional Task Predictor (TP) is trained to select the most appropriate NN from the frozen pool for prediction. We compare ODIP against regularization methods and observe that it yields competitive predictive performance.

Keywords: Continual Learning · Online Domain Incremental Continual Learning

1 Introduction

Though modern Neural Network (NN)s have shown great success in image classification and natural language processing, they assume training data to be Independent and Identically Distributed (IID). Due to this assumption, once confronted with a distribution shift in the input data, the model may undergo costly retraining to preserve old knowledge while adjusting to the new distribution. Without retraining, an NN receiving non-IID data forgets its past knowledge when confronted with a distribution shift. This phenomenon is identified as "catastrophic forgetting" in literature [20,6,9,17].

Continual Learning (CL) currently attempts to minimize this catastrophic forgetting in NNs via replay and regularization methods [17]. Though current replay methods outperform regularization methods in terms of performance, they may not be suitable for situations with memory and privacy constraints on the replay buffer[17,2]. Even though offline CL methods have been proposed, current research mainly focuses on online methods to solve catastrophic forgetting. This allows one to develop continually learning agents which are adaptive. But resilient to catastrophic forgetting.

Online Domain Incremental Continual Learning (ODI-CL) focuses on CL models, which learn from one input distribution to another with minimum catastrophic forgetting. Here the class distribution remains the same. There are many practical applications of this scenario in the modern IoT world. For example, one could use an ODI-CL approach to avoid costly retraining of an X-ray image classification model after a distribution shift in the incoming data due to some hardware changes in the X-ray machine[22]. The same scenario could be valid for many NN models that rely on hardware sensor inputs. Also, on certain ODI-CL settings, replay approaches may be less preferred due to constraints on having a replay buffer.

Considering the practical importance of non-replay ODI-CL, this work proposes an ODI-CL method that alleviates catastrophic forgetting in NNs. But superior to regularization methods. Here a tiny pool of small Convolutional Neural Network (CNN)s are trained online. Once confronted with concept drift, it freezes the best CNN for a given concept, considering the estimated loss of all CNNs. Task Predictor (TP) is trained to pick the best CNN from the frozen pool for prediction. This approach is further extended to automatically detect concept drifts in incoming data using a Task Detection (TD) instead of relying on an external task id signal. Experiment results reveal that both the proposed methods: with and without automatic TD, surpass the performance of current popular regularization methods.

The main contributions of this paper are the following:

1. Online Domain Incremental Pool (ODIP): we introduce a novel method to alleviate catastrophic forgetting for Online Domain Incremental Continual Learning without using instance replay. Here, a small pool of tiny CNNs is trained, and the best one is frozen at the end of each task. Task Predictor is trained to predict the best frozen CNN for evaluation for a given instance. The experiment results reveal that ODIP yields superior accuracy than regularization baselines. Furthermore, an in-depth investigation is done to better understand the effectiveness of different TPs on three ODI-CL datasets.
2. Instead of relying on an external task id signal during prediction, ODIP uses an automatic Task Detection mechanism to detect tasks in the incoming data. This allows ODIP to select the most appropriate frozen network to produce predictions for each instance. ADaptive sliding WINdow (ADWIN) is used to detect drifts in CNN's loss to determine a new task. To the best of our knowledge, this automatic Task Detection for Online Domain Incremental Continual Learning has not been proposed before.

The rest of the paper is organized as follows. The following section presents the current developments in Online Domain Incremental Continual Learning, including some practical use cases. The next section presents the proposed Online Domain Incremental Pool for ODI-CL. The experiments section explains the experimental setup where the proposed method is compared against popular ODI-CL methods on three datasets. It also provides insights into the effectiveness of different Task Predictors. The final section provides conclusions and directions for future research.

2 Related work

The literature has thoroughly documented that an NN receiving non-IID data forgets past knowledge when confronted with a concept shift [20,6,9,17]. Continual Learning (CL) attempts to continually learn with minimal forgetting of past concepts [9,17]. In Online Domain Incremental Continual Learning (ODI-CL), this learning happens online, and the data stream comprises different concepts (distributions) with the same label distribution [17].

