Task-agnostic Continual Learning via Growing Long-Term Memory Networks

Anonymous authors
Paper under double-blind review

Abstract

As our experience shows, humans can learn and deploy a myriad of different skills to tackle the situations they encounter daily. Neural networks, in contrast, have a fixed memory capacity that prevents them from learning more than a few sets of skills before starting to forget them. In this work, we make a step to bridge neural networks with human-like learning capabilities. For this, we propose a model with a growing and open-bounded memory capacity that can be accessed based on the model's current demands. To test this system, we introduce a continual learning task based on language modelling where the model is exposed to multiple languages and domains in sequence, without providing any explicit signal on the type of input it is currently dealing with. The proposed system exhibits improved adaptation skills in that it can recover faster than comparable baselines after a switch in the input language or domain.

1 Introduction

In a classic cartoon by Gary Larson, a student raises his hand to ask the teacher: "Mr. Osborne, may I be excused? My brain is full." (Larson & Martin, 2003). We laugh at this situation because we know it is absurd. Human brains don't just get full. Instead, they seem to be able to keep in their long-term memory massive amounts of information encoding well-acquired knowledge and skills. Furthermore, the information stored in memory is not necessarily relevant at all times. For instance, a person may have a phone call in French in the morning, then go about her daily errands in German, and later write an email in English. Different linguistic knowledge will be required for each of these situations, and context alone, rather than some explicit signal, will dictate what is needed at each given moment.

Vanilla neural network models have been successfully deployed in various applications in the past. However, they rely on fixed sized memories and suffer from the problem known as "catastrophic forgetting" (McCloskey & Cohen, 1989; Ratcliff, 1990), which refers to the fact that previously acquired information is quickly forgotten as novel skills need to be mastered. Earlier work attempted to correct this problem by looking for available capacity on a fixed-sized network that would allow encoding a new solution without affecting previously learned tasks (Kirkpatrick et al., 2017; Zenke et al., 2017; Serrà et al., 2018; Lopez-Paz & Ranzato, 2017; Fernando et al., 2017; Lee et al., 2017). The problem with this approach is that eventually, the system will run out of available capacity. Instead, here we argue for developing models that can grow their internal capacity. While some work has also relied on growing the model to face catastrophic forgetting (Rusu et al., 2016; Li & Hoiem, 2018; Aljundi et al., 2017), they all rely, to the best of our knowledge, on an explicit signal identifying the task that the system is currently solving. Indeed, most work dealing with catastrophic forgetting has evaluated the models on settings often making unrealistic assumptions. Not only they typically provided the model with an explicit identifier for the task at hand, but also tasks featured unnatural properties, such as scrambled pixels, or categories that were incrementally added, but presented sequentially on blocks once and for all, and never encountered again during training. Only recently, some work has started tackling continual learning in a more realistic task-agnostic way (Aljundi et al., 2019). Yet, there are no standard publicly available datasets that can help the evaluation of continual learning systems on more natural settings.

In this paper, we make a two-fold contribution towards task agnostic continual learning. First, we introduce a recurrent neural network that can grow its memory by creating new modules as training progresses. Rather than using all modules simultaneously, or indexing them based on a task identification signal, our model learns to weight their contributions to adapt to the current context. Second, we introduce to the community a multilingual/multidomain language modelling task with switching domains that we hope can fit this bill. We propose two variants of it. The first is a character-based language modelling benchmark with text written in 5 different languages that randomly switch between one another. The second one is a word-based language modelling task, where the text oscillates between 4 different domains. No segmentation signal is given when there is a switch, making the models having to discover it autonomously while they are evaluated for their adaptation skills.

Our experimental results show that our system can switch between different domains faster than comparable neural networks. Furthermore, our model is very general because it does not make any assumption about the type of underlying neural network architecture and thus, it can easily be adopted for tackling other tasks in conjunction with any other neural network system.

2 Growing models

Growth in neural networks has been explored with different perspectives. Here, we present a discussion of the possible avenues for developing neural networks with unbounded memory.

