# 6. One-shot Learning with Memory-Augmented Neural Networks

논문 : https://arxiv.org/pdf/1605.06065.pdf

요약

- Meta-Learning에서 Memory based 기법을 제안한 논문 (Memory based meta learning에 기조가 되는 기법)
- Neural Tuning Network( 2014년도 논문 )와 같이 augmented memory를 사용해서 class representation을 저장/활용함
    - augmented memory에 저장할 class representation을 잘 generalization할 수 있도록 하는 것이 논문의 목표
    - 간단히 LSTM + External Memory로 표현
- model architecture 구성 및 용도
    - LSTM같은 RNN
        - 느린 weight 업데이트를 통한 long-term storage
    - External Memory
        - 에피소드의 데이터 입력을 저장하기 위한 short-term storage

---

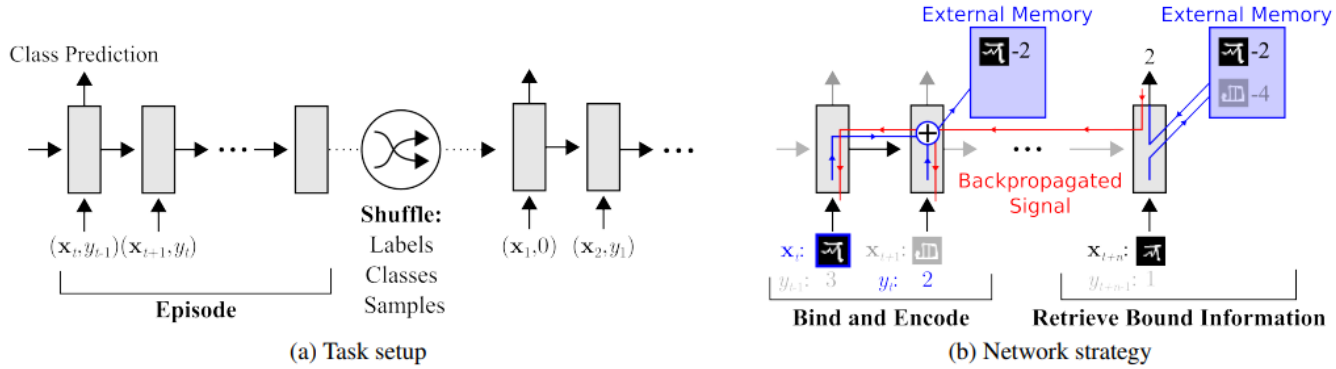



*Figure 1.* Task structure. (a) Omniglot images (or $x$-values for regression), $\mathbf{x}_t$, are presented with time-offset labels (or function values), $y_{t-1}$, to prevent the network from simply mapping the class labels to the output. From episode to episode, the classes to be presented in the episode, their associated labels, and the specific samples are all shuffled. (b) A successful strategy would involve the use of an external memory to store bound sample representation-class label information, which can then be retrieved at a later point for successful classification when a sample from an already-seen class is presented. Specifically, sample data $x_t$ from a particular time step should be bound to the appropriate class label $y_t$, which is presented in the subsequent time step. Later, when a sample from this same class is seen, it should retrieve this bound information from the external memory to make a prediction. Backpropagated error signals from this prediction step will then shape the weight updates from the earlier steps in order to promote this binding strategy.