To avoid catastrophic forgetting in NNs, CL algorithms use two popular approaches: regularization and replay. Regularization algorithms like Elastic Weight Consolidation (EWC) [9] and Learning without Forgetting (LwF) [13] adjust the weights of the network in such a way that it minimizes the overwriting of the weights for the old concept. Elastic Weight Consolidation (EWC) uses a quadratic penalty to regularize updating the network parameters related to the past concept. It uses the diagonal of the Fisher Information Matrix to approximate the importance of the parameters [9]. EWC has some shortcomings: 1) Fisher Information Matrix needs to be stored for each task, 2) requires an extra pass over each task's data at the end of the training [17]. Though different versions of EWC address these concerns [21,4,14], [4] seems to be the one suitable for online CL by keeping a single Fisher Information Matrix calculated by a moving average. Learning without Forgetting (LwF) uses knowledge distillation to preserve knowledge from past tasks. Here, the model related to the old task is kept separate, and a separate model is trained on the current task. When the LwF receives data for a new task (X_n, Y_n), it computes the output (Y_o) from the old model for new data X_n. During training, assuming that $\hat{Y_o}$ and $\hat{Y_n}$ are predicted values for X_n from the old model and new model, LwF attempts to minimize the loss: $\alpha L_{KD}(Y_o, \hat{Y_o}) + L_{CE}(Yn, \hat{Y_n}) + R$. Here L_{KD} is the distillation loss for the old model, and α is the hyper-parameter controlling the strength of old model against the new one. L_{CE} is the cross-entropy loss for the new task. R is the general regularization term. Due to this strong relation between old and new tasks, it may perform poorly in situations where there is a huge difference between old and new tasks distributions [17].

Replay methods present a mix of instances from the old and current concepts to the Neural Network (NN) based on a given policy while training. This reduces the forgetting as the training instances from the old concepts avoid complete overwriting of past concepts' weights. GDumb [19], Experience Replay (ER) [5], and Maximally Interfered Retrieval (MIR) [1] are some of the most popular CL replay methods. Replay Using Memory Indexing (REMIND) [7] takes this approach to another level by storing the internal representations of the instances by the initial frozen part of the network and using a randomly selected set of these internal representations to train the last unfrozen layers of the network. Here, REMIND can store more instances' representations using internal low-dimensional features. In general, these replay approaches are motivated by how the hippocampus in the brain stores and replays high-level representations of the memories to the neocortex to learn from them [12].

Recent research has focused on using ODI-CL methods to avoid costly retraining in practical situations where the model is confronted with concept drift. ODI-CL has been used in X-ray image classification to avoid costly retraining on distribution shifts due to unforeseen shifts in hardware's physical properties [22]. Also, it has been used to mitigate bias in facial expression and action unit recognition across different demographic groups [8]. Furthermore, ODI-CL was used to counter retraining on concept drifts for multi-variate sequential data of critical care patient recordings [2]. The authors highlight some replay methods' infeasibility due to strong privacy requirements in clinical settings. This concern is further highlighted in [17] empirical study as well.

Most of the current ODI-CL methods rely on an explicit end-of-task signal during training. EWC and LwF use this signal to optimize weights, while replay methods like ER use it to update their replay buffer. However, GDumb does not rely on this signal for replay buffer updates. Though [17] identifies ODI-CL as training without the end of the task signal. Practical implementations such as [8] and [2] use the end of the task signal to employ CL methods such as EWC and LwF. However, on the other hand, practical implementation in [22] assumes a gradual distribution shift in the input data distribution where instances from both the new and old tasks could appear in the stream for a certain period.

Our approach (ODIP) initially assumes the presence of an end-of-task signal at training and later proposes a method to detect it automatically. When confronted with concept drift, the proposed method freezes the best NN from a small pool of little networks, and a predictor is trained to choose the best network from the frozen pool for a given evaluation instance. As it avoids using a replay buffer, it is a good candidate for settings with higher privacy requirements.