1. Growth of layers: Early work on Neural Networks used this method to reduce the amount of computational complexity or for escaping local minima (Fahlman & Lebiere, 1990; Hirose et al., 1991). The goal, back then, was using the smallest possible number of hidden units. Here, instead, we are interested in allowing neural networks to grow for endowing them with larger memory capacity. In this sense, this strategy seems limited because all units remain fully connected at all time, forcing the network to access all memories simultaneously.
2. Growth of architecture: A different type of growth could be the one dictated by a different model that decides the characteristics of the learning system, including how many units to put in it. Neural architecture search (Elsken et al., 2018) and, particularly, neuro-evolution (Stanley et al., 2019) provide good examples of this. Note, however, that this type of growth is different from the main problem that we are dealing with here, in which a model needs to be able to extend *itself*.
3. Learned, structured growth: Models, like the Stack-RNNs Joulin & Mikolov (2015) permit the model to create new units, which are placed on a stack data structure, allowing it thus to have a flexible memory to process different problem instances of varying sizes. The model itself learns how many computational resources to use, but so far this has been demonstrated only on toy problems like sequence memorization. Moreover, Stack-RNNs are also unable to quickly recover “memories” from distant past because it would imply cycling through the whole stack.
4. Sparse growth: This is the strategy that we focus on in this paper. The network is grown by blocks or modules. One potential drawback with this strategy is the linear increase of time complexity as the network grows. To prevent this, here we simply limit the maximum number of modules that are kept alive at any given time. Other, more sophisticated, options could employ a Hierarchical Softmax (Goodman, 2001) operation over the modules or Winner-Takes-All types of rules (Srivastava et al., 2013), essentially searching for just the right memories to answer the current situation.

3 Model

The proposed Growing Long-Term Memory Network (GLTMN) is composed of modules operating in concert to compute the network’s output by means of a weighted combination of their predictions. As such, it belongs to a family of architectures that, depending on whether the predictions are additively or multiplicatively combined, are referred as Mixture-of-Experts (Jacobs et al., 1991; Eigen et al., 2013) or Product-of-Experts (Hinton, 1999). Before combining them, all the module’s predictions are weighted by a vector of coefficients that can be produced by another module that is jointly trained. Or system differs in the following ways. First, the modules in our system are subdivided into two main blocks: the short-term memory (STM) and long-term memory (LTM). These two components

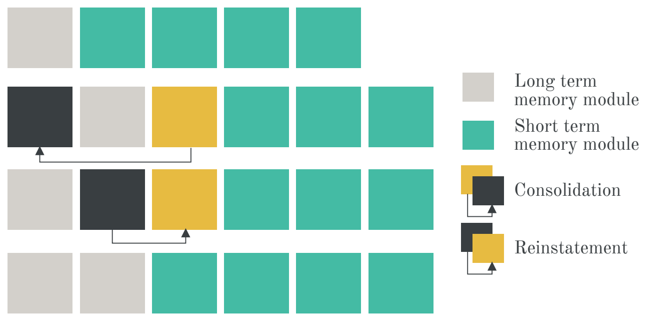


Figure 1: Sketch of the proposed model: Modules in the STM are actively trained while those in LTM remain frozen. To consolidate an STM module into LTM a new copy is created. A frozen LTM module can be reinstated back into STM for further training by overwriting the weights of the STM module that was just consolidated into LTM. When a maximum size limit is reached, the LTM copy of the reinstated module is removed to maintain the model size constant.

differ in the following ways. First, while the STM has a fixed number of modules, the LTM grows incrementally, only being limited in size by the hosting computer memory capacity. Second, while predictions are computed as a standard MoE architecture, using both LTM and STM modules, only the latter ones gets trained on incoming experience. Mixture weights that encompass both LTM and STEM are kept as a separate parameter of the system that is continually trained based on recent experience. Modules in the STM are consolidated into LTM whenever a trigger point is reached (such as, after a given number of examples have been processed) choosing the module that has been contributing the most to the output according to its corresponding mixture weight recent history. At this point, the module is removed from STM and frozen into LTM. Similarly, LTM modules can be reinstated back into STM by picking the module with the highest contribution weight and copying back into STM. When a maximum size is reached, the LTM module that was reinstated into STM is removed, thus keeping the overall size of the model constant (see Figure 1 for a general sketch).

