Learning from few examples

One-shot learning with memory-augmented neural networks

Maciej Żelaszczyk

March 21, 2018

PhD Student in Computer Science
Division of Artificial Intelligence and Computational Methods
Faculty of Mathematics and Information Science

m.zelaszczyk@mini.pw.edu.pl

Warsaw University of Technology

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.
- Data: search engines, social networks.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.
- Data: search engines, social networks.
- Conditions: combination of the above.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.
- Data: search engines, social networks.
- Conditions: combination of the above.
- Success: image recognition, games, speech recognition, translation, etc.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.
- Data: search engines, social networks.
- Conditions: combination of the above.
- Success: image recognition, games, speech recognition, translation, etc.
- Learning relies heavily on extensive datasets.

Traditional deep learning

- Network design: feedforward nets, CNNs, LSTMs, etc.
- Computational resources: GPUs.
- Data: search engines, social networks.
- Conditions: combination of the above.
- Success: image recognition, games, speech recognition, translation, etc.
- Learning relies heavily on extensive datasets.
- Sometimes the net is not as important as the data.

Traditional deep learning

- Backpropagation, stochastic gradient descent.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.
- Weights updated slowly.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.
- Weights updated slowly.
- Gradual changes in network behavior.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.
- Weights updated slowly.
- Gradual changes in network behavior.
- Possibility to freeze network, show new classes and retrain.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.
- Weights updated slowly.
- Gradual changes in network behavior.
- Possibility to freeze network, show new classes and retrain.
- Substantial number of new instances needed.

Traditional deep learning

- Backpropagation, stochastic gradient descent.
- Extensive, incremental learning.
- Weights updated slowly.
- Gradual changes in network behavior.
- Possibility to freeze network, show new classes and retrain.
- Substantial number of new instances needed.
- Possibly inefficient with respect to data.

Different learning paradigm

- Generalize from very few examples.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.
- Quickly adapts to new instances.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.
- Quickly adapts to new instances.
- Single observations shift network behavior dramatically.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.
- Quickly adapts to new instances.
- Single observations shift network behavior dramatically.
- Rapid inference.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.
- Quickly adapts to new instances.
- Single observations shift network behavior dramatically.
- Rapid inference.
- Data efficient to add new classes.

Different learning paradigm

- Generalize from very few examples.
- Network has a degree of general knowledge.
- Quickly adapts to new instances.
- Single observations shift network behavior dramatically.
- Rapid inference.
- Data efficient to add new classes.
- Modular design.

Learning to learn

- Meta-learning [Schmidhuber et al., 1997].

Learning to learn

- Meta-learning [Schmidhuber et al., 1997].
- Various incarnations of the idea.

Learning to learn

- Meta-learning [Schmidhuber et al., 1997].
- Various incarnations of the idea.
- General premise - learning occurs on two levels:

Learning to learn

- Meta-learning [Schmidhuber et al., 1997].
- Various incarnations of the idea.
- General premise - learning occurs on two levels:
  1. Within a task, e.g. bind input data to class in a particular dataset.

Learning to learn

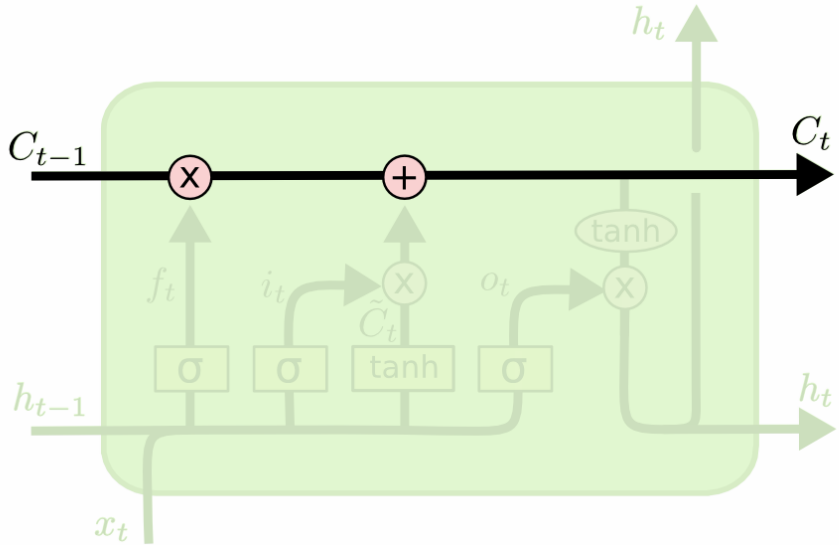
- Meta-learning [Schmidhuber et al., 1997].
- Various incarnations of the idea.
- General premise - learning occurs on two levels:
  1. Within a task, e.g. bind input data to class in a particular dataset.
  2. Across tasks - how task structure varies across target domains.

Learning to learn

- Meta-learning [Schmidhuber et al., 1997].
- Various incarnations of the idea.
- General premise - learning occurs on two levels:
  1. Within a task, e.g. bind input data to class in a particular dataset.
  2. Across tasks - how task structure varies across target domains.
- Several neural net structures seem fit to meta-learn.

Long-short term memory

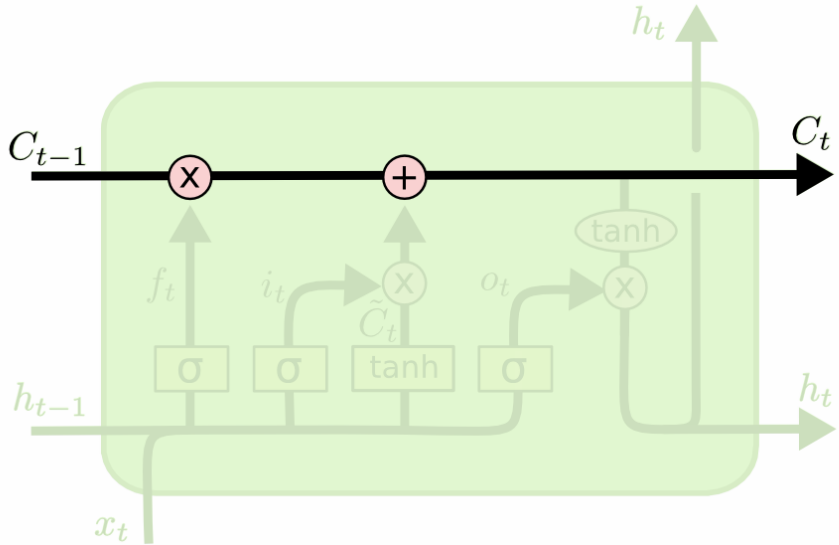
- Introduced to circumvent the vanishing gradient problem [Hochreiter and Schmidhuber, 1997].



