Continual Learning Long Short Term Memory

Xin Guo* University of Delaware guoxin@udel.edu

Yu Tian* Rutgers University yt219@cs.rutgers.edu

Qinghan Xue IBM qinghan.xue@ibm.com

Panos Lampropoulos IBM panosl1@ibm.com

Steven Eliuk IBM steven.eliuk@ibm.com

Kenneth Barner University of Delaware barner@udel.edu

Xiaolong Wang† IBM xiaolong.wang@ibm.com

Abstract

Catastrophic forgetting in neural networks indicates the performance decreasing of deep learning models on previous tasks while learning new tasks. To address this problem, we propose a novel Continual Learning Long Short Term Memory (CL-LSTM) cell in Recurrent Neural Network (RNN) in this paper. CL-LSTM considers not only the state of each individual task's output gates but also the correlation of the states between tasks, so that the deep learning models can incrementally learn new tasks without catastrophically forgetting previously tasks. Experimental results demonstrate significant improvements of CL-LSTM over state-of-the-art approaches on spoken language understanding (SLU) tasks.

1 Introduction

The whole AI community has enjoyed a superior performance boost from the emerging of deep learning technologies, thanks to the availability of big data and computing technologies. One of the most recent, realistic and emerged challenges for deep learning models on streaming data is continual learning capability. When new data is available, re-training brand new models with all the old and new data is the ideal way to achieve high performance on both tasks. However, there are several factors preventing saving old data for the entire lifetime, such as the memory restriction and data governance. When learning without all the old data, the performance on old tasks will drop dramatically, this phenomenon is called catastrophic forgetting (Mcclelland et al., 1995).

Catastrophic forgetting occurs in neural networks due to the stability-plasticity dilemma (Abraham and Robins, 2005), where the network requires sufficient plasticity to capture new tasks, but large weights variations may disrupt previous learned representations. Continual learning methods are proposed to prevent catastrophic forgetting, when only a limited size of old data is available.

Several approaches have been proposed to solve this problem in deep learning field (Awasthi and Sarawagi, 2019; Rusu et al., 2016; Zhizhong Li, 2018; Kirkpatrick et al., 2016; Riemer et al., 2019; Serra et al., 2018; Hou et al., 2018). A popular trend is to use expandable networks to store/learn old/new knowledge then acquire a task ID to select one from all the tasks during the inference stage. (Rusu et al., 2016; Mallya et al., 2018; Yoon et al., 2017; Mallya and Lazebnik, 2017).

In contrast, only a few attempts have been made to address catastrophic forgetting in natural language forgetting (NLP) field. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2016) has been adapted to visual question answering (Greco et al., 2019) and language modeling (Wolf et al., 2019). Progressive Neural Network proposed in reinforcement learning (Rusu et al., 2016) has been adopted to semantic slot filling in (Shen et al., 2019). A continual learning architecture preventing catastrophic forgetting via block-sparsity and orthogonality constraints is presented in (Pasunuru and Bansal, 2019) on diverse sentence-pair classification tasks.

* indicates equal contributions. This work was done during Xin and Yu's internship at IBM.

† Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1817–1822

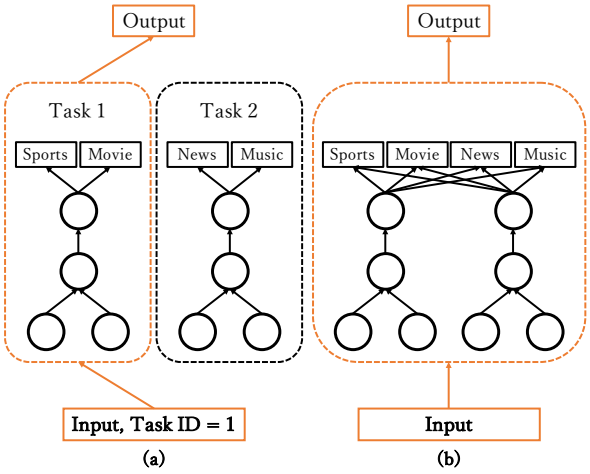

Figure 1: Deep neural networks a) with requirement on task IDs, and b) without requirement on task IDs, in inference stage.