## 모델 구조 (그림 7)

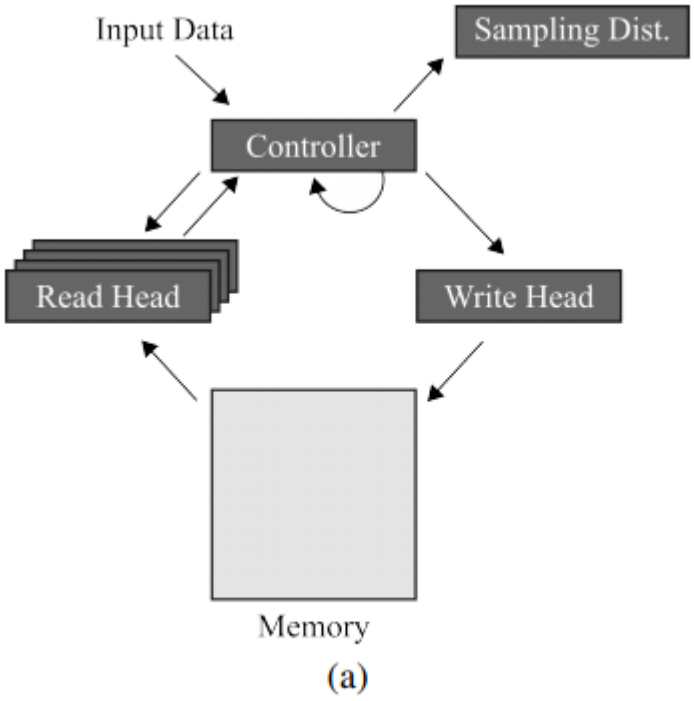
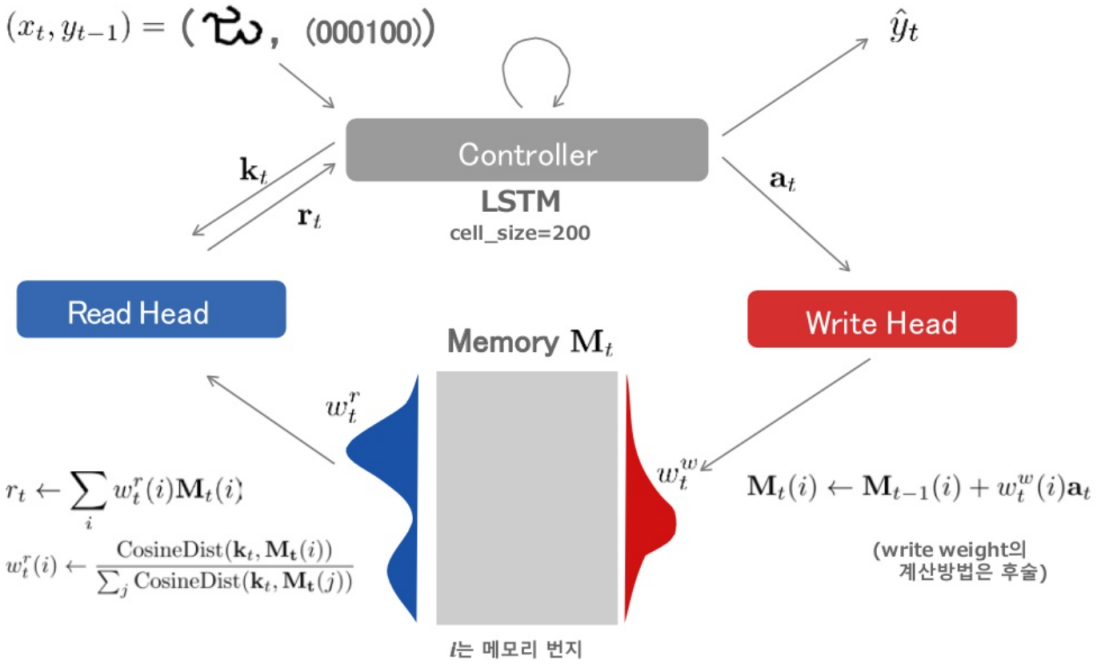



*Figure 7.* MANN Architecture.

## MANN의 구조 : 개념도




- Controller
  - LSTM과 같은 Recurrent model
  - 시간 t에서 들어온 입력 $\mathbf{x}_t$, hidden state $\mathbf{h}_{t-1}$, 시간 t-1에서 예측한 정답 $\mathbf{y}_{t-1}$을 입력으로 key $\mathbf{k}_t$를 연산
    - 학습 단계에서는 $\mathbf{y}_{t-1}$ 값을 teacher-student learning처럼 실제 정답을 사용
    - $\mathbf{k}_t$는 시간 t에 External Memory $\mathbf{M}_t$의 정보를 읽기/쓰기위해 사용됨