Learning More formally, our model is composed of a set of modules $\mathcal{M} = \{M_1, \ldots, M_n\}$, where $M_1, \ldots, M_l$ are the modules in the LTM, and $\{M_{l+1}, \ldots, M_n\}$ are the ones in STM. At the beginning of training all modules belong to STM, thus $l = 0$. The system computes its predictions as follows. When an input x (with target y) is observed, it is fed to all modules $M_{1\ldots n}$, obtaining log-linear output vectors $\hat{\mathbf{y}}^{(1)} = M_1(x), \ldots, \hat{\mathbf{y}}^{(n)} = M_n(x)$. An additional vector of mixture weights $\mathbf{w} \in \mathbb{R}^n$ is used to linearly combine them. The output of the full model $\mathbf{y}$ is computed as a linear combination of the individual modules outputs weighted by the parameters w_i:

$$\hat{\mathbf{y}} = \operatorname{softmax}\left(\sum_{i=1}^{n} w_i * \hat{\mathbf{y}}^{(i)}\right) \tag{1}$$

Note that since we are combining the model unnormalized predictions before the application of the softmax, we are effectively computing a geometric combination of each individual module's unnormalized probabilities: $\exp(\hat{\mathbf{y}}_j) \propto \prod_{i=1}^{n} \exp\left(\hat{\mathbf{y}}_i^{(j)}\right)^{w_i}$. Thus, this model can be seen as a Product of Experts. Compared to a Mixture of Experts, this approach does not require to normalize the output of each individual model, thus being much more efficient to compute.

The predictions are then used to compute the cross-entropy loss $L(\hat{\mathbf{y}}, y)$, which is then used to backpropagate both into the STM modules and the mixture weights, and then optimized independently through gradient descent. In particular, to swiftly adapt the weights, we repeat the training update for $\mathbf{w}$ multiple ($k = 100$) times, whereas we only do it once for the modules. Note that in order to compute the gradients of the loss with respect to the weights after each update there is no need to recompute each module's output and thus, each step is not expensive to compute.

Memory management Every T processed examples, the system *consolidates* a STM module into LTM. For this, it picks the module that has been most active in recent experience, as measured by the absolute mean weight value over the past examples (we use the last 20 batches). To limit the amount of computational power needed, we restricted the maximum total number of modules to $n = 30$. When this limit is reached, another module is removed from LTM and *reinstated* back into STM for further training. We pick the module with the highest mixture weight in absolute value. That is, while STM modules are selected for consolidation based on their past importance so they can be preserved for future use, LTM modules are reinstated based on their present relevance, so they can be further adapted. Note that, despite that, in practice the model has a memory limit, its potential to allocate new memory is unbounded, not unlike modern computer systems where physical memory limits do not affect the way programs are written. In the future, the memory efficiency could be improved by incorporating mechanisms such as distillation (Hinton et al., 2015) to compress the information stored across different modules into a single one.

Last, but not least, note that the above-described model does not make any assumption about the type of architecture used. In our following experiments, we evaluate the model using an LSTM architecture, but there is nothing preventing it to be applied to feed-forward, convolutional or other types of networks; or even a mixture thereof.

3.1 Parametrization for Language Modelling

In this work we instantiate the GLTMN for an online language modelling task. For this we adopt double-layered LSTM networks as modules. Each of these modules observe text as small sequences of tokens, as is standard when applying the backpropagation through time algorithm, and has to predict each of the upcoming tokens by acting in concert with all the other modules[1].

When the system is not allowed to use the LTM, our model reduces to a Products of Experts (PoE) where the mixture coefficients are given by an independent weight vector parameter. To make this architecture comparable to previously proposed neural mixture models (e.g. Eigen et al. (2013)), we consider as a baseline a PoE model where the weights are computed by another network. In particular, we used another LSTM network that looks at the current batch and produces the weights.

4 Experiments

Our experimental setup aims to establish whether our model is able to adapt to a continuous stream of circumstances in which it needs to develop, improve and use a wide range of skills. While most previous work interested in continual learning considered sequences of tasks that were unambiguously identified by a marker given as an extra input to the model, here we are interested in a more realistic setup where only context can dictate which skills are required at a given time. For this, we introduce two lifelong language modelling tasks, where the model is continually exposed to a novel linguistic stream that switches between different languages or domains without any explicit marker signalling the change. More specifically, we propose two language modelling benchmarks: One is word-level and multi-domain whereas the other is character-level and multilingual. Both benchmarks feature conflicting learning signals when moving between domains or languages, making the network susceptible to catastrophic forgetting. A good model should be very good at transitioning between languages or domains, while still maintaining good overall performance for (online) language modelling.