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

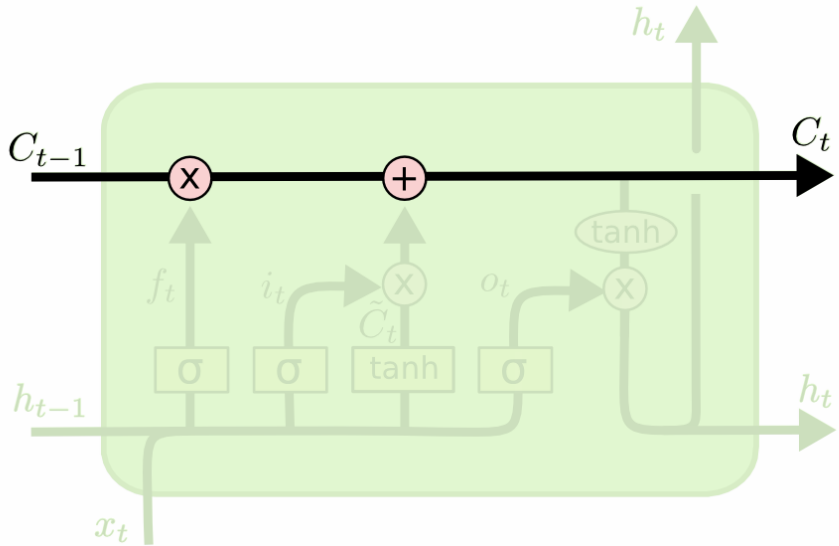
- Introduced to circumvent the vanishing gradient problem [Hochreiter and Schmidhuber, 1997].
- Architecture consists of:



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

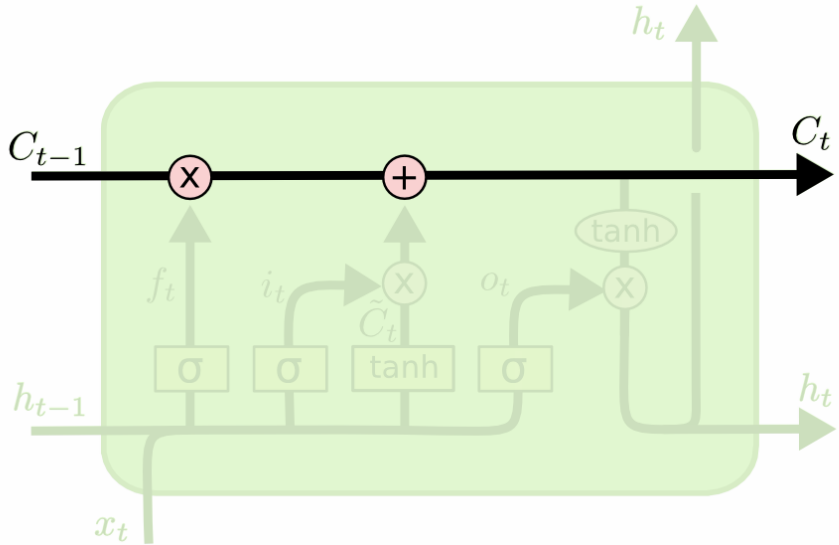
- Introduced to circumvent the vanishing gradient problem [Hochreiter and Schmidhuber, 1997].
- Architecture consists of:
  1. Network weights and activation functions.



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

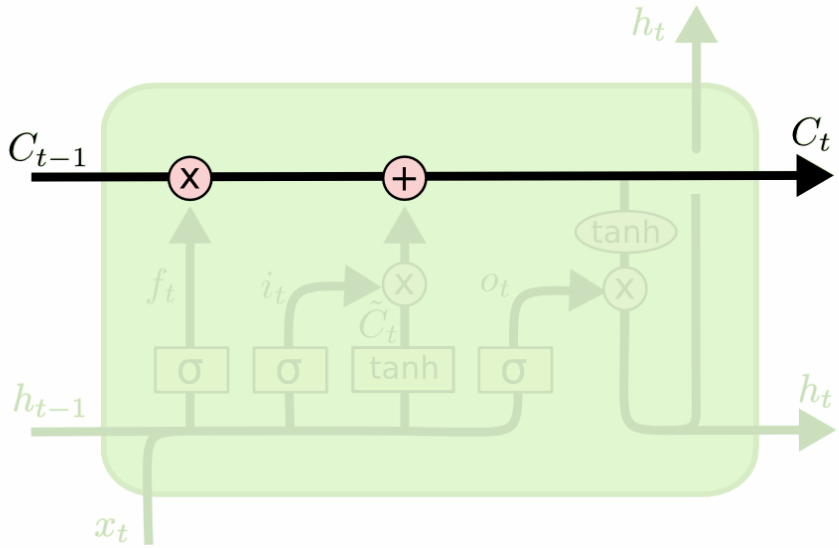
- Introduced to circumvent the vanishing gradient problem [Hochreiter and Schmidhuber, 1997].
- Architecture consists of:
  1. Network weights and activation functions.
  2. State cell.



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

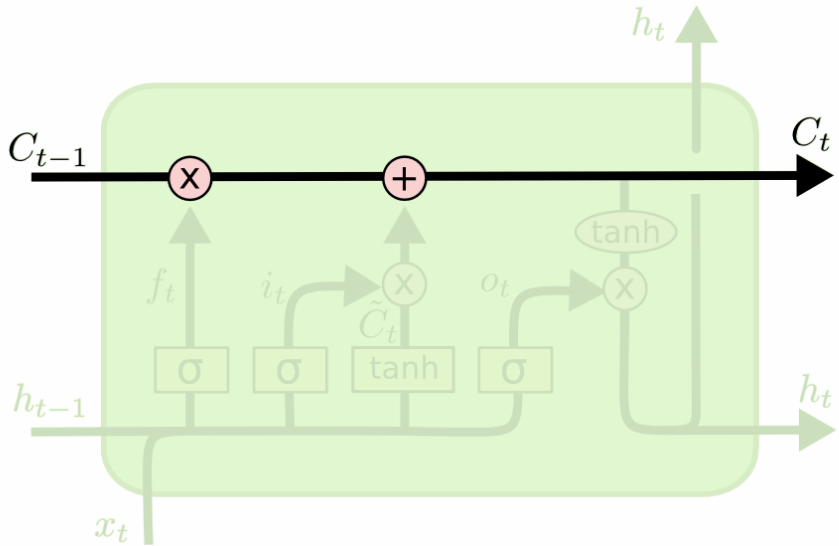
- Dichotomy in design can accomodate two-tier learning.



