Entropy-based Stability-Plasticity for Lifelong Learning

Vladimir Araujo[1,2]*, Julio Hurtado[2,3]*, Alvaro Soto[2], Marie-Francine Moens[1]
[1]KU Leuven, [2]Pontificia Universidad Católica de Chile, [3]University of Pisa
vgaraujo@uc.cl, jahurtado@uc.cl, asoto@ing.puc.cl, sien.moens@kuleuven.be

Abstract

The ability to continuously learn remains elusive for deep learning models. Unlike humans, models cannot accumulate knowledge in their weights when learning new tasks, mainly due to an excess of plasticity and the low incentive to reuse weights when training a new task. To address the stability-plasticity dilemma in neural networks, we propose a novel method called Entropy-based Stability-Plasticity (ESP). Our approach can decide dynamically how much each model layer should be modified via a plasticity factor. We incorporate branch layers and an entropy-based criterion into the model to find such factor. Our experiments in the domains of natural language and vision show the effectiveness of our approach in leveraging prior knowledge by reducing interference. Also, in some cases, it is possible to freeze layers during training leading to speed up in training.

1. Introduction

Humans learn continuously throughout their lives, integrating new information to their knowledge to face new and changing environments. By contrast, artificial neural networks learn in a bounded environment, where the input distribution is assumed fixed. When the input distribution changes, the model must adapt its weights to perform correctly on the new task. Due to those modifications, the model overwrites previously learned patterns, creating interference between old and new tasks, causing a problem known as catastrophic forgetting [29, 34]. This excessive plasticity in the model is part of the stability-plasticity dilemma [31, 44], which addresses the trade-off between modifying the parameters to learn a new task (plasticity) or keeping the parameters constant (stability) to avoid interference between tasks.

Several methods have been proposed to mitigate the stability-plasticity dilemma, focusing mainly on avoiding the catastrophic forgetting problem. Using different techniques to mitigate interference, these methods can be divided into two groups. The first group aims to restrict weight modifications by using regularization functions [1, 19, 49] that minimize the modifications of key weight values. The second group uses gating functions [15, 27, 39] to adaptively activate each weight depending on the context provided by the current task or input instance.

In this work, we follow the first group by proposing a model that aims to restrict weight modifications. We rely on evidence showing that the lower layers of a deep learning model capture general knowledge while the upper layers capture task-specific knowledge [20, 22, 47, 48]. Under this premise, in the case of a lifelong learning scenario, a model should update its layer weights based on how general or specific these layers should be. We propose the Entropy-based Stability-Plasticity (ESP) method, which relies on an entropy-based criterion to decide how much a model has to modify the weights in each of its layers. Specifically, ESP augments each layer of an encoder with a branch layer that computes an entropy-based plasticity factor during the forward pass, and dynamically updates the layer weights based on these plasticity factors during the backward pass. This way, when a new training example arrives, the model calculates how much we can update the weights of the model via a plasticity factor. We found in our experiments that in some cases, our method forces the model to freeze some layers, setting their gradients to zero, which encourages reusing past knowledge and reduces training time.

We demonstrate the effectiveness of our method experimentally by running a diverse set of experiments and comparing our results against well-known baselines. Unlike previous work in the field, we evaluated ESP on both, vision and natural language domains. The code is publicly available for further replicability and future research[1].

2. Related Work

Previous methods have tackle the problem of Continual Learning (CL) using three main strategies. The first group of methods focus on limiting the plasticity of learning new

*Equal contribution.

[1]https://github.com/vgaraujov/ESP-CL

tasks. The typical approach penalizes weight modifications or freezes a subset of the model. This can be achieved by adding weight regularizations [19, 49], using masks to freeze parts of the model [15, 27, 28], or based those regularization on gradients behavior [4, 37].

The second strategy is to use dynamic architectures by increase network capacity and adding extra parameters [10, 36], or by finding new paths of relevant weights to solve each task [12], freezing used weights, and limiting learning of new tasks. Other approaches use different functions as components in the network, either Hypernetworks [45], Deep Artificial Neurons (DANs) [3], Compositional Structures [30, 32], or novel learning strategies [16], so that network components can be more flexible when learning new tasks.

