Progressive Memory Banks for Incremental Domain Adaptation

Nabiha Asghar

Collaborators:
Lili Mou, Kira Selby, Kevin Pantasdo, Pascal Poupart, Xin Jiang

Motivation

- **Domain Adaptation (DA)**: Transfer knowledge from one domain to another (in a machine learning system; especially neural networks)

- ***Incremental* Domain Adaptation (IDA):** Sequentially incoming domains
  - Only have access to data of current domain
  - Build a unified model that performs well on all domains

- **Use-cases of IDA**
  1. Company loses a client and its data, but wants to preserve the ‘knowledge’ in the ML system
  2. Quickly adapt to new domain/data without training from scratch
  3. Don’t know the domain of a data point during inference

Outline

- Prevalent and State-of-the-art DA & IDA methods in NLP
- Proposed Approach: Progressive Memory for IDA
- Theoretical Analysis
- Empirical Experiments
    - Natural Language Inference (Classification)
    - Dialogue Response Generation
- Conclusion

Related Work - DA & IDA

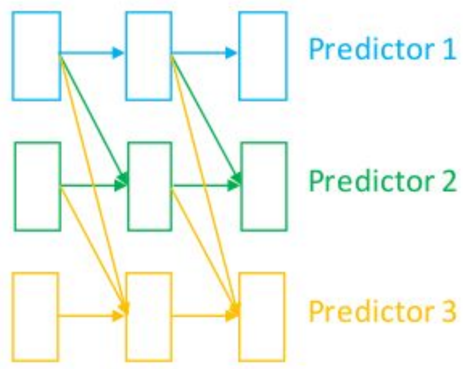
- **Multi-task learning:** Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains

Related Work - DA & IDA

- **Multi-task learning:** Jointly train on all domains
    - Non-incremental DA
    - Expensive to add new domain; needs data for all domains
- **Finetuning:** Sequentially train on all domains
    - Catastrophic forgetting of old domains

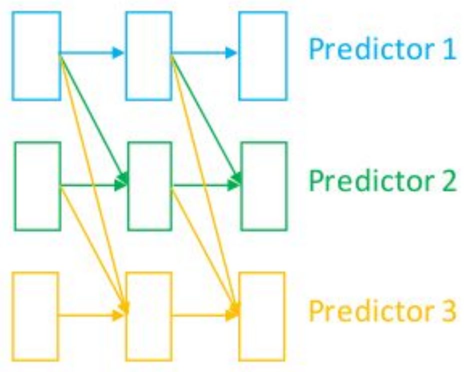
Related Work - DA & IDA

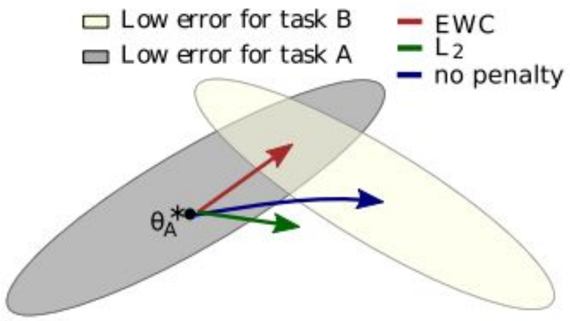
- **Multi-task learning:** Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains
- **Finetuning:** Sequentially train on all domains
  - Catastrophic forgetting of old domains
- **Progressive Neural Networks:** Training with network expansion and partial freezing
  - For prediction, need to know domain of input



