ML Session 06
Memory Networks and Graph Attention Networks

Javen Qinfeng Shi

Associate Professor, The University of Adelaide (UoA)

Director and Founder, Probabilistic Graphical Model Group, UoA

Director of Advanced Reasoning and Learning, Australian Institute of Machine Learning (AIML), UoA

What is a Memory Network?

- Memory Network
  - Reasoning with long-term memory component that can be read and written
- Why not RNN?
  - The memory (encoded by hidden states and weights) are too small
  - Not accurately remember facts from the past (compressed in dense vectors)
  - Not even able to reproduce input as output (Zaremba&Sutskever, 2014)

Brief History

- First introduced in ICLR15 (MemNet) and then NIPS15 (MemN2N)
  - J. Weston, S. Chopra, and A. Bordes. Memory networks, *ICLR*, 2015.
  - S. Sukhbaatar, A. Szlam, J. Weston, R. Fergus, End-To-End Memory Networks, NIPS, 2015
- Can be seen as a simplification of Neural Turing Machine (NTM) arVix 2014
  - A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. arXiv, 2014.
- Many variants, mainly for 2 purposes:
  - Attention mechanism: providing weights over the facts, past examples, regions, objects (often via softmax)
  - Storing facts and/or knowledge

MEMORY NETWORKS

Jason Weston, Sumit Chopra & Antoine Bordes
Facebook AI Research
770 Broadway
New York, USA
`{jase,spchopra,abordes}@fb.com`

Figure 1: Example "story" statements, questions and answers generated by a simple simulation. Answering the question about the location of the milk requires comprehension of the actions "picked up" and "left". The questions also require comprehension of the time elements of the story, e.g., to answer "where was Joe before the office?".

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

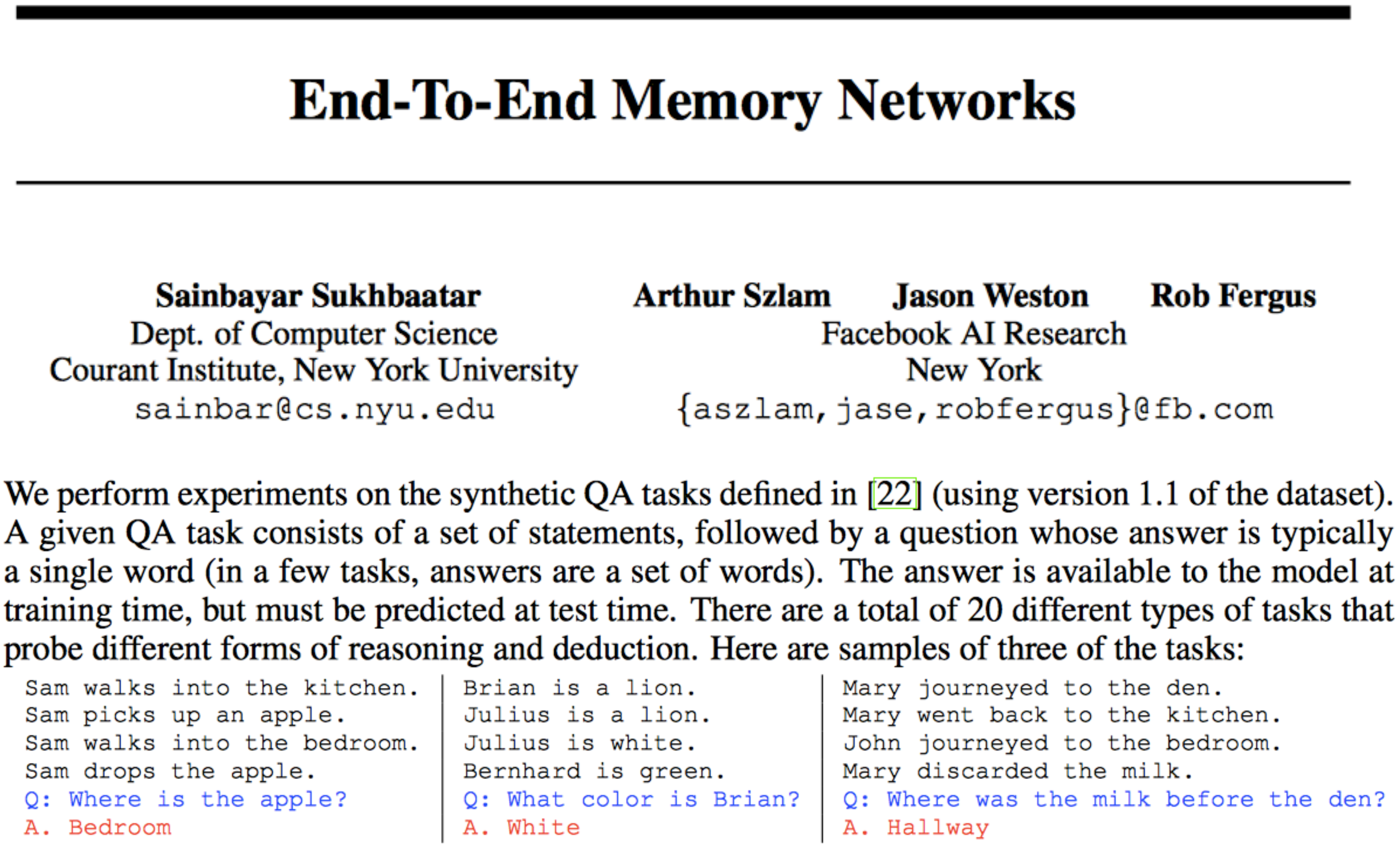
Figure 2: Sample test set predictions (in red) for the simulation in the setting of word-based input and where answers are sentences and an LSTM is used as the R component of the MemNN.

Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living_room.
Where is Dan? A: living room I believe
Where is Joe? A: the bathroom
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now ? A: the milk is in the kitchen
Where is Dan now? A: I think he is in the bedroom
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living_room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? A: I think Joe is in the office

Figure 3: An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 5.2 and had never seen many of these words before, e.g., Bilbo, Frodo and Gollum.

Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring.
Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring.
Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.
Where is the ring? A: Mount-Doom
Where is Bilbo now? A: Grey-havens
Where is Frodo now? A: Shire

Mem N2N (NIPS15)



End-To-End Memory Networks

Sainbayar Sukhbaatar
Dept. of Computer Science
Courant Institute, New York University
sainbar@cs.nyu.edu

Arthur Szlam **Jason Weston** **Rob Fergus**
Facebook AI Research
New York
{aszlam,jase,robfergus}@fb.com

We perform experiments on the synthetic QA tasks defined in [22] (using version 1.1 of the dataset). A given QA task consists of a set of statements, followed by a question whose answer is typically a single word (in a few tasks, answers are a set of words). The answer is available to the model at training time, but must be predicted at test time. There are a total of 20 different types of tasks that probe different forms of reasoning and deduction. Here are samples of three of the tasks:

Sam walks into the kitchen.	Brian is a lion.	Mary journeyed to the den.
Sam picks up an apple.	Julius is a lion.	Mary went back to the kitchen.
Sam walks into the bedroom.	Julius is white.	John journeyed to the bedroom.
Sam drops the apple.	Bernhard is green.	Mary discarded the milk.
Q: Where is the apple?	Q: What color is Brian?	Q: Where was the milk before the den?
A. Bedroom	A. White	A. Hallway