The third strategy is based on memory-based methods. This strategy mitigate catastrophic forgetting by inserting data from past tasks into the training process of new tasks, continuously re-training previous tasks [13], either with raw samples [5, 6, 35], or minimizing gradient interference [4, 25]. Later works such as [23, 40] train generator functions (GANs) or autoencoders [18] to generate elements from past distributions. They seek memory-efficiency by generating examples instead of saving real data. Similarly, [2, 17] seek to be memory-efficient by saving feature vectors of instances from previous tasks, while learning a transformation from the feature space of past tasks to current ones. Other works use memory to create prototypes that can represent classes [7, 35], either for use as distillation or classification vectors.

3. Method

In this work, we consider a lifelong (continual) learning setup. Each task t consists of a new data distribution $D^t = (X^t, Y^t)$, where X^t denotes the input instances and Y^t denotes the instance labels. The goal is to train a classification model $f : X \longrightarrow Y$ using data from a sequence of T tasks: $D = \{D^1, ..., D^T\}$. Following the Class Incremental Learning setup for CL [42], each task is presented sequentially to the model without a task descriptor. Also, in our setup, we only allow each item to be viewed once, such as in online learning scenarios [8, 26].

A model in this configuration consists of an encoder and a decoder. The encoder takes an input x and produces a vector representation. The encoder could be any kind of model, for instance, a Transformer [43] for text classification, or a ResNet [14] for image classification. The decoder is a linear transformation and a softmax layer to predict the class y of an input x. Note that because there is no task descriptor, the decoder predicts across all classes.

Next, we explain the proposed method and how it is incorporated into the learning process.

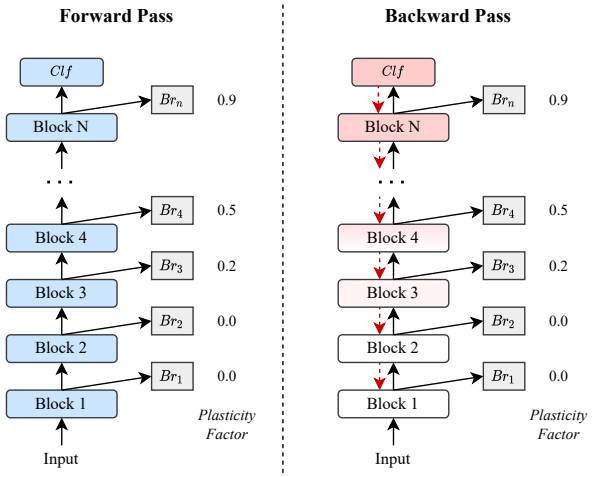


Figure 1. Overview of the method. During the forward step, the backbone processes an example and generates prediction and plasticity factor values for each block (left). During the backward pass, the plasticity factor is used to adjust the final amount of modification each layer will have (right).

3.1. Entropy-based Stability-Plasticity (ESP)

Previous works attempt to solve the stability-plasticity dilemma by slowing down learning on certain weights based on how important they are to previously seen tasks [19, 49]. Although effective, these methods neglect that some layers learn general or task-specific patterns [20, 22, 47, 48] and constantly update the weights of the different layers, resulting in interference with the acquired knowledge. Based on this, ESP addresses the stability-plasticity dilemma using a mechanism that allows the model to decide how much each layer should be updated using an entropy-based criterion. This method favors the reuse of previous knowledge existing in the layers by means of little or no modification of their weights and the specialization of the layers in a specific task by means of a high modification of their weights.

Under the CL setup mentioned above, an encoder could be seen as a stack of processing blocks. As shown in Fig. 1, ESP extends each model block i with a side branch layer Br_i to generate a classification of the input x:

$$\hat{y}_i = W_i^2(\sigma(W_i^1 f_i(x))) \tag{1}$$

where $f_i(x)$ is the output of the block i, W_i^1 and W_i^2 are trainable linear layers, and σ is an activation function.

