Breaking Free From the Stability-Plasticity Dilemma With Incremental Domain Recognition on Sequential Data

Romain Mouret

mouret.romain@gmail.com

July 10, 2021

Abstract

We make the case for identifying the input domain prior to running downstream models and propose an architecture that opens the door to single-head lifelong learning systems that forget at a decreasing rate as the tasks grow in complexity. Our model accurately identifies domains and is compatible with other continual learning algorithms, provided they benefit from knowing the current domain beforehand.

I. Introduction

One of the most reliable approaches to overcoming catastrophic forgetting is to replay the training data[1], usually by storing the training samples or by training a model to randomly generate training samples.

Since replay is expensive computation-wise and often memory-wise, new methods have been developed to mitigate catastrophic forgetting. Many of these methods [2, 3, 4] face a dilemma: the more flexible the weights are, the more likely the model will forget. Conversely, the more stable the weights are, the more difficult it is for the model to fit new data. In practice, under this paradigm, there seems to be no tradeoff suitable for training models over very large time scales, and there is no promising sign hinting that the stability-plasticity tradeoff can become more favorable as we tackle harder tasks with larger models. This is in stark contrast with the success of deep learning, which has shown that scaling up neural networks can lead to human-level performance on some tasks.

II. Prior domain knowledge in lifelong learning

I argue that approaches relying on prior knowledge of the current task or domain [5, 6, 7, 8] compare favorably to domain-unaware methods, but they are often dismissed on the basis that knowing the domain beforehand is not a realistic scenario in lifelong learning. *Prima facie*, this is a valid criticism since domain and task labels are normally not available during inference, and inferring them using a dedicated model is seemingly a problem of the same difficulty as class incremental learning [9] and of similar nature as one-class classification [10] if output scores were to be calibrated between domains. We find ourselves stuck in a circular logic: prior knowledge of the domain would be exceedingly helpful in building a continual learning (CL) system immune to catastrophic forgetting (CF), but a CF-immune CL system is required to robustly identify domains or tasks [7].

I propose to solve this problem by having the CL system detect discrepancies at the abstract level. The abstract level is the last layer of the feature processors. At this level, the values of the latent units no longer depend on the input domain. For example, the abstract representation of an ice cream truck seen on a rainy day should be identical to its representation on a sunny day despite contextual differences, assuming that the feature processors are trained to produce consistent abstractions thanks to a learning process like the one implemented by Algorithms 1 and 2.

This domain-invariance property of abstraction allows the system to make some predictions without already knowing the domain of the current observation, thereby avoiding cir-

cular logic.

III. Architecture

The proposed architecture requires future abstract states to be predictable by current abstract states. As a consequence, observations and targets have to be of a sequential nature. This includes text, time series, videos and various settings in robotics.

In this architecture, the state at time $t+1$ is predicted by two entities. (1) a model or a program called Center that predicts the next state from the current state, and (2) a feature processor P that predicts the next state from observations alone. These two predictions will conflict. We resolve conflicts by choosing the domain hypothesis under which P agrees the most with Center, typically by minimizing the Euclidean distance between Center's prediction and P's prediction.

As a special case of this architecture, we will build P as a collection of disjoint neural nets $P_1, P_2 \dots P_n$. Each network is trained to process a different domain. While this simplifies the training algorithm and allows the system to instantiate domain-specialized networks, it is memory inefficient and misses on the opportunity to mutualize knowledge between domains. It should be noted, however, that parameter sharing[11, 8] and pruning[12] can mitigate these issues.

IV. Algorithm

Below, A denotes the Action network mapping abstract states to targets.

The networks are trained with gradient descent using three loss functions operating on samples drawn from D_d, the distribution of the domain d under consideration.

Target loss For example, cross entropy:

$$l_1(d) = \mathop{\mathbb{E}}_{o \in D_d} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} y(o,t) \log(A(P_d(o_t))) \right]$$

where y maps an observation to its expected binary target.