3 Online Domain Incremental Pool

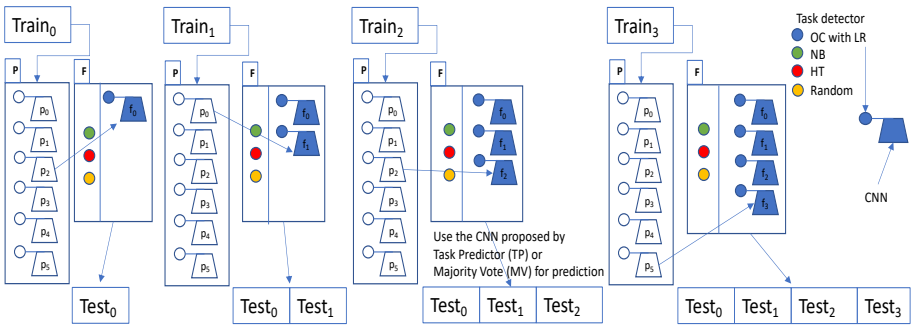


Fig. 1: Proposed Online Domain Incremental Pool (ODIP)

The ODI-CL is defined as the training set composed of multiple concepts of non-IID data, where each concept has a different input distribution with the same label distribution [17]. The goal of the learning algorithm is to minimize

Algorithm 1 TRAIN OC WITH LR

Input: Task Predictor TP: One Class Classifier with Logistic Regression , z: extracted features

1: $score$, $in\_class \leftarrow$ TRAIN $OC(z)$
2: TRAIN $LR(score, in\_class)$

catastrophic forgetting of the past concepts while performing well on the current concept [17,7]. Initially, at training, we assume that the task id that signals the end of a concept is available to the learning model. However, this information is not available to the model during evaluation. Later, the proposed method(ODIP) is extended to discard this external task id signal.

Algorithm 2 ODIP TRAINING ALGORITHM

Input: P: pool of training CNNs, F: pool of frozen CNNs, T: task set, X_t: training set for task t, TP: Task Predictor

Initialize pool $F = \{\}$
for all task $t \in T$ **do**
  for all mini-batch b_t in training set X_t for task t **do**
    $z \leftarrow$ features from mini-batch b_t for task t
    for all learner $p \in P$ **do**
      Compute loss L_p of mini-batch b_t and train CNN_p
      Update ADWIN_p with L_p
      if task predictor TP_p is One Class Classifier with LR **then**
        TRAIN OC WITH LR(TP_p, z)
      end if
    end for
    if task predictor TP is Naive Bayes or Hoeffding Tree **then**
      TRAIN $TP(z, t)$
    end if
  end for
  Append the CNN with lowest loss estimated using ADWIN to F
end for

We propose an Online Domain Incremental Pool (ODIP), where P pool of tiny CNNs are trained for each concept t with a given Task Predictor. The Task Predictors could be None, Naive Bayes (NB), Hoeffding Tree (HT), and One Class Classifier (OC) with Logistic Regression (LR). The Task Predictor is trained for mini-batch b_t using extracted features from a static feature extractor. At the end of each task's training, CNN with the lowest estimated loss is frozen

and added into the frozen pool F. In the special case of OC with LR, the relevant OC with the LR is also part of the frozen CNN. Algorithm 2, along with figure 1, further explains this training approach.

Algorithm 3 PREDICT OC WITH LR

Input: Task Predictor TP: One Class Classifier with Logistic Regression, z: extracted features.

1: $score$, $in\_class \leftarrow$ PREDICT $OC(z)$

Output: PREDICT $LR(score)$

In ODIP, there are two vote aggregation methods for prediction: Weighted Voting (WV) or votes from the best CNN (CNN_{best}). For Weighted Voting, the probabilities of the Task Predictor are used as weights. In the CNN_{best} case, it is either selected randomly from the F pool or the one predicted by Task Predictor. Algorithm 4 further explains this. Recently proposed ODI-CL algorithms rely

Algorithm 4 ODIP PREDICTION ALGORITHM

Input: x_t: instance of task t, F: pool of frozen CNNs, TP: Task Predictor, useWeightedVoting