Later, the vector $\hat{y}_i$ is used to compute the entropy of the prediction probability distribution for each block:

$$E(\hat{y}_i) = \sum \hat{y}_i \log \hat{y}_i \tag{2}$$

Finally, a plasticity factor (PF) is calculated as the complementary entropy value. Note that a $softmax$ function is

applied first to find the proportion of entropy corresponding to each block i.

$$PF = (1 - softmax(E(\hat{y}))) \tag{3}$$

This whole process is slightly similar to previous work using branch classifiers and entropy for early exiting [11,41, 46,51]. However, in this case, the calculated factors provide the proportion by which the weights of each model layer should be modified. Intuitively, a high value of PF (low entropy) leads to a high modification of the weights, specializing them for the task. On the other hand, a low value of PF (high entropy) leads to smaller changes in weights, reducing interference and catastrophic forgetting.

3.2. Training

Analogous to EWC and SI, which have an additional step to find the importance of each weight after each new task, ESP needs to update each block's branch layers Br before training a new task. To do that, we first freeze the encoder and decoder and train only the attached layers with a subset of the training set (e.g., replay set). We train the layers as a classification problem, where the output of each layer is compared with ground truth using a *Cross Entropy* loss.

$$Loss_i = CrossEntropy(\hat{y_i}, y_i) \tag{4}$$

Then, to train the backbone (encoder and decoder), the branch layers must be frozen. The reason for freezing the branch layers is to maintain the optimal quality of the decoder classifier. If the branch layers are not frozen, the model layers will no longer be optimized solely for the decoder classifier, which generally worsens its quality. Empirically, we found that joint optimization of branch layers and the backbone also leads to worst results.

This training step uses the information provided by the frozen branch layers to self-regulate the weight update. During the forward pass (Fig. 1 left), the model generates an output y and each Br generates the corresponding PF. During the backward pass (Fig. 1 right), the PF scales the gradient of the corresponding block to control the modifications that the task wants to make to the model. Note the encoder and decoder are optimized with the same loss function of Eq. (4). The training of the model with ESP is summarized in Algorithm 1.

Algorithm 1: ESP Training Process

Components:

- D^t: Dataset for task t.
- F: Model.
- BC: Branch Layers.

$BC \leftarrow TrainBC(F, D^t)$
for *x,y* **to** D^t **do**
  # Forward pass
  $PF, \hat{y} \leftarrow F(x)$
  $\nabla g \leftarrow Loss(y, \hat{y})$
  # Backward pass
  $\nabla g_{new} \leftarrow UpdateGrad(PF, \nabla g)$
  $F \leftarrow UpdateModel(F, \nabla g_{new})$
end

4. Experiments

In this paper, we test our approach in the domains of natural language and vision. For a fair and consistent comparison, we use the same CL setup (explained in Sec. 3) and the same baselines for both domains.

4.1. Baselines

One of the most reliable approaches to overcoming catastrophic forgetting is **Replay** [13]. It involves storing a subset of previous inputs (e.g., sentences) and mixing them with more recent inputs to update the model. The replay subset is usually a percentage of data randomly taken from the training set of previous tasks. As our baseline, we use a standard replay strategy with commonly used percentages.

To compare our method, we consider several well-known methods that attempt to address the stability-plasticity dilemma to apply to our primary baseline. We provide a brief description of each below:

1. **Stability**: A method that keeps the encoder weights fixed and trains only the decoder (classifier).
2. **Plasticity**: A method with complete freedom to train the encoder and decoder (classifier).
3. **Linear Plasticity**: A method that, similarly to ESP, uses a factor to scale the gradient of each block. The factors are linearly spaced between 0 and 1 with respect to the model's number blocks, where 0 is for the first block and 1 for the last one.
4. **O-EWC**: Online EWC [38] introduces a regularization term involving the Fisher information matrix that indicates the importance of each of the parameters to previous tasks.
5. **SI**: Synaptic Intelligence [49] adjusts the plasticity of the model by regularizing the modification of these important weights with a coefficient.

We also consider the class imbalance issue in our experiments. We train all methods in two scenarios: (1) using ONLY the replay set items, similar to [33] but without using a fixed amount of data, and (2) combining ALL data from the current task with the replay set. In the latter case, there

	Replay				
	10%	**20%**	**30%**	**40%**	**50%**
Stability	67.8	68.8	69.1	69.4	69.7
Plasticity	76.6	76.9	76.9	76.8	76.9
Linear Plasticity	76.8	77.1	76.9	77.0	77.1
O-EWC	76.8	76.8	76.9	76.9	77.1
SI	76.7	76.6	76.9	76.9	77.2
ESP	**76.9**	**77.3**	**77.1**	**77.2**	**77.5**

Table 1. Text classification results using replay set concatenated with ALL the training set.