Mem N2N (NIPS15) Setting

Our model takes a discrete set of inputs $x_1, ..., x_n$ that are to be stored in the memory, a query q, and outputs an answer a. Each of the x_i, q, and a contains symbols coming from a dictionary with V

We perform experiments on the synthetic QA tasks defined in [22] (using version 1.1 of the dataset). A given QA task consists of a set of statements, followed by a question whose answer is typically a single word (in a few tasks, answers are a set of words). The answer is available to the model at training time, but must be predicted at test time. There are a total of 20 different types of tasks that probe different forms of reasoning and deduction. Here are samples of three of the tasks:

```
Sam walks into the kitchen.   | Brian is a lion.          | Mary journeyed to the den.
Sam picks up an apple.        | Julius is a lion.         | Mary went back to the kitchen.
Sam walks into the bedroom.   | Julius is white.          | John journeyed to the bedroom.
Sam drops the apple.          | Bernhard is green.        | Mary discarded the milk.
Q: Where is the apple?        | Q: What color is Brian?   | Q: Where was the milk before the den?
A. Bedroom                    | A. White                  | A. Hallway
```

Mem N2N (NIPS15)

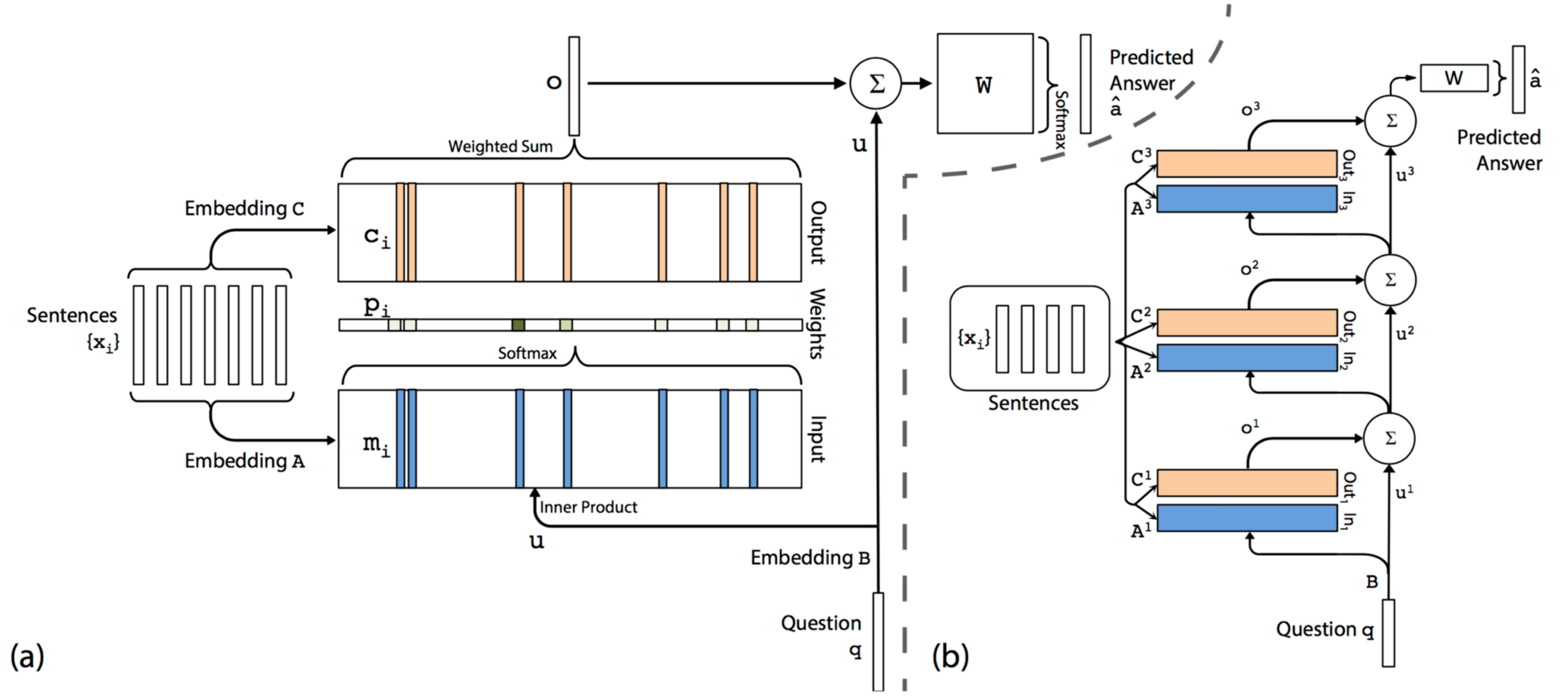


Figure 1: (a): A single layer version of our model. (b): A three layer version of our model. In practice, we can constrain several of the embedding matrices to be the same (see Section 2.2).

$$p_i = \text{Softmax}(u^T m_i). \quad o = \sum_i p_i c_i. \quad \hat{a} = \text{Softmax}(W(o+u))$$

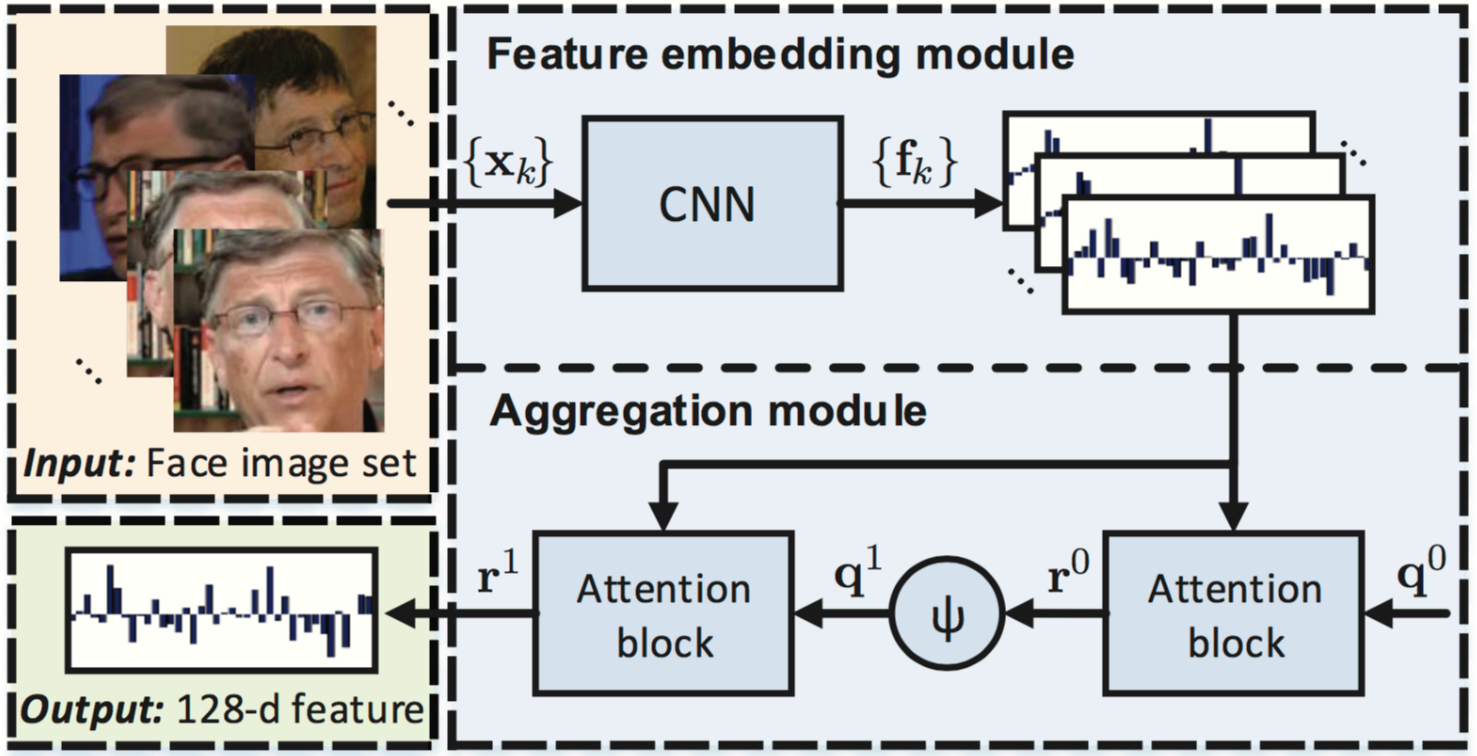
$$\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$$