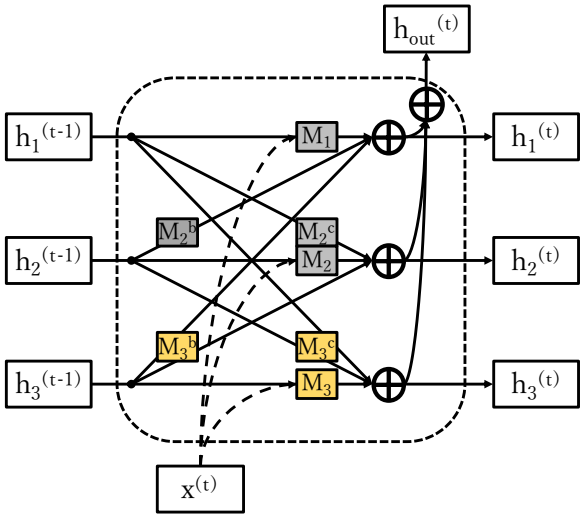

Figure 2: CL-LSTM with three tasks. For the third task, old modules are *frozen* (grey) and M_3, M_3^c, M_3^b (yellow) are trained for information sharing. $h_{out}^{(t)}$ is the aggregation of all hidden states.

To our best knowledge, none of the previous works in NLP considers the interactions between tasks at the LSTM cell level. Moreover, the requirement of task IDs in inference is infeasible and impractical in the real scenarios as shown in Fig. 1. Therefore, a novel Continual Learning Long Short Term Memory (CL-LSTM) cell is proposed to prevent catastrophic forgetting. The contributions of the paper are: (a) a novel LSTM cell for continual learning is proposed. The proposed CL-LSTM includes separate modules for different tasks; (b) each task further has a *broadcast* module to send its hidden states to all of the old tasks, and a *collect* module to take hidden states as inputs from all of the old tasks. Therefore, the output gates of each task integrates information from all tasks; (c) the proposed model doesn't require task IDs to perform inference, which is more practical in real-world scenarios. We evaluate the proposed CL-LSTM on both slot filling and intent detection of spoken language understanding. Experimental results show that the proposed CL-LSTM outperforms state-of-the-arts by a large margin. Code is available at https://github.com/IBM-GCDO/EMNLP-CL-LSTM.

2 Method

2.1 Preliminary: LSTM

As we know, LSTM (Long Short Term Memory) (Hochreiter and Schmidhuber, 1997) operates as a parameterized function R that takes an input vector x_t with a state vector (c_{t-1}, h_{t-1}) and returns a state vector $(c_t, h_t) = R(x_t, c_{t-1}, h_{t-1})$. Specifically, it incorporates a *gating mechanism*, taking the form:

$$f_t = W^f x_t + U^f h_{t-1} + b^f, \quad (1)$$
$$i_t = W^i x_t + U^i h_{t-1} + b^i, \quad (2)$$
$$o_t = W^o x_t + U^o h_{t-1} + b^o, \quad (3)$$
$$\tilde{c}_t = W^c x_t + U^c h_{t-1} + b^c, \quad (4)$$

where Ws and Us are learnable matrices, bs are biases. If we integrate Ws and Us into one single matrix W, combine bs into b, then by concatenating x_t and h_{t-1} together, we have:

$$[f_t, i_t, o_t, \tilde{c}_t] = W[x_t, h_{t-1}] + b. \quad (5)$$

The outputs c_t and h_t can be obtained from:

$$c_t = \sigma(f_t) \circ c_{t-1} + \sigma(i_t) \circ \tanh(\tilde{c}_t), \quad (6)$$
$$h_t = \sigma(o_t) \circ g(c_t), \quad (7)$$

where σ indicates the sigmoid function, $\circ$ represents the Hadamard product, g can be either tanh or the identity function. In this paper, we are interested in the hidden states: for a standard LSTM cell with parameters $\{W, b\}$ included within one module M, the update of h_t can be represented as:

$$h_t = M(x_t, h_{t-1}). \quad (8)$$

2.2 CL-LSTM

As discussed above, model parameters $\{W, b\}$ in the standard LSTM cell keep updating once the

given cell starts to learn the new task, which makes it difficult to avoid catastrophic forgetting. To mitigate this phenomena, we propose a novel cell named CL-LSTM as illustrated in Fig. 2, which is mainly composed of the following components:

Task-oriented Modules. Assuming that the model is going to learn K tasks sequentially. The training data is $X = \{X_1, X_2, ..., X_K\}$, where X_k denotes the training dataset for the k^{th} task. There are C_k different classes included in task k. When the first task comes, CL-LSTM starts with a single module $M_1 = \{W_1, b_1\}$. M_1 is updated like a standard LSTM with the training data $x \in X_1$:

$$h_1^{(t)} = M_1(x^{(t)}, h_1^{(t-1)}), t \in \{1, 2, \cdots, T\}, \quad (9)$$

where $h_1^{(t)}$ is the hidden state at timestamp t, T represents the length of sequential data x, $c_1^{(t)}$ is updated by Eq. 6. When starting to work on a new task $k > 1$, parameters of old tasks ($M_{<k}$) are *frozen* and new module $M_k = \{W_k, b_k\}$ is created. This design allows the model to keep old information in an expandable way.

Hidden State Sharing Modules. We design a communication mechanism to allow the information sharing in hidden states across different tasks. Specifically, when it goes to task $k > 1$, a *broadcast* module $M_k^b = \{W_k^b, b_k^b\}$ is created to send hidden states of task k to all previous ($< k$) modules. On the reverse information flow, a *collect* module $M_k^c = \{W_k^c, b_k^c\}$ is created for task k to collect all hidden states from all previous modules. For any $1 \leq j \leq k$, the hidden states of module j are updated by:

$$h_j^{(t)} = M_j(x^t, h_j^{(t-1)}) + \sum_{1 \leq i < j} M_j^c(h_i^{(t-1)}) + \sum_{j < l \leq k} M_l^b(h_l^{(t-1)}), t \in \{1, 2, \cdots, T\}, \quad (10)$$

where $h_j^{(t)}$ is the updated hidden state of module j with additional information sharing. Note that at task k, M_j^c and M_j^b are *frozen* for all $j < k$. The intuition of broadcast and collect module is: when learning a new task k, M_k^c can learn how to aggregate *weighted* previous knowledge to accelerate and improve the knowledge learning of task k. And via M_k^b, the knowledge of task k can broadcast to previous modules, facilitating the task separations as well as enhancing the performance of old tasks.

Hidden States and Outputs. At k^{th} task, we have k hidden states at timestamp t: $h_i^{(t)}, i \in \{1, 2, \cdots, k\}$. To avoid using task ID to select different modules for different tasks during inference, we directly feed the input data to *all* modules and aggregate the knowledge from $\forall k \leq K$ tasks, an unique *output* hidden state $h_{out_k}^{(t)}$ is obtained by:

$$h_{out_k}^{(t)} = h_1^{(t)} + h_2^{(t)} + \cdots + h_k^{(t)}. \quad (11)$$

Note that different from standard LSTM, here $h_{out_k}^{(t)}$ is the summation of all modules' hidden states.

Dataset	ATIS	SNIPS	WR	RT	MV
Train	4,478	13,084	30,521	6,894	8,797
Valid	500	700	8,621	1,521	2,443
Test	893	700	4,181	766	978
# Slot	119	71	28	17	25
# Intent	22	7	12	1	1

Table 1: Dataset statistics on train, valid, test sets, and number of slot and intent labels.

3 Experiments

In this section, CL-LSTM is evaluated on Spoken Language Understanding (SLU) tasks in continual learning framework. SLU mainly includes two goals: slot filling and intent detection. Slot filling is a sequence labelling problem which maps each sentence to a sequence of slot labels with the same length, while intent detection is a classification problem where each sentence has one intent label.

3.1 Datasets

We evaluate the performance of the proposed CL-LSTM on five datasets (Table 1): Airline Travel Information Systems (ATIS) (Hemphill et al., 1990), Snips (Coucke et al., 2018), Weather Reminder (WR) (Wea), MIT Corpus Movie (MV) (MIT, a) and MIT Corpus Restaurant (RT) (MIT, b). ATIS, Snips and WR datasets have both slot and intent labels while RT and MV have slot labels only.