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

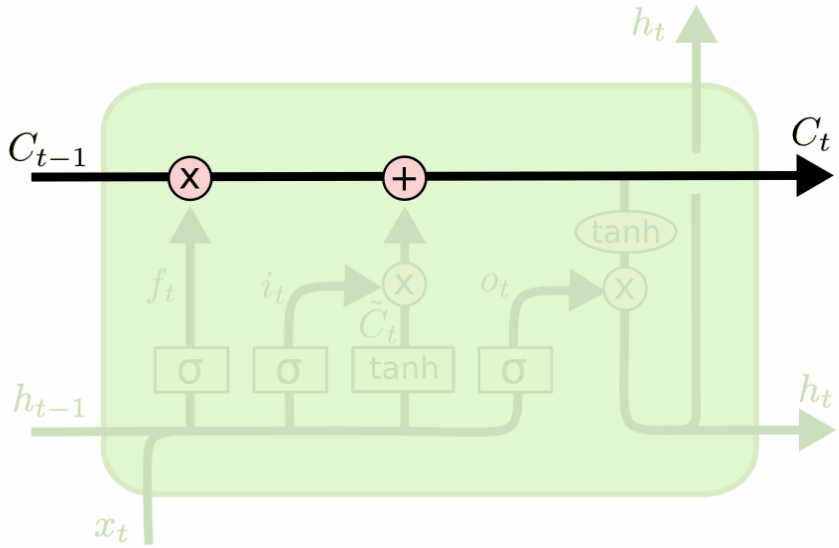
- Dichotomy in design can accomodate two-tier learning.
- Weights used to learn across datasets, memory cell used to cache representations.



Source: Olah, C., *Understanding LSTM Networks*

Long-short term memory

- Dichotomy in design can accomodate two-tier learning.
- Weights used to learn across datasets, memory cell used to cache representations.
- Learns never-before-seen quadratic functions with low number of data samples [Hochreiter et al., 2001].



Source: Olah, C., *Understanding LSTM Networks*

Limits of LSTMs

A scalable solution needs to meet several requirements:

Limits of LSTMs

A scalable solution needs to meet several requirements:

1. Stable memory.

Limits of LSTMs

A scalable solution needs to meet several requirements:

1. Stable memory.
2. Addressable content.

Limits of LSTMs

A scalable solution needs to meet several requirements:

1. Stable memory.
2. Addressable content.
3. No. of parameters independent of size of memory.

Limits of LSTMs

LSTMs don't satisfy these conditions:

Source: Graves, A., *IJCNN 2017 Plenary Talk: Frontiers in Recurrent Neural Network Research*

Limits of LSTMs

LSTMs don't satisfy these conditions:

1. In practice, hidden state $\mathbf{h_t}$ is modified at each time step.

Source: Graves, A., *IJCNN 2017 Plenary Talk: Frontiers in Recurrent Neural Network Research*

Limits of LSTMs

LSTMs don't satisfy these conditions:

1. In practice, hidden state $\mathbf{h_t}$ is modified at each time step.
2. Increasing the size of memory is equivalent to expanding the vector $\mathbf{h_t}$ and the whole network. No. of weights grows at least linearly with required memory.

Source: Graves, A., *IJCNN 2017 Plenary Talk: Frontiers in Recurrent Neural Network Research*

Limits of LSTMs

LSTMs don't satisfy these conditions:

1. In practice, hidden state $\mathbf{h_t}$ is modified at each time step.
2. Increasing the size of memory is equivalent to expanding the vector $\mathbf{h_t}$ and the whole network. No. of weights grows at least linearly with required memory.
3. Location and content are intertwined. Not easy to extract content.

Source: Graves, A., *IJCNN 2017 Plenary Talk: Frontiers in Recurrent Neural Network Research*

MANN

We could use memory-augmented neural networks (MANNs). One example would be a Neural Turing machine (NTM) / Differentiable neural computer (DNC) architecture:

MANN

We could use memory-augmented neural networks (MANNs). One example would be a Neural Turing machine (NTM) / Differentiable neural computer (DNC) architecture:

1. External memory matrix is relatively stable.

MANN

We could use memory-augmented neural networks (MANNs). One example would be a Neural Turing machine (NTM) / Differentiable neural computer (DNC) architecture:

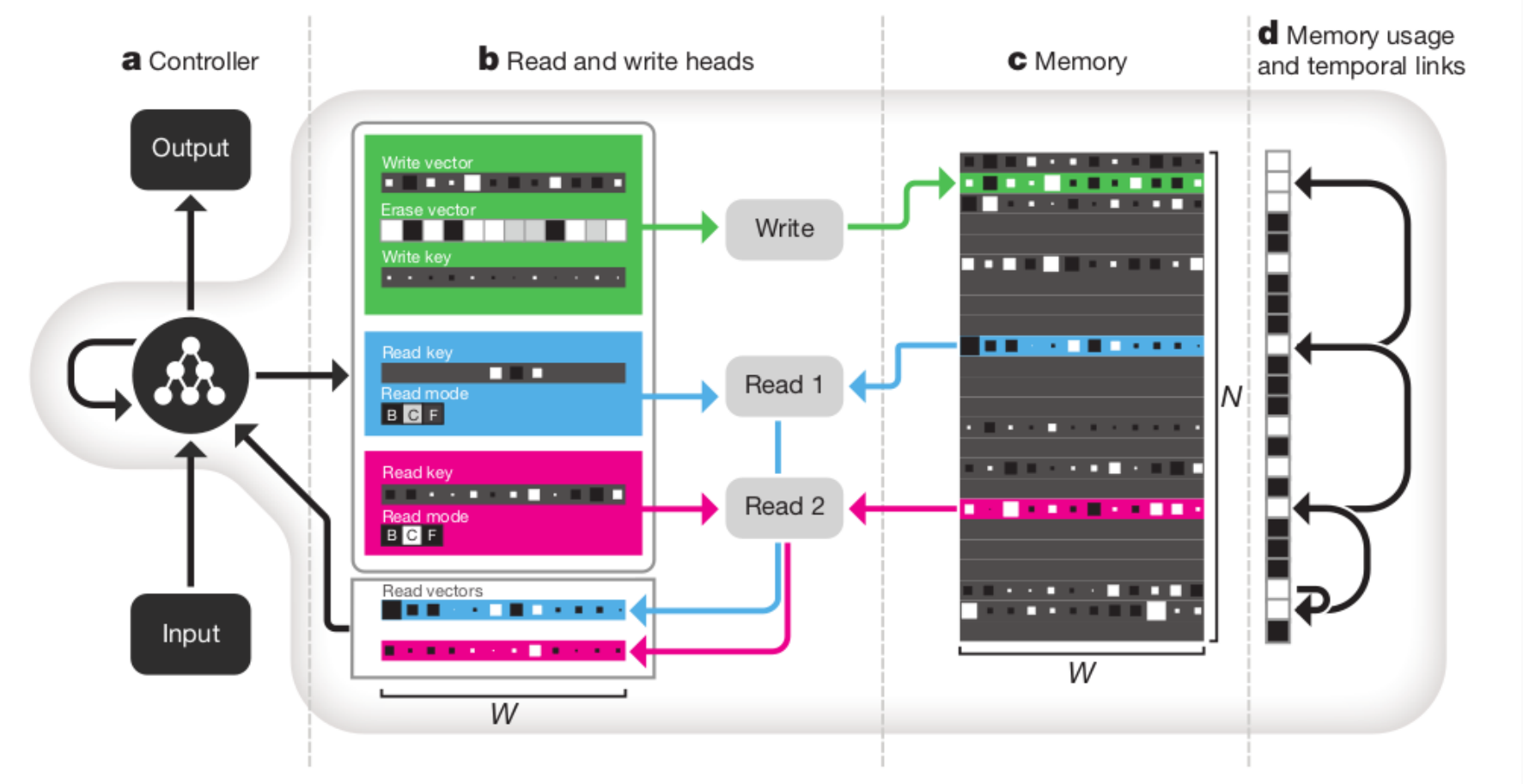
1. External memory matrix is relatively stable.
2. Size of memory not directly related to size of network.

MANN

We could use memory-augmented neural networks (MANNs). One example would be a Neural Turing machine (NTM) / Differentiable neural computer (DNC) architecture:

1. External memory matrix is relatively stable.
2. Size of memory not directly related to size of network.
3. Content-based and usage-based addressing.

Differentiable neural computer



Source: [Graves et al., 2016]

Differentiable neural computer

- Network architecture supports meta-learning.

Differentiable neural computer

- Network architecture supports meta-learning.
- Weights of the controller updated to learn structure across datasets.

Differentiable neural computer

- Network architecture supports meta-learning.
- Weights of the controller updated to learn structure across datasets.
- Input stored in external memory matrix, recalled to make dataset-specific predictions.

Differentiable neural computer

- Network architecture supports meta-learning.
- Weights of the controller updated to learn structure across datasets.
- Input stored in external memory matrix, recalled to make dataset-specific predictions.
- Weight updates allow us to extract representations of data, memory enables rapid binding of information.

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.
- Meta-learning approach: choose parameters θ^* to minimize expected cost $\mathcal{L}$ across a distribution of datasets $p(D)$:

$$\theta^* = \text{argmin}_\theta E_{D \sim p(D)} \left[\mathcal{L}(D; \theta) \right]$$

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.
- Meta-learning approach: choose parameters θ^* to minimize expected cost $\mathcal{L}$ across a distribution of datasets $p(D)$:

$$\theta^* = \text{argmin}_\theta E_{D \sim p(D)} \left[\mathcal{L}(D; \theta) \right]$$

- An episode is a presentation of dataset

$$D = \{d_t\}_{t=1}^T = \{(\mathbf{x_t}, y_t)\}_{t=1}^T$$

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.
- Meta-learning approach: choose parameters θ^* to minimize expected cost $\mathcal{L}$ across a distribution of datasets $p(D)$:

$$\theta^* = \text{argmin}_\theta E_{D \sim p(D)} \left[\mathcal{L}(D; \theta) \right]$$

- An episode is a presentation of dataset

$$D = \{d_t\}_{t=1}^T = \{(\mathbf{x_t}, y_t)\}_{t=1}^T$$

- For classification, $\mathbf{x_t}$ is the input data, y_t is the label.

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.
- Meta-learning approach: choose parameters θ^* to minimize expected cost $\mathcal{L}$ across a distribution of datasets $p(D)$:

$$\theta^* = \text{argmin}_\theta E_{D \sim p(D)} \left[\mathcal{L}(D; \theta) \right]$$

- An episode is a presentation of dataset

$$D = \{d_t\}_{t=1}^{T} = \{(\mathbf{x_t}, y_t)\}_{t=1}^{T}$$

- For classification, $\mathbf{x_t}$ is the input data, y_t is the label.
- Data is presented to the network as follows:

$$(\mathbf{x_1}, \text{null}), (\mathbf{x_2}, y_1), \ldots, (\mathbf{x_T}, y_{T-1})$$

Meta-learning setup

- Traditional approach: choose parameters θ to minimize cost $\mathcal{L}$ on dataset D.
- Meta-learning approach: choose parameters θ^* to minimize expected cost $\mathcal{L}$ across a distribution of datasets $p(D)$:

$$\theta^* = \text{argmin}_\theta E_{D \sim p(D)} \left[\mathcal{L}(D; \theta) \right]$$

- An episode is a presentation of dataset

$$D = \{d_t\}_{t=1}^T = \{(\mathbf{x_t}, y_t)\}_{t=1}^T$$

- For classification, $\mathbf{x_t}$ is the input data, y_t is the label.
- Data is presented to the network as follows:

$$(\mathbf{x_1}, \text{null}), (\mathbf{x_2}, y_1), \ldots, (\mathbf{x_T}, y_{T-1})$$

- At time t the correct label for the previous sample y_{t-1} is provided along with a new query $\mathbf{x_t}$.