Attention mechanisms

- Attention mechanisms have become almost a *de facto* standard in many sequence-based tasks (Bah- danau et al., 2015; Gehring et al., 2016).
- One of the benefits of attention mechanisms is that they allow for
  - dealing with variable sized inputs
  - focusing on the most relevant parts of the input to make decisions.

An example: attention model to recognise faces in a video (sequence)

Jiaolong Yang, Gang Hua, et al., Neural Aggregation Networks for Video face Recognition, CVPR 2017



$$\mathbf{r} = \sum_k a_k \mathbf{f}_k .$$

$$e_k = \mathbf{q}^T \mathbf{f}_k$$

$$a_k = \frac{\exp(e_k)}{\sum_j \exp(e_j)} .$$

Figure 1. Our network architecture for video face recognition. All input face images $\{\mathbf{x}_k\}$ are processed by a feature embedding

Question

- What is the essence of a traditional Memory Network?
- Answer (using verbal or whiteboard):
  - Retrieving and storing
  - Inner product gives a score (which can be normalised by softmax)
  - Inner product can be applied directly to the query and candidates vectors, or their transformations (via a matrix, or encoder)

- What if the input is not a vector, a tensor, or a sequence?
- What if it is a graph?
- What if every graph can be different, and you wish what you have learnt can be applied/ generalised to a different graph?

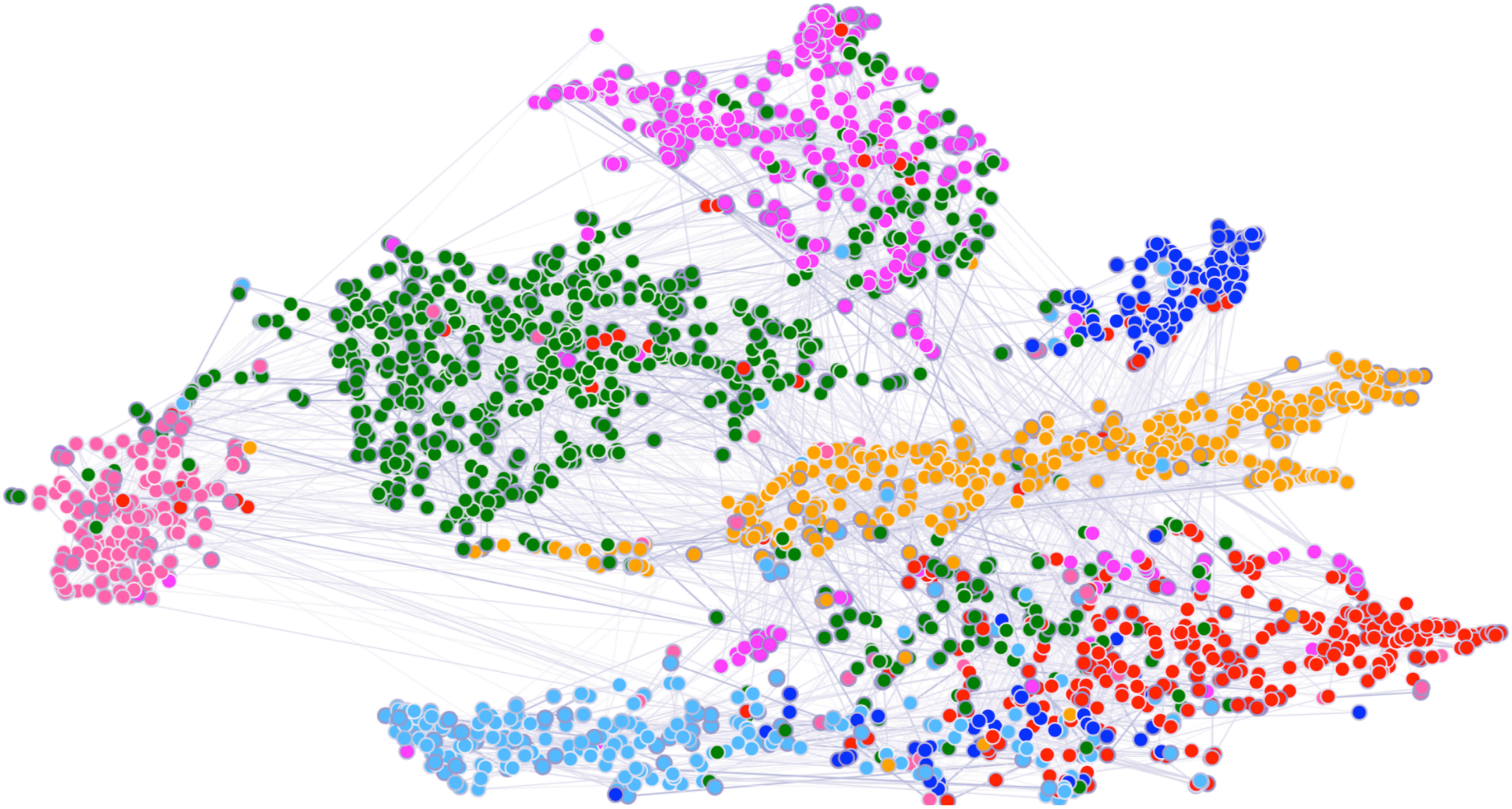
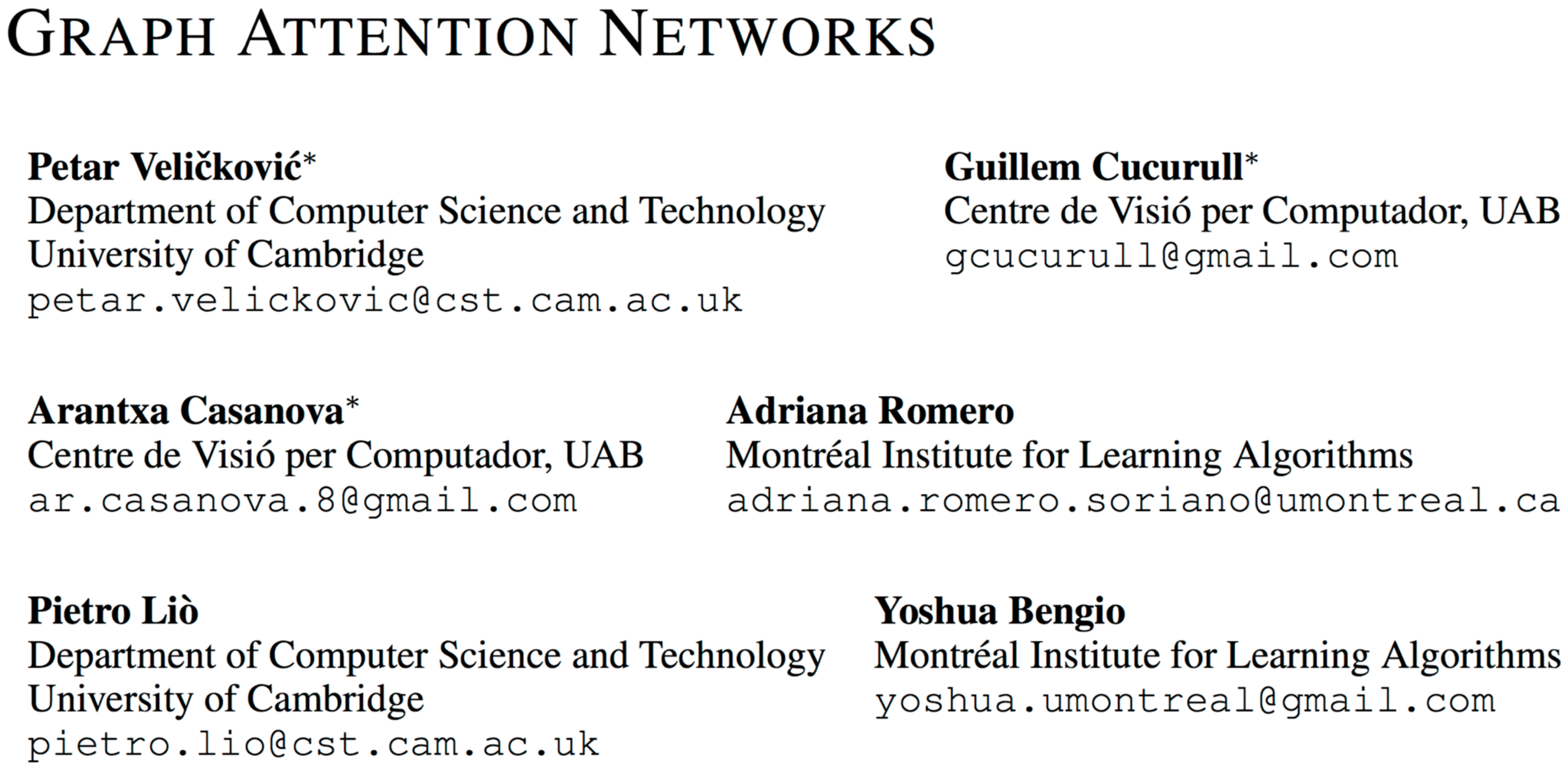


Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads ($\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k}$).

Dataset	**Type**	**Nodes**	**Edges**	**Classes**	**Features**	**Label rate**
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

ICLR18



GRAPH ATTENTION NETWORKS

Petar Veličković*
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*
Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*
Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero
Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

Yoshua Bengio
Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

GRAPH ATTENTIONAL LAYER

- Input: a set of node features

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

- Output: a set of node features (with potentially different dimensionality)

$$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

Attention coefficients *per* neighborhood

- For each node i, and we only compute e_ij if j is i's neighbor (N_i, determined by the graph)

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$$

a *weight matrix*, $\mathbf{W} \in \mathbb{R}^{F' \times F}$, is applied to every node.

- Normalize it via softmax

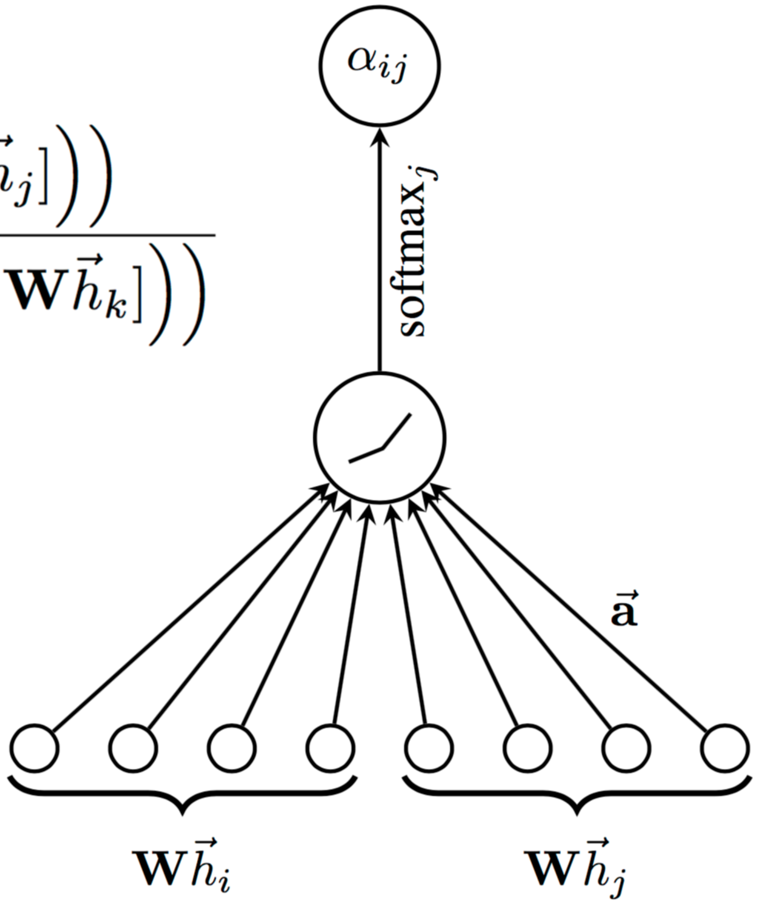
$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

a is a single layer feedforward network

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right).$$

T is transpose, and II is concatenation



Multi-head attention

- Learn to a single, a and W, to perform well, is difficult, or at least requires more hand engineering
- Let's try to use K indepedent a, W (a^k, W^k, k = 1, ..., K) (like drawing raffle tickets multiple times to increase the chance that one of the drawing is lucky)
- Why indepedent?
  - Paralisable
  - Easier
  - Won't be effected much by some bad drawing previously