1: $z \leftarrow$ features from instance x_t of task t
2: **if** useWeightedVoting **then**
3: **if** TP is Majority Vote **then**
4: $votes \leftarrow 1/|F| \sum_{f=1}^{|F|} \text{PREDICT}(f, x_t)$
5: **else if** TP is One Class Classifier with LR **then**
6: $votes \leftarrow 1/|F| \sum_{f=1}^{|F|} \text{PREDICT OC WITH LR}(TP_f, z) \times \text{PREDICT}(f, x_t)$
7: **else**
8: $votes \leftarrow 1/|F| \sum_{f=1}^{|F|} \text{PREDICT}(TP, z) \times \text{PREDICT}(f, x_t)$
9: **end if**
10: **else**
11: **if** TP is Random **then**
12: Select $\text{CNN}_{selected}$ randomly from pool F
13: **else if** TP One Class Classifier with LR **then**
14: $\text{CNN}_{selected} \leftarrow \arg\max_{f \in F} \text{PREDICT OC WITH LR}(TP_f, z)$
15: **else**
16: $\text{CNN}_{selected} \leftarrow \arg\max_{f \in F} \text{PREDICT}(TP_f, z)$
17: **end if**
18: $votes \leftarrow$ PREDICT($\text{CNN}_{selected}$, x_t)
19: **end if**

Output: $votes$

on an explicit end of the task signal (task id) to identify the start of a new task. ODIP is also relying on these explicit task ids to distinguish different tasks. This reliance on an explicit task id may preclude one from employing current

ODI-CL algorithms in real-life settings where it may be challenging to identify such a signal explicitly.

Algorithm 5 ODIP TRAINING ALGORITHM WITH AUTO TASK DETECTION

Input: P: pool of training CNNs, F: pool of frozen CNNs, T: task set, X_t: training set for task t, TP: Task Predictor

Initialize pool $F = \{\}$
Initialize $taskId = 0$
for all task $t \in T$ **do**
  for all mini-batch b_t in training set X_t for task t **do**
    $taskEnd \leftarrow false$
    $z \leftarrow$ features from mini-batch b_t for task t
    for all learner $p \in P$ **do**
      Compute the loss L_p of mini-batch b_t and train CNN_p
      Update ADWIN_p with L_p
      if ADWIN_p detects change **then**
        $taskEnd \leftarrow true$
      end if
      if task predictor TP_p is One Class Classifier with LR **then**
        TRAIN OC WITH LR(TP_p, z)
      end if
    end for
    if task predictor TP is Naive Bayes or Hoeffding Tree **then**
      TRAIN $TP(z, taskId)$
    end if
    if $taskEnd$ **then**
      $taskId \leftarrow taskId + 1$
      Append the CNN with lowest loss estimated using ADWIN to F
    end if
  end for
end for

ODIP is extended to identify concept drifts in the incoming stream automatically. ADaptive sliding WINdow (ADWIN) [3] is used as a task detector. ADWIN has nice properties where it uses exponential histograms for memory efficiency and discards the buffer related to the previous concept once confronted with a drift. Every CNN in P pool has its ADWIN. They are updated with each CNN's loss after training. Once updated, a new task is identified if any ADWIN detects a drift in the loss. Here a drift in the loss is assumed to be related to the drift in the input stream. Algorithm 5 explains this training with automatic Task Detection in detail. In the experiments, the effectiveness of ODIP was compared against popular regularization baselines.

Table 1: Datasets

Dataset	Number of tasks	Number of Classes	Channels, H, W
CORe50	11	10	3, 32, 32
RotatedCIFAR10	4	10	3, 32, 32
RotatedMNIST	4	10	1, 28, 28

4 Experiments

The experiments attempt to understand the effectiveness of ODIP against popular regularization baselines. They also attempt to identify the effectiveness of Task Predictors. Lastly, they attempt to determine the effectiveness of ODIP with automatic Task Detection against regularization baselines.