Related Work - DA & IDA

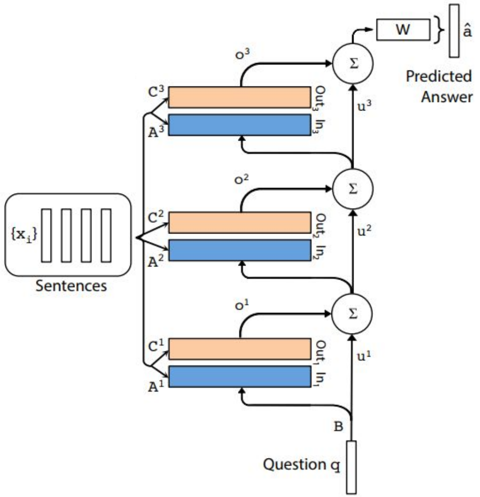
- **Multi-task learning:** Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains
- **Finetuning:** Sequentially train on all domains
  - Catastrophic forgetting of old domains
- **Progressive Neural Networks:** Training with network expansion and partial freezing
  - For prediction, need to know domain of input
- **Elastic Weight Consolidation (EWC):** Finetuning with regularization
  - Control learning on weights important for older domains
  - keeps the weights in a neighborhood of one possible minimizer of the empirical risk of the first task
  - needs to store a large number of parameters



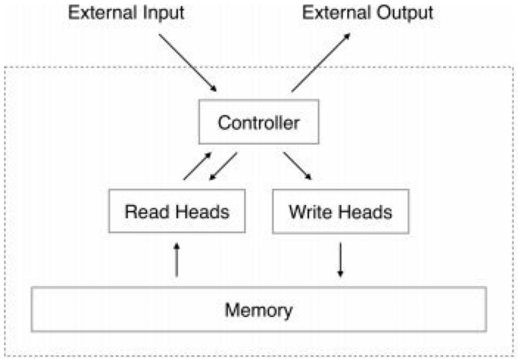


Related Work - Memory Networks

- **End-to-end memory network**
  - Assign a memory slot to an input sentence/sample
  - Assign a memory slot to one history

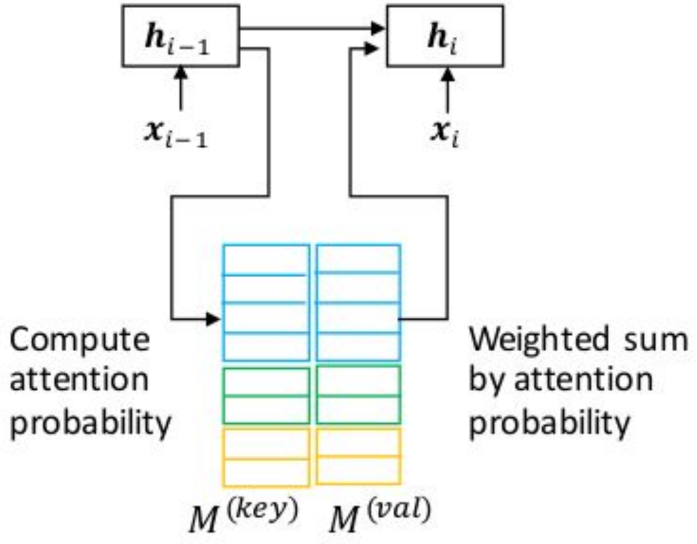


- **Neural Turing Machine**
  - Memory is not directly parameterized; read/written by neural controller
  - Serves as temporary scratch paper; does not store knowledge



Proposed Approach - Progressive Memory

- Incrementally increase model capacity (by increasing memory size)
- Memory slots store knowledge in distributed fashion
- We adopt key-value memory



Progressive Memory

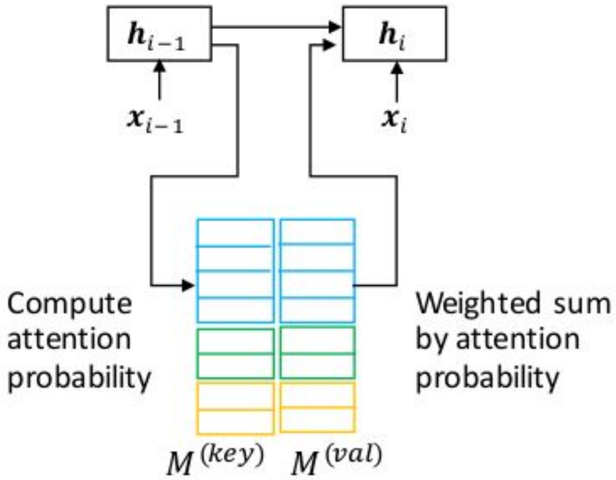
At time step i:

The RNN state is given by $\boldsymbol{h}_i = \mathrm{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i)$

The memory mechanism computes an attention probability $\boldsymbol{\alpha}_i$ by

$$\widetilde{\alpha}_{i,j} = \exp\{\boldsymbol{h}_{i-1}^\top \boldsymbol{m}_j^{(\mathrm{key})}\}$$

$$\alpha_{i,j} = \frac{\widetilde{\alpha}_{i,j}}{\sum_{j'=1}^{N} \widetilde{\alpha}_{i,j'}}$$



$\boldsymbol{m}_j^{(\mathrm{key})}$: key vector of j'th memory slot (N in total)

Retrieve memory content by weighted sum (by attention probability) of all memory values:

$$\boldsymbol{c}_i = \sum_{j=1}^{N} \alpha_{i,j} \boldsymbol{m}_j^{(\mathrm{val})}$$

$\boldsymbol{m}_j^{(\mathrm{val})}$: value vector of j'th memory slot

$$\boxed{\boldsymbol{h}_i = \mathrm{RNN}(\boldsymbol{h}_{i-1}, [\boldsymbol{x}_i, \boldsymbol{c}_i])}$$

Progressive Memory

At time step i:

The RNN state is given by $\boldsymbol{h}_i = \mathrm{RNN}(\boldsymbol{h}_{i-1}, \boldsymbol{x}_i)$

The memory mechanism computes an attention probability $\boldsymbol{\alpha}_i$ by

$$\widetilde{\alpha}_{i,j} = \exp\{\boldsymbol{h}_{i-1}^\top \boldsymbol{m}_j^{(\text{key})}\}$$

$$\alpha_{i,j} = \frac{\widetilde{\alpha}_{i,j}}{\sum_{j'=1}^{N} \widetilde{\alpha}_{i,j'}}$$

$\boldsymbol{m}_j^{(\text{key})}$: key vector of j’th memory slot (N in total)

Retrieve memory content by weighted sum (by attention probability) of all memory values:

$$\boldsymbol{c}_i = \sum_{j=1}^{N} \alpha_{i,j} \boldsymbol{m}_j^{(\text{val})}$$

$\boldsymbol{m}_j^{(\text{val})}$: value vector of j’th memory slot

For IDA:

Add M slots to original N slots

$$\alpha_{i,j}^{(\text{expand})} = \frac{\widetilde{\alpha}_{i,j}}{\sum_{j'=1}^{N+M} \widetilde{\alpha}_{i,j'}}$$

$$\boldsymbol{c}_i^{(\text{expand})} = \sum_{j=1}^{N+M} \alpha_{i,j}^{(\text{expand})} \boldsymbol{m}_j^{(\text{val})}$$

Algorithm

Algorithm 1: Progressive Memory for IDA

Input: A sequence of domains $D_0, D_1, \cdots, D_n$
Output: A model performing well on all domains
Initialize a memory-augmented RNN
Train the model on D_0
for $D_1, \cdots, D_n$ **do**
  Expand the memory with new slots
  Load RNN weights and existing memory banks
  Train the model by updating all parameters
end
Return: The resulting model

