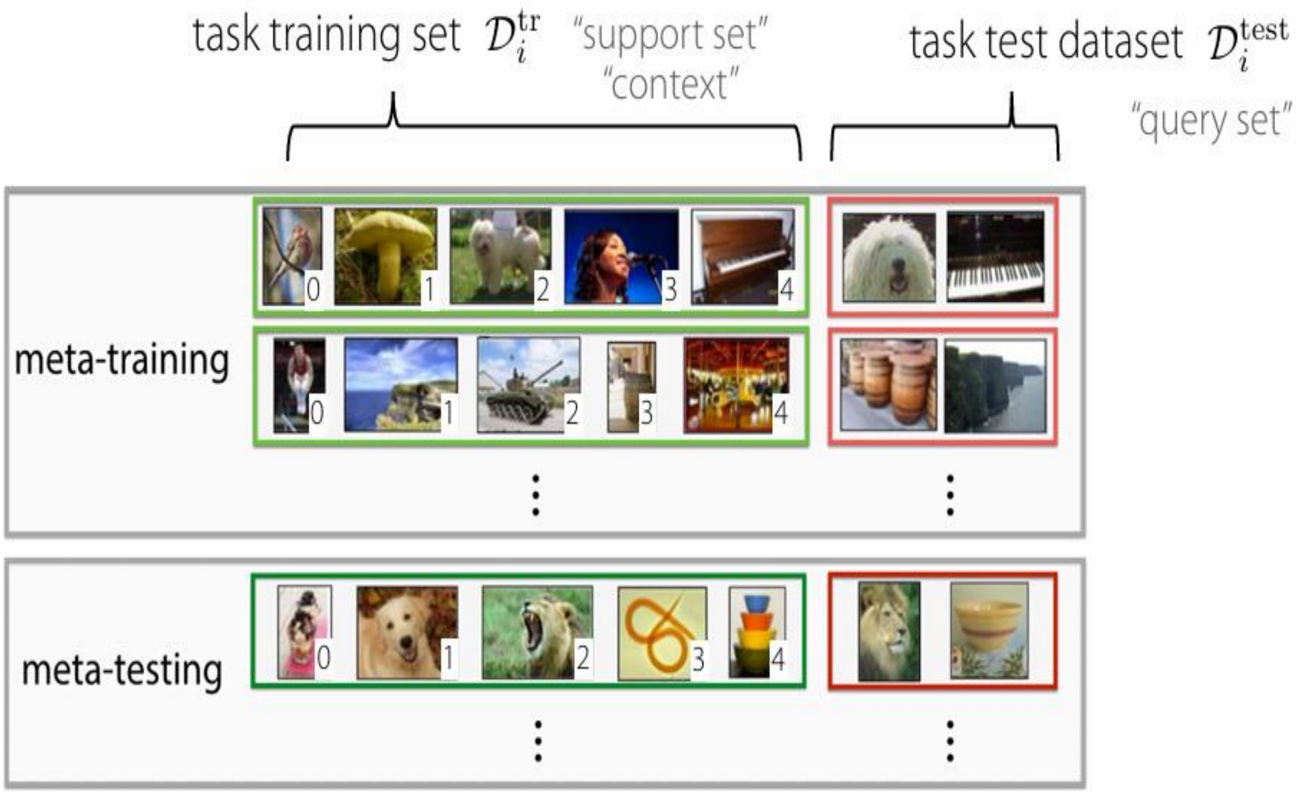
Meta Learning

Meta-Learning Problem

Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

Some terminology



k-shot learning: learning with **k** examples per class
(or **k** examples total for regression)

N-way classification: choosing between N classes

Two ways to view meta-learning algorithms

Mechanistic view

- Deep network that can read in an entire dataset and make predictions for new datapoints
- Training this network uses a meta-dataset, which itself consists of many datasets, each for a different task

Probabilistic view

- Extract shared prior knowledge from a set of tasks that allows efficient learning of new tasks
- Learning a new task uses this prior and (small) training set to infer most likely posterior parameters

Probabilistic View

learn *meta-parameters* θ: $p(\theta|\mathcal{D}_{\text{meta-train}})$

whatever we need to know about $\mathcal{D}_{\text{meta-train}}$ to solve new tasks

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \ldots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

$$\mathcal{D}_i^{\text{tr}} = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\}$$

meta-learning: $\theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$

adaptation: $\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}^{\text{tr}}, \theta^\star)$

$\Updownarrow$

$$\phi^\star = f_{\theta^\star}(\mathcal{D}^{\text{tr}})$$

$$\phi^\star = f_{\theta^\star}(\mathcal{D}^{\text{tr}})$$

meta-learning: $\theta^\star = \max_\theta \sum_{i=1}^{n} \log p(\phi_i|\mathcal{D}_i^{\text{ts}})$

where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

Probabilistic View

How to *design* a meta-learning algorithm

1. Choose a form of $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$

2. Choose how to optimize θ w.r.t. max-likelihood objective using $\mathcal{D}_{\text{meta-train}}$

Mechanistic View

Supervised Learning:

Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$ Data: $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})_i\}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Meta-Supervised Learning:

Inputs: $\underbrace{\mathcal{D}^{\mathrm{tr}}}_{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}$ $\mathbf{x}_{\mathrm{test}}$ Outputs: $\mathbf{y}_{\mathrm{test}}$ Data: $\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_i\}$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

$$\mathbf{y}_{\mathrm{test}} = f(\mathcal{D}^{\mathrm{tr}}, \mathbf{x}_{\mathrm{test}}; \theta)$$

Mechanistic View

How to design a meta-learning algorithm

1. Choose a form of $f_\theta(\mathcal{D}^{\mathrm{tr}}, \mathbf{x}^{\mathrm{ts}})$

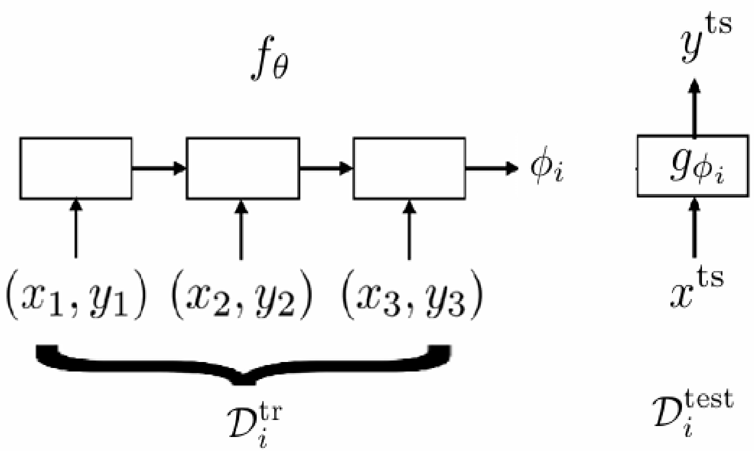
2. Choose how to optimize θ w.r.t. max-likelihood objective using meta-training data

meta-parameters

Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\mathrm{tr}})$ "learner"

Predict test points with $\mathbf{y}^{\mathrm{ts}} = g_{\phi_i}(\mathbf{x}^{\mathrm{ts}})$



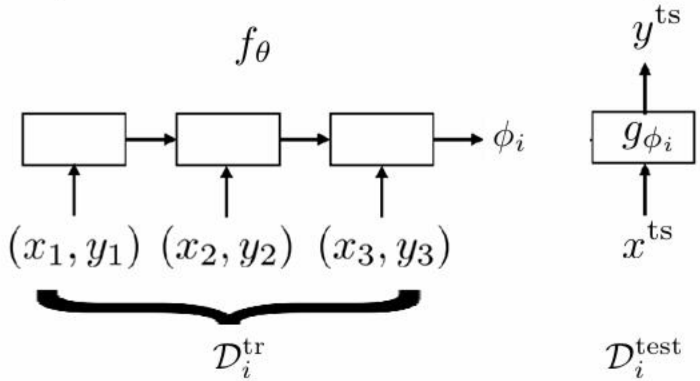
Train with standard supervised learning!

$$\min_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\mathrm{test}}} - \log g_{\phi_i}(y \mid x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})}$$

$$\min_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\mathrm{tr}}), \mathcal{D}_i^{\mathrm{ts}})$$

Black-Box Adaptation

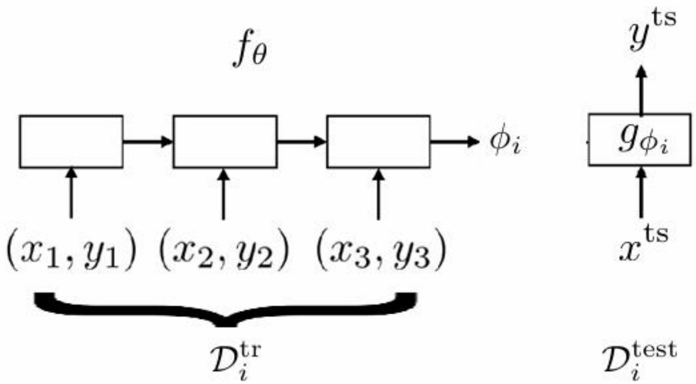
Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\mathrm{tr}})$.



1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$

Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$.