Meta-learning setup

- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.

Meta-learning setup

- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.
- Labels shuffled from dataset to dataset.

Meta-learning setup

- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.
- Labels shuffled from dataset to dataset.
- Network has to store representations in memory until class labels are presented, bind them and store for later use.

Meta-learning setup

- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.
- Labels shuffled from dataset to dataset.
- Network has to store representations in memory until class labels are presented, bind them and store for later use.
- Ideal performance: guess for first-seen class, use of memory to perfectly classify this class going forward.

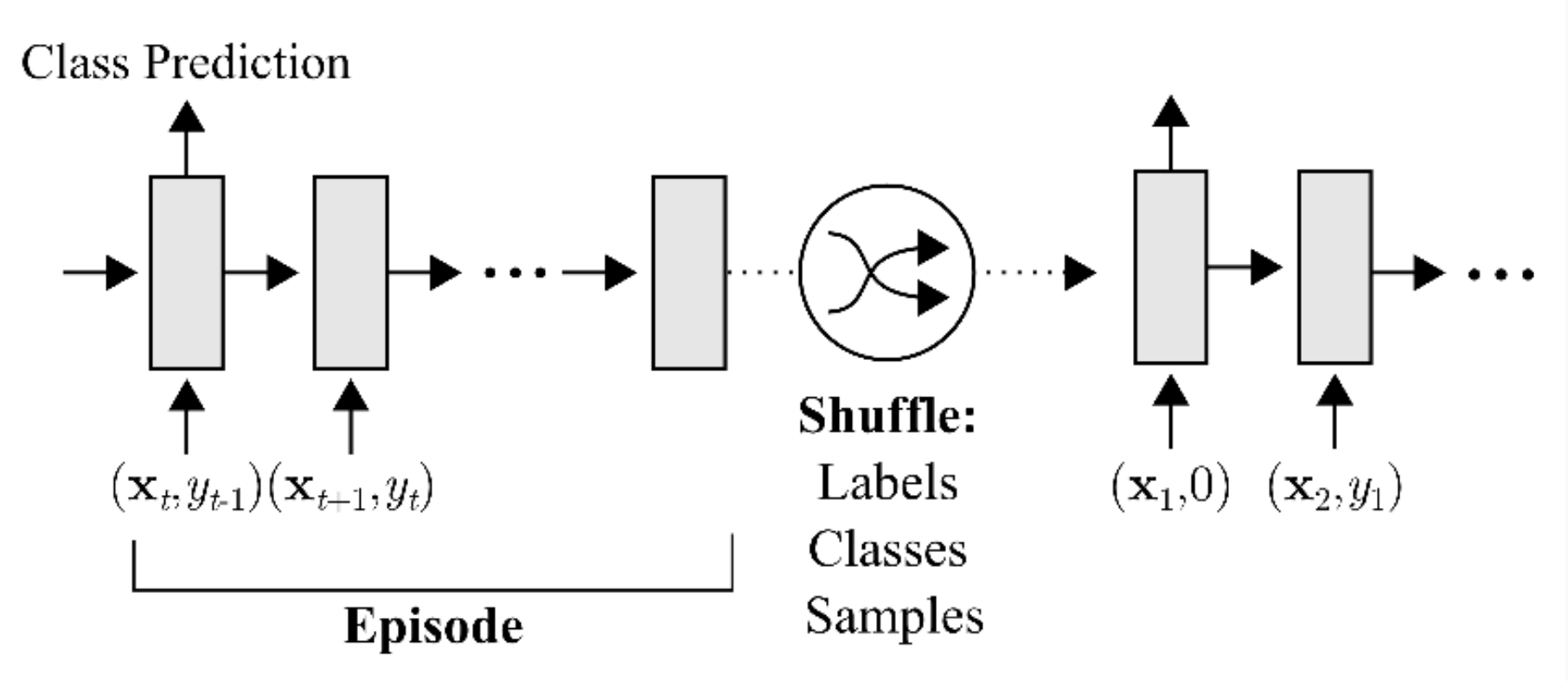
Meta-learning setup

- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.
- Labels shuffled from dataset to dataset.
- Network has to store representations in memory until class labels are presented, bind them and store for later use.
- Ideal performance: guess for first-seen class, use of memory to perfectly classify this class going forward.
- System models the predictive distribution $p\left(y_t|\mathbf{x_t}, D_{1:t-1}; \theta\right)$.