3.2 Experimental Settings

Two experimental settings are proposed to fully utilize these multi-goal datasets to evaluate catastrophic forgetting in continual learning.

Exp1: in order to perform both slot filling and intent detection simultaneously, each method is evaluated on three tasks sequentially: ATIS→SNIPS→WR, where each dataset is a task.

Exp2: in order to use all the datasets, each method is evaluated on five tasks for slot filling only with task order: ATIS→SNIPS→WR→RT→MV.

When training a new task, only N exemplars (training samples) from previous tasks are kept.

3.3 Training Details

We use $h_{out_k} = \{h^{(1)}_{out_k}, h^{(2)}_{out_k}, ..., h^{(T)}_{out_k}\}$ for slot filling, and the final state $h^{(T)}_{out_k}$ for intent detection. Specifically, the predictions (p_{slot}, p_{intent}) are made by adding fully connect layers $\mathcal{F}_{slot}$ and $\mathcal{F}_{intent}$ to sequential hidden outputs h_{out_k} and final hidden outputs $h^{(T)}_{out_k}$, respectively:

$$p_{slot} = \mathcal{F}_{slot}(h_{out_k}), \quad (12)$$

$$p_{intent} = \mathcal{F}_{intent}(h^{(T)}_{out_k}). \quad (13)$$

Model parameters are updated with cross-entropy loss. $\mathcal{F}_{slot}, \mathcal{F}_{intent}$ are always trainable to allow information sharing among different tasks.

3.4 Evaluation Metrics

F1-score and classification accuracy are reported for slot filling and intent detection, respectively. Semantic accuracy as defined in (Schuster and Paliwal, 1997) is used to evaluate the combined performance of both slot filling and intent detection.

In order to evaluate the overall model performance, after training the last task, averaged metrics are computed on the test datasets of *all* the tasks. Average metrics show models' effectiveness on preventing catastrophic forgetting, since performance drop on old tasks will lead to a lower average.

3.5 Baseline Models

Four baseline methods including fine-tuning, joint training, Learning Without Forgetting (LWF) (Zhizhong Li, 2018) and EWC (Kirkpatrick et al., 2016) are used to compare with the proposed CL-LSTM. Specifically, fine-tuning loads trained model on previous task to initialize model parameters; Joint training trains with the training data of all the tasks in each experiment and serves as the upper bound; LWF is a state-of-the-art continual learning method which can be adapted to language understanding tasks; EWC has been adapted to natural language understanding tasks such as visual question answering and language modeling.

3.6 Implementation Details

To perform a fair comparison, a bidirectional LSTM (Bi-LSTM) is used as the model structure for these baseline methods. All models are implemented with TensorFlow 1.13.1. During training, each sentence is a sequences of words, and we convert each word into a 128 dimensional word embeddings before feeding into the LSTM cell. We set number of neurons to be 64 for the LSTM cell, M_k is a $[128 + 64, 64 \times 4]$ matrix for both the baseline Bi-LSTM and CL-LSTM. For CL-LSTM, the broadcast module M^b_k and collect module M^c_k are $[64, 64 \times 4]$ matrices. $\mathcal{F}_{slot}$ and $\mathcal{F}_{intent}$ are fully connected layers take 64-dimensional $h^{(\cdot)}_{out_k}$ as input, and output vectors with dimensions same as the number of slot labels and intent labels, respectively.

All models are trained with Adam optimisation (Kingma and Ba, 2014) method. The learning rate is initially set to be 0.001 and updated with a decay rate of 0.05. For each task, model is trained for 100 epochs, and the best performed model on the current task is selected. Besides, using Ubuntu-18.04.1 system with 2 GPUs (NVIDIA V100-SXM2-16gb), CL-LSTM takes (i) 8 hours to run 100 epochs for 3 tasks experiment; (ii) 20 hours to run 100 epochs for 5 tasks experiment.

