ONLINE CONTINUAL LEARNING USING ENHANCED RANDOM VECTOR FUNCTIONAL LINK NETWORKS

Cheryl Sze Yin Wong, Guo Yang, Arulmurugan Ambikapathi, Ramasamy Savitha

Institute for Infocomm Research (I2R), A-STAR
1 Fusionopolis Way, Singapore 138632

ABSTRACT

We propose an online continual learning algorithm based on an enhanced Random Vector Functional Link Network (OCL-eRVFL), that learns a sequence of tasks continually, where each task is defined by streaming data with each sample arriving once and only once. As data for a new task in domain incremental or class incremental setting streams in, the output weights of an eRVFL is updated through Recursive least squares, such that the representations for the past tasks are not catastrophically forgotten. As the recursive least square update is based only on the currently streaming sample, samples are not stored. Hence, unlike state-of-the-art OCL that avoid catastrophic forgetting through memory replay of samples from past task, the proposed OCL-eRVFL needs no extra memory. The proposed OCL-eRVFL is evaluated on streaming split CIFAR10, split CIFAR100 and split CIFAR10/100 image classification data sets within a class and domain incremental setting. Performance results show that the proposed OCL-eRVFL efficiently learns a sequence of tasks with streaming data, without additional memory expense.

Index Terms— random vector functional link networks, recursive least squares, online continual learning

1. INTRODUCTION

Deep neural networks have shown tremendous progress in classification, but they do not generalize beyond their training data distribution. However, data in the real-world environment is evolving, and requires models that adapt their representations with changing data distributions. Recently, there is an increased interest in developing continual learning (CL) algorithms, where a model learns a sequence of tasks that are defined by increments in domain and/or class [1]. More recently, online continual learning (OCL) algorithms, which learn a sequence of tasks with streaming data for each task available once and only once, are being developed. [2, 3].

On the other hand, traditional deep learning approaches require training over several epochs to learn and converge to a model for accurate inference. Therefore, SOTA methods for OCL are replay-based, requiring samples to be stored for repeated training [2, 3]. However, the number of samples stored is limited by the available memory. Hence, there is a need for OCL approaches without additional memory.

In this work, we propose an OCL algorithm based on an enhanced Random Vector Functional Link networks (OCL-eRVFL). The universal approximation [4] and the convergence of RVFL has been theoretically proven [5]. As a single hidden layer RVFL cannot model complex relationships in the data, we enhance the RVFL with a pre-trained layer to facilitate extraction of high level abstraction of the original inputs. In the first task, the weights in RVFL mapping the relationship between the extracted features and the class labels are computed using pseudo-inverse. As the samples of subsequent tasks stream in, the weights are updated using the theoretically substantiated Recursive Least Squares (RLS) [6] in an online fashion, without having to save samples of any task. The proposed OCL-eRVFL is evaluated on the standard multi task settings that include split CIFAR10, split CIFAR100 and split CIFAR10/100 dataset, under the premise of OCL. The results show that OCL-eRVFL performs better than the state-of-the-art (SOTA) algorithms in OCL. Moreover, unlike other OCL methods that require saving subset of samples for replay during training successive tasks, the proposed OCL-eRVFL does not need *any* samples to be stored from previous or current tasks (thanks to online learning), and therefore require *no additional memory*.

The main novelties of OCL-eRVFL are: (1) RLS for class-incremental learning, (2) RVFL for OCL and (3) OCL without memory replay, yet reducing catastrophic forgetting.

2. RELATED WORKS

CL using deep neural networks are usually based on regularisation, architectural and rehearsal (replay) methods [7]. Regularization strategies for CL [8, 9] preserve representations of learnt task, by penalizing strong adaptations of the weights for subsequent new tasks. Architectural strategies for CL [10, 11] modify the structure of neural networks by pruning redundant weights and adding neurons towards efficient network utilization and efficient representation of all tasks.