Meta-learning setup

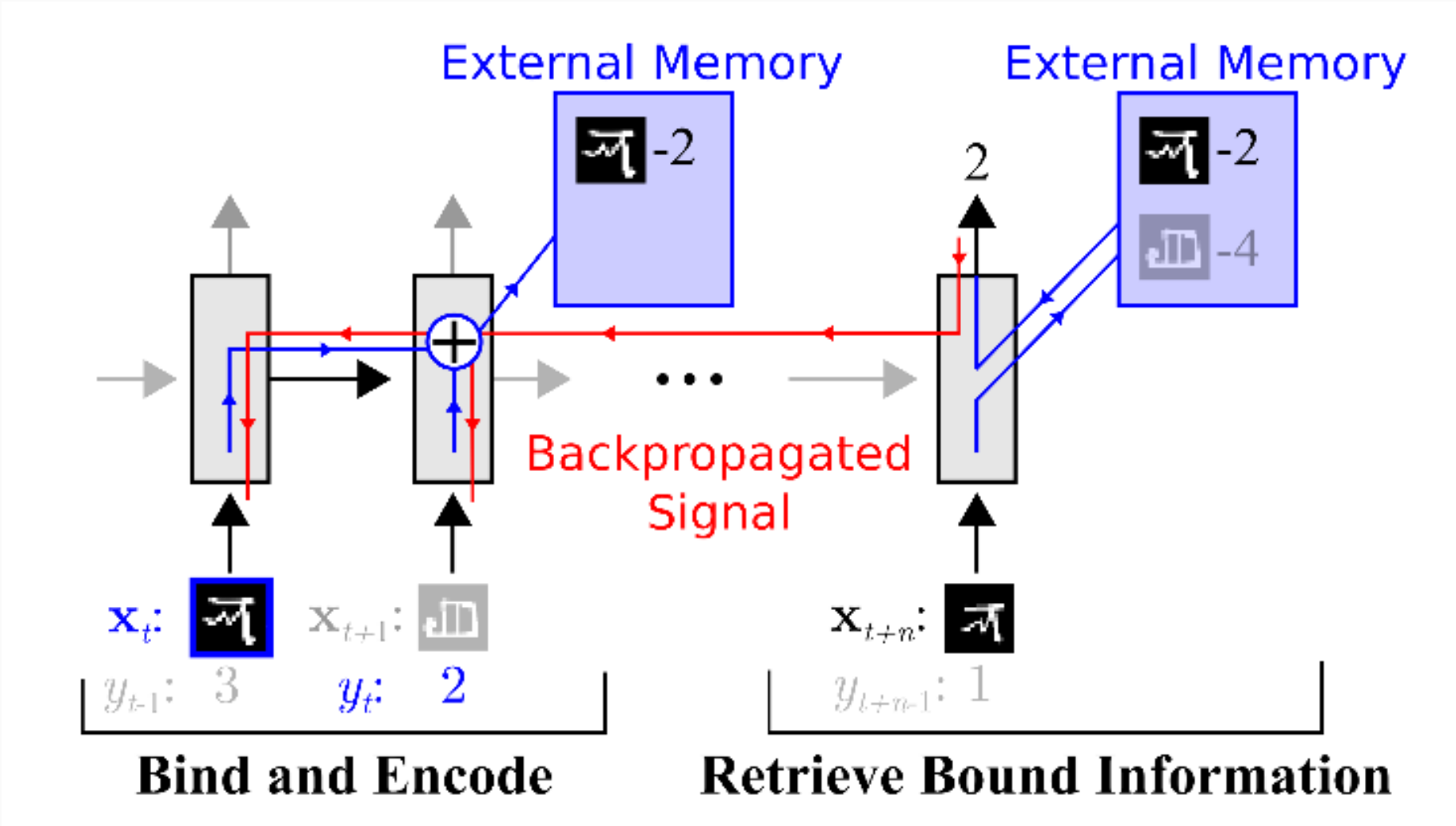
- At time t the network is asked to output label y_t for query $\mathbf{x_t}$.
- Labels shuffled from dataset to dataset.
- Network has to store representations in memory until class labels are presented, bind them and store for later use.
- Ideal performance: guess for first-seen class, use of memory to perfectly classify this class going forward.
- System models the predictive distribution $p\left(y_t|\mathbf{x_t}, D_{1:t-1}; \theta\right)$.
- There is exploitable structure: a meta-learning model would learn to bind input to appropriate class regardless of particular input data or label.

Meta-learning setup



Source: [Santoro et al., 2016]

Meta-learning setup



Source: [Santoro et al., 2016]

Dataset

Omniglot dataset:

- Image classification dataset.

Dataset

Omniglot dataset:

- Image classification dataset.
- 1,623 classes.

Dataset

Omniglot dataset:

- Image classification dataset.
- 1,623 classes.
- Few examples per class.

Dataset

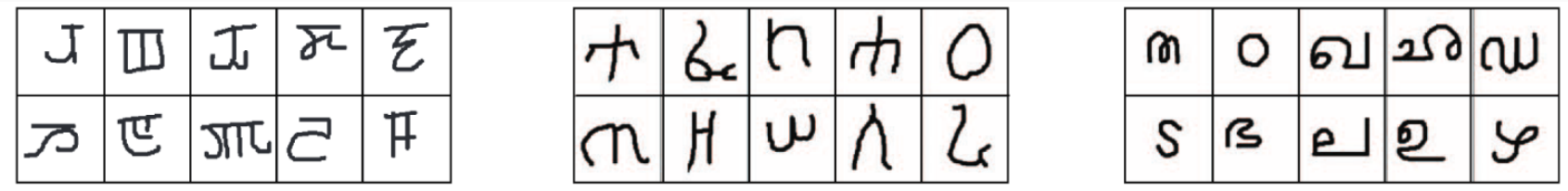
Omniglot dataset:

- Image classification dataset.
- 1,623 classes.
- Few examples per class.
- "Transpose of MNIST."

Dataset

Omniglot dataset:

- Image classification dataset.
- 1,623 classes.
- Few examples per class.
- "Transpose of MNIST."



Source: [Lake et al., 2015]

Experimental setup

- DNC/NTM parametrized by θ.

Experimental setup

- DNC/NTM parametrized by θ.
- Choose parameters θ^* to minimize expected cost $\mathcal{L}$ across samples from the Omniglot dataset.

Experimental setup

- DNC/NTM parametrized by θ.
- Choose parameters θ^* to minimize expected cost $\mathcal{L}$ across samples from the Omniglot dataset.
- For classification, $\mathbf{x_t}$ is the raw pixel input, y_t is the label.

Experimental setup

- DNC/NTM parametrized by θ.
- Choose parameters θ^* to minimize expected cost $\mathcal{L}$ across samples from the Omniglot dataset.
- For classification, $\mathbf{x_t}$ is the raw pixel input, y_t is the label.
- Data is presented to the network as follows:

$$(\mathbf{x_1}, \text{null}), (\mathbf{x_2}, y_1), \ldots, (\mathbf{x_T}, y_{T-1})$$

Experimental setup

- DNC/NTM parametrized by θ.
- Choose parameters θ^* to minimize expected cost $\mathcal{L}$ across samples from the Omniglot dataset.
- For classification, $\mathbf{x_t}$ is the raw pixel input, y_t is the label.
- Data is presented to the network as follows:

$$(\mathbf{x_1}, \text{null}), (\mathbf{x_2}, y_1), \ldots, (\mathbf{x_T}, y_{T-1})$$

- Network output is a softmax layer producing $\mathbf{p_t}$ with elements:

$$p_t(i) = \frac{\exp\left(\mathbf{W}^{op}(i)\mathbf{o_t}\right)}{\sum_j \exp\left(\mathbf{W}^{op}(j)\mathbf{o_t}\right)}$$

Experimental setup

- DNC/NTM parametrized by θ.
- Choose parameters θ^* to minimize expected cost $\mathcal{L}$ across samples from the Omniglot dataset.
- For classification, $\mathbf{x_t}$ is the raw pixel input, y_t is the label.
- Data is presented to the network as follows:

$$(\mathbf{x_1}, \text{null}), (\mathbf{x_2}, y_1), \ldots, (\mathbf{x_T}, y_{T-1})$$