1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

$\mathcal{D}_i^{\text{tr}}$ $\mathcal{D}_i^{\text{test}}$

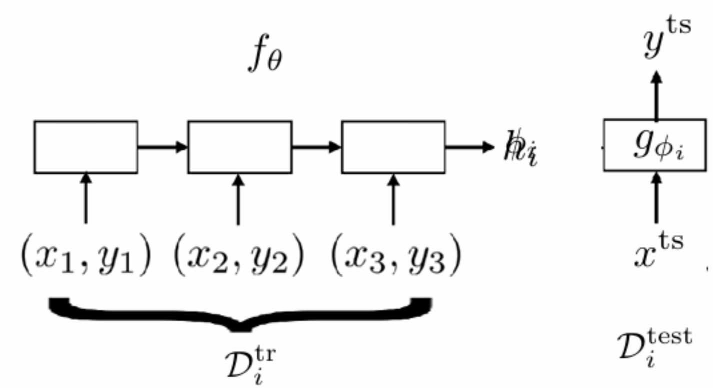
Black-Box Adaptation

Key idea: Train a neural network to represent $\phi_i = f_\theta(\mathcal{D}_i^{\mathrm{tr}})$.

Challenge

Outputting all neural net parameters does not seem scalable?

Idea: Do not need to output **all** parameters of neural net, only sufficient statistics



Meta Networks

- It learns meta-level knowledge across tasks and shifts its inductive biases via fast parameterization for rapid generalization.

Meta Networks

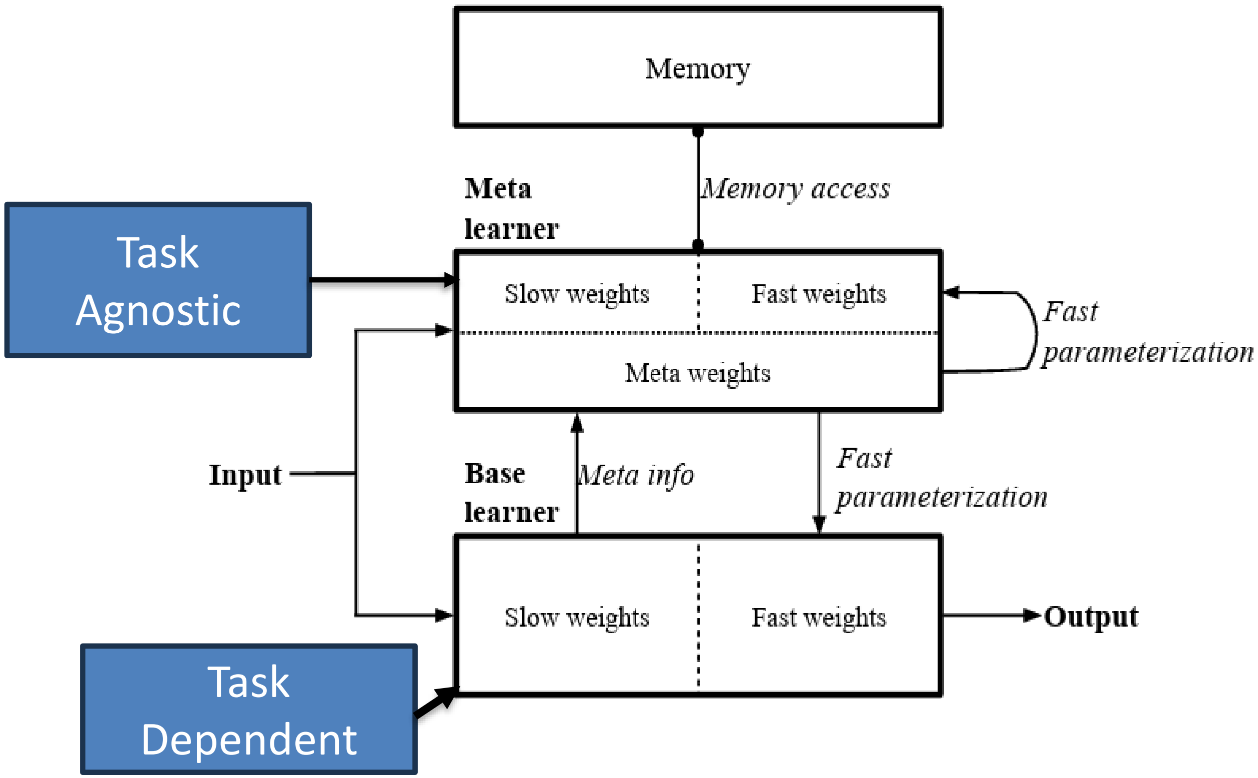


Figure 1. Overall architecture of Meta Networks.

Meta Networks

Three main procedures:

1) Acquisition of meta information,
2) Generation of fast weights
3) Optimization of slow weights,

Meta Learner

The meta learner consists of a dynamic representation learning function u and fast weight generation functions m and d. The function u has a representation learning objective and constructs embeddings of inputs in each task space by using task-level fast weights. The weight generation functions m and d are responsible for processing the meta information and generating the example and task-level fast weights.