	Replay				
	10%	**20%**	**30%**	**40%**	**50%**
Stability	63.3	66.8	68.0	68.6	68.9
Plasticity	74.9	75.6	76.3	76.6	76.8
Linear Plasticity	74.7	75.8	76.2	76.6	76.9
O-EWC	74.3	75.7	76.2	76.5	76.3
SI	74.6	76.0	76.2	76.6	76.9
ESP	**75.0**	**76.1**	**76.6**	**76.9**	**77.3**

Table 2. Text classification results using ONLY replay set.

	Replay				
	1%	**2%**	**3%**	**4%**	**5%**
Stability	15.9	17.5	18.0	20.3	19.9
Plasticity	20.1	29.1	36.6	40.1	42.3
Linear Plasticity	20.3	26.1	28.9	34.6	39.7
O-EWC	16.5	19.7	25.4	29.3	34.6
SI	**23.0**	29.9	38.8	**40.4**	**43.8**
ESP	21.8	**31.8**	**39.1**	38.6	40.6

Table 3. Image classification results using replay set concatenated with ALL the training set.

	Replay				
	1%	**2%**	**3%**	**4%**	**5%**
Stability	15.8	22.9	26.3	28.3	29.0
Plasticity	28.0	37.3	42.6	46.6	**50.3**
Linear Plasticity	24.6	32.6	39.6	43.6	45.5
O-EWC	**29.2**	37.8	40.1	45.7	47.5
SI	28.9	**38.5**	**44.0**	**47.1**	49.6
ESP	26.0	36.1	41.2	46.2	48.5

Table 4. Image classification results using ONLY replay set.

could be a significant class imbalance, but a more considerable amount of data would be available to train the model.

4.2. Natural Language

4.2.1 Implementation Details

We use BERT [9], a Transformer-based [43] pre-trained language model, as the encoder. As decoder, following original BERT model, we use the first token (special token [CLS]) of the sequence and a classifier to predict the class. In addition, we use the default BERT vocabulary in our experiments. We use Adam optimizer with a learning rate of $3e^{-5}$ and a training batch of size 32.

4.2.2 Datasets

We use publicly available text classification datasets from [50]: (1) AGNews classification, (2) Yelp sentiment analysis, (3) Amazon sentiment analysis, (4) DBPedia article classification and (5) Yahoo questions and answers categorization. We follow the same data processing described in [8]. In total, we have 575,000 training examples and 38,000 test examples with 33 classes from all datasets. In addition, we use the originally proposed dataset orders:

(i) Yelp → AGNews → DBPedia → Amazon → Yahoo
(ii) DBPedia → Yahoo → AGNews → Amazon → Yelp
(iii) Yelp → Yahoo → Amazon → DBpedia → AGNews
(iv) AGNews → Yelp → Amazon → Yahoo → DBpedia

4.2.3 Results

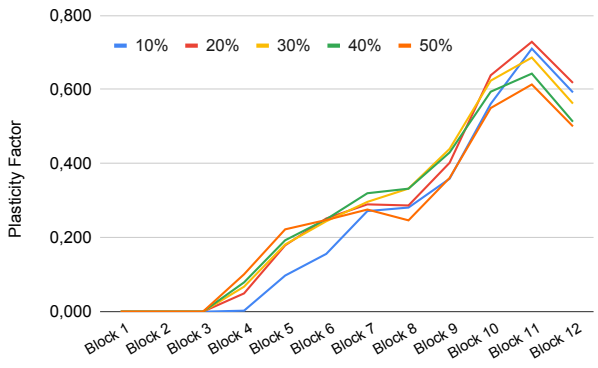
Our results in the natural language experiments are shown in Tab. 1 and Tab. 2. The Plasticity model performs well compared to the Stability version. This was expected because the Stability model limits the flexibility of the model to acquire new knowledge. Interestingly, the Linear Plasticity model outperforms the Plasticity model in almost all experiments, supporting the hypothesis that lower blocks need minor updates, and modifying higher blocks leads to better results. On the other hand, we find that O-EWC and SI perform similarly or even worse than the Plasticity model in some cases. This is because these methods perform better under a setup in which a task id is provided.