Training Considerations

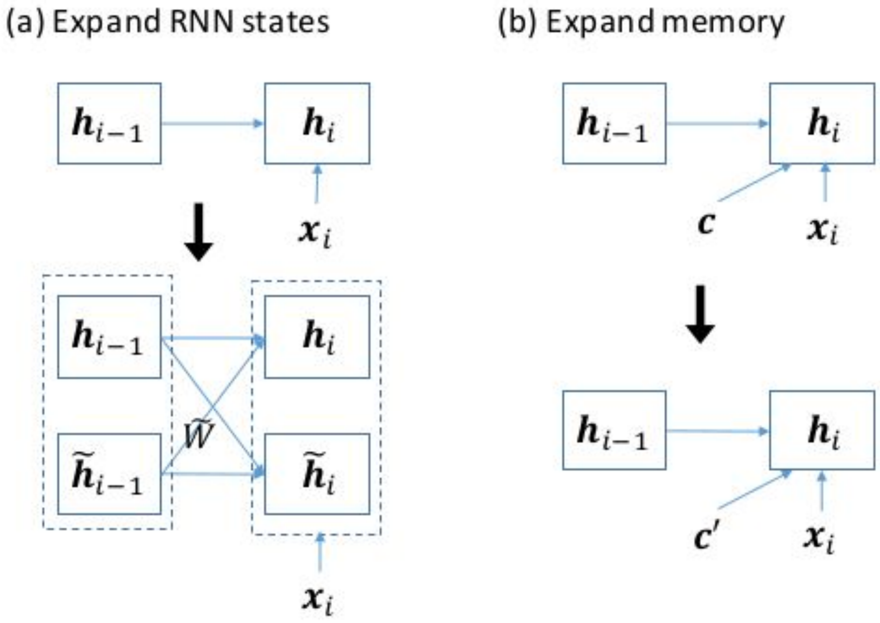
- Freezing learned params ***versus*** Finetuning learned params
    - Empirical results are better for latter
- Finetuning w/o increasing memory ***versus*** Finetuning w/ increasing memory
    - increased model capacity helps to learn new domain with less overriding of the previously learned model. Empirical results confirm this.
- Expanding hidden states ***versus*** Expanding Memory
    - An alternate way of increasing model capacity
    - similar to the progressive neural network, except that all weights are fine-tuned and there are connections from new states to existing states.
    - Theoretical and empirical results show latter is better

Expanding hidden states vs Expanding Memory

Theorem 1. *Let RNN have vanilla transition with the linear activation function, and let the RNN state at the last step $\boldsymbol{h}_{i-1}$ be fixed. For a particular data point, if the memory attention satisfies $\sum_{j=N+1}^{N+M} \widetilde{\alpha}_{i,j} \leq \sum_{j=1}^{N} \widetilde{\alpha}_{i,j}$, then memory expansion yields a lower expected mean squared difference in $\boldsymbol{h}_i$ than RNN state expansion, under reasonable assumptions. That is,*

$$\mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{m})} - \boldsymbol{h}_i\|^2\right] \leq \mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{s})} - \boldsymbol{h}_i\|^2\right]$$

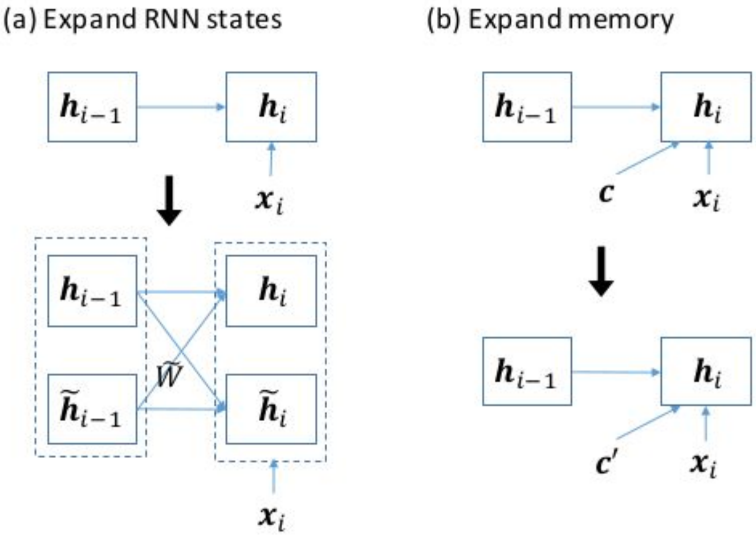
where $\boldsymbol{h}_i^{(\mathrm{m})}$ refers to the hidden states if the memory is expanded. $\boldsymbol{h}_i^{(\mathrm{s})}$ refers to the original dimensions of the RNN states, if we expand the size of RNN states themselves.