This work was supported by the Institute for Infocomm Research and Industry Alignment Fund Prepositioning (Health & Biomedical Sciences) H19/01/a0/023, A*STAR, Singapore.

Rehearsal strategies [12, 13, 14, 15, 2] are aimed at replaying images or representations of the past tasks to remember them, while adapting the new tasks.

On the other hand, online learning algorithms [16, 17] are able to learn by *seeing a sample only once*. In OS-ELM [17], the weights in the networks can be updated through derived mathematical calculations without the need for previous data samples. Furthermore, incremental learning through structural adaptions to network were explored in [16].

All the aforementioned algorithms are aimed at learning a sequence of tasks or streaming data. However, a sequence of task, where data for each task is streaming, is a common data behaviour in real-world environments, requiring OCL algorithms. As the existing OCL algorithms are memory intensive, we propose a new strategy for continual learning to learn a sequence of tasks, where samples within each task are streaming one by one and only used once.

3. PROBLEM SETUP: ONLINE CONTINUAL LEARNING

This paper considers the standard **disjoint task formulation, class-incremental and online continual learning (OCL)** scenario [2]. A class-incremental disjoint task (CI) formulation describes a CL scenario where every new task has novel and unique classes. The OCL scenario entails learning a sequence of CI tasks, with streaming data within tasks.

For a sequence of l tasks $(T_1, \dots T_l)$, the data for each task is denoted by $\mathcal{D}_t;\ t = 1, \dots, l$. Each sample occurs in input-output pairs $(\mathbf{X}_t^j, y_t^j); j = 1, \dots, N_t$, where N_t is the number of samples in each task. We consider the input $\mathbf{X}_t^j$ as an image and the corresponding class label $\mathbf{y}_t^j \in R^{n_c}$ as one-hot encoded unit vector, where n_c is the total number of classes for all tasks encountered till the current task. We denote the predicted label vector for image $\mathbf{X}_t^j$ as $\widehat{\mathbf{y}}_t^j \in R^{n_c}$.

Let the training and testing data of task t be $\mathcal{D}_t$ is divided into $\mathcal{D}_t^{train}$ and $\mathcal{D}_t^{test}$, respectively. The data for each task $\mathcal{D}_t^{train}$ arrives sequentially. During the task t, the data from the previous task $\mathcal{D}_{t-1}^{train}$ are unavailable. Furthermore, under the online setup, each sample $(\mathbf{X}_t^j, y_t^j)$ for task t arrives one-by-one, and is available only once for training. Let f be a prediction model parameterized by $\boldsymbol{\theta}_t$ at task t. Then, the model predicts

$$\widehat{\mathbf{y}}_t = f(\mathbf{x}_t, \boldsymbol{\theta}_t) \tag{1}$$

The objective of the learning model is to learn $\boldsymbol{\theta}_t$ such that it can make accurate inference for data from any task $T_1, \dots, T_t$, without the need for task labels.

4. OCL-RVFL

The proposed algorithm OCL-eRVFL is illustrated in Fig. 1. The architecture of the network is divided into two parts: (1) A pre-trained network with weights that remains unchanged throughout all tasks and (2) An enhanced random vector functional link network with output weights $W^{m[t]}$ that gets updated online with streaming data samples.