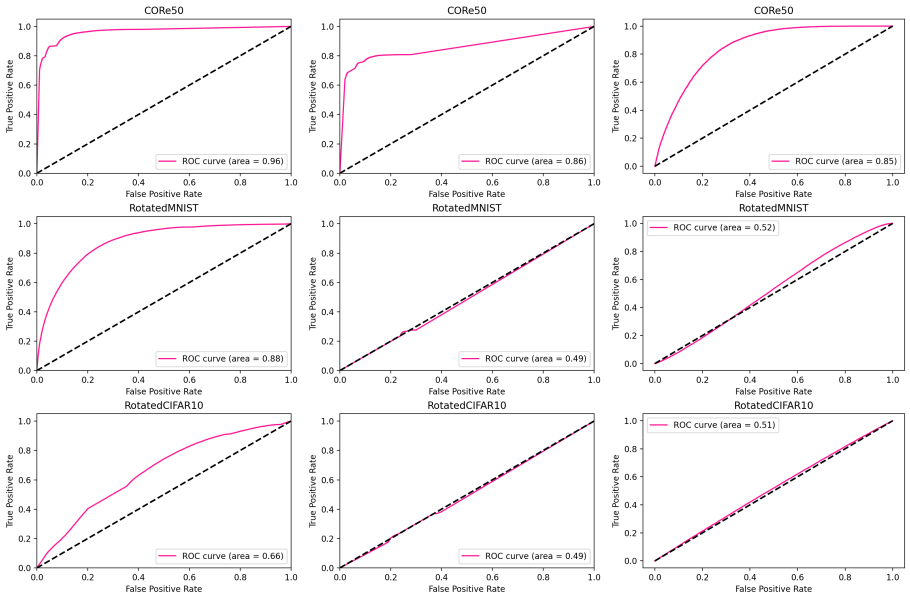
Different versions of ODIP were compared against regularization baselines: LwF and EWC. The replay methods were not considered in the baselines as the setting avoids using a replay buffer. The baselines use CNNs with 4.3 times the parameters (144234) than in ODIP experiments (33450). For ODIP, ResNet-18 was used as the static feature extractor and flattened last layer features were used to train the TPs. Five types of TPs were used in the experiments: random, Majority Vote (MV), NB, HT[1], and OC[2] with LR. Also, two types of vote aggregation methods were considered in the experiments: WV and the use of votes from CNN_{best}. Furthermore, two variants of automatic Task Predictor were considered in the experiments: α) include the best training CNN for prediction when the frozen pool is empty, β) include the best training CNN for prediction when the frozen pool is empty OR when the predicted network is related to the current concept. P pool size for ODIP was set to 6 CNNs. In the experiments, we also considered a hypothetical scenario of ODIP, where the task id is available at evaluation, and it is used to determine the correct frozen CNN. This is presented as the "Tid known" in the results. This allows one to determine the hypothetical upper bound of ODIP.

Three datasets were used in the experiments: CORe50 [15], RotatedCIFAR10, and RotatedMNIST. With RotatedCIFAR10 and RotatedMNIST, 90°rotations (0°, 90°, 180°, -90°) of the original images from CIFAR10 [10] and MNIST [11] were considered separate tasks. Altogether there were four tasks in those two datasets. With CORe50, 11 distinct sessions (8 indoor and 3 outdoor) of the same object were considered separate tasks: tasks 0-2,4-8 indoor, tasks 3,9, and 10 outdoor. Here 10 object categories were considered as the class labels. In the above datasets, all classes were presented in all the tasks. Such rearranging was done to the original datasets to adhere to the ODI-CL definition described in [17].

All experiments were run using Avalanche [16] Continual Learning platform. Average accuracy and forgetting defined in [17] are used in the evaluation. All

[1] Use skmultiflow[18] online versions of NB and HT

[2] Online One-Class SVM. ODIP source code available at github.



(a) Naive Bayes (b) Hoeffding Tree (c) OC with LR

Fig. 2: Effectiveness of Task Predictors. ROC curves for predicted task id and AUC scores for all the Task Predictors

Table 2: Average accuracy after training on the last task

dataset	Baselines		ODIP								
	EWC	LwF	**Tid known**	Random	MV	HT_{WV}	OC_{WV}	NB_{WV}	NB_{NoWV}	$NB_{TD\alpha}$	$NB_{TD\beta}$
CORe50	0.41	0.41	0.69	0.42	0.53	0.63	0.56	**0.66**	0.61	0.44	0.47
RotatedCIFAR10	0.44	**0.48**	0.48	0.38	0.45	0.44	0.46	0.43	0.40	0.40	0.42
RotatedMNIST	0.51	0.72	0.97	0.48	0.66	0.53	0.65	**0.79**	0.78	**0.79**	**0.79**
Avg	0.45	0.54	0.72	0.42	0.55	0.53	0.56	**0.63**	0.60	0.54	0.56

experiments were run three times, and relevant averages and standard deviations were considered in the evaluation. The standard deviations were omitted from this manuscript due to space constraints.