To prove: $\mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{m})} - \boldsymbol{h}_i\|^2\right] \le \mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{s})} - \boldsymbol{h}_i\|^2\right]$

Suppose the original hidden state $\boldsymbol{h}_i$ is D-dimensional. We assume each memory slot is d-dimensional, and that the additional RNN units when expanding the hidden state are also d-dimensional. We further assume every variable in the expanded memory and expanded weights are iid with zero mean and vari ance σ^2. Finally, every variable in the learned memory slots, i.e., m_{jk}, follows the same distribution (zero mean, variance σ^2). This assumption may not be true after the network is trained, but is useful for proving theorems.

$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{s})} - \boldsymbol{h}_i\|^2\right] \\
&= \mathbb{E}\left[\|\widetilde{W} \cdot \widetilde{\boldsymbol{h}}_{i-1}\|^2\right] \\
&= \sum_{j=1}^{D} \sum_{i=1}^{d} \mathbb{E}\left[\left(\widetilde{w}_{jk}\right)^2\right] \mathbb{E}\left[\left(\widetilde{h}_{i-1}[k]\right)^2\right] \\
&= D \cdot d \cdot \mathrm{Var}\big(w\big) \cdot \mathrm{Var}(h) \\
&= Dd\sigma^2\sigma^2
\end{aligned}$$



$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{h}_i^{(\mathrm{m})} - \boldsymbol{h}_i\|^2\right] \\
&= \mathbb{E}\left[\big\|W_{(c)}\Delta\boldsymbol{c}\|^2\right] \quad \text{where} \quad \Delta\boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{c}' - \boldsymbol{c} \\
&= Dd\sigma^2\mathrm{Var}\big(\Delta c_k\big)
\end{aligned}$$

$W_{(\mathrm{c})}$ is the weight matrix connecting attention content to RNN states.

It remains to show that $\mathrm{Var}(\Delta c_k) \le \sigma^2$

$$\Delta \boldsymbol{c} = \boldsymbol{c}' - \boldsymbol{c}$$

$$= \sum_{j=1}^{N} (\alpha'_j - \alpha_j)\boldsymbol{m}_j + \sum_{j=N+1}^{N+M} \alpha'_j \boldsymbol{m}_j = \sum_{j=1}^{N+M} \beta_j \boldsymbol{m}_j$$

$$\beta_j \stackrel{\text{def}}{=} \begin{cases} \dfrac{-\widetilde{\alpha}_j \frac{\widetilde{\alpha}_{N+1}+\cdots+\widetilde{\alpha}_{N+M}}{\widetilde{\alpha}_1+\cdots+\widetilde{\alpha}_N}}{\widetilde{\alpha}_1 + \cdots + \widetilde{\alpha}_{N+M}}, & \text{if } 1 \le j \le N \\ \dfrac{\widetilde{\alpha}_j}{\widetilde{\alpha}_1 + \cdots + \widetilde{\alpha}_{N+M}}, & \text{if } N+1 \le j \le N+M \end{cases}$$

$$\begin{aligned} \mathrm{Var}(\Delta c_k) &= \mathbb{E}[(c'_k - c_k)^2] \quad \forall 1 \le k \le d \\ &= \frac{1}{d}\,\mathbb{E}\left[\|\boldsymbol{c}' - \boldsymbol{c}\|^2\right] \\ &= \frac{1}{d}\,\mathbb{E}\left[\sum_{k=1}^{d}\left(\sum_{j=1}^{N+M} \beta_j m_{jk}\right)^2\right] \\ &\le \sigma^2\,\mathbb{E}\left[\sum_{j=1}^{N+M} (\alpha'_j)^2\right] \\ &\le \sigma^2 \qquad \square \end{aligned}$$

Memory	Unnormalized measure	Original attn. prob.	Expanded attn. prob.
$\boldsymbol{m}_1$	$\tilde{\alpha}_1$	α_1	α'_1
$\boldsymbol{m}_2$	$\tilde{\alpha}_2$	α_2	α'_2
...	...	...	...
$\boldsymbol{m}_N$	$\tilde{\alpha}_N$	α_N	α'_N
$\boldsymbol{m}_{N+1}$	$\tilde{\alpha}_{N+1}$		α'_{N+1}
...	...		...
$\boldsymbol{m}_{N+M}$	$\tilde{\alpha}_{N+M}$		α'_{N+M}

Figure 3: Attention probabilities before and after memory expansion.