In contrast, ESP outperforms all baselines when trained on all experiments. We found that the increase in performance is consistent in the ONLY and the ALL scenarios. Overall, ESP achieves an accuracy gain of 0.34 and 0.44 points on average (across all replay percentages) over SI and O-EWC, respectively. ESP also exceeds Linear Plasticity, which means that dynamic plasticity factors are useful to avoid forgetting.

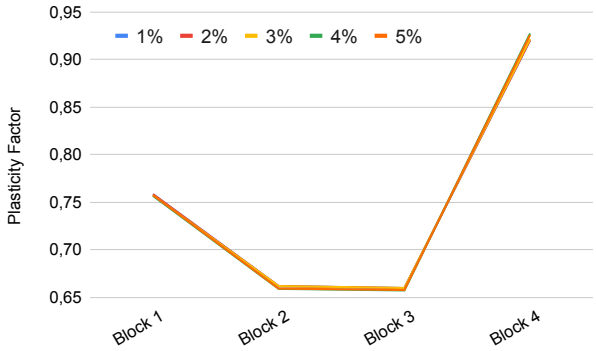
4.3. Vision

4.3.1 Implementation Details

For the visual experiments, as encoder we use a pre-trained ResNet-18 [14], and following previous works, a linear classifier for the decoder. Instead of using a branch clas-



(a) Natural language results with BERT base model.



(b) Vision results with ResNet-18 model.

Figure 2. Average plasticity factor per block across all tasks. % represents the percentage amount of replay.

sifier for each layer, here we use a branch classifier for each block of ResNet. The output of each block is reduced to one element per channel by averaging the values of the activation maps, this vector goes through the branch functions to find the scores.

We use SGD as our optimizer, using a learning rate of $1e^{-3}$ and momentum factor equal to 0.9. We run all of our experiments using a batch size of 32. For EWC and SI, we try different values of regularization coefficient, at the end we use 2000 for EWC and 0.1 for SI.

4.3.2 Datasets

Following previous works [49], we use CIFAR10 [21] equally divided into 5 tasks. We use the implementation from Avalanche [24] to generate the different sequence. We run each experiments 3 times with different seeds and we average the results.

4.3.3 Results

Similar to the natural language experiments, we use a pre-trained model, specifically a ResNet pre-trained on ImageNet. However, a big difference between text and images is the number of elements saved in the replay set. As images weigh more than phrases, we do experiments saving between 1 and 5% of previous tasks.

Results shown in Tab. 3 and Tab. 4, shows that regularization methods have better results than Stability, Plasticity and Linear Plasticity. On the other hand, modifying the weights without a constraint does not achieve good results either. In general, there is no clear advantage between the regularization methods in either of the two scenarios. We hypothesize the results diverge from the natural language experiments for two reasons: The first is the difference in the number of blocks, indicating that ESP may take advantage of deeper networks. The second is because there is a difference in the input distribution between ImageNet and CIFAR10, we expand this hypothesis a bit more in Sec. 4.4.

The motivation behind regularization methods is to reduce the plasticity of the model to prevent forgetting. By minimizing the modification of relevant weights in the future training process, these methods force future tasks to reuse knowledge even if those patterns are irrelevant or hurtful to new tasks. For this reason, we believe it is essential for these methods to learn representations that may be useful across tasks. For example, if the model weights are too specific for a task and we freeze all layers, the model would not find a correct classification.

Given the above, we believe that one reason why regularization methods do not perform well in Class Incremental scenarios is the inability to find good representations and thus the need to start from a pre-trained model. To prove this hypothesis and compare our results, we change the pre-trained ResNet-18 to one initialize randomly. The results show that none of the three regularizing methods has good results, being outperformed by the Plasticity method in almost all replay percentages. The advantage of this method is that it has complete flexibility to adjust the weights. This advantage leads to the weights to learn new representations, not tied to representations particular to the previous tasks. It is of little use to reduce the modification of past relevant weights if they can not be reused for future tasks.