Table 2 contains the average accuracy of each method after training on the last task. As one can see from the table, ODIP NB_{WV} produces the best results. ODIP Random and EWC seem to yield poor results. In general, all methods with Weighted Voting produced good results compared to the two baselines. However, weights from a good Task Predictor seem to boost the performance significantly. Also, ODIP $NB_{TD\beta}$, which has automatic Task Detection, yields better results than regularization baselines. It is also on-par or better than the other ODIP methods, which use task ids, except for NB. Considering the hypothetical "Tid known" scenario, it is evident that just selecting the correct

Table 3: Average forgetting after training on the last task

dataset	Baselines		ODIP								
	EWC	LwF	**Tid known**	Random	MV	HT_{WV}	OC_{WV}	NB_{WV}	NB_{NoWV}	$NB_{TD\alpha}$	$NB_{TD\beta}$
CORe50	0.10	0.07	0.00	0.01	-0.02	0.02	**-0.03**	0.01	0.00	0.00	0.20
RotatedCIFAR10	0.10	0.00	0.00	0.01	**-0.03**	0.05	-0.03	-0.01	0.00	-0.03	-0.01
RotatedMNIST	0.63	0.24	0.00	0.20	0.16	0.59	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**
Avg	0.28	0.11	0.00	0.07	0.04	0.22	**0.02**	0.04	0.04	0.03	0.10

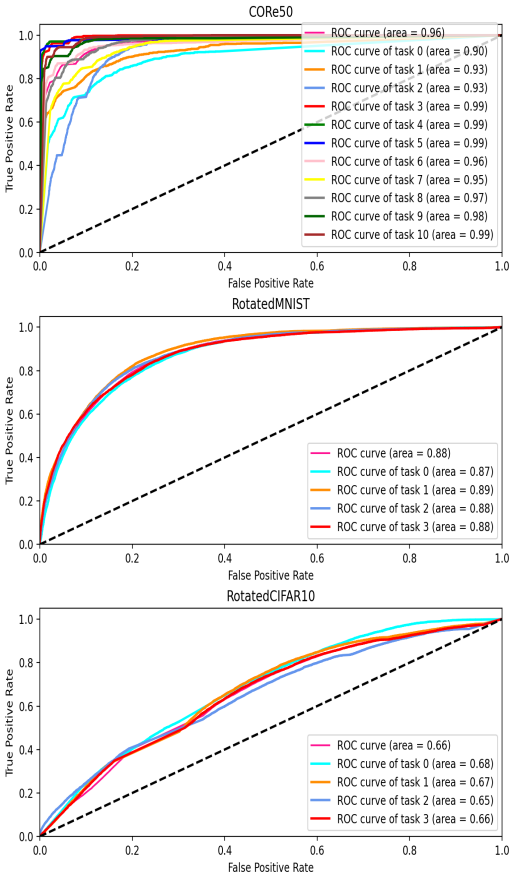


Fig. 3: Effectiveness of Naive Bayes as a TP. ROC curves and AUC scores for predicted task id for each task.

frozen CNN is sufficient to outperform current baselines by a considerable margin. This is further evident in table 3, with "Tid known" having a zero average forgetting across all datasets after training on the last task. Note here that a smaller average forgetting is better.

To further understand the effectiveness of the Task Predictors, the predicted task id was compared against the true task id in non-auto-TD mode against all datasets. This comparison was made for all evaluation instances after training on the last task. Figure 2 shows the ROC curves for predicted task id and the relevant AUC scores for each TP on each dataset. According to the figure, it is clear that NB is a better Task Predictor for all datasets. This further strengthens the overall strong NB results in table 2. Figure 3 further explains the effectiveness of NB as a Task Predictor when predicting each task in a given dataset. From the per-task ROC curves and AUC scores, it is clear that NB performs similarly on all the tasks for a given dataset. Nevertheless, it does perform slightly better on certain tasks. This is evident in CORe50, with NB performing slightly better for tasks 3,4,5,9, and 10. This suggests, in general, that NB is a good Task Predictor.