Base Learner

It estimates the main task via task loss.

$$\mathcal{L}_i = loss_{task}(b(W, x'_i), y'_i)$$

$$\nabla_i = \nabla_W \mathcal{L}_i$$

Layer Augmentation

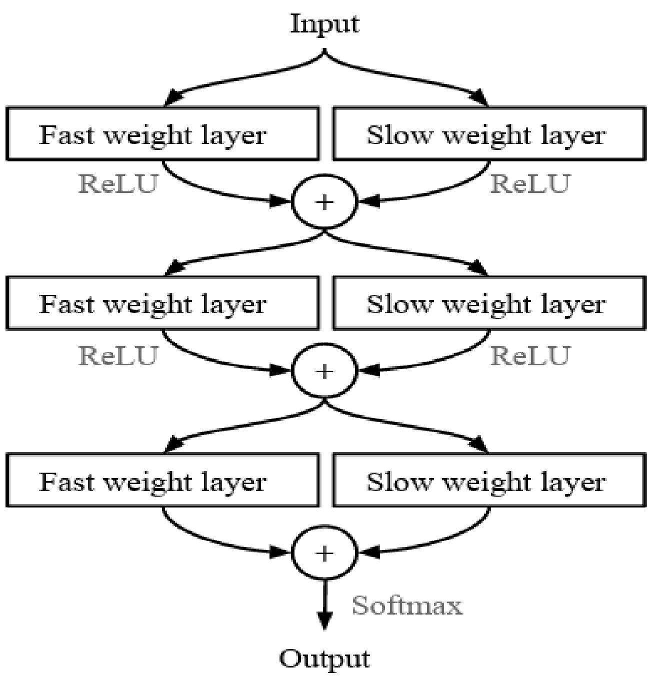


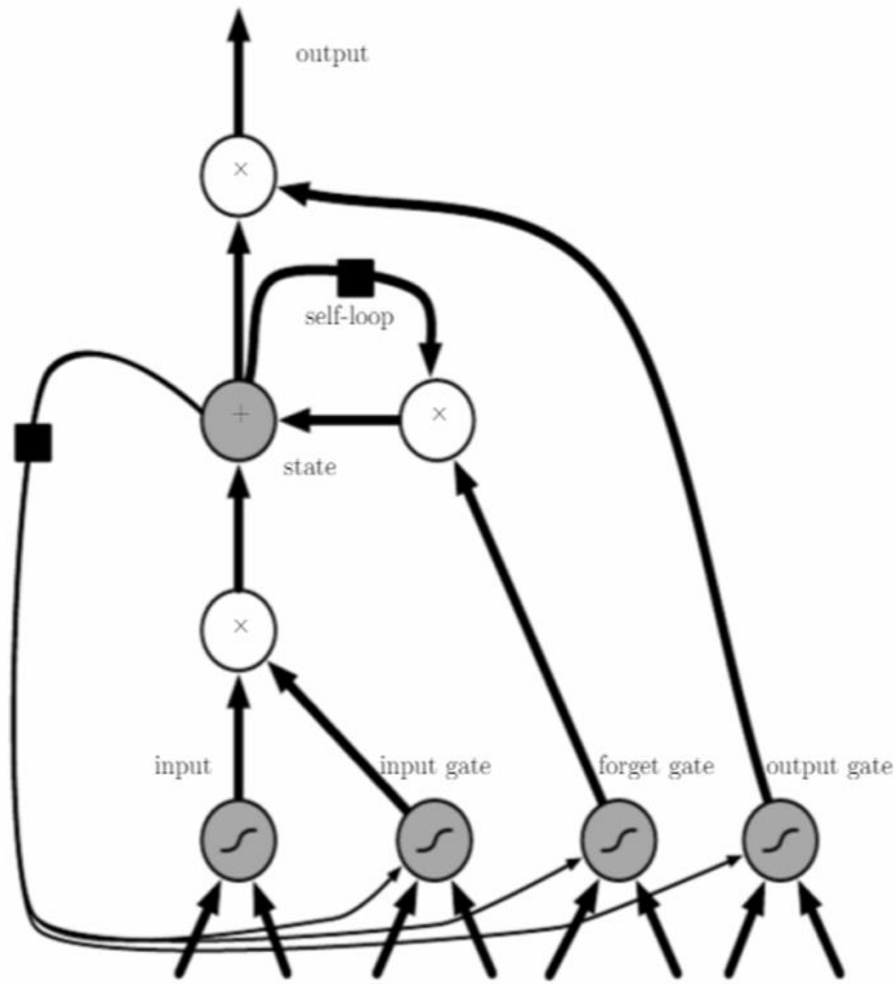
Figure 2. A layer augmented MLP

Meta Learning Dataset

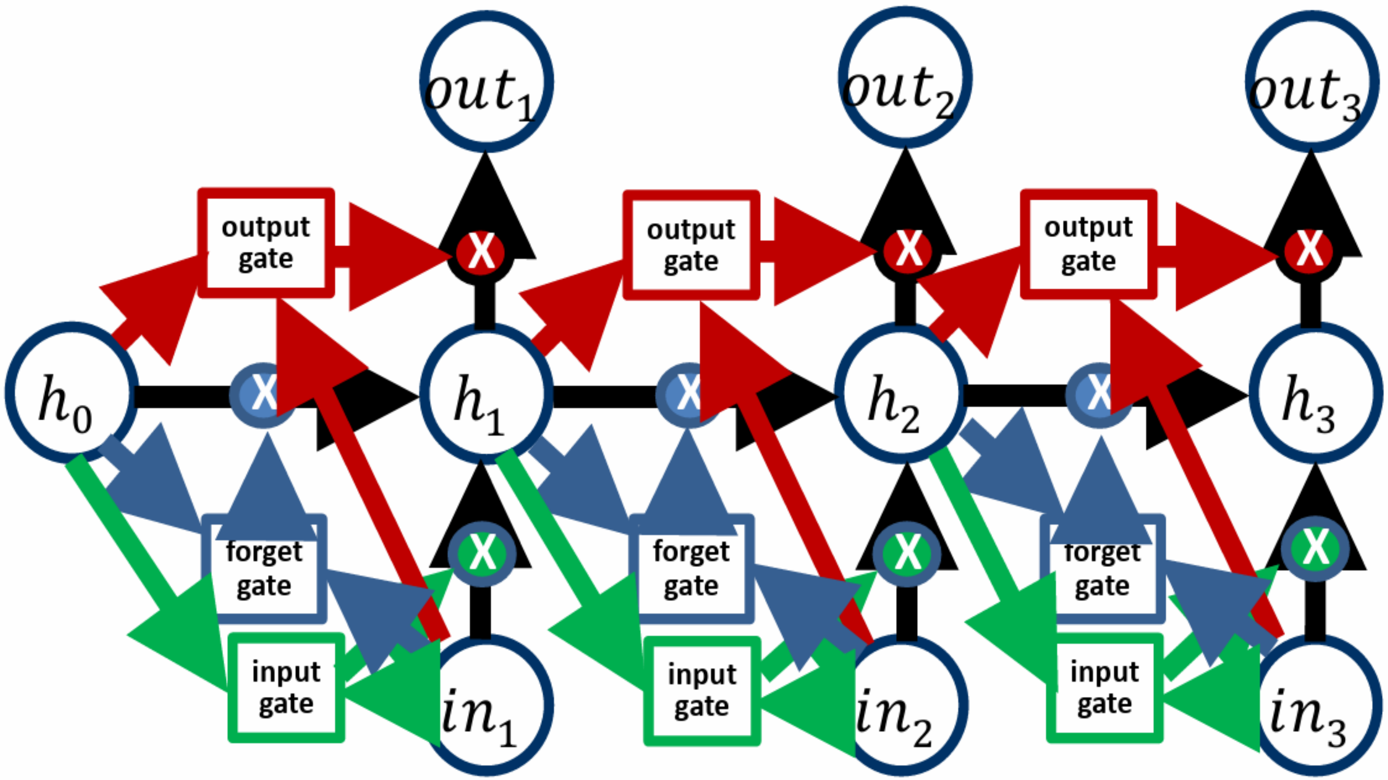
	Omniglot	miniImageNet
Classes	1623	100
Examples for each class	20	600
Image size	28 × 28	84 × 84
Training dataset classes	1200+ new classes	64
Validation dataset classes	No	16

LSTM

- Special gated structure to control memorization and forgetting in RNNs
- Mitigate gradient vanishing
- Facilitate long term memory