- **External memory 읽기** (식 2, 3, 4) : $\mathbf{k}_t$와 External Memory에 저장된 i번째 행 벡터 $\mathbf{M}_t(i)$와 코사인 유사도를 구해 attention machanism을 적용

- i의 개수 : 시간 t에서의 class num
- 식 2 : key에 대한 class 별 cosine similarity를 구하는 식
- 식 3 : softmax로 class 별 element-wise multiple 비율을 구하는 식
- 식 4 : weighted-sum으로 시간 t에서의 memory $\mathbf{r}_t$ 를 구하는 식
    - $\mathbf{r}_t$ : inference에서 controller의 최종 output
    - softmax output layer / dense layer로 classification에 사용 / 시간 t+1에서의 hidden state로 사용

$$K\big(\mathbf{k}_t, \mathbf{M}_t(i)\big) = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\| \mathbf{k}_t \| \| \mathbf{M}_t(i) \|}, \qquad (2)$$

$$w_t^r(i) \leftarrow \frac{\exp\big(K\big(\mathbf{k}_t, \mathbf{M}_t(i)\big)\big)}{\sum_j \exp\big(K\big(\mathbf{k}_t, \mathbf{M}_t(j)\big)\big)}. \qquad (3)$$

$$\mathbf{r}_t \leftarrow \sum_i w_t^r(i)\mathbf{M}_t(i). \qquad (4)$$

- **External memory 쓰기 : Least Recently Used Access (LRUA)**
    - Recurrent model에서 사용하는 attention machanism의 경우 입력이 sequencial 하기 때문에 입력 순서가 attention 계산에 영향이 있음
        - 이 논문이 모델을 학습할 때는 순서가 의미없이 랜덤으로 섞임
            - ex) 동일한 이미지 5장의 입력 순서가 바뀌면 학습에 영향을 줌
    - 논문은 Least Recently Used Access를 제안
        - 최근에 가장 적게 사용된 위치(class representation)와 최근에 사용한 위치 중 하나를 External memory에 write
            - rarely-used location : '얼만큼 잘 안써졌는가?' 를 계산하기 위해 식 (6) 사용
            - last used location : 식 (3)의 비율로 확인
        - 식 5 : External Memory의 행 별 사용이력을 저장하기위해 시간 t에서의 weights scaled according to usage weights $\mathbf{w}_t^u$ 를 계산
            - $\mathbf{w}_t^r$ : 식 (3)에서의 class 별 read weight
            - $\mathbf{w}_t^w$ : 식 (7)에서의 class 별 write weight
            - 사용 이력을 저장함으로써 얼마나 자주 사용됐는지 덜사용됐는지 계산하기 위함
        - 식 6 : 시간 t에서의 External memory에 대한 least-used weight $\mathbf{w}_t^{lu}$ 계산식
            - $m(\mathbf{w}_t^u, n)$ : 식 (5)로 연산한 $\mathbf{w}_t^u$ vector에서 n번째로 작은 요소
            - $\mathbf{w}_t^u$ 의 i번째 요소 값인 $w_t^u(i)$ 이 $m(\mathbf{w}_t^u, n)$ 보다 작다면?
                - 새로운 값으로 갱신할 수 있도록 1
                - 아니면 기존값을 유지하면서 갱신할 수 있도록 0
        - 식 7 : 시간 t에서의 write weight $\mathbf{w}_t^w$ 연산식
            - $\sigma$ : sigmoid function
            - $\alpha$ : scalar gate parameter
            - 시간 t-1에서의 read weight와 least-used weight를 사용해서 계산함
            - 이는 그림 1에서 보듯 정답 y의 값을 t+1시점에 넘겨주기 때문으로 추정
        - 식 8 : 시간 t에서의 write weight로 External memory의 각 행의 백터 갱신 $\mathbf{M}_t(i)$
            - 시간 t에서의 key 백터를 write weight 만큼 Externel memory의 행별로 갱신

$$\mathbf{w}_t^u \leftarrow \gamma \mathbf{w}_{t-1}^u + \mathbf{w}_t^r + \mathbf{w}_t^w. \qquad (5)$$

