Differentiable Reasoning over Long Stories
Assessing Systematic Generalisation in Neural Models

Wanshui Li and Pasquale Minervini
University College London, London WC1E 6BT, UK

Abstract. Contemporary neural networks have achieved a series of developments and successes in many aspects; however, when exposed to data outside the training distribution, they may fail to predict correct answers. In this work, we were concerned about this generalisation issue and thus analysed a broad set of models systematically and robustly over long stories. Related experiments were conducted based on the CLUTRR, which is a diagnostic benchmark suite that can analyse generalisation of natural language understanding (NLU) systems by training over small story graphs and testing on larger ones. In order to handle the multi-relational story graph, we consider two classes of neural models: “E-GNN”, the graph-based models that can process graph-structured data and consider the edge attributes simultaneously; and “L-Graph”, the sequence-based models which can process linearized version of the graphs. We performed an extensive empirical evaluation, and we found that the modified recurrent neural network yield surprisingly accurate results across every systematic generalisation tasks which outperform the modified graph neural network, while the latter produced more robust models.

Keywords: Systematic Generalisation · Robust Reasoning · Neuro-symbolic Models.

1 Introduction

Since the earliest “artificial neuron” model proposed based on studies of the human brain and nervous system, neural networks have achieved a series of developments and even notable successes in some commercially essential areas, such as image captioning, machine translation and video games. However, several shortcomings in modern deep neural networks are emphasised by Garnelo and Shanahan [8]:

– Data inefficiency and high sample complexity. In order to be effective, contemporary neural models usually need large volumes of training data, such as BERT [6] and GPT2 [23] with the pre-trained basis which comes from an incredible amount of data.
– Poor generalisation. It is a challenge to predict correct answers when neural models are evaluated on examples outside of the training distribution, and even just small invisible changes to the inputs can entirely derail predictions [9].
Meanwhile, the ability of NLU systems is increasingly being questioned — and neural networks more generally — to generalise systematically and robustly [1]. For instance, the brittleness of NLU systems to adversarial examples [15], the failure of exhibiting reasoning and generalisation capabilities but only exploiting statistical artefacts in datasets [11], the difficulty of incorporating the statistics of the natural language instead of reasoning on those large pre-trained models [22] and the substantial performance gap of generalisation and robustness between state-of-art NLU models and a Graph Neural Networks (GNN) model [27]. It seems that modern neural models capture the wrong pattern and do not understand the content of data, which is far away from the expected reasoning ability mimicking from the human brain. In other words, both of this aspect should be emphasised so that we can build up a better model.

In order to evaluate and compare each model’s ability of systematic generalisation and robust reasoning, we use the benchmark suite name Compositional Language Understanding and Text-based Relational Reasoning (CLUTRR), which contains a broad set of semi-synthetic stories involving hypothetical families, and the goal is to reason the relationship between two entities when given a story [27]. Actually, [27] already analysed several models, including those shocking baselines models we have known such as BERT and Graph Attention Networks (GAT) [32]. They found that existing NLU systems show poor results both on generalisation and robustness comparing GAT who directly works on symbolic inputs. For instance, interestingly line graphs in the paper showed us that in most situations, BERT stays the lowest accuracy among all text-based models and also a graph neural model, GAT; when trained on noisy data, only GAT can effectively capture the robust reasoning strategies. Both of these phenomena show us that there is a gap between unstructured text inputs and structured symbolic inputs.

Therefore, motivated by that unexpected results and inspired by the neuro-symbolic reasoning models [6,4,8], we explore two types of models, the graph-based model and the sequence-based model. Each of them has different forms of symbolic inputs, and they are trained over the CLUTRR datasets to evaluate their generalisation and robustness performance compared with texted-based models. Briefly speaking, graph-based models are those models with a graph as input, such that in CLUTRR datasets, entities and relationships are modelled as nodes and edges to form graphs. To deal with graph-type input, we usually use GNN models such as GAT, Graph Convolutional Networks (GCN) [17]. In fact, their architectures need to be modified so that we can feed the multi-relational graph into models. For sequence-based models, they are the revised version of those typical sequence encoding models such as Convolutional Neural Networks (CNN) [19], Recurrent Neural Networks (RNN) [24]. Their entities and relationships are modelled into graphs and then concatenated into sequences.
2 Related Work

Neuro-symbolic Models. Recent advances in deep learning show an improvement of expressive capacity, and significant results have been achieved in some perception tasks, such as action recognition and event detection. Nevertheless, it is widely accepted that an intelligent system needs combining both perception and cognition. Highly cognitive tasks, such as abstracting, reasoning and explaining, are closely related to symbol systems which usually cannot adapt to complex high-dimensional space. Neuro-symbolic models combine the advantages of the deep learning model with symbolic methods, thereby significantly reducing the search space of symbolic methods such as program synthesis [5,35, cf.].

There are many applications and methods of neuro-symbolic reasoning in natural language processing (NLP), basically involved complex question answering (QA), that is, given a complicated question, the answer is inferred from the context. Usually, the process is to split the tricky question into several sub-questions and then to use the neuro-symbolic reasoning model to get the results, which requires the abilities of question understanding, information extraction of context, and symbolic reasoning.

For instance, Gupta et al. [10] proposed Neural Module Networks (NMN) to solve tricky question answering tasks, where the reasoning process over texts involves natural language reasoning and symbolic reasoning. Natural language reasoning can be seen as the process of information extraction from texts, and symbolic reasoning is the reasoning process based on the extracted, structured information. Compared with NMN customising modules based on tasks, Compositional Attention Networks (MAC) [14] is a soft-attention architecture (i.e. computing all corresponding attention weight with all the data) providing a more universal and reusable architectures with shared parameters, and it is an end to end differentiable. Neural State Machine (NSM) [13] has a similar reasoning mechanism that in MAC, but its expression way is the probability distribution of a scene graph based on the given content, which shows powerful generalisation capacity under multi-tasks. Neuro-Symbolic Concept Learner (NS-CL) [20] has three modules: extract and express target in fixed-length vectors, similarity comparison for the object and contents, execution and analysis with a curriculum learning approach, where the modules are efficient and even achieve good results over a fewer quantity of data.

By combining neural networks with symbolic systems, models can be improved effectively in their shortcomings aspects, such as promoting data efficiency due to reusable property in multiple tasks; becoming interpretable or human understanding to some extent because of human-like thinking process; facilitating generalisation capacity because of high-level, abstract representations. In our work, we modify the architectures of GNN and sequence encoding models (i.e. CNN, RNN) to achieve the neuro-symbolic reasoning process (see Chapter 3), and based on their types of input, we separate them into two classes: the graph-based model and the sequence-based model.
3 Assessing Systematic Generalisation

Two types of model, the E-GNN (graph-based) model and the L-Graph (sequence-based) model, are used to work on CLUTTR datasets to evaluate the performance of systematic generalisation and robust reasoning as well.

3.1 Datasets

As in [27], we use the same pre-generated datasets to evaluate our models and it is also easier and more obvious to compare our models’ performance with that in the paper. There are totally six datasets separated into two groups: two of them (named "data_d089907f8" and "data_db9b8f04") are used for systematic generalisation evaluation, and the rest of them (named "data_7c5b0e70", "data_6b8f2a1", "data_523348e6" and "data_d83ecc3e") are used for robust reasoning evaluation.

We test our models’ systematic generalisation capacity with clauses of length $k = 2, 3 \cdots, 10$ for both datasets but training process has two regimes: clauses of length in "data_d089907f8" are $k = 2, 3$ and in "data_db9b8f04" are $k = 2, 3, 4$. An example of a train and test instance in CLUTTR in Fig. 1.