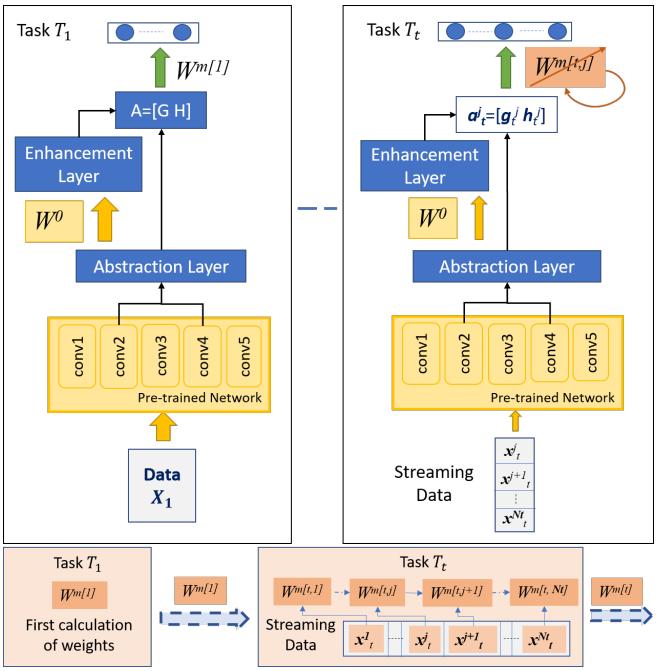

Fig. 1. Online Continual Learning using Enhanced Random Vector Functional Link Network. Yellow boxes indicates network weights that are frozen. Thin black lines represents concatenation of matrices. Green arrow indicates the output weights that are being updated as data streams in. Note: The pre-trained network is not fixed and can be chosen according to data.

4.1. Pre-trained Network

A pre-trained network is used to extract generalised useful features from the input samples, irrespective of the dataset and the task labels. This pre-trained network is a deep neural network that has been trained on a preferably rich heterogeneous dataset or with the samples from the first task. In our experiments, we use a conventional ResNet18 that has been pre-trained on the standard ImageNet dataset. In particular, the responses of the 2nd and 4th convolutional layers (more deeper layers will have lesser information due to max pooling operation) constitute the abstraction layer of the enhanced Random Vector functional link (RVFL) network.

4.2. An enhanced RVFL network

A traditional RVFL network has a single hidden layer, in which the weights and biases of the hidden neurons ($\mathbf{W}^0$) are randomly generated and the output weights ($\mathbf{W}^m$) are computed through a closed form solution.

Dataset	Images per class	Tasks	Class per task
split CIFAR-10	5000	5	2
split CIFAR-100	500	20	5
split CIFAR 10/100	5000 / 500	6	[10, 20 × 5]

Table 1. Dataset description

The enhanced RVFL (eRVFL) comprises of an abstraction layer, an enhancement layer and the output layer. In OCL-eRVFL, the concatenated feature layer consists of the features $\mathbf{g}_t^j \in R^{n_f}$, where n_f features are extracted from the pre-trained network (abstraction layer) and non-linearly transformed features from the enhancement layer $\mathbf{h}_t^j \in R^{n_k}$, where n_k is the dimension of enhancement layer. The weights $\mathbf{W}^0$ between $\mathbf{g}_t^j$ and $\mathbf{h}_t^j$ are randomly generated and are kept unchanged for the complete sequence of tasks. Hence, only the weights between the concatenated features and the output layer $\mathbf{W}^{m[t]}$ is computed for task T_t. For the first task, these weights $\mathbf{W}^{m[1]}$ are computed using pseudo-inverse, as follows. Let $\mathbf{G} \in R^{N_1 \times n_f}$ and $\mathbf{H} \in R^{N_1 \times n_k}$ be the row-wise concatenated matrices of the represented features and the enhancement layers over all the training samples in task T_1, respectively. Then,

$$\mathbf{M}_1 = [\mathbf{A}^T \cdot \mathbf{A}]^{-1} \tag{2}$$

where $\mathbf{A} = [\mathbf{G} \quad \mathbf{H}] \in R^{N_1 \times (n_f + n_k)}$ is the concatenation of features in the abstraction and the enhancement layers for Task 1. The corresponding weights for this task 1($\mathbf{W}^{m[1]}$) can be computed as

$$\mathbf{W}^{m[1]} = \mathbf{M}_1 \cdot \mathbf{A}^T \cdot \mathbf{Y} \tag{3}$$

where $\mathbf{M}_1 \cdot \mathbf{A}^T$ is the Moore-Penrose pseudo inverse of $\mathbf{A}$, and $\mathbf{Y} \in R^{N_1 \times n_c}$ is the row-wise concatenated matrix containing the one-hot encoded labels of all the training samples.