4.1 Tasks and Dataset

We are interested in modelling the continual adaptation to incoming non-i.i.d. data, a situation that is closer to the learning experience of any human being. Therefore, the traditional train-test split approach is not adequate here. Instead, we adopt an online learning paradigm. This means that at each time step the model receives an instance x_t and makes a prediction $\hat{y}_t$. Then, the true target y_t will be observed, with the model incurring in a loss $L(\hat{y}_t, y_t)$. After reporting this loss, the model is trained, possibly for more than a single iteration, on the just observed example. The goal is minimizing the cumulative loss $\sum_{t=1}^{T} L(\hat{y}_t, y_t)$.

For our first text dataset we build on parts of the news corpus developed for the 2009 Workshop of Machine Translation (Callison-Burch et al., 2009). For this benchmark, we selected five languages:

[1]An implementation of our models together with the corresponding evaluation code will be made available at `http://anonymized/`

English, French, Spanish, German and Czech because they all have similar character sets, while also showing interesting linguistic variability thanks to belonging to three different Indo-European branches: Romance (French and Spanish), Germanic (English and German) and Slavic (Czech). Compared to earlier multilingual corpora (Kawakami et al., 2017), our dataset was carefully constructed to include only linguistically valid character sets, in order to prevent non-linguistic noise from interfering with our experiments. For this, we removed all lines from the input that containing characters appearing less than 100 times on the full orpus. The resulting character vocabulary is no bigger than 215 characters.

The second dataset is an English multi-domain dataset. For this, we used four different source corpora: news (same as above), europarl (Koehn, 2005), the Toronto Book Corpus (Zhu et al., 2015) and Wikipedia (Merity et al., 2016). We kept in the vocabulary the top 25K words for each corpus, which after merging yielded a vocabulary size of 58K words.

We then split the corpus in fragments coming from different languages or domains with lengths randomly sampled from a (truncated) exponential distribution. Thanks to the memorylessness property of this distribution, it is virtually impossible to estimate when the next switch is going to happen. For the multilingual dataset, we extracted 1M and 10M-characters-long randomly alternating combinations of 100 sequences, 10 for each language, with lengths sampled from a (truncated) exponential distribution with means $\lambda = 10\text{k}$ and $\lambda = 100\text{k}$ characters, respectively, and a 10M-characters-long one with 1000 sequences with mean length of 10k characters. For the multi-domain dataset we followed the same procedure, extracting 100 alternating sequences with mean lengths of $\lambda = 10\text{k}$ and $\lambda = 20\text{k}$, for a total of 1M and 2M words. We used a smaller corpus in this last case because to allow for faster experimentation as the models have now to predict over a larger vocabulary, and thus they require more training time.

4.2 Experimental Setup

We set the maximum size of the GLTMN to 30 LSTM modules having two layers and 200 hidden units each. We first compared it to the performance of a single equally-sized LSTM network, allowing us to measure the advantage of any mixture model with respect to any given single module. Second, we included as a further baseline another double-layered LSTM with 1300 units which has the same amount of parameters as our fully grown model on the multilingual task.

As reference points, we also trained independent LSTMs, one for each domain or language (thus, using for diagnostic purposes a "forbidden" domain supervision signal), enabling us to compare the performance of our model to a situation where there is no forgetting from conflicting learning signals, but also where there is no possibility of transferring learned representations across possibly related domains.

Finally, we evaluated a PoE, as described in Section 3.1. This is a model whose mixture coefficients are produced by a simple LSTM module with 10 hidden units. We also experimented with passing the weights through a softmax layer, thus enforcing a convex rather than a linear combination of the modules, but this did not prove useful neither for the GLTMN nor the PoE. We tuned the hyperparameters of all the models on a development set for each corpus. Among the ones that we considered for the GLTMN was the size of the STM choosing between 10, 15, 20 or all 30 modules. In the latter case, the model reduces to a PoE with a weights parametrized by a trainable vector of coefficients. Using 20 modules for the STM proved to be the best-performing strategy. Details of the hyperparamter search for the models are included in Appendix A.1.