Abstract state prediction loss

$$l_2(d) = \mathop{\mathbb{E}}_{o \in D_d} \left[\frac{1}{|o|-1} \sum_{t=1}^{|o|-1} \|Center(P_d(o_t)) - P_d(o_{t+1})\| \right]$$

State amplitude loss

$$l_3(d) = 1 - \mathop{\mathbb{E}}_{o \in D_d} \left[\frac{1}{|o|} \sum_{t=1}^{|o|} \|tanh(P_d(o_t))\| \right]$$

l_3 compensates for l_2's tendency to push amplitudes down.

Algorithm 1: Pre-training

Data: Small set X of arbitrary pre-training domains
do
  $d \leftarrow$ random domain from X ;
  train P_d conjointly with A and Center, minimizing $l_1(d) + l_2(d) + l_3(d)$ on one random sequence from d
until *convergence*;
Freeze A and P_d for every $d \in X$;
do
  $d \leftarrow$ random domain from X ;
  Fine-tune Center minimizing $l_2(d)$ on one random sequence from d
until *convergence*;
Freeze Center ;
Throw away P_d for every $d \in X$

Algorithm 2: Incremental training

foreach *domain* d **do**
  train a new processor P_d minimizing $l_1(d) + l_2(d)$
end

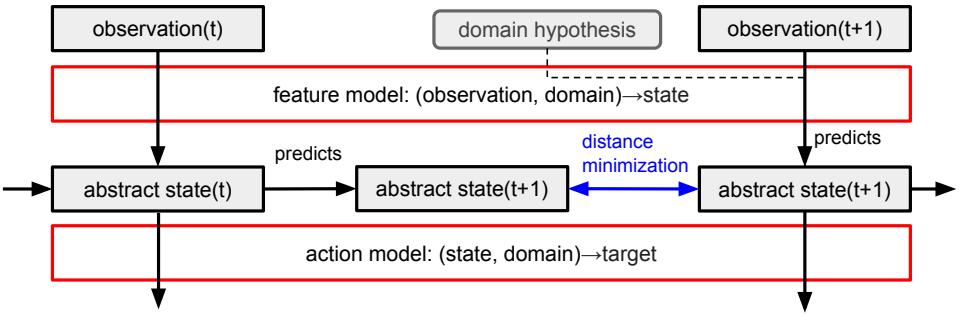

Figure 1: Architecture

Algorithm 3: Inference

Data: Observation o_{t+1} and state S_t
Result: S_{t+1}, inferred domain $\hat{d}$ and target T_{t+1}

$\hat{d} \leftarrow \text{argmin}_d \| P_d(o_{t+1}) - Center(S_t) \|$;
$S_{t+1} \leftarrow P_{\hat{d}}(o_{t+1})$;
$T_{t+1} \leftarrow A(S_{t+1}, \hat{d})$

V. Results

We will showcase the architecture with a language model trained on a dataset of cooking recipes [13]. Words of the recipes are randomly mapped to an integer between 0 and 4096 using a hash function. We create 7 domains by randomly permuting the mapping between words and indices, and pre-train the full system, including the abstraction layer, from one domain only.

To facilitate batch processing, our setup doesn't allow observations of the same sequence to belong to different domains. Under these circumstances, it would be too easy to take advantage of the perfect correlation between the domain at time t and the domain at time $t+1$, so we are purposefully ignoring this correlation while inferring the domain of the current observation.

Feature processors are implemented as masked convolution networks [14]. Center and A are implemented as multi-layer perceptrons.

As shown on Figure 2, the domain identification error rate drops as we increase the size of the abstraction layer. From size=16 to size=32, it is divided by 3.3. From size=32 to size=48, it is further divided by 2.9. At size = 48, the average domain identification accuracy is 99.23%, nearly eliminating catastrophic forgetting for non-overlapping neural networks on 7 domains or less.