4.3. Online update using Recursive Least Squares (RLS)

As the samples for the subsequent new tasks stream in, recursive least squares is used to update the weights $\mathbf{W}^{m[t]}$, $t = 2, \ldots, l$ based on each sample j in task t, as shown below:

$$\mathbf{M}_t^j = \mathbf{M}_t^{j-1} - \frac{\mathbf{M}_t^{j-1} \cdot \mathbf{a}_t^j \cdot \mathbf{a}_t^{j^T} \cdot \mathbf{M}_t^{j-1}}{1 + \mathbf{a}_t^{j^T} \cdot \mathbf{M}_t^{j-1} \cdot \mathbf{a}_t^j}, \ \forall j, \ \forall t \tag{4}$$

$$\mathbf{W}^{m[t,j]} = \mathbf{W}^{m[t,j-1]} + \mathbf{M}_t^j \cdot \mathbf{a}_t^j \cdot (y_t^j - \mathbf{a}_t^{j^T} \cdot \mathbf{W}^{m[t,j-1]}) \tag{5}$$

where $\mathbf{a}_t^j = [\mathbf{g}_t^j; \mathbf{h}_t^j] \in R^{(n_f + n_k)}$ is the input layer representation of sample j in task t. $\mathbf{W}^{m[t,j]}$ is the learnt output weights from OCL-eRVFL, for sample j from task t.

Algorithm 1: OCL-eRVFL

Data: $\mathcal{D}_t$; $t = 1, \ldots, l$
Result: $\mathbf{W}^{m[t]}$, $\hat{\mathbf{y}}_\mathbf{s}, \forall s = 1, \ldots, t$
Feature Extractor
1. Choose a pre-trained network for feature extraction.
2. Randomly generate weights W^0 between feature and enhancement layers.
for $t = 1, \ldots, l$ **do**
  Training Phase
  Input : $\mathbf{x}_t^j, y_t^j \in \mathcal{D}_t^{train}$
  if *t=1* **then**
    Derive $\mathbf{A}$ feature representations using the feature extractor.
    Calculate $\mathbf{M}_1$ using Eq 2.
    Calculate $\mathbf{W}^{m[1]}$ using Eq 3.
  else
    for $j = 1, \ldots, N_t^{train}$ **do**
      Derive $\mathbf{a_t^j}$ using the feature extractor.
      Update $\mathbf{M}_t^j$ using Eq 4.
      Update $\mathbf{W}^{m[t,j]}$ using Eq 5.
    end
  end
  Testing Phase
  for $s = 1, \ldots, t$ **do**
    Input : $\mathbf{x}_s^j \in \mathcal{D}_s^{test}$
    Output: $\hat{\mathbf{y}}_\mathbf{s}$
    Derive $\mathbf{A}$ feature representations using the feature extractor.
    Make predictions $\hat{\mathbf{y}}_\mathbf{s}$ using OCL-eRVFL with $\mathbf{W}^{m[t]}$.
  end
end

5. EXPERIMENTS

We evaluate the proposed OCL-eRVFL on the CIFAR10 and CIFAR100 datasets in OCL settings under three different scenarios - (a) the split CIFAR10, (b) the split CIFAR100, and the (c) spilt CIFAR10/100. More details on the datasets would be given in Section 5.1. The pre-trained network that is used in our experiments is the ResNet18 (pre-trained on ImageNet) available in the PyTorch library. The number of enhancement nodes n_k is increased in the experiments until the accuracy plateaus. Accordingly, $n_k = 50, 500, 20$ for scenarios (a), (b), and (c), respectively.

The OCL-eRVFL is evaluated using the average accuracy on the fixed test set for all tasks ($T_1, \ldots T_t$) after the training for the last task T_t. The OCL-eRVFL is compared against SOTA OCL algorithms for split CIFAR10 [2] and for split CIFAR100 [3]. For split CIFAR10/100, there is no published works on experiments conducted in OCL scenario, hence we compare it to offline continual learning methods [18, 19].