![Fig. 1. An example of a train and test instance in CLUTTR. Training over instance with clauses of length $k=2$ (left), and testing over instance with clauses of length $k=10$ (right). All names are replaced with capital alphabet since replaceable and unrelated in the process; relationships are shown between two nodes beside the line. Task (or query) is to identify the relationship between two nodes linked by red dash based on the given supporting facts.

1 https://github.com/facebookresearch/cluttr/](https://github.com/facebookresearch/cluttr/)
3.2 E-GNN

In order to handle multi-relational graphs and better learn the relationship between two entities, we modified the normal Graph Neural Networks (GNN) architecture. Entities and relationships are modelled as nodes and edges, respectively in the graph, and we concatenate relationships information on the entities so that the model can also consider the edge attributes during encoding. In other words, the graph-based mechanism is based on the supporting facts to embed a query and seek the target, so we can also present this type of model in the form of:

$$\hat{y} = \text{softmax} \left( W[\text{emb}(F) \parallel \text{emb}(Q | F)] \right)$$

, where $\text{emb}(\cdot)$ denotes an embedding process working on a set of ground facts, such that the supporting facts ($F$) or a query conditioning on supporting facts ($Q | F$); $\parallel$ means a concatenation of two items; $W$ represents weight matrix; and $\hat{y}$ represents the distribution of all possible relationships we get after softmax function, and then we will choose the highest probability relationship type as the predicted target.

After processing with GNN models, $\text{emb}(F)$ is the representation of each node in the graph with a shaped $[B \times N \times \text{emb}_{\text{dim}}]$, where $B$ is the batch size, $N$ is the number of nodes and $\text{emb}_{\text{dim}}$ is the embedding dimension. $\text{emb}(Q | F)$ is the representation of each query and it comes from $\text{emb}(F)$. Here, we gather the representation of needed nodes in the query from $\text{emb}(F)$ and it shaped in $[B \times (2 \times \text{emb}_{\text{dim}})]$, where ”2” means two entities/nodes in a query. Besides, when we conduct concatenating, $\text{emb}(F)$ is reshaped into $[B \times \text{emb}_{\text{dim}}]$ by taking average on the nodes, and we get the shape $[B \times (3 \times \text{emb}_{\text{dim}})]$ in the end. To discuss this in more detail, we need to look at the message passing framework used in the GNN model.

Message Passing Framework When we handle the graph data, we normally will conduct convolution operation in a neighbourhood aggregation or message passing scheme, which can be expressed as

$$x^{(k)}_i = \text{UPDATE} \left( x_i, \text{AGGR}_{j \in N(i)} \text{MESSAGE}^{(k)}(x^{(k-1)}_i, x^{(k-1)}_j, e_{ij}) \right)$$

, where $x^{(k-1)}_i$ denotes the state of the current node $i$ in the $(k-1)$th layer; $x^{(k-1)}_j$ denotes the state of the neighbour node $j$ in the $(k-1)$th layer; $e_{ij}$ denotes the edge features from node $j$ to node $i$; $N(i)$ denotes the neighbourhood set of node $i$ (i.e. 1-hop neighbours); MESSAGE denotes differentiable message generation function and it can be in various ways (i.e. Multi Layer Perceptrons), and obtain the embedding vector of each pair of the nodes and edge; AGGR denotes the aggregation function (i.e. sum, mean, max) which is differentiable permutation invariant; UPDATE denotes the state update function which usually conducts bias term adding, linear transformation or multi-head processing (i.e. concatenating) to update what we obtain from aggregation process.
In short, this framework has two steps: Gather the state message from neighbour nodes, and then generate the current node’s message by applying the specific aggregation function; based on the message so far, update the state of the target node.

In our work, the modification can be shown in two parts. In the MESSAGE part, we have

$$\Theta_{x_j} \rightarrow (\Theta_{x_j}) \| e_{ij}$$

, where \( \Theta \) is the weight matrix and "\( \| \)" means concatenation. We concatenate the edge representation \( e_{ij} \) to each corresponding node \( x_j \), and then in the UPDATE part, an edge update matrix will be multiplied to get the dimension we expect (i.e. the number of relationship types).

The graph-based models we used in this work: Graph Convolutional Networks (GCN) [17], Graph Attentional Networks (GAT) [32], Simple Graph Convolutional Networks (SGCN) [34], Attention-based Graph Neural Networks (AGNN) [29], and a special one, Relational Graph Convolutional Networks (RGCN) [25]. Noticed that RGCN does not need to be modified since it already considered the relationship between two entities, we put it here as it belongs to one kind of graph-based model.

### 3.3 L-Graph

Instead of conditioning the supporting facts to represent the query in the graph-based model, we process the facts and query independently. Inputs in the sequence-based encoder are the subject-predicate-object (SPO) triples by linearising a relational graph that extracted from supporting facts within a story, as in [21]. Actually supporting facts within a story can be seen as a reasoning path, we only pay attention to nodes(i.e. entities) and edges (i.e. relationship) among them. Therefore, subject and object correspond to two entities, and a predicate is a relationship between these two entities. After extracting all SPO triples in a story, we then directly put them (i.e. SPO sequence) into standard sequence models (i.e. CNN, RNN), since they are ordered and can be seen as a short/key sentence compared to the original story. Similarly, we represent the query into the SO sequence which similar to SPO sequence but without the predicate/target (i.e. relationship) and then put it into sequence models as well. At last, after concatenating and softmax function, we can predict the relationship by choosing the highest value.

To be specific. The sequence-based model does not process query conditioning on the supporting facts, in contrast, supporting facts and query are handled independently:

$$\hat{y} = softmax(W[emb(F) \| emb(Q)])$$

where "\( emb \)" denotes an embedding process working on a set of ground facts, such that the supporting facts \( (F) \) or a query \( (Q) \); "\( \| \)" means a concatenation of the two items; \( W \) represents weight matrix; and \( \hat{y} \) represents the distribution of all possible relationships we get after softmax function, and we will choose the highest probability relationship type as the predicted target.
Here, $emb(F)$ is from the result of processing SPO sequences with sequence encoding models, such as CNN and RNN, while $emb(Q)$ without predicate in the processing inputs (i.e. SO sequences). The shape of them both are $[B \times emb_{dim}^\ast]$, where $B$ is the batch size, $emb_{dim}^\ast$ is the embedding dimension for each story (consist of SPO sequences and SO sequences) based on sequence models, and $[B \times (2 \times emb_{dim}^\ast)]$ after concatenation. For example, when the LSTM [12] is used with hidden size = 100, we can get $[B \times 100]$ for both processes and get $[B \times 200]$ after concatenating, and if we use the Bidirectional Long Short-Term Memory Networks (Bi-LSTM) [30], $emb_{dim}^\ast$ will become twice of that in the LSTM.

We consider several sequence encoding models, namely Recurrent Neural Networks (RNN) [24], Long Short-Term Memory Networks (LSTM) [12], Gated Recurrent Units (GRU) [2], Bidirectional Recurrent Neural Networks (Bi-RNN) [26], Bidirectional Long Short-Term Memory Networks (Bi-LSTM) [30], Bidirectional Gated Recurrent Units (Bi-GRU) [33], Convolutional Neural Networks (CNN) [19], CNN with Highway Encoders (CNNH) [16], Multi-Headed Attention Networks (MHA) [31]. All related codes can be easily built up based on ”torch.nn” or the NLP research library, named ”AllenNLP” [7].

4 Experimental Results

We build up several baselines for evaluating the systematic generalisation and robust reasoning capacity on the pre-generated CLUTTR datasets. Models’ performance is comparing to each other, including ”text-based” models [27] mentioned in the paper. The prefix ”graph.” using in the sequence-based model (L-Graph) is to distinguish from the text-based model.