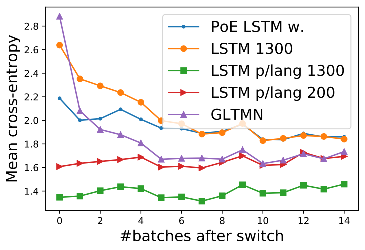
4.3 Metrics

We are interested in measuring whether the growing model brings in any advantage at recovering information that the network had learned before, while remaining competitive in terms of overall performance. To measure these aspects, we propose the following metrics:

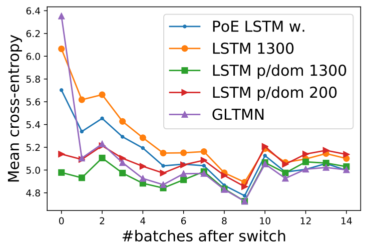
Online perplexity This is the general perplexity over the data measured during model training. Note that since the task is framed as an online learning one, the training loss serves as a test measure because no example is seen twice.

	multilingual				multi-domain			
	Perplexity		Confusion		Perplexity		Confusion	
	λ=10k	λ=100k	λ=10k	λ=100k	λ=10k	λ=20k	λ=10k	λ=20k
Ind. LSTM 1300	7.31	4.49	1.05	1.17	375	297	1.25	1.47
Ind. LSTM 200	8.36	5.82	1.22	1.28	419	342	1.32	1.56
LSTM 1300	**7.44**	5.31	9.37	30.3	373	279	7.5	9.54
LSTM 200	10.3	6.82	11.5	43.4	430	353	5.29	9.26
PoE	7.74	5.34	9.27	26.3	**289**	**233**	6.6	8.31
GLTMN	7.88	**5.27**	**4.97**	**15.59**	311	241	**4.73**	**6.04**

Table 1: Models perplexities and post-switch confusion metrics for the character-level multilingual and word-level multi-domain datasets.



(a) multilingual (λ = 100k)



(b) multi-domain (λ = 20k)

Figure 2: Mean cross-entropy for the 15 batches after a switch, averaged over all such occurrences.

Post-switch confusion: When a distribution shift is experienced, a neural network that suffers from forgetting typically produces a spike. With this measure we aim at capturing how large was this spike. Let L_{avg} be the average cross-entropy loss of the model between a switch occurring at $t = 0$ and the following one, and let t_{avg} be the time step at which the model touches this level for the first time. Then, we define confusion as:

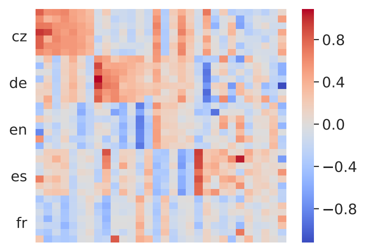
$$C = \sum_{t=0}^{t_{avg}} \frac{\exp(L_i)}{\exp(L_{avg})} = \sum_{t=0}^{t_{avg}} \exp(L_i - L_{avg})$$

That is, confusion computes the number of time steps weighted by the relative perplexity increase during which the model remains above the average loss for that sequence. We also complement this measure with plots that illustrate this process.

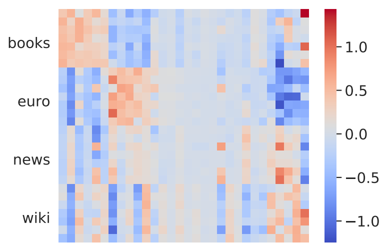
In order to observe the asymptotic behaviour of the models, we restrict our analysis by reporting measures pertaining only to the second half of the data.

4.4 Results

We report our experimental results for both the multilingual task and for the multi-domain data in Table 1. Results disaggregated by domain or language are reported in Appendix A.2. There are several aspects that are worth pointing out in these results. First, we can see that the results in overall online perplexity are mixed. The PoE with LSTM weights scores the biggest number of wins (two), followed by ours and a plain LSTM, winning one each. In general, we can conclude that there is a slight advantage for the PoE model, which may be due to its higher learning capacity during stable non-switching periods, but the GLTMN performs almost on-par with it. Moreover, when looking at the surprisal measures, we can see that the GLTMN excels at recovering after changes in the distribution. This observation can be graphically confirmed by looking at Figure 2. As it can be seen, the GLTMN recovers much faster than the stand-alone LSTM and the PoE models. It does spike, however, on the first batch, although this is completely expected. Recall that weights of the model are optimized according to each new batch. Thus, before the fist batch of the new data was observed it was not possible for it to adapt.