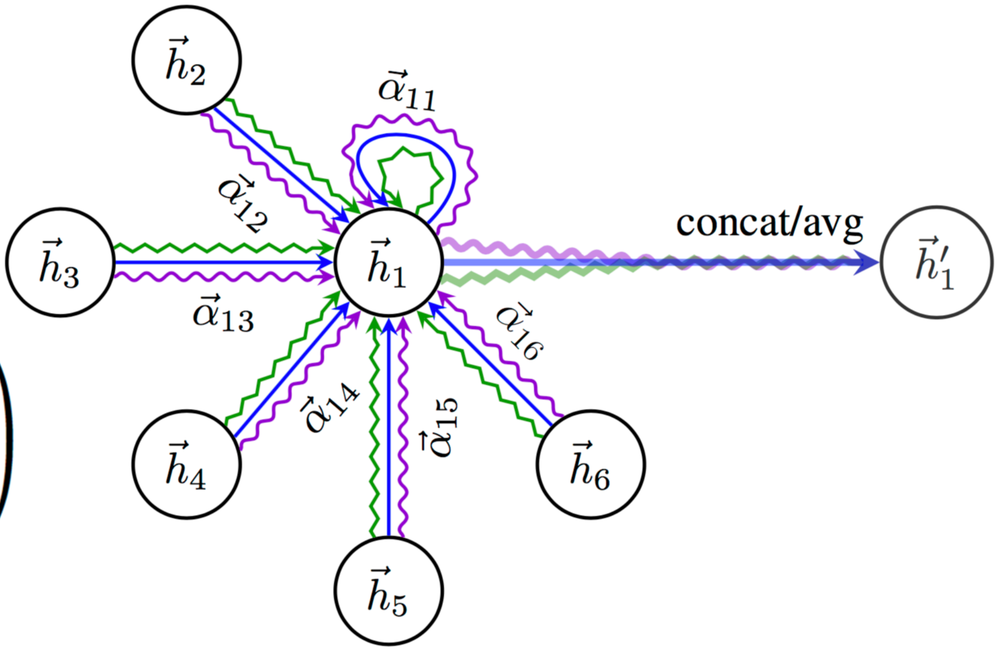
2 schemes:

Concatenation:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

Averaging:

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$



Benefits

- It does not need to know the entire graph upfront.
- A learned model can be applied to any unseen arbitrary graph.

Datasets

Transductive learning We utilize three standard citation network benchmark datasets—Cora, Citeseer and Pubmed (Sen et al., 2008)—and closely follow the transductive experimental setup of Yang et al. (2016). In all of these datasets, nodes correspond to documents and edges to (undirected) citations. Node features correspond to elements of a bag-of-words representation of a document. Each node has a class label. We allow for only 20 nodes per class to be used for training—however, honoring the transductive setup, the training algorithm has access to all of the nodes' feature vectors. The predictive power of the trained models is evaluated on 1000 test nodes, and we use 500 additional nodes for validation purposes (the same ones as used by Kipf & Welling (2017)). The Cora dataset contains 2708 nodes, 5429 edges, 7 classes and 1433 features per node. The Citeseer dataset contains 3327 nodes, 4732 edges, 6 classes and 3703 features per node. The Pubmed dataset contains 19717 nodes, 44338 edges, 3 classes and 500 features per node.

Inductive learning We make use of a protein-protein interaction (PPI) dataset that consists of graphs corresponding to different human tissues (Zitnik & Leskovec, 2017). The dataset contains 20 graphs for training, 2 for validation and 2 for testing. Critically, testing graphs remain *completely unobserved* during training. To construct the graphs, we used the preprocessed data provided by Hamilton et al. (2017). The average number of nodes per graph is 2372. Each node has 50 features that are composed of positional gene sets, motif gene sets and immunological signatures. There are 121 labels for each node set from gene ontology, collected from the Molecular Signatures Database (Subramanian et al., 2005), and a node can possess several labels simultaneously.