Competing Methods

- Multi-task learning (Non-IDA)
- Finetuning with fixed memory
- **Finetuning with increasing memory**
- Finetuning with expanding hidden states
- Progressive Neural Network
- Elastic Weight Consolidation (EWC)

Competing Methods

- Multi-task learning (Non-IDA)
- Finetuning with fixed memory*
- **Finetuning with increasing memory***
- Finetuning with expanding hidden states*
- Progressive Neural Network
- Elastic Weight Consolidation (EWC)

* with and without additional vocabulary

Experiment I - Natural Language Inference

- Classification Task
    - Determine the relationship between two sentences (*entailment*, *contradiction* or *neutral*)
- Dataset: MultiNLI Corpus (~400K labelled samples)
    - 5 domains: Fiction, Government, Slate, Telephone, Travel
- Base Model: BiLSTM network with pretrained GloVe embeddings

Experiment I - Results

#Line	Model	Trained on/by	% Accuracy on S	% Accuracy on T
1	RNN	S	$65.01^{\Downarrow}$	$61.23^{\Downarrow}$
2	RNN	T	$56.46^{\Downarrow}$	$66.49^{\Downarrow}$
3	RNN+ Mem	S	$65.41^{\Downarrow}$	$60.87^{\Downarrow}$
4	RNN+ Mem	T	$56.77^{\Downarrow}$	$67.01^{\Downarrow}$
5	RNN+ Mem	S+T	$66.02^{\downarrow}$	70.00
6	RNN + Mem	S→T (F)	$65.62^{\downarrow}$	$69.90^{\downarrow}$
7	RNN + Mem	S→T (F+M)	66.23	70.21
8	RNN + Mem	S→T (F+M+V)	**67.55**	**70.82**
9	RNN + Mem	S→T (F+H)	$64.09^{\Downarrow}$	$68.35^{\Downarrow}$
10	RNN + Mem	S→T (F+H+V)	$63.68^{\Downarrow}$	$68.02^{\Downarrow}$
11	RNN + Mem	S→T (EWC)	$66.02^{\Downarrow}$	$64.10^{\Downarrow}$
12	RNN + Mem	S→T (Progressive)	$64.47^{\Downarrow}$	$68.25^{\Downarrow}$

For the statistical test (compared with Line 8), $\uparrow, \downarrow$: $p < 0.05$ and $\Uparrow, \Downarrow$: $p < 0.01$.

Experiment I - Results

Group	Setting	Fic	Gov	Slate	Tel	Travel
Non-IDA	In-domain training	65.41$^{\Downarrow}$	67.01$^{\Downarrow}$	59.30$^{\Downarrow}$	67.20$^{\Downarrow}$	64.70$^{\Downarrow}$
	Fic + Gov + Slate + Tel + Travel (multi-task)	**70.60**$^{\uparrow}$	**73.30**	63.80	69.15	67.07$^{\downarrow}$
IDA	Fic → Gov → Slate → Tel → Travel (F+V)	67.24$^{\downarrow}$	70.82$^{\Downarrow}$	62.41$^{\downarrow}$	67.62$^{\downarrow}$	**68.39**
	Fic → Gov → Slate → Tel → Travel (F+V+M)	*69.36*	*72.47*	**63.96**	**69.74**	**68.39**
	Fic → Gov → Slate → Tel → Travel (EWC)	67.12$^{\Downarrow}$	68.71$^{\Downarrow}$	59.90$^{\Downarrow}$	66.09$^{\Downarrow}$	65.70$^{\Downarrow}$
	Fic → Gov → Slate → Tel → Travel (Progressive)	65.22$^{\Downarrow}$	67.87$^{\Downarrow}$	61.13$^{\Downarrow}$	66.96$^{\Downarrow}$	67.90