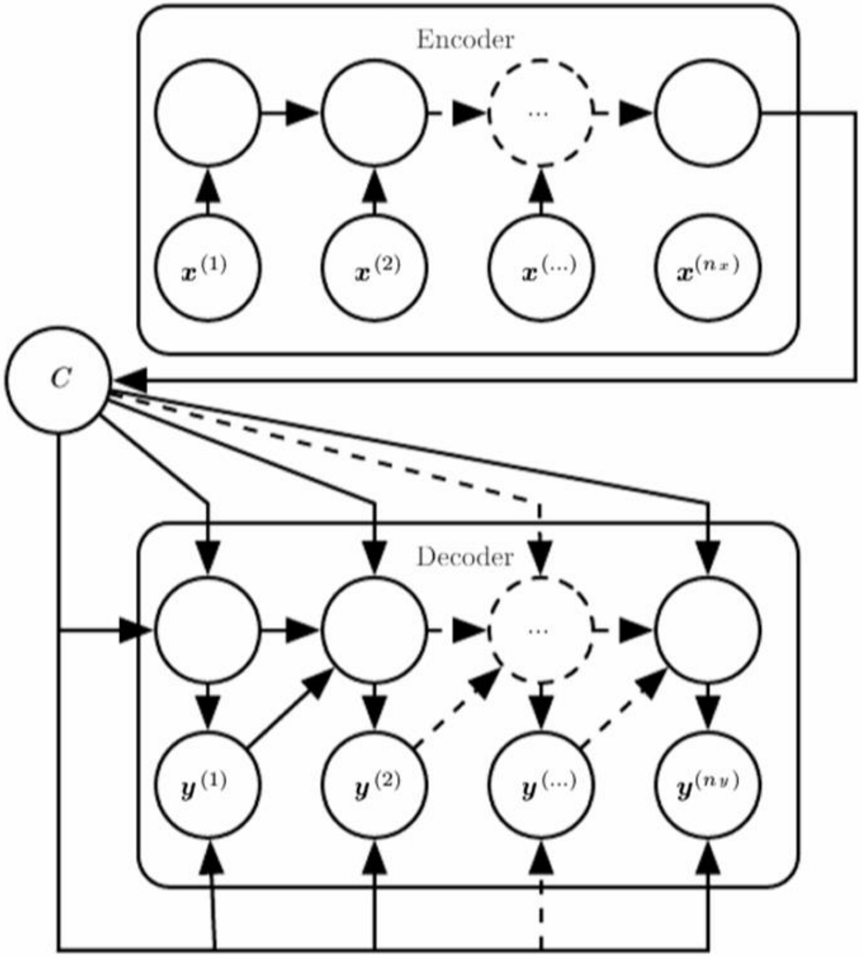
Unrolled LSTM



Encoder Decoder Model

- Also known as sequence2sequence
  - $x^{(i)}$: i^{th} input
  - $y^{(i)}$: i^{th} output
  - c: context (embedding)

- Usage:
  - Machine translation
  - Question answering
  - Dialog



Attention

- Mechanism for alignment in machine translation, image captioning, memory addressing, etc.
- Attention in machine translation: align each output word with relevant input words by computing a softmax of the inputs
  - Context vector c_i: weighted sum of input encodings h_j