5 Conclusion

The proposed ODIP produces competitive results for ODI-CL in comparison to regularization-based approaches. The extended version can detect tasks in ODI-CL automatically. ODIP with and without automatic Task Detection produces competitive results compared to current popular regularization baselines: LwF and EWC. This makes ODIP as a good replacement for regularization methods in the ODI-CL setting. It could be further improved to have a fixed frozen pool size. Then information from the Task Predictor could be used to identify the CNN to train and then replace when the frozen pool is full.

References

1. Aljundi, R., Caccia, L., Belilovsky, E., Caccia, M., Lin, M., Charlin, L., Tuytelaars, T.: Online continual learning with maximally interfered retrieval. arXiv preprint arXiv:1908.04742 (2019)
2. Armstrong, J., Clifton, D.: Continual learning of longitudinal health records. arXiv preprint arXiv:2112.11944 (2021)
3. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. pp. 443–448. SIAM (2007)
4. Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H.: Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 532–547 (2018)
5. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486 (2019)
6. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
7. Hayes, T.L., Kafle, K., Shrestha, R., Acharya, M., Kanan, C.: Remind your neural network to prevent catastrophic forgetting. In: European Conference on Computer Vision. pp. 466–483. Springer (2020)

8. Kara, O., Churamani, N., Gunes, H.: Towards fair affective robotics: Continual learning for mitigating bias in facial expression and action unit recognition. arXiv preprint arXiv:2103.09233 (2021)
9. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
12. Lewis, P.A., Durrant, S.J.: Overlapping memory replay during sleep builds cognitive schemata. Trends in cognitive sciences **15**(8), 343–351 (2011)
13. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
14. Liu, X., Masana, M., Herranz, L., Van de Weijer, J., Lopez, A.M., Bagdanov, A.D.: Rotate your networks: Better weight consolidation and less catastrophic forgetting. In: International Conference on Pattern Recognition (ICPR). IEEE (2018)
15. Lomonaco, V., Maltoni, D.: Core50: a new dataset and benchmark for continuous object recognition. In: Conference on Robot Learning. pp. 17–26. PMLR (2017)
16. Lomonaco, V., Pellegrini, L., Cossu, A., Carta, A., Graffieti, G., Hayes, T.L., Lange, M.D., Masana, M., Pomponi, J., van de Ven, G., Mundt, M., She, Q., Cooper, K., Forest, J., Belouadah, E., Calderara, S., Parisi, G.I., Cuzzolin, F., Tolias, A., Scardapane, S., Antiga, L., Amhad, S., Popescu, A., Kanan, C., van de Weijer, J., Tuytelaars, T., Bacciu, D., Maltoni, D.: Avalanche: an end-to-end library for continual learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2nd Continual Learning in Computer Vision Workshop (2021)
17. Mai, Z., Li, R., Jeong, J., Quispe, D., Kim, H., Sanner, S.: Online continual learning in image classification: An empirical survey. Neurocomputing **469**, 28–51 (2022). https://doi.org/https://doi.org/10.1016/j.neucom.2021.10.021, https://www.sciencedirect.com/science/article/pii/S0925231221014995
18. Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-multiflow: A multi-output streaming framework. Journal of Machine Learning Research **19**(72), 1–5 (2018), http://jmlr.org/papers/v19/18-251.html
19. Prabhu, A., Torr, P.H., Dokania, P.K.: Gdumb: A simple approach that questions our progress in continual learning. In: European conference on computer vision. pp. 524–540. Springer (2020)
20. Ratcliff, R.: Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. Psychological review **97**(2), 285 (1990)
21. Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y.W., Pascanu, R., Hadsell, R.: Progress & compress: A scalable framework for continual learning. In: International Conference on Machine Learning. pp. 4528–4537. PMLR (2018)
22. Srivastava, S., Yaqub, M., Nandakumar, K., Ge, Z., Mahapatra, D.: Continual domain incremental learning for chest x-ray classification in low-resource clinical settings. In: Albarqouni, S., Cardoso, M.J., Dou, Q., Kamnitsas, K., Khanal, B., Rekik, I., Rieke, N., Sheet, D., Tsaftaris, S., Xu, D., Xu, Z. (eds.) Domain Adaptation and Representation Transfer, and Affordable Healthcare and AI for Resource Diverse Global Health. pp. 226–238. Springer International Publishing, Cham (2021)