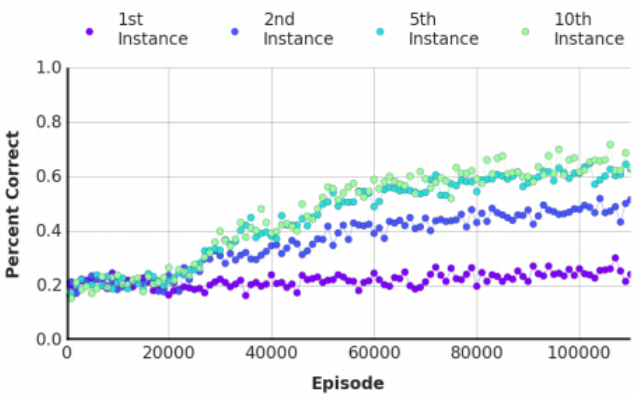
$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n) \end{cases}, \qquad (6)$$

$$\mathbf{w}_t^w \leftarrow \sigma(\alpha)\mathbf{w}_{t-1}^r + (1 - \sigma(\alpha))\mathbf{w}_{t-1}^{lu}. \qquad (7)$$

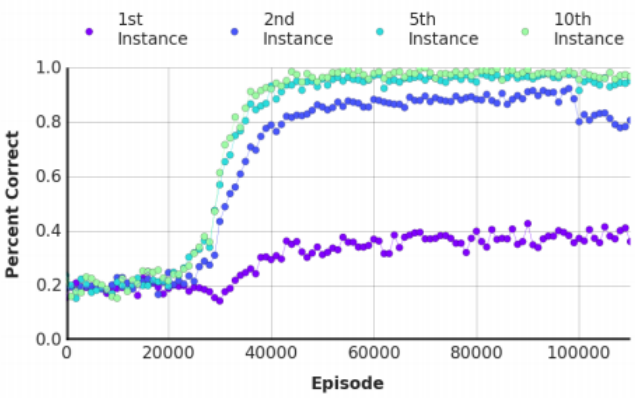
$$\mathbf{M}_t(i) \leftarrow \mathbf{M}_{t-1}(i) + w_t^w(i)\mathbf{k}_t, \forall i \qquad (8)$$

## 실험

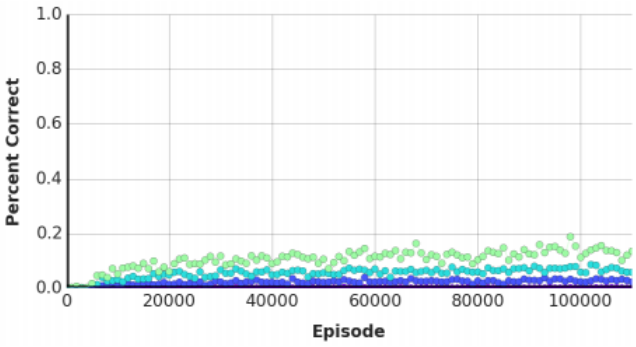
- 기존기법들과 성능비교
  - instance = N-shot learning을 표현
    - ex) 1st instance = 1-shot learning
  - 사용 데이터셋
    - Omniglot (classification 용)
    - Gaussian process에서 샘플링한 데이터 (regression 용)