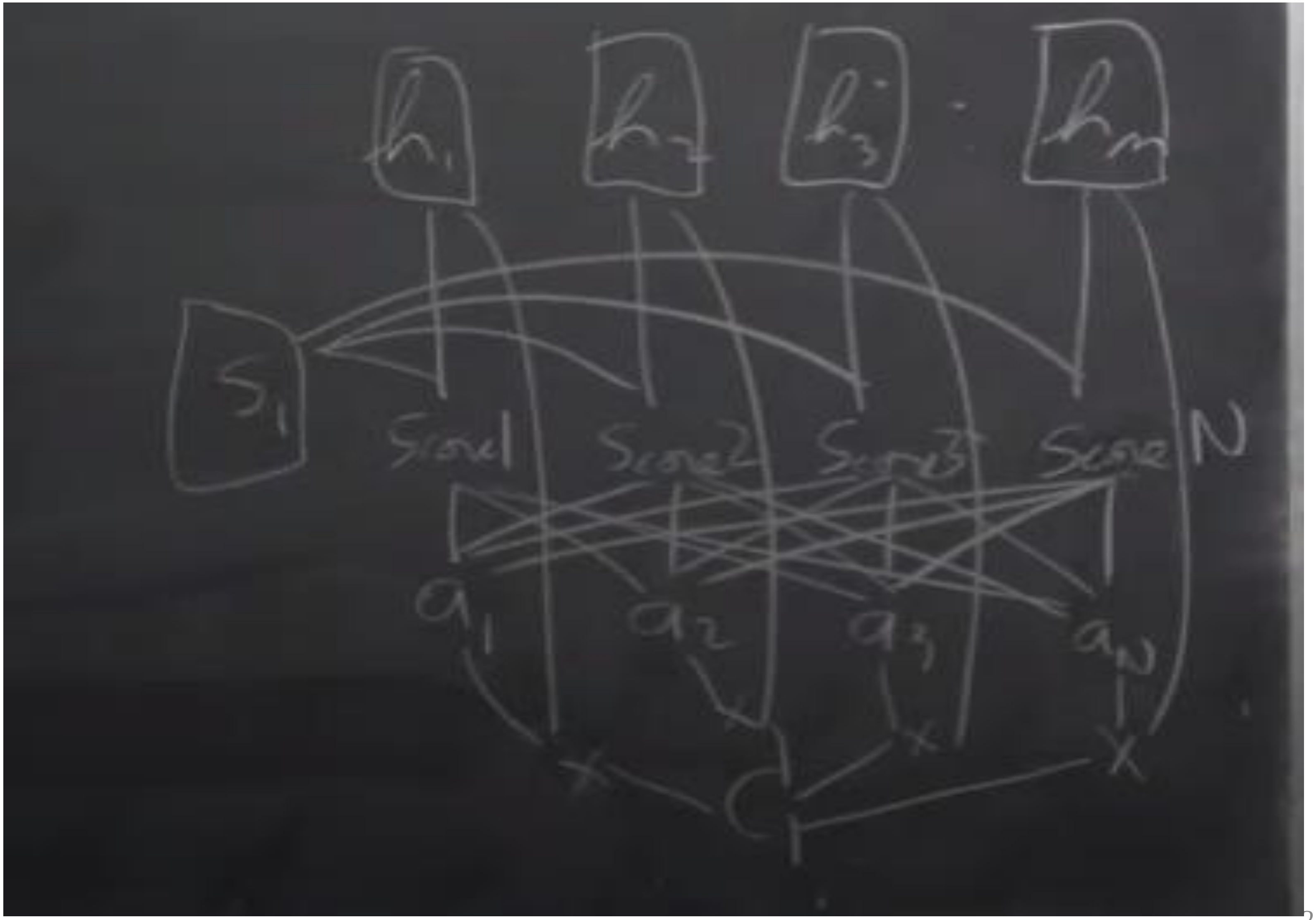
$$c_i = \sum_j a_{ij} h_j$$

  - Where a_{ij} is an alignment weight between input encoding h_j and output encoding s_i

$$a_{ij} = \frac{\exp\left(alignment(s_i, h_j)\right)}{\sum_{j'} \exp(alignment(s_i, h_{j'}))} \quad \text{(softmax)}$$

  - Alignment example: $alignment\left(s_i, h_j\right) = s_i^T h_j$

Attention

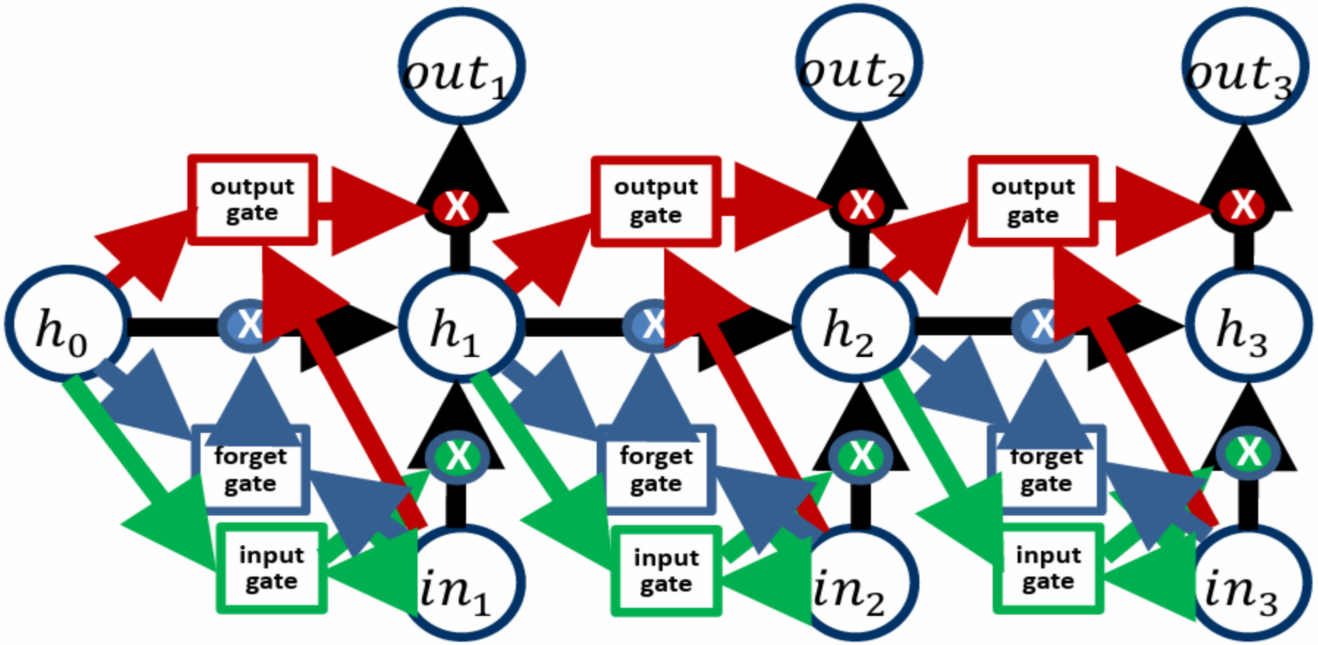


Dialog System

- Suppose we have a database of message-response pairs
  - Store database in memory
  - Key-value pairs: embeddings of message-response pairs (m_i, r_i)
- Use attention mechanism to answer query
  - Embed query: q
  - Measure alignment of query with each message: $a_i = q^T m_i$
  - Compute softmax distribution: $p_i = \exp(a_i) / \sum_j \exp(a_j)$
  - Compute response: $r = \sum_i p_i r_i$
  - Decode response
- End-to-end memory networks (Sukhbaatar, Szlam, Weston, Fergus; NIPS 2015)

General read/write memory

- Replace hidden units by addressable memory
- Replace output gate by attention mechanism
- Generalize input and forget gates to vectors that perform specific operations on each record in the memory



Meta Learning with Memory Augmented Neural Networks

Task (or episode): A dataset of input-output pairs.