5.1. Dataset

We provide the details of the dataset used in the OCL settings in Table 1. The images in all these datasets are of dimension $32 \times 32 \times 3$. In the sequence of split CIFAR 10/100 dataset, there is a total of 6 tasks. The first task consists of the 10 classes of the CIFAR-10 dataset. The subsequent 5 tasks consist of 20 classes each from the CIFAR-100 dataset.

5.2. Performance Study: Split CIFAR10/Split CIFAR100:

Algorithm	Num of replay samples	Accuracy
GEM [12]	1000	17.5 ± 1.6
iCARL [13]	1000	32.4 ± 2.1
ER [15]	1000	41.3 ± 1.9
ER-MIR [14]	1000	47.6 ± 1.1
ER5 [14]	1000	42.4 ± 1.1
ER-MIR5 [14]	1000	49.3 ± 0.1
GDumb [2]	1000	61.3 ± 1.7
OCL-eRVFL (Ours)	-	$\mathbf{73.2 \pm 0.1}$

Table 2. Split CIFAR-10: Performance results of OCL-eRVFL in comparison with SOTA OCL methods. Results for SOTA algorithms are extracted from [2].

Table 2 presents OCL-eRVFL against the state of the art (SOTA) algorithms in OCL settings for split CIFAR10. For fair comparison, the architecture of the SOTA algorithms is also ResNet18. As typical deep neural networks require training in multiple epochs to learn, and hence the respective continual learning algorithms compared here require replay mechanisms. Replay samples in the OCL methods are allowed to be fed to the neural network multiple times as they are kept in the memory. With the use of RVFL with recursive least squares as an online updating mechanism, our OCL-eRVFL outperforms the best of the SOTA algorithms, by more than 11%.

Algorithm	Num of replay samples	Accuracy
Offline [3]	50000	49.7 ± 2.6
ER [15]	10000	18.4 ± 1.4
GSS [20]	10000	19.3 ± 0.7
iCaRL[13]	5000	19.2 ± 1.1
CN-DPM [21]	1000	14.0 ± 1.7
GDumb [2]	10000	28.8 ± 0.9
OCL-eRVFL (Ours)	-	39.1 ± 1.1

Table 3. Split CIFAR-100: Performance comparison of OCL-eRVFL with SOTA OCL methods. Results for SOTA algorithms are extracted from [3].

Table 3 presents performance of OCL-eRVFL against the SOTA methods, in OCL settings for split CIFAR100. The architecture of the SOTA algorithms is the reduced ResNet18 [3]. OCL-eRVFL is able to achieve an accuracy of 39.1%, which is higher than the SOTA algorithm by over 10%. It should be noted that the offline training of $\approx$ 49.7% can be considered as the upper bound accuracy.

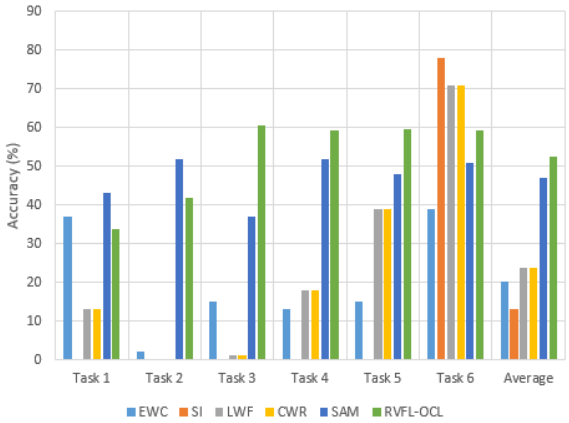

Fig. 2. Split CIFAR-10/100: Performance results of OCL-eRVFL against SOTA methods.