4.4. Further Analysis

This section discusses the plasticity factors resulting from our experiments in both domains. Fig. 2 shows the average of the plasticity factors for our experiments on natural language (Fig. 2a) and vision (Fig. 2b).

We use a pre-trained BERT, a 12 block model for natural language. Interestingly, the plasticity factor of the lower layers (1 to 4) is 0, and the factor constantly increases for the upper layers. Which means null modification in the

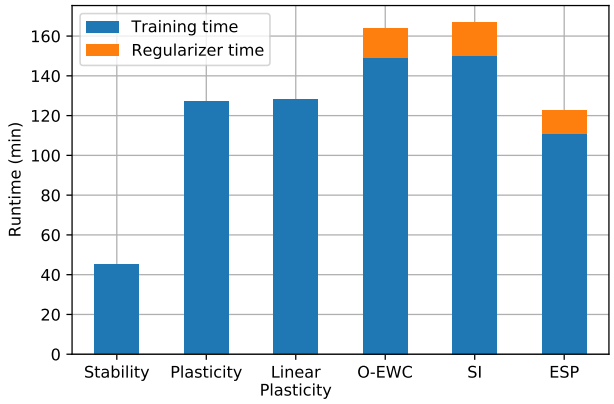


Figure 3. Training time comparison (in minutes) for all models with a replay of 20%. The blue color corresponds to the training time of the backbone and the orange color corresponds to the additional time that the regularizer takes.

lower layers and high modification in the upper layers. Our method leverages existing general knowledge in lower layers, which is general knowledge, while updating upper layers with task-specific knowledge.

Regarding vision, we use a pre-trained ResNet-18, a four-block model. Here our method behaves slightly similar to the natural language model, where the block with higher modifications is the last one. However, unlike the natural language results, the first block has a higher plasticity factor than blocks 2 and 3. This result may be due to the data used for pre-train the model. The BERT model is pre-trained on a massive corpus of different domains and topics, promoting the lower layers to be general for any task. On the other hand, ResNet-18 was pre-trained on ImageNet, which has much higher resolution images than CIFAR10. The basic patterns are expected to be different between both datasets, explaining the high plasticity factor of the first block.

In general, the plasticity factors for natural language and vision are higher at the last layers. However, no one reaches complete plasticity, which means those layers retain some specialization acquired in previous tasks. Also, Fig. 2 it shows that different amounts of percentages of replay sets have similar results, indicating that the plasticity of the network mainly depends on the current input.

Finally, we argue that ESP could be computationally efficient for cases like natural language experiments. Fig. 3 shows the training runtime of ESP and baselines under the 20% replay setup. We use a GPU NVIDIA GeForce RTX 3090 and a CPU AMD EPYC 7502 for these experiments. The Stability model is the more efficient, with $\sim$45 minutes, because it only updates the decoder layer. The Plasticity and Linear Plasticity model take $\sim$127 minutes because the entire backbone is trained. On the other hand, O-EWC and SI methods take a total time of $\sim$164 and $\sim$167 minutes. For a fair comparison, we divide the total execution time into the backbone training time (blue) and the regularizer time (orange) because both models include an additional process to calculate the importance of the weights. O-EWC takes $\sim$149 minutes of backbone training time and $\sim$15 minutes to find the importance of each weight. SI takes $\sim$150 minutes of backbone training time and $\sim$17 minutes of regularization time. Note that the backbone training time of these methods is superior to the plasticity model because they include an additional calculation loss based on the importance of the previously calculated weights.

Concerning our method, ESP finishes its training in $\sim$123 minutes. Analogous to O-EWC and SI, ESP has an additional process to tune the branch layer, which takes $\sim$12 minutes. This time is similar compared to O-EWC and SI regularizer time. However, if we compare the training time of the spine, ESP is remarkably efficient. ESP takes $\sim$111 minutes, which is less than other methods, including the Plasticity model. It happens because ESP sometimes forces the model not to update some layers, allowing those layers to be frozen on the fly, resulting in decreased training time.