Method	**50**	**100**	**200**	**300**	**500**
Joint Training	89.91	89.91	89.91	89.91	89.91
Fine-tune	65.85	72.24	78.69	82.23	85.31
LWF	65.51	73.48	79.38	82.03	84.30
EWC	61.22	67.22	76.99	79.42	82.55
CL-LSTM$^-$	71.29	78.38	83.00	84.65	87.36
CL-LSTM	**74.74**	**79.96**	**83.97**	**85.54**	87.68
CL-LSTM$^+$	74.43	79.81	83.88	85.20	**87.73**

Table 2: Results of Exp1 on *F1-score* along with exemplar size from 50 to 500 samples.

Method	**50**	**100**	**200**	**300**	**500**
Joint Training	95.05	95.05	95.05	95.05	95.05
Fine-tune	76.15	81.55	86.20	88.24	91.19
LWF	78.26	81.43	86.40	87.16	90.23
EWC	76.94	81.55	86.27	88.02	90.21
CL-LSTM$^-$	78.69	82.12	85.56	87.63	90.76
CL-LSTM	**79.10**	**82.49**	86.48	87.91	91.15
CL-LSTM$^+$	78.84	81.79	**87.59**	**88.58**	**91.23**

Table 3: Results of Exp1 on *intent accuracy* along with exemplar size from 50 to 500 samples.

3.7 Experimental Results

For Exp1, experimental results on F1-score, intent accuracy and semantic accuracy along with the exemplar size are shown in Table 2∼4, respectively. Note that the results of joint training are invariant to the number of exemplar size since it refers to the training with all the training data of all the tasks.

We also evaluate another two versions of the proposed CL-LSTM which are CL-LSTM$^-$ and

Method	**50**	**100**	**200**	**300**	**500**
Joint Training	76.92	76.92	76.92	76.92	76.92
Fine-tune	40.46	49.53	55.08	60.23	67.57
LWF	40.79	48.23	56.85	59.93	66.24
EWC	37.84	43.01	54.05	56.52	61.69
CL-LSTM$^-$	45.73	54.92	62.59	64.08	70.37
CL-LSTM	**50.46**	**57.84**	**63.81**	**65.36**	70.99
CL-LSTM$^+$	50.36	56.96	63.67	64.91	**71.00**

Table 4: Results of Exp1 on *semantic accuracy* along with exemplar size from 50 to 500 samples.

Method	**50**	**100**	**200**	**300**	**500**
Joint Training	75.86	75.86	75.86	75.86	75.86
Fine-tune	39.82	52.50	54.52	57.03	68.04
LWF	38.42	51.83	54.15	57.14	68.42
EWC	46.90	**55.69**	**62.42**	**67.08**	71.25
CL-LSTM	48.26	53.34	55.62	60.77	70.82
CL-LSTM$^+$	**49.49**	52.80	55.85	61.84	**71.75**

Table 5: Results of Exp2 on *F1-score* along with exemplar size from 50 to 500 samples.

CL-LSTM$^+$. CL-LSTM$^-$ is an ablated model, where hidden state sharing modules are not included and only task-oriented modules are used. CL-LSTM$^+$ is a more complicated design, where different broadcast and collect modules are created for every *pair* of tasks, please refer to Supplementary for more details.

Experimental results show that CL-LSTM and CL-LSTM$^+$ outperform state-of-the-art methods (fine-tuning, LWF and EWC models). The results in Table 4 also show that the proposed CL-LSTM models outperform baseline methods on semantic accuracy by a margin of 3.42% when exemplar size is 500, and 9.67% when exemplar size is 50. As semantic accuracy evaluates joint performance of slot filling and intent detection, it indicates that CL-LSTM is promising for continual learning, especially with limited size of exemplars.

For Exp2 in Table 5, we observe CL-LSTM has best performance with the most and least exemplars, while EWC shows advantages in other cases. Compare to the results in Table 2∼4, EWC is problematic when it has to maintain the weights for *both* slot filling and intent detection. In addition, CL-LSTM is orthogonal to EWC, so EWC can be applied on top of CL-LSTM to further improve the performance.