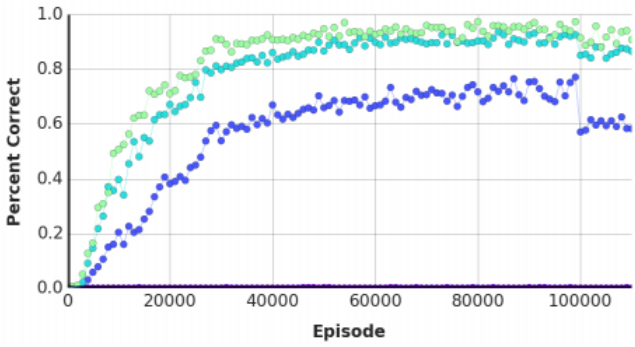

(a) LSTM, five random classes/episode, one-hot vector labels




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




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

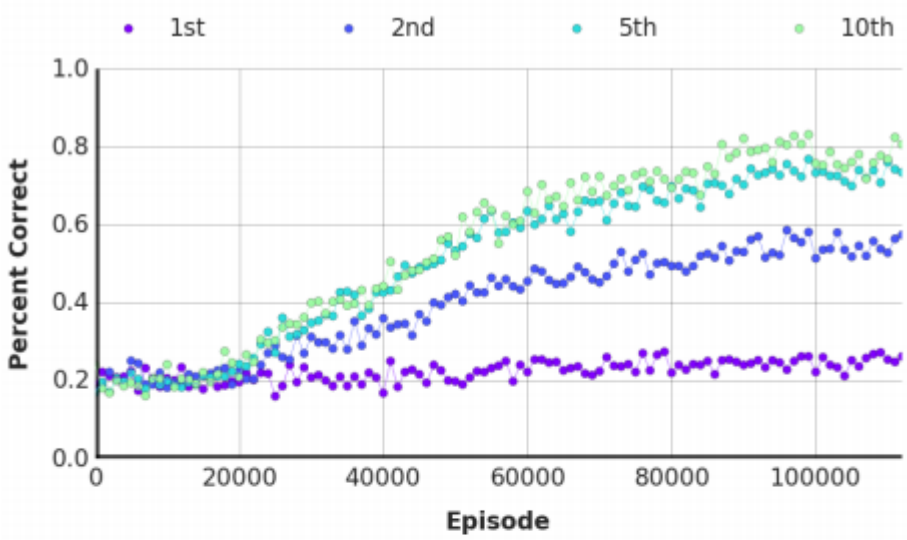



(d) MANN, fifteen classes/episode, five-character string labels

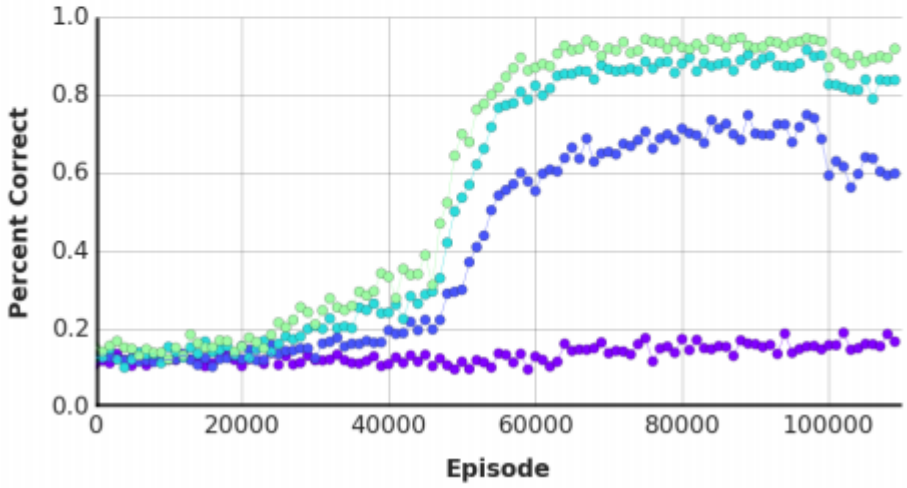
*Figure 2.* Omniglot classification. The network was given either five (a-b) or up to fifteen (c-d) random classes per episode, which were of length 50 or 100 respectively. Labels were one-hot vectors in (a-b), and five-character strings in (c-d). In (b), first instance accuracy is above chance, indicating that the MANN is performing "educated guesses" for new classes based on the classes it has already seen and stored in memory. In (c-d), first instance accuracy is poor, as is expected, since it must make a guess from 3125 random strings. Second instance accuracy, however, approaches 80% during training for the MANN (d). At the 100,000 episode mark the network was tested, without further learning, on distinct classes withheld from the training set, and exhibited comparable performance.

*Table 1.* Test-set classification accuracies for humans compared to machine algorithms trained on the Omniglot dataset, using one-hot encodings of labels and five classes presented per episode.

| 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** |




(a) Five classes per episode




(b) Ten classes per episode