5. Conclusion

In this paper, we introduced ESP, a method based on an entropy-based criterion to decide how much a model has to modify the weights of each of its layers. ESP augments each block of an encoder with branch layers that computes an entropy-based plasticity factor used to update layer weights dynamically. Our experiments in the natural language and vision domains show the effectiveness of our model in leveraging prior knowledge by not updating lower layers and specializing other layers by updating higher layers. In addition, we show that in the case of the natural language model, our method promotes computational efficiency since it forces not to update some layers.

Among the ideas for future work, we consider testing the hypothesis that ESP works better on networks with more blocks than a Resnet-18, such as vision Transformers. Also, we would like to extend ESP to an utterly online scenario.

Acknowledgement

We thanks the reviewers for their positive remarks and some valuable suggestions. This work was supported by the European Research Council Advanced Grant 788506 and the National Center for Artificial Intelligence CENIA FB210017, Basal ANID.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings*

of the European Conference on Computer Vision (ECCV), pages 139–154, 2018. 1

[2] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptive quantization modules. In *International Conference on Machine Learning*, pages 1240–1250. PMLR, 2020. 2

[3] Blake Camp, Jaya Krishna Mandivarapu, and Rolando Estrada. Continual learning with deep artificial neurons. *arXiv preprint arXiv:2011.07035*, 2020. 2

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[5] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and M Ranzato. Continual learning with tiny episodic memories. *International Conference on Machine Learning*, 2019. 2

[6] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1952–1961. PMLR, 13–18 Jul 2020. 2

[7] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8250–8259, 2021. 2

[8] Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2, 4

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 4

[10] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. *Proceedings of the European Conference on Computer Vision*, 2020. 2

[11] Cristobal Eyzaguirre, Felipe del Rio, Vladimir Araujo, and Alvaro Soto. DACT-BERT: Differentiable adaptive computation time for an efficient BERT inference. In *NLP Power! The First Workshop on Efficient Benchmarking in NLP*, 2022. 3

[12] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 2

[13] Tyler L. Hayes, Giri P. Krishnan, Maxim Bazhenov, Hava T. Siegelmann, Terrence J. Sejnowski, and Christopher Kanan. Replay in Deep Learning: Current Approaches and Missing Biological Elements. *Neural Computation*, 33(11):2908–2950, 10 2021. 2, 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[15] Julio Hurtado, Hans Lobel, and Alvaro Soto. Overcoming catastrophic forgetting using sparse coding and meta learning. *IEEE Access*, 9:88279–88290, 2021. 1, 2

[16] Julio Hurtado, Alain Raymond, and Alvaro Soto. Optimizing reusable knowledge for continual learning via metalearning. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[17] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *Proceedings of the European Conference on Computer Vision*. Springer, 2020. 2

[18] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018. 2

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, Mar. 2017. 1, 2

[20] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. 1, 2

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *http://www.cs.toronto.edu/ kriz/cifar.html*, 2009. 5

[22] Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning, 2019. 1, 2

[23] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *International Joint Conference on Neural Networks*. IEEE, 2019. 2

[24] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolias, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. 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. 5

[25] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 2

[26] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022. 2

[27] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 1, 2

[28] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[29] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[30] Jorge A Mendez and Eric Eaton. Lifelong learning of compositional structures. In *International Conference on Learning Representations*, 2021. 2

[31] Martial Mermillod, Aurélia Bugaiska, and Patrick BONIN. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*, 4, 2013. 1

[32] Oleksiy Ostapenko, Pau Rodriguez, Massimo Caccia, and Laurent Charlin. Continual learning via local module composition. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[33] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pages 524–540. Springer, 2020. 3

[34] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990. 1

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 2

[36] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2

[37] Gobinda Saha and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021. 2

[38] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018. 3

[39] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018. 1

[40] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in neural information processing systems*, 2017. 2

[41] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016. 3

[42] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 2

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 4

[44] Pieter Verbeke and Tom Verguts. Learning to synchronize: How biological agents can couple neural task modules for dealing with the stability-plasticity dilemma. *PLOS Computational Biology*, 15(8):1–25, 08 2019. 1

[45] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020. 2

[46] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. 3

[47] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 1, 2

[48] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833. Springer International Publishing, 2014. 1, 2

[49] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3987–3995. JMLR.org, 2017. 1, 2, 3, 5

[50] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. 4

[51] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020. 3