4.1 Systematic Generalisation Evaluation

We have conducted experiments on two pre-generated CLUTTR datasets, ”data_089907f8” and ”data_db9b8f0”, to compare each models’ systematic generalisation capacity. Due to limited pages, only partly results under maximising validation accuracy metric based on ”data_db9b8f0” are illustrated. Numerical details for all models under both minimising validation loss metric and maximising validation accuracy metric did not show here, as well as optimal hyperparameters.
Fig. 2. Systematic generalisation performance of graph-based models when trained on clauses of length $k = 2$, 3, 4.

In ascending order, the mean test accuracy of graph-based models are: 0.7255 (gcn), 0.7812 (sgcn), 0.7858 (gat), 0.8258 (rgcn) and 0.8442 (agnn); the test accuracy of clauses length $k = 10$ are: 0.3196 (gcn), 0.4591 (gat), 0.4975 (sgcn), 0.571 (rgcn) and 0.617 (rgcn).

The gap between two types of model is quite evident in Fig. 2. Graph-based models have more than 0.9 accuracies on clauses length 2, 3 and 4, and then drop significantly till clauses length 10. Among them, AGNN can be regarded as the best model; it obtains the highest value across most of the situations and even at length 10 with a figure of 0.617. Different from the monotonic decline in the graph-based models, over the length from 4 to 10, the text-based models’ testing accuracy remained level. Although the graph-based model performs much better than text-based in most scenarios, they decrease very fast and at most 0.6 accuracies can be obtained at length 10 and the worst one only about 0.3.
Fig. 3. Systematic generalisation performance of common sequence models when trained on clauses of length $k = 2, 3, 4$.

In ascending order, the mean test accuracy of sequence-based models are: 0.3023 (graph_cnn), 0.3265 (graph_cnnh), 0.3764 (graph_boe), 0.9447 (graph_lstm), 0.9571 (graph_rn) and 0.9721 (graph_gru); the test accuracy of clauses length $k = 10$ are: 0.1473 (graph_cnn), 0.1509 (graph_boe), 0.171 (graph_cnnh), 0.8679 (graph_lstm), 0.9116 (graph_rn) and 0.9241 (graph_gru).

RNN, LSTM and GRU perform much stable with high test accuracy in Fig. 3 (trained on clauses of length $k = 2, 3, 4$), with nearly 0.9 test accuracy on clauses length 10 stories. However, BOE, CNN and CNNH decline in an unbelievable speed, from nearly 1.0 test accuracy on clauses length 2 to only about 0.1 on clauses length 3, while test-based models do not show much difference.

4.2 Robust Reasoning Evaluation

We also have conducted experiments on four pre-generated CLUTRR datasets, "data_7c5b0e70", "data_06b8f2a1", "data_523348e6" and "data_d83ecc3e", to compare each models’ robust reasoning capacity. Numerical details for all models
under both minimising validation loss metric and maximising validation accuracy metric did not show here, and optimal hyperparameters as well.

5 Conclusions

In this paper, we implemented and evaluated two neuro-symbolic reasoning architectures for testing systematic generalisation and inductive reasoning capabilities of NLU systems, namely the E-GNN (Graph-based) model and the L-Graph (Sequence-based) model. Both two types of models were trained on the CLUTRR datasets, and we demonstrated and evaluated quite a lot of experiments results in both generalisation and robustness aspects.

Most models can perform well with fewer clauses length inputs, however, as the length of clauses increasing in the test stage, all models’ performance decline monotonically, especially for graph-based models, which proves the challenge of "zero-shot" systematic generalisation [18,28]. Among them, those sequence-based models with recurrent neural networks outperform the graph-based model, the typical sequence model, and the multi-head attention networks model when considering generalisation capacity, with around 90% on clauses length 10 when trained over clauses of length 2, 3 and 4. Despite that, their robustness performance cannot as good as that of graph-based models and sometimes even fail to obtain any correct kinship.

These results highlight that graph-based models with attention architectures can catch the patterns of the data and prevent from diverse noise influence; sequence-based models can make good use of SPO sequences to predict the kinship between entities; however, we can also notice that there is a fatal flaw in this strategy—very sensitive to the order of sequences; bidirectional settings and multi-heads attention architecture did not have much more advantage than the original ones; although models were trained on data with noise added, most of them may be able to get some linguistic cues (i.e. gender) and predict well under that type of noise scenarios on clauses of length $k = 2$, but testing on clauses of length $k = 3$ is still a challenge for some models.

Structured input plays a significant role in all those models. Compared with text-based models, the performance of models with structured inputs improves significantly, especially for models with recurrent neural networks, such as LSTM and GRU, show us an unbelievable accuracy during the systematic generalisation tests. Moreover, this phenomenon also proves that there is a gap between reasoning models trained on structured input or unstructured natural language.

References

1. Bahdanau, D., Murty, S., Noukhovitch, M., Nguyen, T.H., de Vries, H., Courville, A.: Systematic generalization: What is required and can it be learned? In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=RkezXnA9YX
2. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Stroudsburg, PA, USA (2019). https://doi.org/10.18653/v1/N19-1423

4. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research 61, 1–64 (2018)

5. Gan, C., Li, Y., Li, H., Sun, C., Gong, B.: Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1811–1820 (2017)

6. d’Avila Garcez, A.S., Besold, T.R., Raedt, L.D., Földiák, P., Hitzer, P., Icard, T., Kühlberger, K., Lamb, L.C., Miikkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: Contributions and challenges. AAAI Press (2015)

7. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.: AllenNLP: A deep semantic natural language processing platform. In: Proceedings of Workshop for NLP Open Source Software (NLP-OSS). pp. 1–6. Association for Computational Linguistics, Melbourne, Australia (Jul 2018). https://doi.org/10.18653/v1/W18-2501

8. Garnelo, M., Shanahan, M.: Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. Current Opinion in Behavioral Sciences 29, 17–23 (2019)

9. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

10. Gupta, N., Lin, K., Roth, D., Singh, S., Gardner, M.: Neural module networks for reasoning over text. arXiv preprint arXiv:1912.04971 (2019)

11. Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., Smith, N.A.: Annotation artifacts in natural language inference data. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). p. 107–112. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/N18-2017

12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (Nov 1997). https://doi.org/10.1162/neco.1997.9.8.1735

13. Hudson, D., Manning, C.D.: Learning by abstraction: The neural state machine. In: Advances in Neural Information Processing Systems. pp. 5903–5916 (2019)

14. Hudson, D.A., Manning, C.D.: Compositional attention networks for machine reasoning. arXiv:1803.03067 [cs] (Apr 2018), http://arxiv.org/abs/1803.03067

15. Jia, R., Liang, P.: Adversarial examples for evaluating reading comprehension systems. CoRR abs/1707.07328 (2017), http://arxiv.org/abs/1707.07328