Memory: A data structure that can be used to store information. The memory is typically a matrix of real-valued numbers.

Read and write heads: Mechanisms that allow the network to access and modify memory. The read head reads a value from memory, and the write head writes a value to memory.

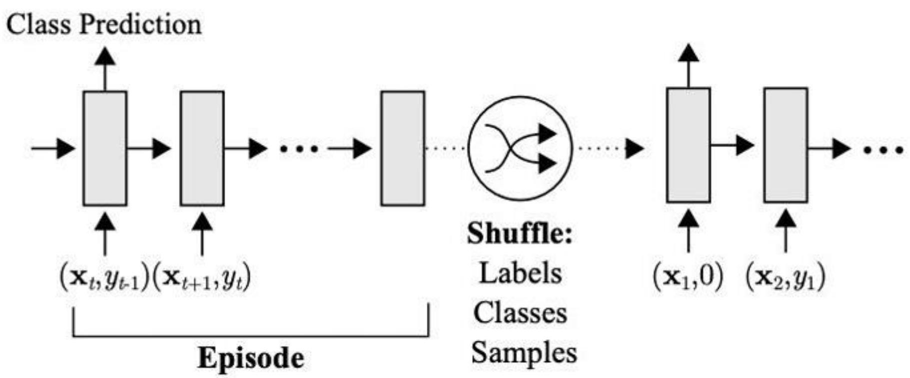
Controller: A neural network that interacts with memory to perform tasks. The controller is responsible for reading and writing to memory, and for generating outputs based on the contents of memory.

Meta Learning with Memory Augmented Neural Networks

- Learn to do classification on unseen class
- Learn the sample-class binding on memory instead of weights

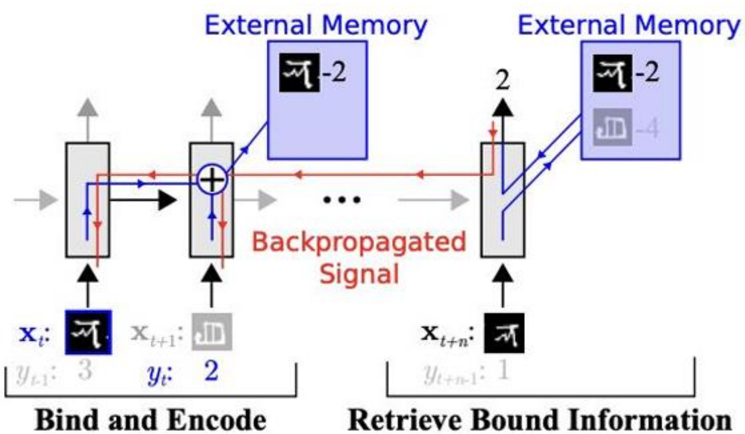
Meta Learning with Memory Augmented Neural Networks

- y_t (label) is present in a temporally offset manner.



(a) Task setup

Meta Learning with Memory Augmented Neural Networks (ML-MANN)



(b) Network strategy

- It must learn to hold data samples in memory until the appropriate labels are presented at the next time-step, after which sample-class information can be bound and stored for later use.

Predictive Distribution

$$p(y_t|\mathbf{x}_t, D_{1:t-1}; \theta),$$

ML-MANN

- Controller is either LSTM or feed forward network.
- The controller interacts with an external memory module using read and write heads, which act to retrieve representations from memory or place them into memory, respectively.

ML-MANN

Key produce by the controller

Cosine similarity measure

$$K\left(\mathbf{k}_t, \mathbf{M}_t(i)\right) = \frac{\mathbf{k}_t \cdot \mathbf{M}_t(i)}{\parallel \mathbf{k}_t \parallel \parallel \mathbf{M}_t(i) \parallel},$$

Memory (i) from a row

Weighted read vector

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

Memory (r) is reterived

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

ML-MANN (Write)

- New information is written into rarely-used locations, preserving recently encoded information, or it is written to the last used location, which can function as an update of the memory with newer, possibly more relevant information.

ML-MANN

- **Rapid Learning:** Demonstrates rapid assimilation of new data for accurate predictions with minimal samples.
- **Episodic Memory:** Utilizes episodic memory to efficiently store and retrieve past task information.
- **Outperformance:** Outperforms gradient-based networks in one-shot learning tasks with limited data.
- **Key-Value Memory:** Efficiently retrieves relevant information using key-value memory, enhancing task adaptation.
- **Enhanced Memory Access:** Introduces a novel memory access method focusing on content, further boosting performance.
- **Human-Like Learning:** Indicates potential to bridge the gap between machine and human learning with flexible adaptation and inductive transfer abilities.

Thank You