3.8 Ablation Study

As listed in Table 2∼4, the ablated model CL-LSTM$^-$ outperforms fine-tuning, LWF and EWC models on most of the metrics, showing that *freezing* previous modules can keep old knowledge. However, both of the CL-LSTM and CL-LSTM$^+$ are better than CL-LSTM$^-$, illustrating that rather than a simple aggregation (Eq. 11), the information sharing between tasks (Eq. 10) benefits both old and new tasks, which is important in continual learning. Using only one broadcast and one collect module for each task instead of specific models for every pair of tasks, performance of CL-LSTM is comparable to CL-LSTM$^+$, showing that a simplified broadcast/collect design may avoid over-fitting, especially in fewer tasks.

module	task 0	task 1	task 2	avg
h_0	43.85	0	0	14.62
h_1	48.18	28.32	0	25.50
h_2	41.41	9.77	46.47	32.55

Table 6: Semantic accuracies of using each module's output as prediction in Exp1 setting.

3.9 Analysis of Module Aggregation

In Eq. 11, the output of each module $h_i^{(t)}$ are aggregated into an unified output $h_{out_k}^{(t)}$. The benefit of this design is that the fusion frees the dependence on task IDs during inference. A detailed analysis is provided here to further illustrate the superior performance of using $h_{out_k}^{(t)}$, by comparing to models that directly use $h_i^{(t)}$ as the output. Specifically, after training our model on ATIS→SNIPS→WR with Eq. 11, we predict on test sets of task 0,1,2 with $h_i = \{h_i^{(1)}, ..., h_i^{(T)}\}$ separately, the semantic accuracies for 50 exemplars are shown in Table 6. We can see: 1) h_i only has predictive power for task $\leq i$ (as h_i is trained with task $\leq i$ data, then being frozen); 2) Compared to h_i, our h_{out} has better average semantic accuracy (CL-LSTM achieves 50.46% in table 4 for 3 tasks), which shows that h_{out} takes the advantage of information aggregation. Note that the recurrent architecture makes it possible for accuracy of h_{out} greater than maximum accuracy among $h_i, i = 0, 1, 2$.

4 Conclusion

In this paper, we propose a novel CL-LSTM cell to alleviate catastrophic forgetting problem in continual learning frameworks. Experimental results have demonstrated that adding broadcast and collect modules can help keeping old knowledge as well as learning new knowledge. Superior performance is achieved by CL-LSTM over other related works on spoken language understanding tasks.

References

Facebook Multi-language Dataset (Weather Reminder). https://github.com/sz128/slot_filling_and_intent_detection_of_SLU/tree/master/data/multilingual_task_oriented_data/en.

a. *MIT Corpus Movie Dataset*. https://groups.csail.mit.edu/sls/research.

b. *MIT Corpus Restaurant Dataset*. https://groups.csail.mit.edu/sls/research.

Wickliffe C Abraham and Anthony Robins. 2005. Memory retention–the synaptic stability versus plasticity dilemma. *Trends in neurosciences*, 28(2).

Abhijeet Awasthi and Sunita Sarawagi. 2019. Continual learning with neural networks: A review. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, CoDS-COMAD '19, New York, NY, USA. ACM.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *ArXiv*, abs/1805.10190.

Claudio Greco, Barbara Plank, Raquel Fernández, and Raffaella Bernardi. 2019. Psycholinguistics meets continual learning: Measuring catastrophic forgetting in visual question answering. In *ACL*.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2018. Lifelong learning via progressive distillation and retrospection. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

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

James Kirkpatrick, Razvan Pascanu, Neil C. 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. 2016. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *The European Conference on Computer Vision (ECCV)*.

Arun Mallya and Svetlana Lazebnik. 2017. Packnet: Adding multiple tasks to a single network by iterative pruning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

James Mcclelland, Bruce Mcnaughton, and Randall O'Reilly. 1995. 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.

Ramakanth Pasunuru and Mohit Bansal. 2019. Continual and multi-task architecture search. *CoRR*, abs/1906.05226.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *In International Conference on Learning Representations (ICLR)*.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *CoRR*, abs/1606.04671.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11).

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning*.

Yilin Shen, Xiangyu Zeng, and Hongxia Jin. 2019. A progressive model to enable continual learning for semantic slot filling. 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)*.

Thomas Wolf, Julien Chaumond, and Clement Delangue. 2019. Continuous learning in a hierarchical multiscale neural network. In *ACL*.

Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *ArXiv*, abs/1708.01547.

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