Results

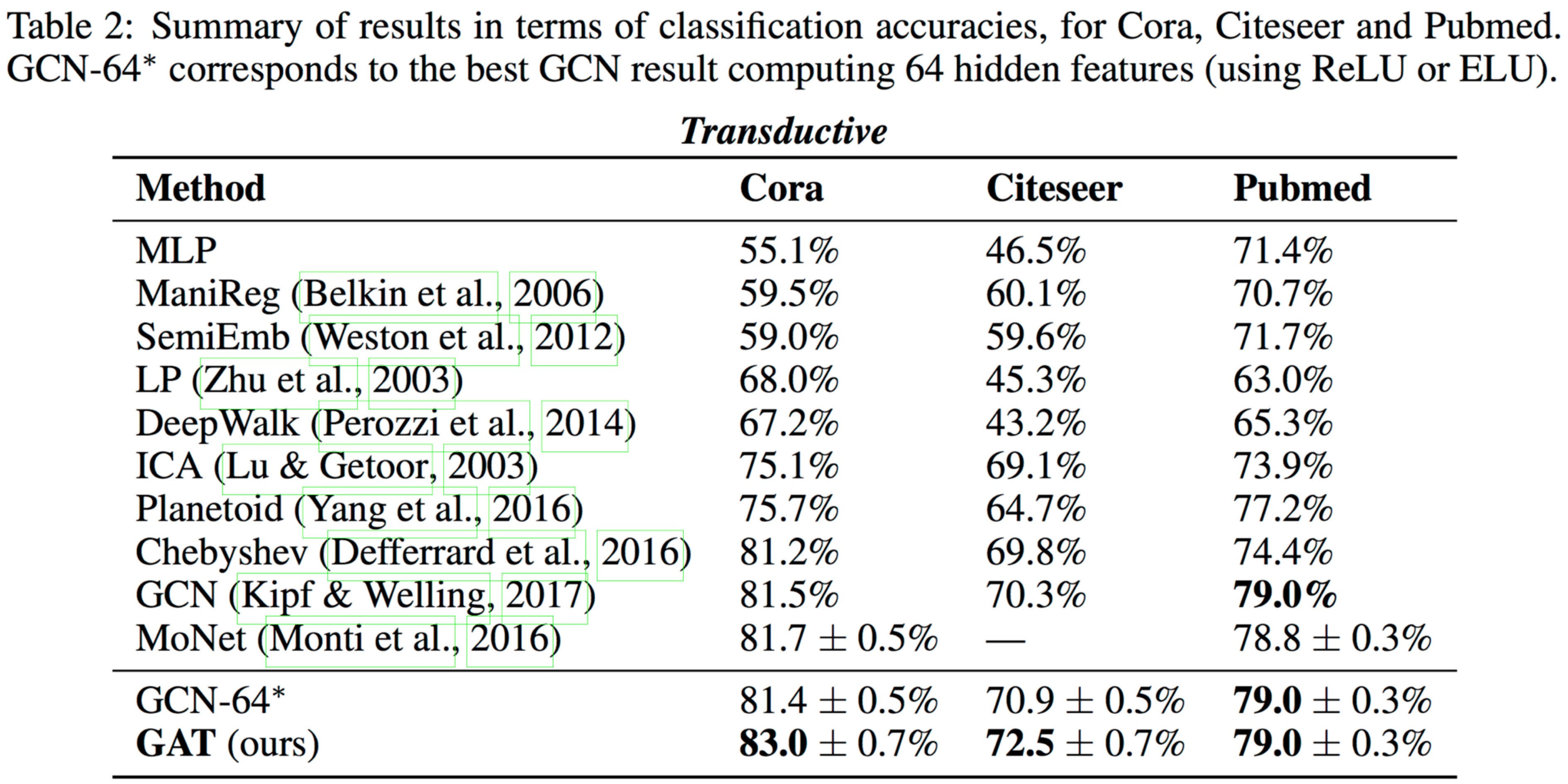


Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

Results

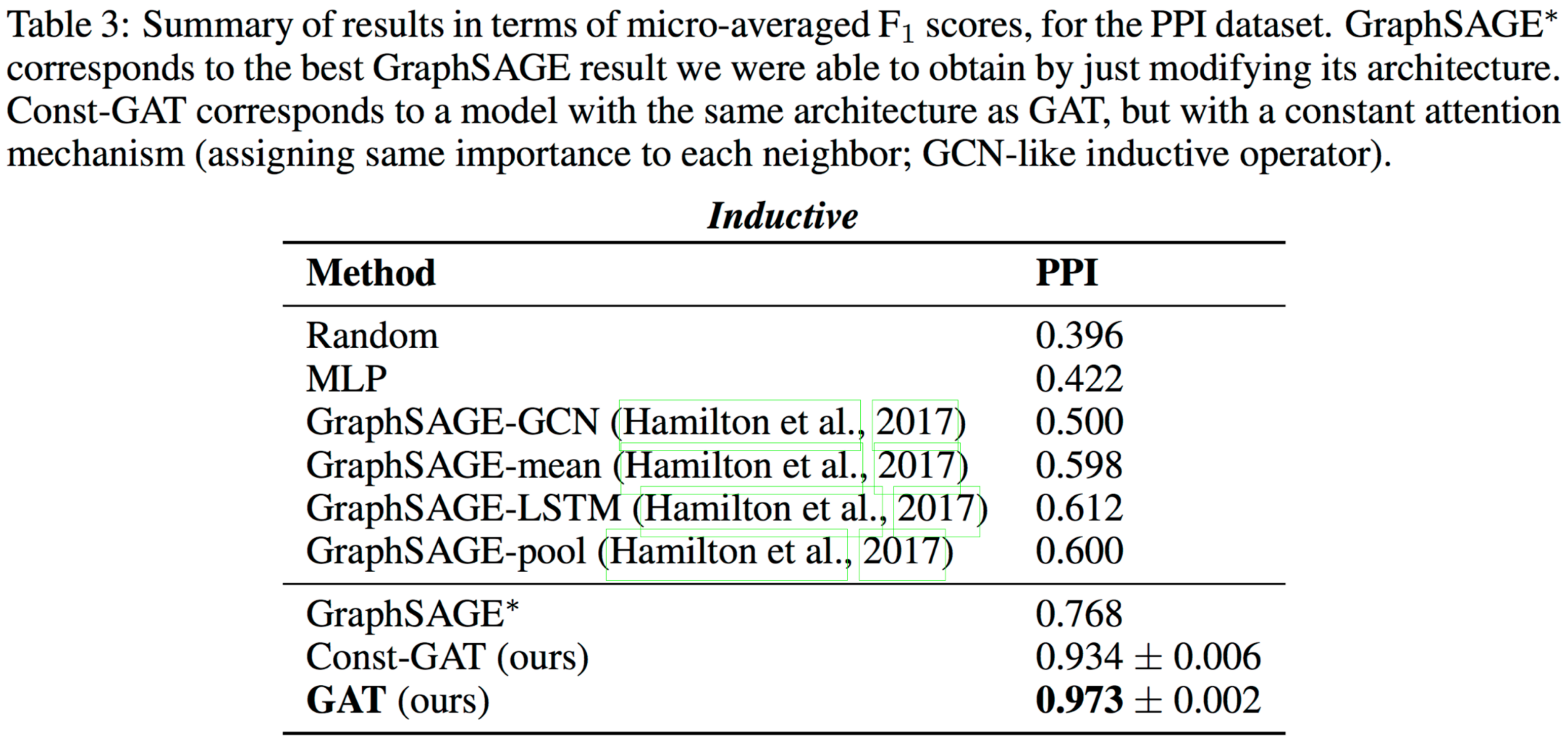


Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

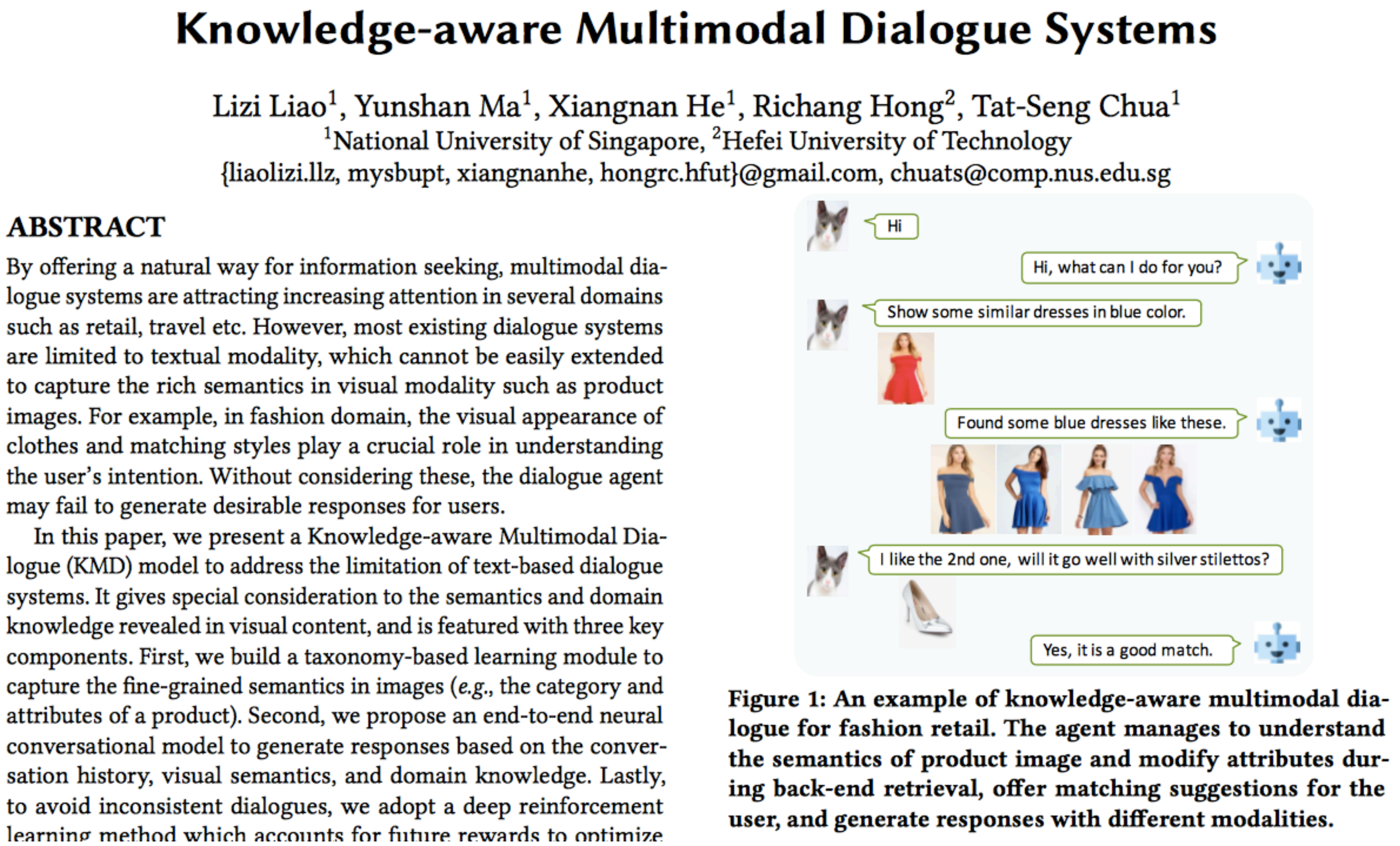
Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Reasoning with knowledge?