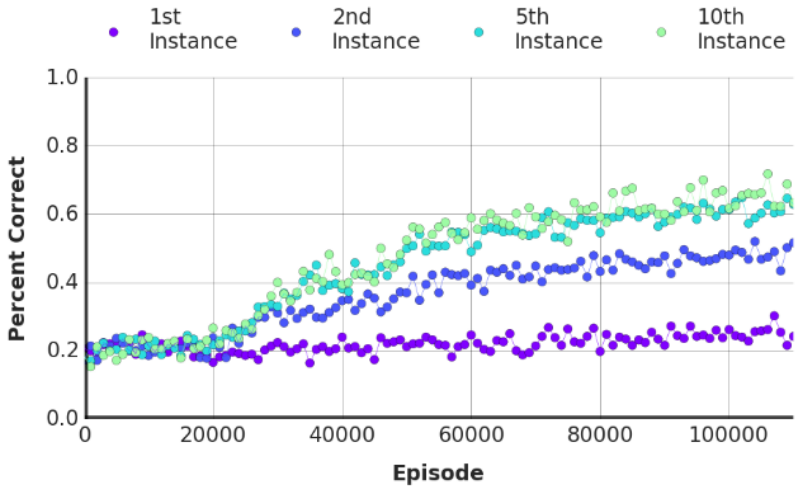
- Network output is a softmax layer producing $\mathbf{p_t}$ with elements:

$$p_t(i) = \frac{\exp\left(\mathbf{W}^{op}(i)\mathbf{o_t}\right)}{\sum_j \exp\left(\mathbf{W}^{op}(j)\mathbf{o_t}\right)}$$

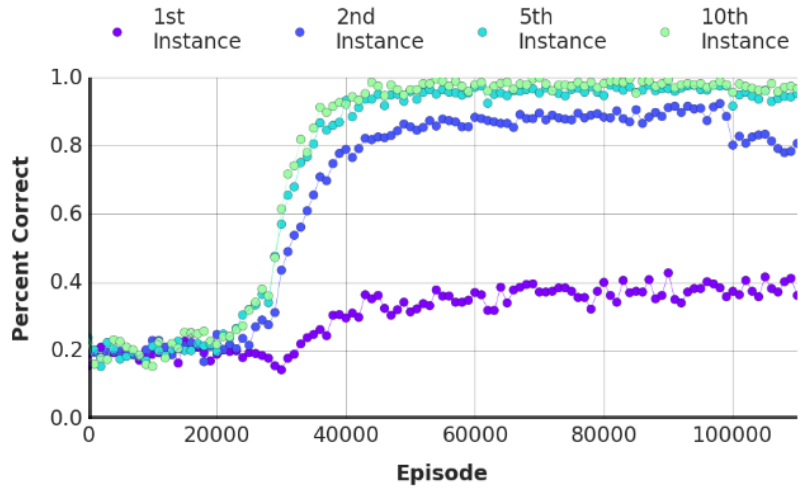
- For one-hot labels, episode loss is

$$\mathcal{L}(\theta) = -\sum_t \mathbf{y_t^T} \log \mathbf{p_t}$$

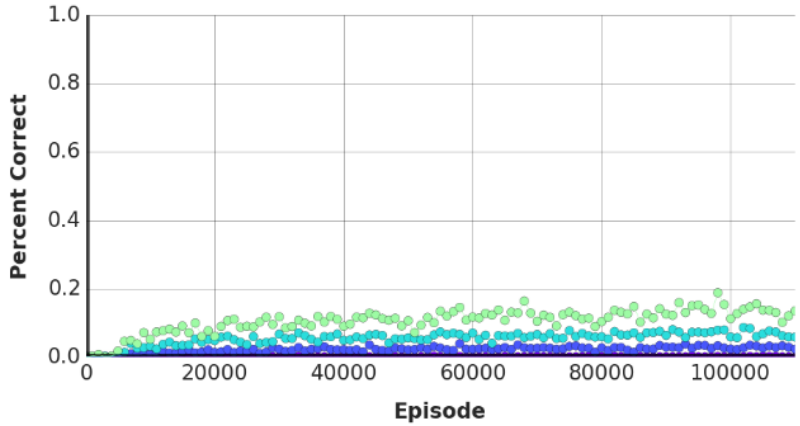
Experimental results



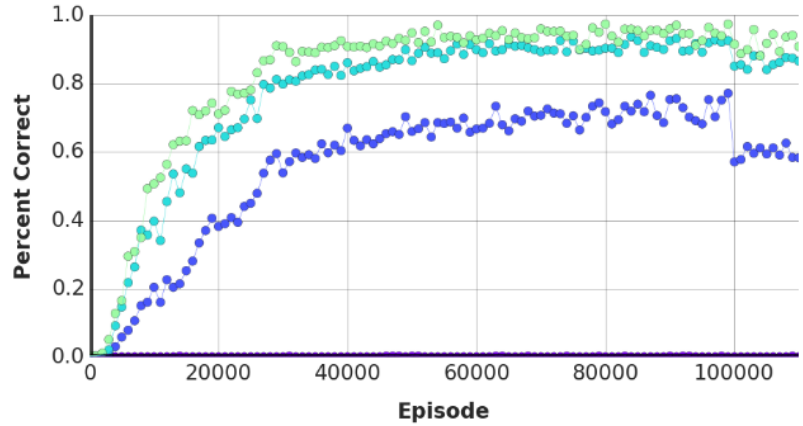
(a) LSTM, five random classes/episode, one-hot vector labels



(b) MANN, five random classes/episode, one-hot vector labels



(c) LSTM, fifteen classes/episode, five-character string labels



(d) MANN, fifteen classes/episode, five-character string labels

Source: [Santoro et al., 2016]