16. Kim, Y., Jeon, Y., Son, T., Rush, A.M.: Character-aware neural language models. arXiv preprint arXiv:1508.06615 (2015)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Lake, B., Baroni, M.: Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In: International Conference on Machine Learning. pp. 2873–2882. PMLR (2018)
19. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1, 541–551 (1989)
20. Mao, J., Gan, C., Kohli, P., Tenenbaum, J.B., Wu, J.: The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. arXiv preprint arXiv:1904.12584 (2019)
21. Minervini, P., Riedel, S., Stenetorp, P., Grefenstette, E., Rocktäschel, T.: Learning reasoning strategies in end-to-end differentiable proving. arXiv preprint arXiv:2007.06477 (2020)
22. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. CoRR abs/1802.05365 (2018), http://arxiv.org/abs/1802.05365
23. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog 1(8), 9 (2019)
24. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (Oct 1986), https://doi.org/10.1038/323533a0, https://www.nature.com/articles/323533a0
25. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. pp. 593–607. Springer (2018)
26. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing 45(11), 2673–2681 (1997)
27. Sinha, K., Sodhani, S., Dong, J., Pineau, J., Hamilton, W.L.: Clutrr: A diagnostic benchmark for inductive reasoning from text. arXiv preprint arXiv:1908.06177 (2019)
28. Sodhani, S., Chandar, S., Bengio, Y.: On training recurrent neural networks for lifelong learning. CoRR, abs/1811.07017 (2018)
29. Thekumparampil, K.K., Wang, C., Oh, S., Li, L.J.: Attention-based graph neural network for semi-supervised learning. arXiv preprint arXiv:1803.03735 (2018)
30. Thireou, T., Reczko, M.: Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. IEEE/ACM transactions on computational biology and bioinformatics 4(3), 441–446 (2007)
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
32. Velicković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
33. Vukotić, V., Raymond, C., Gravier, G.: A step beyond local observations with a dialog aware bidirectional GRU network for Spoken Language Understanding. In: Interspeech. San Francisco, United States (Sep 2016), https://hal.inria.fr/hal-01351733
34. Wu, F., Zhang, T., Souza Jr, A.H.d., Fifty, C., Yu, T., Weinberger, K.Q.: Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153 (2019)
35. Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P., Tenenbaum, J.: Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In: Advances in neural information processing systems. pp. 1031–1042 (2018)
A Kinship and Knowledge Base

| Kinship         |ｵﾊﾞﾔｼｹｲﾀﾞ | ﾆｯｸﾏｻﾔ | ﾁﾇﾀｹｼｮｳ | ﾄﾘﾊﾗｹｲﾀﾞ |
|-----------------|-------------|-------------|-------------|-------------|
| Grandfather     | Grandmother | Father      | Mother      |             |
| Uncle           | Aunt        | Father-in-law| Mother-in-law|
| Brother-in-law   | Sister-in-law| Husband     | Wife        |             |
| Brother         | Sister      | Nephew      | Niece       |             |
| Son             | Daughter    | Son-in-law  | Daughter-in-law |
| Grandson        | Granddaughter |          |             |             |
Table 2. Knowledge Base (KB) used in the CLUTRR

| Head (H = [α*]) | Body (B = [α1, α2]) |
|-----------------|---------------------|
| [grand, X, Z]  | [[child, X, Y], [child, Y, Z]] |
| [grand, X, Z]  | [[SO, X, Y], [grand, Y, Z]] |
| [grand, X, Z]  | [[grand, X, Y], [sibling, Y, Z]] |
| [inv-grand, X, Z] | [[inv-child, X, Y], [inv-child, Y, Z]] |
| [inv-grand, X, Z] | [[sibling, X, Y], [inv-grand, Y, Z]] |
| [child, X, Z]  | [[child, X, Y], [sibling, Y, Z]] |
| [child, X, Z]  | [[SO, X, Y], [child, Y, Z]] |
| [inv-child, X, Z] | [[sibling, X, Y], [inv-child, Y, Z]] |
| [inv-child, X, Z] | [[child, X, Y], [inv-grand, Y, Z]] |
| [sibling, X, Z] | [[child, X, Y], [inv-un, Y, Z]] |
| [sibling, X, Z] | [[inv-child, X, Y], [child, Y, Z]] |
| [sibling, X, Z] | [[sibling, X, Y], [sibling, Y, Z]] |
| [in-law, X, Z] | [[child, X, Y], [SO, Y, Z]] |
| [inv-in-law, X, Z] | [[SO, X, Y], [inv-child, Y, Z]] |
| [un, X, Z]     | [[sibling, X, Y], [child, Y, Z]] |
| [inv-un, X, Y] | [[inv-child, X, Y], [sibling, Y, Z]] |

B Datasets

Table 3. Details of pre-generated datasets in the CLUTRR

| Type          | Dataset     | Training         | Testing          |
|---------------|-------------|------------------|------------------|
| Generalization| data_069907F8 | Clean (k = 2, 3) | Clean (k = 2, 3, ... 10) |
|              | data_65986F4 | Clean (k = 2, 3, 4) | Clean (k = 2, 3, ... 10) |
| Robustness    | data_7c9d6e70 | Clean (k = 2, 3) | ALL (k = 2, 3) |
|              | data_068f2f21 | Supporting (k = 2, 3) | ALL (k = 2, 3) |
|              | data_52338e65 | Irrelevant (k = 2, 3) | ALL (k = 2, 3) |
|              | data_833ec3e6 | Disconnected (k = 2, 3) | ALL (k = 2, 3) |

k: clause length  ALL: clean, supporting, irrelevant and disconnected
Table 4. An example in the CLUTRR datasets

|       |                                                                 |
|-------|-----------------------------------------------------------------|
| id    | f4161421-bf6e-4165-9133-07f1d0040e87e                           |
| story | [Dorothy]'s brother [Michael] and her went to get ice cream. [Michael] is the proud father of the lovely [Donald] |
| query | "(Donald', 'Dorothy')"                                         |
| target| aunt                                                             |
| text_query | [Donald] asked his aunt [Dorothy] what college he should attend.' |
| clean_story | [Michael] is the proud father of the lovely [Donald] [Dorothy]'s brother [Michael] and her went to get ice cream. |
| proof_state | [{{'Donald', 'aunt', 'Dorothy'}}, {{'Donald', 'father', 'Michael'}, ('Michael', 'sister', 'Dorothy')}] |
| f_comb | father-sister                                                   |
| task_name | task_1.2                                                     |
| story_edges | [(0, 1), (1, 2)]                                               |
| edge_types | ['father', 'sister']                                           |
| query_edge | (0, 2)                                                         |
| genders | Donald: male, Michael: male, Dorothy: female                   |
| syn_story |                                                              |
| node_mapping | {7: 0, 2: 1, 3: 2}                                           |
| task_split | train                                                          |

C Abbreviations

For brevity, we use some abbreviations to represent hyperparameters and models in our work and code. The prefix "graph" using in the sequence-based model is to distinguish from the text-based model.
Table 5. The Abbreviations in this work

| Type          | Notation   | Meaning   | Notation   | Meaning |
|---------------|------------|-----------|------------|---------|
| **Graph-based** | gcn        | GCN       | gat        | GAT     |
|               | sgcnc      | SGCN      | agnn       | AGNN    |
|               | rgcn       | RGCN      |            |         |
| **Sequence-based** | graph_boe | BOE       | graph_cnn  | CNN     |
|               | graph_cnnh | CNNH      | graph_rnn  | RNN     |
|               | graph_lstm | LSTM      | graph_gru  | GRU     |
|               | graph_birnn| Bi-RNN    | graph_bilstm| Bi-LSTM |
|               | graph_bigru| Bi-GRU    | graph_intra| MHA     |
|               | graph_multihead | MHA | graph_stack | MHA     |
| **Text-based**   | rn         | RN        | mac        | MAC     |
|               | bilstm     | Bi-LSTM   | bilstm_mean| Bi-LSTM |
|               | bilstm_atten | Bi-LSTM |            |         |
| **Hyper-parameter** | ned   | The dimension of node embedding |         |         |
|               | eed        | The dimension of edge embedding |         |         |
|               | hd         | The dimension of hidden size   |         |         |
|               | fi         | The number of filters          |         |         |
|               | hi         | The number of highway layers   |         |         |
|               | he         | The number of heads            |         |         |
|               | ep         | The number of epochs           |         |         |
## D Systematic Generalisation