5.3. CIFAR-10/100: Comparison with SOTA methods

Fig. 2 presents the performance results of OCL-eRVFL with SOTA methods in the split CIFAR10/100 scenario. It is observed that the OCL-eRVFL is at least 4% better than its closest contender, the Self Attention Meta-learner (SAM) [18]. The samples in task 1 (CIFAR-10) is downsampled to 500 samples per class (a total of 5000 samples) to balance the data across all tasks OCL-eRVFL. The results show the effectiveness of OCL-eRVFL even when compared to algorithms without the OCL criteria.

6. CONCLUSIONS

This paper proposes a novel OCL algorithm OCL-eRVFL based on combining a pre-trained network with RVFL network. For the first time in the literature, the proposed OCL-eRVFLis the OCL algorithm that does not need to store additional samples. The OCL-eRVFL updates its weights through RLS, as data for tasks stream in. OCL-eRVFL is evaluated on split CIFAR10, split CIFAR100 and split CIFAR10/100 datasets. Results show that the algorithm is better than SOTA algorithms in online continual learning scenarios. Some of the related research directions that are currently under investigation include, theoretical analysis on the online updating mechanism RLS and finding possibilities for extending the capacity of the current network to better represent the new data when they stream in.

7. REFERENCES

[1] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[2] Ameya Prabhu, Philip Torr, and Puneet Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *The European Conference on Computer Vision (ECCV)*, August 2020.

[3] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner, "Online continual learning in image classification: An empirical survey," *CoRR*, vol. abs/2101.10423, 2021.

[4] Le Zhang and P.N. Suganthan, "A comprehensive evaluation of random vector functional link networks," *Information Sciences*, vol. 367-368, pp. 1094–1105, 2016.

[5] Deanna Needell, Aaron A. Nelson, Rayan Saab, and Palina Salanevich, "Random vector functional link networks for function approximation on manifolds," 2020.

[6] E.K.P. Chong and S.H. Zak, *An Introduction to Optimization*, Wiley Series in Discrete Mathematics and Optimization. Wiley, 2013.

[7] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.

[8] Zhizhong Li and Derek Hoiem, "Learning without forgetting," 2017.

[9] 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," 2017.

[10] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell, "Progressive neural networks," 2016.

[11] Ching-Yi Hung, Cheng-Hao Tu, Cheng-En Wu, Chien-Hung Chen, Yi-Ming Chan, and Chu-Song Chen, "Compacting, picking and growing for unforgetting continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 13647–13657.

[12] David Lopez-Paz and Marc'Aurelio Ranzato, "Gradient episodic memory for continual learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2017, NIPS'17, p. 6470–6479, Curran Associates Inc.

[13] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5533–5542.

[14] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[15] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. 2019, vol. 32, Curran Associates, Inc.

[16] Ramasamy Savitha, ArulMurugan Ambikapathi, and Kanagasabai Rajaraman, "Online rbm: Growing restricted boltzmann machine on the fly for unsupervised representation," *Applied Soft Computing*, vol. 92, pp. 106278, 2020.

[17] Guang-Bin Huang, Nan-Ying Liang, Hai-Jun Rong, Paramasivan Saratchandran, and Narasimhan Sundararajan, "On-line sequential extreme learning machine," in *IASTED International Conference on Computational Intelligence, Calgary, Alberta, Canada, July 4-6, 2005*, M. H. Hamza, Ed. 2005, pp. 232–237, IASTED/ACTA Press.

[18] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy, "Self-attention meta-learner for continual learning," *CoRR*, vol. abs/2101.12136, 2021.

[19] Davide Maltoni and Vincenzo Lomonaco, "Continuous learning in single-incremental-task scenarios," *Neural Networks*, vol. 116, pp. 56–73, 2019.

[20] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio, "Gradient based sample selection for online continual learning," in *NeurIPS*, 2019.

[21] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim, "A neural dirichlet process mixture model for task-free continual learning," in *International Conference on Learning Representations*, 2020.