(a) multilingual ($\lambda = 10\text{k}$)



(b) multi-domain ($\lambda = 10\text{k}$)

Figure 3: Weight values for GLTMN computed over the last seven sequences observed by the model.

We also note that while on the multilingual data, a LSTM trained independently on each different language (first row in the table) exhibits the lowest perplexity, this does not hold in the multi-domain corpus. This shows that while there is limited room for transferring knowledge in the multilingual case, in consonance with previous results (Dhar & Bisazza, 2018). In contrast, the multi-domain setting provides plenty of opportunities for transferring knowledge across each domain, and thus task-agnostic systems can benefit them. The main takeaways are that enhancing the neural network with a modular structure (e.g. PoE, GLTMN) brings an improvement to the general performance, but the LTM component helps the network to recover faster after the task switch.

4.5 Analysis

To get some further insights into the workings of the model we analyzed its weight vectors in the multilingual and the multi-domain tasks ($\lambda = 10\text{k}$) for the last seven exposures to each language or domain. We then averaged all the weight vectors on every batch for each of these seven linguistic sequences. In Figure 3 we show all these vectors sorted by the corresponding type of sequence, while weight indices were sorted by means of a clustering scheme to highlight modules that are working in tandem. Note that any of these modules could be in LTM or STM at any given time.

In the multilingual case, we can see that there are a large number of modules allocated both to Czech and German. This may be explained by the fact that these are the two hardest domains as measured by the disaggregated perplexities (see Appendix for details). While there is some transfer of modules between languages, this remains quite limited, consistently with our observation above that models trained independently on each linguistic stream reach the lowest online perplexity. In contrast, for the multi-domain corpus, the clustering is much less clear. While the weights for the book corpus and the europarl domains seem to be mostly anti correlated and quite idiosyncratic, the modules acting on the news and wiki domains seem to be more distributed. This may also be explaining our findings that knowledge transfer helps in this task.

5 Related work

Our work is related to the efforts aimed at solving the problem of catastrophic forgetting in neural networks (McCloskey & Cohen, 1989; Ratcliff, 1990; French, 1999; Goodfellow et al., 2013), which have received considerable attention in the machine learning community. These approaches can be categorized into mainly two different branches: Those that keep the neural network size fixed and attempt to correct the learning procedure to avoid forgetting (Kirkpatrick et al., 2017; Zenke et al., 2017; Serrà et al., 2018; Lopez-Paz & Ranzato, 2017; Fernando et al., 2017; Lee et al., 2017), and those that allow the system to grow new modules to account for novel tasks (Rusu et al., 2016; Li & Hoiem, 2018; Aljundi et al., 2017; d'Autume et al., 2019). Our work is closer to the second stem.

Also, close to our approach are mixture of experts systems (Jacobs et al., 1991; Eigen et al., 2013; Shazeer et al., 2017; Wang et al., 2018; Yang et al., 2017; Bakhtin et al., 2018), in particular the product of experts approach (Hinton, 1999).

Other models with unbounded memory were proposed in the past (Fahlman & Lebiere, 1990; Joulin & Mikolov, 2015; Rusu et al., 2016; Li & Hoiem, 2018; Aljundi et al., 2017), although not all of them were studied in the context of continual learning, as we are doing here, and those who were assumed training tasks to be properly identified, as previously noted.

Similar to our work are the models enahnced with a memory component, such as: memory networks (Sukhbaatar et al., 2015), stack RNNs (Joulin & Mikolov, 2015) and neural turing machines (Graves et al., 2014) which show that having a structured memory helps with learning longer dependencies and remembering. While our approach has some similarities, the proposed model saves fully connected modules which can save into the memory not only data but also the algorithms learned by the modules.

The interaction between recent and remote memory has been extensively studied in the neuroscientific literature (McClelland et al., 1995). We do not claim any direct connection between our model and how the human brain works, but we borrow the terms associated with consolidation and reinstatement of memories, as they fit quite neatly into our context.

Finally, our problem formulation is an instance of neural-network-assisted language modelling (Bengio et al., 2003; Mikolov et al., 2010) and character level language modeling (Sutskever et al., 2011; Mikolov et al., 2012; Graves, 2013; Bojanowski et al., 2015). Some models conceived for language modeling can extend their memories to support fast-changing statistics from the recent past, as in cache models (Grave et al., 2017; Merity et al., 2016). Also, some other work has extended these models towards the multilingual setting (Östling & Tiedemann, 2016). Here, we adapt these problems to a life-long learning setup where different languages can be conceived as different tasks. Differently form cache models, context switching implies retrieving a vast set of skills from a relatively distant past.