### D.1 Clauses of Length $k = 2, 3$

**Table 6.** Optimal hyperparameters for training on clauses of length $k = 2, 3$ under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss | Model | Max val acc |
|--------------|-------|-------------|
| ned | eed | hd | fi | hi | he | ep | ned | eed | hd | fi | hi | he | ep |
| 100 | 1000 | 100 |
| 100 | 1500 | 3 | 100 |
| 50 | 1000 | 100 |
| 1000 | 500 | 100 |
| 500 | 200 | 100 |
| 1000 | 1000 | 100 |
| 10 | 10 | 10 | 100 |
| 1500 | 1500 | 6 | 100 |
| 100 | 100 | 256 | 100 |
| 500 | 500 | 256 | 100 |
| 2000 | 2000 | 1024 | 100 |
| 2000 | 2000 | 256 | 100 |
| 2000 | 2000 | 512 | 100 |
| 2000 | 2000 | 1024 | 100 |
| 2000 | 2000 | 128 | 100 |
| 2000 | 2000 | 512 | 10 | 100 |
| 2000 | 2000 | 1024 | 4 | 100 |
Table 7. Training on clauses of length $k = 2, 3$ under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Method       | n_train | n_test | Mean test accuracy | 1 test accuracy | 2 test accuracy | 3 test accuracy | 4 test accuracy | 5 test accuracy | 6 test accuracy | 7 test accuracy | 8 test accuracy | 9 test accuracy | 10 test accuracy |
|--------------|---------|--------|--------------------|----------------|----------------|----------------|---------------|----------------|----------------|----------------|----------------|----------------|----------------|-----------------|
| gnn          | 0.008   | 0.9944 | 0.9803             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| gat          | 0.0017  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| sggcn        | 0.0024  | 0.9979 | 0.9608             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| sggcn-mv     | 0.0024  | 0.9979 | 0.9608             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_bsc    | 0.0012  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_min    | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_mhgr   | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_min    | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_multit  | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| graph_stack  | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| m            | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| mac          | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| bilstm        | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
| bilstm_attn  | 0.0032  | 0.9945 | 0.9809             | 0.9582         | 0.7923         | 0.6285         | 0.5129        | 0.392          | 0.3385         | 0.3666         | 0.2013         | 0.0127         | 0.0141          |
Table 8. Training on clauses of length $k = 2, 3$ under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model   | Mean test acc | 2 test acc | 3 test acc | 4 test acc | 5 test acc | 6 test acc | 7 test acc | 8 test acc | 9 test acc | 10 test acc |
|---------|---------------|------------|------------|------------|------------|------------|------------|------------|------------|-------------|
| gpn     | 0.913         | 0.5640     | 0.9775     | 0.7798     | 0.6306     | 0.5986     | 0.3924     | 0.2875     | 0.3637     | 0.2972      |
| gsp     | 0.962         | 0.6831     | 0.9731     | 0.8431     | 0.7233     | 0.5876     | 0.4046     | 0.4203     | 0.2860     | 0.2806      |
| spcn    | 0.815         | 0.6125     | 0.95       | 0.7473     | 0.6001     | 0.5095     | 0.4556     | 0.4285     | 0.4762     | 0.3664      |
| spn     | 0.944         | 0.9567     | 0.9716     | 0.9625     | 0.9821     | 0.7004     | 0.6487     | 0.4707     | 0.4809     | 0.3299      |
| rpn     | 0.913         | 0.6431     | 0.9665     | 0.7808     | 0.6721     | 0.5486     | 0.4605     | 0.4373     | 0.4322     | 0.4003      |
| graph_hce | 0.4481       | 0.3123     | 0.8005     | 0.4906     | 0.2421     | 0.204      | 0.1942     | 0.0885     | 0.0830     | 0.1221      |
| graph_cnn | 0.7149       | 0.3272     | 0.5125     | 0.3928     | 0.2891     | 0.3465     | 0.232       | 0.229      | 0.2877     | 0.1400      |
| graph_rnn | 0.7003       | 0.3245     | 0.4205     | 0.3246     | 0.2382     | 0.1395     | 0.1676     | 0.1835     | 0.1712     | 0.0554      |
| graph_mn | 0.9022       | 0.9161     | 0.8946     | 0.8441     | 0.7426     | 0.6156     | 0.504       | 0.4653     | 0.4663     | 0.3072      |
| graph_tlm | 0.9047       | 0.8667     | 0.95       | 0.8743     | 0.9603     | 0.8752     | 0.813       | 0.796      | 0.7462     | 0.6505      |
| graph_gru | 0.9261       | 0.9207     | 0.9352     | 0.9727     | 0.9217     | 0.8495     | 0.785       | 0.730      | 0.6237     | 0.3632      |
| graph_bas | 0.7979       | 0.7495     | 0.9775     | 0.9222     | 0.9014     | 0.7222     | 0.6311     | 0.647      | 0.6236     | 0.4053      |
| graph_hstan | 0.7979      | 0.7495     | 0.9775     | 0.9222     | 0.9014     | 0.7222     | 0.6311     | 0.647      | 0.6236     | 0.4053      |
| graph_hlstm | 0.8094     | 0.7731     | 0.9625     | 0.9572     | 0.9951     | 0.7674     | 0.6415     | 0.6685     | 0.5464     | 0.5229      |
| graph_intra | 0.9021      | 0.9028     | 0.9256     | 0.6794     | 0.5891     | 0.4969     | 0.4255     | 0.5255     | 0.4708     | 0.3776      |
| graph_multih | 0.8022      | 0.8631     | 0.9632     | 0.7882     | 0.6757     | 0.5775     | 0.477       | 0.4202     | 0.2880     | 0.1932      |
| graph_stack | 0.7609     | 0.9756     | 0.9537     | 0.9182     | 0.9069     | 0.7724     | 0.6721     | 0.666       | 0.5884     | 0.4608      |
| m       | 1.1311       | 0.3899     | 0.5042     | 0.5974     | 0.4215     | 0.4093     | 0.3294     | 0.3111     | 0.28       | 0.2803      |
| mac     | 1.0877       | 0.4832     | 0.5042     | 0.4631     | 0.3766     | 0.2699     | 0.2283     | 0.2032     | 0.2232     | 0.2309      |
| blstn   | 2.9044       | 0.7677     | 0.8147     | 0.7344     | 0.6441     | 0.5733     | 0.5169     | 0.4752     | 0.2984     | 0.2673      |
| blstn_moe | 1.5513      | 0.7897     | 0.9758     | 0.8709     | 0.7528     | 0.6974     | 0.6268     | 0.5964     | 0.2792     | 0.2939      |
| blstn_att | 1.4113      | 0.8104     | 0.9474     | 0.7158     | 0.5206     | 0.4302     | 0.304      | 0.2699     | 0.2809     | 0.2072      |
D.2 Clauses of Length $k = 2, 3, 4$