- Memory networks can store past observation. Can they store facts and knowledge and reasons with them?
- Yes, but only to a small degree.

MM18



Knowledge-aware Multimodal Dialogue Systems

Lizi Liao[1], Yunshan Ma[1], Xiangnan He[1], Richang Hong[2], Tat-Seng Chua[1]
[1]National University of Singapore, [2]Hefei University of Technology
{liaolizi.llz, mysbupt, xiangnanhe, hongrc.hfut}@gmail.com, chuats@comp.nus.edu.sg

ABSTRACT

By offering a natural way for information seeking, multimodal dialogue systems are attracting increasing attention in several domains such as retail, travel etc. However, most existing dialogue systems are limited to textual modality, which cannot be easily extended to capture the rich semantics in visual modality such as product images. For example, in fashion domain, the visual appearance of clothes and matching styles play a crucial role in understanding the user's intention. Without considering these, the dialogue agent may fail to generate desirable responses for users.

In this paper, we present a Knowledge-aware Multimodal Dialogue (KMD) model to address the limitation of text-based dialogue systems. It gives special consideration to the semantics and domain knowledge revealed in visual content, and is featured with three key components. First, we build a taxonomy-based learning module to capture the fine-grained semantics in images (*e.g.*, the category and attributes of a product). Second, we propose an end-to-end neural conversational model to generate responses based on the conversation history, visual semantics, and domain knowledge. Lastly, to avoid inconsistent dialogues, we adopt a deep reinforcement learning method which accounts for future rewards to optimize

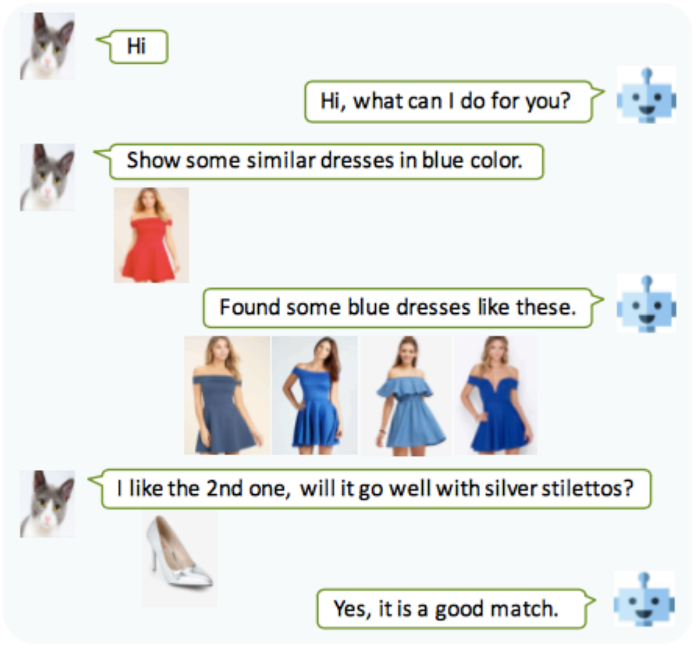

Hi
Hi, what can I do for you?
Show some similar dresses in blue color.
Found some blue dresses like these.
I like the 2nd one, will it go well with silver stilettos?
Yes, it is a good match.

Figure 1: An example of knowledge-aware multimodal dialogue for fashion retail. The agent manages to understand the semantics of product image and modify attributes during back-end retrieval, offer matching suggestions for the user, and generate responses with different modalities.

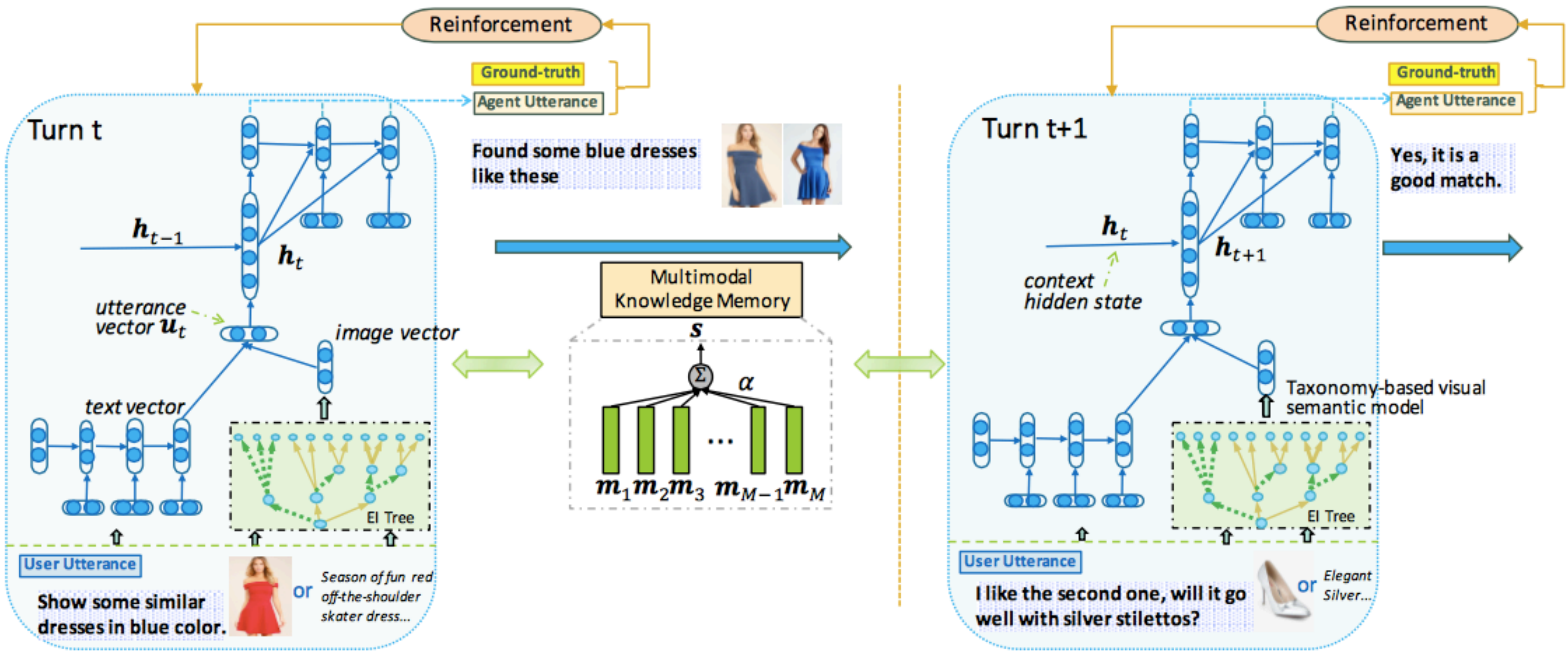
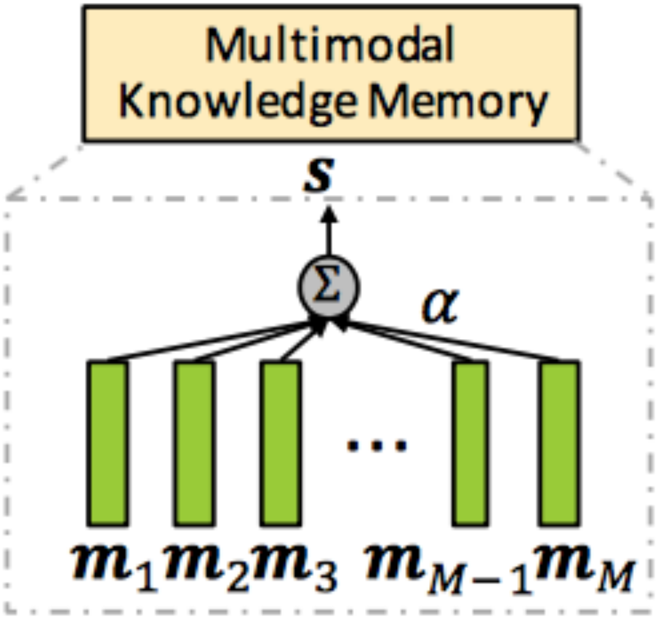