6 Conclusions

We believe that developing more flexible forms of artificial intelligence will probably require flexible memory capabilities that can only be delivered by models capable of growth. Here we have proposed a method based on growing full-fledged modules over time. We explored a particular instantiation of this architecture in which modules are grown at a constant rate and consolidated into a long-term memory (LTM). Once the model has reached a maximum size, memories can be still be consolidated into LTM by reinstating LTM modules back into STM (see Figure 1).

Furthermore, we introduced to the community two lifelong language modelling tasks. One, character-based and multilingual, and other, word-based on multiple domains. Our experiments confirm the efficacy of our Growing LTM model, showing that it can learn to adapt much faster than comparable baselines without suffering in terms of its overall performance.

The proposed system is very flexible, allowing it to be used with any neural network architecture. While here we have studied it in the lifelong language modeling setting, we believe that the system will also show promising results in other domains with similar requirements, such as robotics –where the model can learn to deal with different kinds of terrains– or image recognition –where it can learn different kinds of visual information depending on the contextual requirements (Rebuffi et al., 2017).

In the future, mechanisms that exploit the structure of the input data for associating it with the relevant sets of models (Aljundi et al., 2017; Milan et al., 2016) can be explored. Furthermore, we plan to study mechanisms that would allow the model to decide when to grow, rather than keeping a constant schedule. In the long term, the model should be capable of deciding how to structure its long-term memory and whether or not to grow it, as Stack-RNNs (Joulin & Mikolov, 2015) do to grow the working memory. Moreover, we are interested in exploring how communication between memories can be enabled through a central routing mechanism, in a similar fashion to the model proposed by Hafner et al. (2017).

To conclude, in this work we have given a step –and we hope that more will follow– in providing neural networks with flexible memory structures. We expect that further pursuing this goal will pave the way towards developing more general learning systems and, fundamentally, that in the future neural networks will no longer need to be excused from class just because their weights are full.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, pp. 7120–7129, 2017.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

Anton Bakhtin, Arthur Szlam, Marc'Aurelio Ranzato, and Edouard Grave. Lightweight adaptive mixture of neural and n-gram language models. *arXiv preprint arXiv:1804.07705*, 2018.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Piotr Bojanowski, Armand Joulin, and Tomas Mikolov. Alternative structures for character-level rnns. *arXiv preprint arXiv:1511.06303*, 2015.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pp. 1–28, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626431.1626433`.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. Episodic memory in lifelong language learning. *arXiv preprint arXiv:1906.01076*, 2019.

Prajit Dhar and Arianna Bisazza. Does syntactic knowledge in multilingual language models transfer across languages? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 374–377, 2018.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Advances in neural information processing systems*, pp. 524–532, 1990.

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.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Joshua Goodman. Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pp. 561–564. IEEE, 2001.

Edouard Grave, Moustapha M Cisse, and Armand Joulin. Unbounded cache model for online language modeling with open vocabulary. In *Advances in Neural Information Processing Systems*, pp. 6042–6052, 2017.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Danijar Hafner, Alexander Irpan, James Davidson, and Nicolas Heess. Learning hierarchical information flow with recurrent neural modules. In *Advances in Neural Information Processing Systems*, pp. 6724–6733, 2017.

GE Hinton. Products of experts. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*, volume 1, pp. 1–6. IET, 1999.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yoshio Hirose, Koichi Yamashita, and Shimpei Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4(1):61–66, 1991.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems*, pp. 190–198, 2015.

Kazuya Kawakami, Chris Dyer, and Phil Blunsom. Learning to create and reuse words in open-vocabulary neural language modeling. *arXiv preprint arXiv:1704.06986*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, pp. 201611835, 2017.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pp. 79–86, 2005.

G. Larson and S. Martin. *The Complete Far Side: 1980-1994*. Andrews McMeel Publishing, 2003. ISBN 9780740721137.

Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, pp. 4652–4662, 2017.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*, 2012.

Kieran Milan, Joel Veness, James Kirkpatrick, Michael Bowling, Anna Koop, and Demis Hassabis. The forget-me-not process. In *Advances in Neural Information Processing Systems*, pp. 3702–3710, 2016.