Table 9. Optimal hyperparameters for training on clauses of length $k = 2, 3, 4$ under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss | Model  | Max val acc |
|--------------|--------|-------------|
| ned  eed hd fi hi he ep | gcn 100 1500 | ned  eed hd fi hi he ep |
| 50 1500      | 100 gcn | 100 1500     |
| 200 200      | 8 100 gat | 200 500 10 100 |
| 50 1000      | 100 sgc | 200 100 |
| 100 200      | 100 agnn | 100 100 |
| 100 50       | 100 rgcn | 100 100 |
| 100 100      | 100 graph_bow | 100 100 |
| 1500 1500 10 | 100 graph_cnn | 100 1000 100 |
| 10 10 0 100  | 100 graph_cnnh | 10 10 0 100 |
| 500 500 256  | 100 graph_rnn | 100 1000 1024 100 |
| 100 100 1024 | 100 graph_lstm | 10 10 32 100 |
| 2000 2000 1024 | 100 graph_gru | 50 50 32 100 |
| 1500 1500 128 | 100 graph_birnn | 1500 1500 128 100 |
| 200 200 1024 | 100 graph_bilstm | 100 100 32 100 |
| 50 50 512  | 100 graph_bigru | 10 10 64 100 |
| 2000 2000 128 | 100 graph_intra | 2000 2000 128 100 |
| 1000 1000 32 | 8 100 graph_multihead | 1000 1000 32 8 100 |
| 2000 2000 1024 | 8 100 graph_stack | 2000 2000 1024 8 100 |
Table 10. Training on clauses of length $k = 2, 3, 4$ under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

|        | Mean | train accuracy | 2 test accuracy | 3 test accuracy | 4 test accuracy | 5 test accuracy | 6 test accuracy | 7 test accuracy | 8 test accuracy | 9 test accuracy | 10 test accuracy |
|--------|------|----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| gcn    | 0.788| (0.702)        | 0.823           | 0.790           | 0.769           | 0.748           | 0.731           | 0.731           | 0.729           | 0.731           | 0.731           |
| gat    | 0.791| (0.702)        | 0.823           | 0.790           | 0.769           | 0.748           | 0.731           | 0.731           | 0.729           | 0.731           | 0.731           |
| sgcn   | 0.842| (0.702)        | 0.823           | 0.790           | 0.769           | 0.748           | 0.731           | 0.731           | 0.729           | 0.731           | 0.731           |
| afni   | 0.852| (0.702)        | 0.823           | 0.790           | 0.769           | 0.748           | 0.731           | 0.731           | 0.729           | 0.731           | 0.731           |
| rgn    | 0.612| (0.702)        | 0.823           | 0.790           | 0.769           | 0.748           | 0.731           | 0.731           | 0.729           | 0.731           | 0.731           |

**graph_loss**
- 0.670 (0.702)
- 0.670 (0.702)
- 0.670 (0.702)

**graph_cnn**
- 0.815 (0.702)
- 0.815 (0.702)
- 0.815 (0.702)

**graph_cnh**
- 0.825 (0.702)
- 0.825 (0.702)
- 0.825 (0.702)

**graph_rnn**
- 0.832 (0.702)
- 0.832 (0.702)
- 0.832 (0.702)

**graph_lstm**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_gru**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_bim**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_bilstm**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_birnn**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_bilstm**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_birnn**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**graph_stack**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**m**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**nac**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**blstn**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**blstn_mae**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)

**blstn_attn**
- 0.835 (0.702)
- 0.835 (0.702)
- 0.835 (0.702)
Table 11. Training on clauses of length $k = 2, 3, 4$ under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. |
|---|---|---|---|---|---|---|---|---|---|
| gnn | 0.7658 (0.0040) | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) |
| gat | 0.5962 (0.0062) | 0.7658 (0.0040) | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) |
| sagnn | 0.7658 (0.0040) | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) |
| rgnn | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) | 0.1864 (0.0050) |
| graph_how | 0.7658 (0.0040) | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) |
| graph_con | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) |
| graph_cnn | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) | 0.1864 (0.0050) |
| graph_rnn | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) |
| graph_bilstm | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) |
| graph_lstm | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) | 0.1864 (0.0050) |
| graph_multi_head | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) |
| graph_stack | 0.9034 (0.0012) | 0.8018 (0.0012) | 0.5658 (0.0040) | 0.5149 (0.0040) | 0.4618 (0.0040) | 0.3156 (0.0052) | 0.2776 (0.0050) | 0.2437 (0.0050) | 0.2092 (0.0050) |
| m | 1.3372 (0.0321) | 0.7163 (0.0034) | 0.3349 (0.0006) | 0.1568 (0.0024) | 0.0561 (0.0012) | 0.0053 (0.0012) | 0.0004 (0.0012) | 0.0003 (0.0012) | 0.0002 (0.0012) |
| mac | 1.5846 (0.0130) | 0.7456 (0.0034) | 0.3349 (0.0006) | 0.1568 (0.0024) | 0.0561 (0.0012) | 0.0053 (0.0012) | 0.0004 (0.0012) | 0.0003 (0.0012) | 0.0002 (0.0012) |
| bilstm | 2.7936 (0.0542) | 0.6794 (0.0034) | 0.3349 (0.0006) | 0.1568 (0.0024) | 0.0561 (0.0012) | 0.0053 (0.0012) | 0.0004 (0.0012) | 0.0003 (0.0012) | 0.0002 (0.0012) |
| blstm_attention | 1.8526 (0.0542) | 0.7456 (0.0034) | 0.3349 (0.0006) | 0.1568 (0.0024) | 0.0561 (0.0012) | 0.0053 (0.0012) | 0.0004 (0.0012) | 0.0003 (0.0012) | 0.0002 (0.0012) |
## E Robust Reasoning

### E.1 Clean Facts without Noise

Table 12. Optimal hyperparameters for training on the clean facts without noise under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss | Model  | Max val acc |
|--------------|--------|-------------|
| *ned* | *eed* | *hd* | *fl* | *hi* | *he* | *ep* | *ned* | *eed* | *hd* | *fl* | *hi* | *he* | *ep* |
| 100 | 1500 | 100 | gcn | 100 | 200 | 100 |
| 500 | 200 | 6 | 100 | gat | 500 | 100 | 10 |
| 1000 | 2000 | 100 | sgcn | 200 | 200 | 100 |
| 200 | 512 | 100 | agnn | 500 | 64 | 100 |
| 500 | 32 | 100 | rgcn | 200 | 500 | 100 |
| 1000 | 1000 | 100 | graph boe | 50 | 50 | 100 |
| 10 | 10 | 8 | 100 | graph_cnn | 10 | 10 | 8 | 100 |
| 1000 | 1000 | 10 | 100 | graph cnnh | 1500 | 1500 | 8 | 100 |
| 2000 | 2000 | 128 | 100 | graph rnn | 2000 | 2000 | 128 | 100 |
| 2000 | 2000 | 512 | 100 | graph lstm | 1000 | 1000 | 1024 | 100 |
| 1500 | 1500 | 1024 | 100 | graph gru | 50 | 50 | 1024 | 100 |
| 1000 | 1000 | 256 | 100 | graph birmn | 500 | 500 | 256 | 100 |
| 1500 | 1500 | 256 | 100 | graph_blstm | 1500 | 1500 | 1024 | 100 |
| 200 | 200 | 1024 | 100 | graph_bigru | 2000 | 2000 | 512 | 100 |
| 2000 | 2000 | 128 | 100 | graph_intra | 2000 | 2000 | 128 | 100 |
| 1500 | 1500 | 32 | 6 | 100 | graph_multihed | 1000 | 1000 | 32 | 5 | 100 |
| 2000 | 2000 | 1024 | 1 | 100 | graph_stack | 2000 | 2000 | 1024 | 4 | 100 |
Table 13. Training on the clean facts without noise under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Method   | 1,2,3_trail_n_loss | 1,2,3_trail_n_acc | Mean_test_accuracy | 1_test_acc | 2_test_acc | 3_test_acc | 3,3_test_acc | 4,3_test_acc |
|----------|---------------------|-------------------|--------------------|------------|------------|------------|--------------|--------------|
| gcnn     | 0.0062 (0.0051)     | 0.9953 (0.0015)   | 0.8017 (0.0023)    | 1.0 (1.0)  | 1.0 (1.0)  | 0.5025 (0.2964) | 0.776 (0.1222) | 0.72 (0.1174) |
| gat      | 0.01 (0.0016)       | 0.9943 (0.0015)   | 0.9655 (0.0025)    | 1.0 (1.0)  | 1.0 (1.0)  | 0.8805 (0.2205) | 0.962 (0.0535) | 0.985 (0.0474) |
| sgcn     | 0.0065 (0.0021)     | 0.9963 (0.0014)   | 0.839 (0.0059)     | 1.0 (1.0)  | 1.0 (1.0)  | 0.339 (0.2739) | 0.94 (0.0773) | 0.916 (0.0579) |
| agnn     | 0.0082 (0.0014)     | 0.9957 (0.0012)   | 0.9707 (0.0043)    | 1.0 (1.0)  | 1.0 (1.0)  | 0.8535 (0.2217) | 1.0 (1.0)    | 1.0 (1.0)    |
| rgcn     | 0.0075 (0.0011)     | 0.9959 (0.001)    | 0.9862 (0.0067)    | 1.0 (1.0)  | 1.0 (1.0)  | 0.991 (0.0205) | 1.0 (1.0)    | 1.0 (1.0)    |
| graph_boe | 0.4112 (0.008)     | 0.7392 (0.0072)   | 0.2699 (0.0147)    | 0.5 (0.0)  | 0.4075 (0.10) | 0.021 (0.0339) | 0.246 (0.0709) | 0.175 (0.0474) |
| graph_cnn | 0.6933 (0.0286)     | 0.6662 (0.0161)   | 0.2674 (0.1154)    | 0.15 (0.2415) | 0.329 (0.0655) | 0.371 (0.3142) | 0.244 (0.1077) | 0.243 (0.1012) |
| graph_cnn | 0.6993 (0.0197)     | 0.6732 (0.0149)   | 0.399 (0.0787)     | 0.5 (0.0)  | 0.205 (0.0237) | 0.6045 (0.1939) | 0.327 (0.1122) | 0.318 (0.1147) |
| graph_rnn | 0.0134 (0.0076)     | 0.9944 (0.0023)   | 0.4 (0.0)          | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_lstm | 0.0065 (0.0013)    | 0.9955 (0.0013)   | 0.4 (0.0)          | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_gru | 0.0094 (0.0032)     | 0.9965 (0.0014)   | 0.4 (0.0)          | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_bimn | 0.0148 (0.0019)    | 0.994 (0.002)     | 0.4064 (0.0265)    | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.013 (0.0411) | 0.029 (0.0597) |
| graph_bilstm | 0.0063 (0.0011)   | 0.9957 (0.0013)   | 0.4 (0.0)          | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_bignn | 0.0099 (0.0057)   | 0.9952 (0.0023)   | 0.3966 (0.0044)    | 1.0 (0.0)  | 0.953 (0.021) | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_intra | 0.0052 (0.0017)   | 0.9709 (0.0025)   | 0.9693 (0.0094)    | 1.0 (0.0)  | 0.979 (0.0472) | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_multihead | 0.0104 (0.0039)   | 0.994 (0.0019)    | 0.4 (0.0)          | 1.0 (0.0)  | 1.0 (0.0)  | 0.0 (0.0) | 0.0 (0.0) | 0.0 (0.0) |
| graph_stack | 0.0067 (0.0027)   | 0.970 (0.0052)    | 0.9697 (0.0111)    | 0.9095 (0.0136) | 0.9812 (0.0226) | 0.002 (0.0005) | 0.013 (0.0225) | 0.000 (0.0227) |