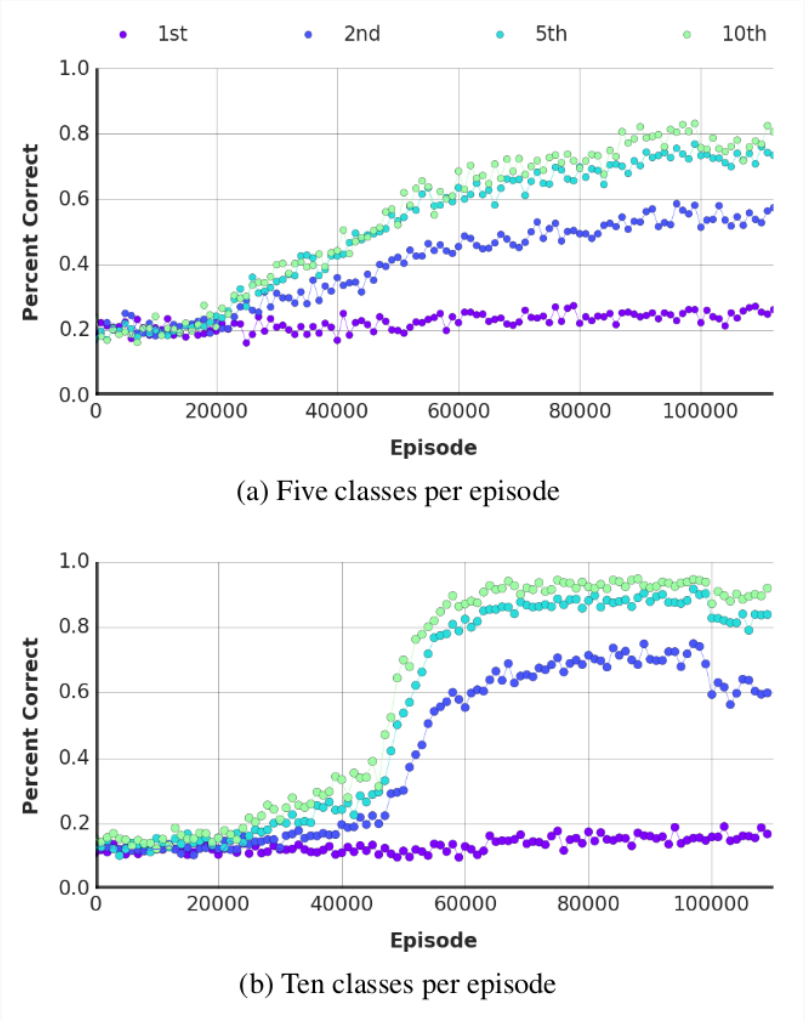
Experimental results

Model	Instance (% Correct)					
	1st	2nd	3rd	4th	5th	10th
Human	34.5	57.3	70.1	71.8	81.4	92.4
Feedforward	24.4	19.6	21.1	19.9	22.8	19.5
LSTM	24.4	49.5	55.3	61.0	63.6	62.5
MANN	**36.4**	**82.8**	**91.0**	**92.6**	**94.9**	**98.1**

Source: [Santoro et al., 2016]

Experimental results

- Persistent memory interference.



(a) Five classes per episode

(b) Ten classes per episode

Source: [Santoro et al., 2016]

Experimental results

Model	Controller	# of Classes	Instance (% Correct)					
			1st	2nd	3rd	4th	5th	10th
kNN (raw pixels)	–	5	4.0	36.7	41.9	45.7	48.1	57.0
kNN (deep features)	–	5	4.0	51.9	61.0	66.3	69.3	77.5
Feedforward	–	5	0.0	0.2	0.0	0.2	0.0	0.0
LSTM	–	5	0.0	9.0	14.2	16.9	21.8	25.5
MANN	Feedforward	5	0.0	8.0	16.2	25.2	30.9	46.8
MANN	LSTM	5	0.0	**69.5**	**80.4**	**87.9**	**88.4**	**93.1**
kNN (raw pixels)	–	15	0.5	18.7	23.3	26.5	29.1	37.0
kNN (deep features)	–	15	0.4	32.7	41.2	47.1	50.6	60.0
Feedforward	–	15	0.0	0.1	0.0	0.0	0.0	0.0
LSTM	–	15	0.0	2.2	2.9	4.3	5.6	12.7
MANN (LRUA)	Feedforward	15	0.1	12.8	22.3	28.8	32.2	43.4
MANN (LRUA)	LSTM	15	0.1	**62.6**	**79.3**	**86.6**	**88.7**	**95.3**
MANN (NTM)	LSTM	15	0.0	35.4	61.2	71.7	77.7	88.4

Source: [Santoro et al., 2016]

Experimental results

- It is possible to learn from very few instances.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.
- Weakly inspired by how humans approach such a task.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.
- Weakly inspired by how humans approach such a task.
- Very narrow problem.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.
- Weakly inspired by how humans approach such a task.
- Very narrow problem.
- Structured input, temporal offset.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.
- Weakly inspired by how humans approach such a task.
- Very narrow problem.
- Structured input, temporal offset.
- Memory interference.

Experimental results

- It is possible to learn from very few instances.
- Meta-learning can extract relevant task structure.
- DNC/NTMs learn quicker than LSTMs.
- Another type of problem where DNCs are advantageous.
- Weakly inspired by how humans approach such a task.
- Very narrow problem.
- Structured input, temporal offset.
- Memory interference.
- Specific architecture.

Future work

- Meta-learning to find a suitable memory-addressing procedure.

Future work

- Meta-learning to find a suitable memory-addressing procedure.
- Learning across tasks, not different samples from one task.

Future work

- Meta-learning to find a suitable memory-addressing procedure.
- Learning across tasks, not different samples from one task.
- Active learning.

Future work

- Meta-learning to find a suitable memory-addressing procedure.
- Learning across tasks, not different samples from one task.
- Active learning.
- Attention mechanisms.

Graves, A., Wayne, G., et al. (2016).
Hybrid computing using a neural network with dynamic external memory.
Nature, 538:471–476.

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural Computation, 9(8):1735–1780.

Hochreiter, S., Younger, A. S., and Conwell, P. R. (2001).
Learning to learn using gradient descent.
In Dorffner, G., Bischof, H., and Hornik, K., editors, *Artificial Neural Networks - ICANN 2001, International Conference Vienna, Austria, August 21-25, 2001 Proceedings*, pages 87–94.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015).

Human-level concept learning through probabilistic program induction.

Science, 350:1332–1338.

 Santoro, A., Bartunov, S., et al. (2016).

One-shot learning with memory-augmented neural networks.

arXiv.

 Schmidhuber, J., Zhao, J., and Wiering, M. (1997).

Shifting inductive bias with success-story algo-rithm, adaptive levin search, and incremental self-improvement.

Machine Learning, 28(1):105–130.