Robert Östling and Jörg Tiedemann. Continuous multilinguality with language vectors. *arXiv preprint arXiv:1612.07486*, 2016.

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

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pp. 506–516, 2017.

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.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pp. 2310–2318, 2013.

Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of ICML 2011*, pp. 1017–1024, 2011.

Xin Wang, Fisher Yu, Ruth Wang, Yi-An Ma, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. Deep mixture of experts via shallow embedding. *arXiv preprint arXiv:1806.01531*, 2018.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *arXiv preprint arXiv:1703.04200*, 2017.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

A Appendix

A.1 Hyperparameter search

We used for all models a dropout rate of 0.2, Adam optimizer (Kingma & Ba, 2014) with a learning rate of $1e-3$ and embedding sizes of 200 units for all models. Both the GLTMN and the PoE have no more than 30 modules. For GLTMN we tuned the STM size (choosing between 10, 15 and 20), the period after which a consolidation/reinstatement cycle is triggered (5k, 10k and 20k characters) and the number of learning iterations on each observation (1 or 2). For the PoE models we tuned the learning rate independently for the modules and for the weight computation (10^{-2} or 10^{-3} in each case) and the number of learning iterations on each observation (1 or 2). For the simple LSTM models, we tuned the number of learning iterations (1, 2 or 5) and the learning rate (10^{-3}, 5×10^{-3} or 10^{-2}).

A.2 Results at the language and domain level

	Perplexity					Post-switch Confusion				
	es	fr	en	cz	de	es	fr	en	cz	de
	$\lambda = 10$k									
LSTM p/lang 1300	6.89	6.26	6.19	9.8	8.34	0.90	1.35	0.38	0.76	1.61
LSTM p/lang 200	7.86	7.31	7.45	10.99	9.05	1.19	1.06	0.94	0.65	1.96
LSTM 1300	6.9	6.9	7.1	10.9	6.9	8.44	7.15	9.84	11.47	10.4
LSTM 200	9.43	9.05	9.32	15.2	10.4	12.2	14.2	11	12.2	8.91
PoE	7.24	6.71	6.76	11.05	8.08	8.32	11.6	9.44	10.14	7.89
GLTMN (ours)	7.35	6.79	6.9	11.08	8.39	5.34	5.55	3.88	5.23	4.76
	$\lambda = 100$k									
LSTM p/lang 1300	4.07	4.04	4.21	5.76	4.74	0.84	0.98	2.08	0.48	1.54
LSTM p/lang 200	5.27	5.35	5.33	7.57	6.05	0.74	1.28	1.02	1.19	1.97
LSTM 1300	4.89	4.77	4.91	7.15	5.41	31.6	41	36	24.4	23.6
LSTM 200	6.32	6.03	5.96	9.61	7.09	41.7	45	45.7	50	37.4
PoE	4.94	4.77	4.90	7.15	5.51	28.8	29.3	30.7	18.6	25.6
GLTMN (ours)	4.86	4.74	4.79	7.05	5.46	13.7	21.2	18.1	14.7	12.9

Table 2: Disaggregated multilingual character level results.

	Perplexity				Post-switch Confusion			
	euro	news	books	wiki	euro	news	books	wiki
	$\lambda = 10$k							
LSTM p/dom 1300	370	674	152	367	2.18	0.52	2.18	1.39
LSTM p/dom 200	417	774	166	401	0.94	0.53	1.82	1.88
LSTM 1300	355	636	171	368	6.45	5.68	9.43	8.27
LSTM 200	424	706	199	430	5.3	4.39	4.82	6.41
PoE	265	519	140	256	5.89	4.21	8.09	7.84
GLTMN (ours)	288	551	150	278	4.39	2.63	5	6.46
	$\lambda = 20$k							
LSTM p/dom 1300	271	522	136	291	1.18	0.72	1.99	1.88
LSTM p/dom 200	334	597	152	321	1.84	0.49	2.02	1.73
LSTM 1300	244	471	144	267	9.77	8.56	10.84	8.97
LSTM 200	322	550	186	362	10.13	7.82	11.17	7.94
PoE	205	396	126	208	6.82	7.99	9.76	8.71
GLTMN (ours)	214	419	129	209	6.68	4.15	7.54	5.65

Table 3: Disaggregated multi-domain word-level results.