Figure 2: The knowledge-aware multimodal dialogue framework (KMD). Two turns of conversations are illustrated. The agent leverages taxonomy-based visual semantic model to understand user utterances in different forms. It generates various forms of responses enriched with extracted domain knowledge. Deep reinforcement learning measures the goodness of a response through a reinforcement signal and optimizes the long-term rewards that characterize a good conversation.

We then incorporate these knowledge into encoder state. Note that $\mathbf{g}_i$ refers to the vector representation of knowledge i, we have:



$$\mathbf{m}_i = \mathbf{A}\mathbf{g}_i \tag{1}$$

$$\mathbf{o}_i = \mathbf{B}\mathbf{g}_i \tag{2}$$

$$\alpha_i = \frac{exp(\mathbf{h}_t^T \mathbf{m}_i)}{\sum_{k=0}^{M} exp(\mathbf{h}_t^T \mathbf{m}_k)} \tag{3}$$

$$\mathbf{s} = \sum_{i=1}^{M} \alpha_i \mathbf{o}_i \tag{4}$$

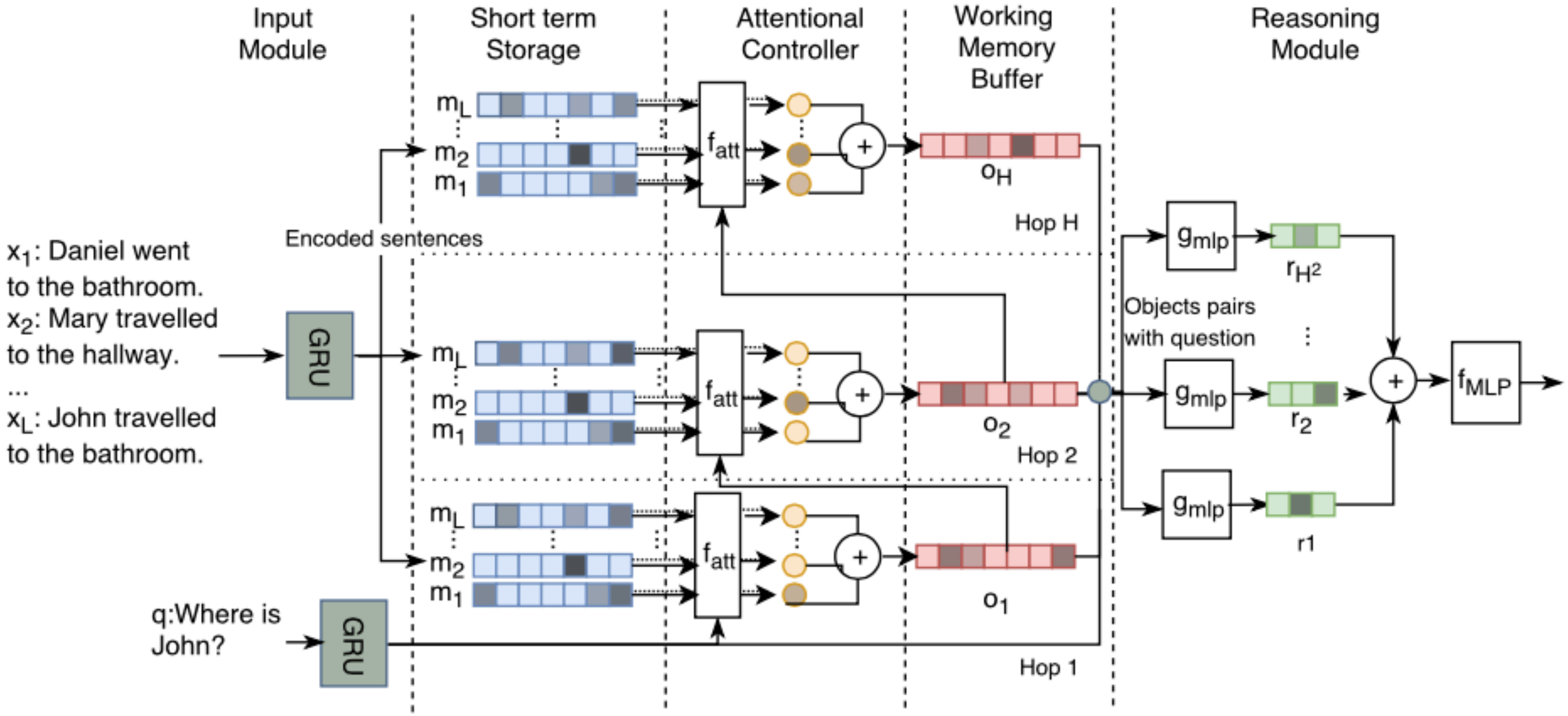
where M is the total number of knowledge entries. $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d \times v}$ are the parameters of the memory network. The former one embeds the knowledge $\mathbf{g}_i$ into memory representation while the later one trans-

ACL18

Working Memory Networks: Augmenting Memory Networks with a Relational Reasoning Module

Juan Pavez*, Héctor Allende
Department of Informatics
Federico Santa María
Technical University

Héctor Allende-Cid
Escuela de Ingeniería Informática
Pontífica Universidad Católica
de Valparaíso



Reasoning Module

The outputs stored in the working memory buffer are passed to the reasoning module. The reasoning module used in this work is a Relation Network (RN). In the RN the output vectors are concatenated in pairs together with the question vector. Each pair is passed through a neural network g_θ and all the outputs of the network are added to produce a single vector. Then, the sum is passed to a final neural network f_ϕ:

$$r = f_\phi\left(\sum_{i,j} g_\theta([o_i; o_j; u])\right), \tag{4}$$

- We will cover relation networks (RNs) and statistical relational learning (SRL) in future.