Experiment II - Dialogue Response Generation

- Generation Task
  - Given an input sentence, generate an appropriate output sentence
- Datasets
  - Source Domain: Cornell Movie Corpus (~220K labelled samples)
  - Target Domain: Ubuntu Dialogue Corpus (~15K labelled samples)
- Base Model: Seq2Seq with decoder-to-encoder-attention

Experiment II - Results

# Line	Model	Trained on/by	BLEU-2 on S	BLEU-2 on T	W2V-Sim on S	W2V-Sim on T
1	RNN	S	2.842$^{\Uparrow}$	0.738$^{\Downarrow}$	0.480$^{\Downarrow}$	0.456$^{\Downarrow}$
2	RNN	T	0.795$^{\Downarrow}$	1.265$^{\Downarrow}$	0.454$^{\Downarrow}$	0.480$^{\Downarrow}$
3	RNN+ Mem	S	**3.074**$^{\Uparrow}$	0.712$^{\Downarrow}$	0.498$^{\Downarrow}$	0.471$^{\Downarrow}$
4	RNN+ Mem	T	0.920$^{\Downarrow}$	1.287$^{\Downarrow}$	0.462$^{\Downarrow}$	0.487$^{\Downarrow}$
5	RNN+ Mem	S+T	2.650$^{\Uparrow}$	0.889$^{\Downarrow}$	0.471$^{\Downarrow}$	0.462$^{\Downarrow}$
6	RNN + Mem	S→T (F)	1.210$^{\Downarrow}$	1.101$^{\Downarrow}$	0.509$^{\Downarrow}$	0.514$^{\Downarrow}$
7	RNN + Mem	S→T (F+M)	1.435$^{\Downarrow}$	1.207$^{\Downarrow}$	**0.526**	0.522
8	RNN + Mem	S→T (F+M+V)	*1.637*	**1.652**	0.522	**0.525**
9	RNN + Mem	S→T (F+H)	1.036$^{\Downarrow}$	1.606$^{\downarrow}$	0.503$^{\Downarrow}$	0.495$^{\Downarrow}$
10	RNN + Mem	S→T (F+H+V)	1.257$^{\Downarrow}$	1.419$^{\Downarrow}$	0.504$^{\Downarrow}$	0.492$^{\Downarrow}$
11	RNN + Mem	S→T (EWC)	1.397$^{\Downarrow}$	1.382$^{\downarrow}$	0.513$^{\Downarrow}$	0.514$^{\Downarrow}$
12	RNN + Mem	S→T (Progressive)	1.299$^{\Downarrow}$	1.408$^{\downarrow}$	0.502$^{\Downarrow}$	0.503$^{\Downarrow}$

Conclusion

- Proposed progressive memory for IDA (Incremental Domain Adaptation)
- Outperforms other IDA approaches
- Empirical results show it avoids catastrophic forgetting
- Theoretical results show it is better than other ways of capacity expansion

- Details: https://arxiv.org/pdf/1811.00239.pdf

References

- **EWC:** Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. PNAS, 2017.
- **Progressive Neural Networks:** Rusu et al. “Progressive neural networks”. arXiv:1606.04671, 2016.
- **End-to-end Memory Networks:** Sukhbaatar et al. “End-to-end memory networks”. NIPS, 2015.
- **Neural Turing Machine:** Graves et al. “Hybrid computing using a neural network with dynamic external memory”. Nature, 2016.
- **MultiNLI corpus:** Williams et al. “A broad-coverage challenge corpus for sentence understanding through inference”. NAACL, 2018.
- **Cornell Movie Corpus:** Danescu and Lee. “Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs”. CMCL, 2011.
- **Ubuntu Dialogue Corpus:** Lowe et al. “The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems”. SIGDIAL, 2015.