| m         | 0.8925 (0.0306)     | 0.7378 (0.0067)   | 0.4557 (0.0047)    | 0.0965 (0.0085) | 0.561 (0.0586) | 0.5688 (0.0975) | 0.6018 (0.0803) | 0.4643 (0.0809) |
| mac       | 0.7389 (0.0302)     | 0.719 (0.0067)    | 0.4008 (0.0473)    | 0.2201 (0.1327) | 0.5064 (0.0424) | 0.5905 (0.0701) | 0.5888 (0.0722) | 0.4962 (0.0519) |
| blstm     | 0.89 (0.024)        | 0.6493 (0.0135)   | 0.457 (0.0031)     | 0.1453 (0.0974) | 0.4014 (0.0788) | 0.5429 (0.0705) | 0.5086 (0.0739) | 0.435 (0.0791) |
| blstm_mean | 0.8241 (0.0259)     | 0.6992 (0.0173)   | 0.3785 (0.0066)    | 0.1237 (0.0759) | 0.4595 (0.0811) | 0.3523 (0.2345) | 0.513 (0.0639) | 0.443 (0.0536) |
| blstm_atten | 0.7814 (0.0278)    | 0.7161 (0.0268)   | 0.4415 (0.0641)    | 0.1895 (0.0065) | 0.442 (0.0932) | 0.614 (0.2402) | 0.526 (0.0300) | 0.436 (0.0606) |
Table 14. Training on the clean facts without noise under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| method  | Mean test accuracy | 1.2 test acc | 1.3 test acc | 2.3 test acc | 3.3 test acc | 4.3 test acc |
|---------|--------------------|--------------|--------------|--------------|--------------|--------------|
| gen     | 0.996 (±0.0109)    | 0.956 (±0.0009) | 0.964 (±0.007) | 1.0 (±0.0)  | 0.947 (±0.037) | 0.891 (±0.0203) |
| gat     | 0.999 (±0.0202)    | 0.996 (±0.0007) | 0.996 (±0.0107) | 1.0 (±0.0)  | 0.986 (±0.0203) | 0.985 (±0.0147) |
| sgn     | 0.996 (±0.0023)    | 0.996 (±0.0007) | 0.993 (±0.0093) | 1.0 (±0.0)  | 0.986 (±0.0419) | 0.985 (±0.0221) |
| sgnm    | 0.994 (±0.0065)    | 0.996 (±0.0007) | 0.967 (±0.0069) | 1.0 (±0.0)  | 0.865 (±0.046)  | 0.992 (±0.0523) |
| rgnn    | 0.991 (±0.0010)    | 0.994 (±0.0009) | 0.848 (±0.0272) | 1.0 (±0.0)  | 0.424 (±0.2059) | 1.0 (±0.0)  |

graph_bro: 0.448 (±0.0092)  0.751 (±0.0035)  0.328 (±0.0289)  0.5 (±0.0)  0.512 (±0.0071)  0.0 (±0.0)  0.722 (±0.0074)  0.23 (±0.0077)
graph_cnn: 0.7047 (±0.0088)  0.6735 (±0.0141)  0.3946 (±0.0261)  0.2 (±0.0)  0.321 (±0.0067)  0.414 (±0.0992)  0.266 (±0.1345)  0.272 (±0.1073)
graph_cnnh: 0.6881 (±0.0029)  0.6797 (±0.0143)  0.3134 (±0.0133)  0.5 (±0.0)  0.178 (±0.0026)  0.366 (±0.0305)  0.27 (±0.1156)  0.23 (±0.1049)
graph_rnn: 0.694 (±0.0093)  0.956 (±0.0016)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_lstm: 0.689 (±0.0091)  0.996 (±0.0014)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_gru: 0.586 (±0.0099)  0.962 (±0.0015)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_binn: 0.915 (±0.0015)  0.992 (±0.0017)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_bilstm: 0.307 (±0.0042)  0.991 (±0.0011)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_bigru: 0.3011 (±0.0021)  0.996 (±0.0011)  0.3988 (±0.0004)  1.0 (±0.0)  0.963 (±0.0038)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_intra: 0.303 (±0.0029)  0.991 (±0.0024)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_multi: 0.3129 (±0.0027)  0.996 (±0.0012)  0.4 (±0.0)  1.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)  0.0 (±0.0)
graph_stack: 0.0295 (±0.0019)  0.9785 (±0.0033)  0.3931 (±0.0077)  0.9181 (±0.0279)  0.9265 (±0.0034)  0.0015 (±0.0048)  0.003 (±0.0095)

rm: 1.0081 (±0.0129)  0.792 (±0.0407)  0.4978 (±0.035)  0.1653 (±0.0123)  0.575 (±0.0075)  0.5945 (±0.0035)  0.6312 (±0.0065)  0.5323 (±0.0055)
mac: 1.0654 (±0.1114)  0.829 (±0.0410)  0.5687 (±0.0373)  0.3886 (±0.0474)  0.5807 (±0.0643)  0.584 (±0.0635)  0.6177 (±0.0854)  0.625 (±0.0494)
bilstm: 2.0409 (±0.3185)  0.7577 (±0.0335)  0.523 (±0.0407)  0.3047 (±0.0382)  0.4729 (±0.0494)  0.5061 (±0.0869)  0.513 (±0.0756)  0.469
bilstm mean: 1.3045 (±0.1348)  0.806 (±0.0283)  0.4089 (±0.0505)  0.4132 (±0.1236)  0.35 (±0.0615)  0.3215 (±0.1672)  0.497 (±0.0572)  0.439
bilstmatten: 1.2706 (±0.1068)  0.624 (±0.01)  0.4945 (±0.0569)  0.4974 (±0.0758)  0.414 (±0.0800)  0.6125 (±0.2328)  0.5 (±0.0656)  0.448 (±0.0573)
E.2 Supporting Facts Noise