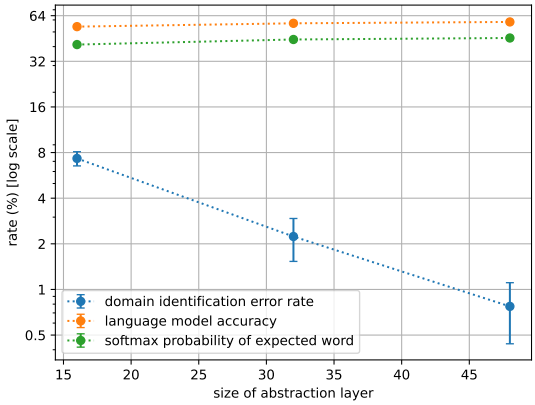

Figure 2: Model evaluation

The accuracy of the language model doesn't significantly increase as the abstraction layer gets wider. In this instance, this goes to show that the high accuracy of the domain identification module is not a mere consequence of increasing the accuracy of the entire system.

The PyTorch code is available on github [15].

VI. Limitations and future work

i. Training domain boundaries

Since it is possible to identify domains during inference, one can utilize a similar process and a distance threshold to detect unknown domains at training time. The remaining unsolved problem is to detect when two unknown domains are the same.

ii. Broader range of CL scenarios

Center and A are frozen after training. This is suitable for domain incremental learning [9] but doesn't leave any room for new abstractions and new targets. It is worth considering relaxing the constraints imposed by the frozen networks to let processors build new abstractions.

This will require replacing Center with a continually trained model. We are not finding ourselves in the same circular problem as the one we started with, as Center solely operates on abstract representations, which have a known, controllable shape and occupy significantly less storage space than raw sensory inputs.

iii. Dynamic state alignment

One could improve domain inference by allowing P to temporarily nudge its weights to better match Center's output. Instead of minimum Euclidean distance, the domain selection criterion would be the smallest change of weights required to align the two predictions.

VII. Conclusion

The main advantage of the proposed approach is that discrepancies get easier to detect as latent layers get wider, insofar as the additional model capacity is not exclusively allocated to new prunable or redundant units [16]. To avoid redundancy, the abstraction layer should be sized in proportion to the complexity of the tasks at hand. In other words, as we tackle more complex tasks, we are licensed to scale up models and their abstraction layer, which in turn lowers the likelihood of catastrophic forgetting.

References

[1] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *arXiv:2104.04132*, 2021.

[2] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[3] S. Wang, X. Li, J. Sun, and Z. Xu, "Training networks in null space of feature covariance for continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 184–193, June 2021.

[4] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995, PMLR, 06–11 Aug 2017.

[5] J. Serra, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 4548–4557, PMLR, 10–15 Jul 2018.

[6] M. Masana, T. Tuytelaars, and J. van de Weijer, "Ternary feature masks: Zero-forgetting for task-incremental learning,"

in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 3570–3579, June 2021.

[7] D. Abati, J. Tomczak, T. Blankevoort, S. Calderara, R. Cucchiara, and B. E. Bejnordi, "Conditional channel gated networks for task-aware continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] J. Pilault, A. E. hattami, and C. Pal, "Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data," in *International Conference on Learning Representations*, 2021.

[9] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv:1904.07734*, 2019.

[10] P. Perera, P. Oza, and V. M. Patel, "One-class classification: A survey," *arXiv:2101.03064*, 2021.

[11] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, "Supermasks in superposition," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 15173–15184, Curran Associates, Inc., 2020.

[12] J. Peng, H. Jiang, Z. Li, E. Guo, X. Wan, M. Deng, Q. Zhu, and H. Li, "Overcoming catastrophic forgetting by soft parameter pruning," *arXiv:1812.01640*, 2018.

[13] R. T. Lee, "Recipe Dataset." `https://eightportions.com/datasets/Recipes/`, 2018.

[14] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning*, pp. 1747–1756, PMLR, 2016.

[15] R. Mouret, "Domain Identification." `https://github.com/rom1mouret/domain_identification`, 2021.

[16] S. Casper, X. Boix, V. D'Amario, L. Guo, M. Schrimpf, K. Vinken, and G. Kreiman, "Frivolous units: Wider networks are not really that wide," in *AAAI*, 2021.