Table 15. Optimal hyperparameters for training on the supporting facts noise under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss | Model  | Max val acc |
|--------------|--------|-------------|
| ned  eed  hd  fi  hi  he  ep | | ned  eed  hd  fi  hi  he  ep |
| 50  500 | 100 | gcn | 50  500 | 100 |
| 500  100 | 6  100 | gat | 500  10 | 8  100 |
| 100  512 | 100 | sgcnn | 50  1024 | 100 |
| 200  1024 | 100 | agnn | 200  256 | 100 |
| 500  32 | 100 | rgcn | 50  500 | 100 |
| 1500  1500 | 100 | graph_boe | 200  200 | 100 |
| 10  10  8 | 100 | graph_cnn | 500  500 | 6  100 |
| 1500  1500 | 8 | graph_cnn | 1500  1500 | 8 | 100 |
| 1000  1000  256 | 100 | graph_rnn | 1500  1500  1024 | 100 |
| 2000  2000  256 | 100 | graph_lstm | 10  10  1024 | 100 |
| 1500  1500  32 | 100 | graph_gru | 10  10  1024 | 100 |
| 200  200  64 | 100 | graph_birnn | 1500  1500  1024 | 100 |
| 1000  1000  32 | 100 | graph_bilstm | 1000  1000  32 | 100 |
| 1000  1000  64 | 100 | graph_bigru | 200  200  1024 | 100 |
| 2000  2000  64 | 100 | graph_intra | 2000  2000  64 | 100 |
| 1000  1000  32 | 10  100 | graph_multithead | 1000  1000  32 | 10  100 |
| 2000  2000  1024 | 8  100 | graph_stack | 2000  2000  1024 | 2  100 |
Table 16. Training on the supporting facts noise under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model       | 2.2.2.3 trln/n_loss | 2.2.2.3 trln/n_acc | Mean test accuracy | 1.3 test_acc | 2.3 test_acc | 3.3 test_acc | 4.3 test_acc |
|-------------|----------------------|--------------------|--------------------|--------------|--------------|--------------|--------------|
| gcnn        | 0.9438 (±0.0002)    | 0.9533 (±0.0003)   | 0.7456 (±0.0074)   | 0.4235 (±0.2089) | 1.0 (±0.0)   | 0.95 (±0.0)   | 0.6465 (±0.0002) |
| gat         | 0.9982 (±0.0001)    | 0.9954 (±0.0005)   | 0.9326 (±0.0049)   | 0.965 (±0.0021)  | 1.0 (±0.0)   | 0.962 (±0.0110) | 0.766 (±0.2634)  |
| sgcnn       | 0.9292 (±0.0023)    | 0.9868 (±0.0010)   | 0.7232 (±0.0054)   | 0.2945 (±0.0029)  | 1.0 (±0.0)   | 0.9336 (±0.0042) | 0.489 (±0.1712)  |
| agnn        | 0.0073 (±0.0002)    | 0.9557 (±0.0011)   | 0.8869 (±0.0045)   | 0.981 (±0.0029)  | 1.0 (±0.0)   | 0.9456 (±0.0033) | 0.562 (±0.215)   |
| rgcn        | 0.0071 (±0.0014)    | 0.9588 (±0.0011)   | 0.9534 (±0.0009)   | 0.957 (±0.0015)   | 1.0 (±0.0)   | 0.971 (±0.0008)  | 0.919 (±0.5613)  |
| ncn         | 0.5245 (±0.0147)    | 0.7276 (±0.0019)   | 0.3376 (±0.0042)   | 0.011 (±0.0023)   | 1.0 (±0.0)   | 0.3866 (±0.0121) | 0.123 (±0.1763)  |
| graph_cnn   | 0.576 (±0.0018)     | 0.4325 (±0.0013)   | 0.518 (±0.2226)    | 0.7 (±0.2581)     | 0.2874 (±0.0503) | 0.385 (±0.2772) | 0.762 (±0.0786)  |
| graph_mh    | 0.6666 (±0.0029)    | 0.4291 (±0.0063)   | 0.4715 (±0.2469)   | 0.9 (±0.2108)     | 0.2529 (±0.0337) | 0.290 (±0.2688) | 0.223 (±0.0499)  |
| graph_rn    | 0.9231 (±0.0028)    | 0.912 (±0.0088)    | 0.612 (±0.1571)    | 1.0 (±0.0)       | 0.9252 (±0.003)  | 0.170 (±0.2355) | 0.161 (±0.1085)  |
| graph_lstm  | 0.9538 (±0.0003)    | 0.9099 (±0.0026)   | 0.9519 (±0.0009)   | 0.959 (±0.0049)   | 1.0 (±0.0)   | 0.8968 (±0.0026) | 0.2325 (±0.1849) |
| graph_gru   | 0.9194 (±0.0036)    | 0.923 (±0.0097)    | 0.6752 (±0.0703)   | 1.0 (±0.0)       | 0.8862 (±0.0067) | 0.325 (±0.1848) | 0.522 (±0.0593)  |
| graph_birnn | 0.9724 (±0.0097)    | 0.916 (±0.0067)    | 0.4306 (±0.0036)   | 0.9035 (±0.1907)  | 1.0 (±0.0)   | 0.8404 (±0.1742) | 0.136 (±0.2234)  |
| graph_bilstm| 0.9443 (±0.0031)    | 0.913 (±0.0056)    | 0.5183 (±0.1318)   | 1.0 (±0.0)       | 0.901 (±0.0079)  | 0.173 (±0.0485) | 0.393 (±0.105)   |
| graph_bilgru| 0.9038 (±0.0035)    | 0.9482 (±0.0049)   | 0.7315 (±0.0595)   | 1.0 (±0.0)       | 0.8768 (±0.0021) | 0.189 (±0.1438) | 0.315 (±0.2415)  |
| graph_trne  | 0.9271 (±0.0037)    | 0.9359 (±0.0039)   | 0.5347 (±0.1781)   | 1.0 (±0.0)       | 0.8319 (±0.0049) | 0.236 (±0.2447) | 0.412 (±0.2206)  |
| graph_multi | 0.9411 (±0.0016)    | 0.9362 (±0.0053)   | 0.9594 (±0.0808)   | 1.0 (±0.0)       | 0.8994 (±0.0074) | 0.337 (±0.2591) | 0.35 (±0.1565)   |
| graph_stack | 0.8415 (±0.0049)    | 0.8265 (±0.0017)   | 0.401 (±0.0035)    | 0.041 (±0.0046)   | 0.8447 (±0.1206) | 0.7463 (±0.0892) | 0.110 (±0.0291)  |

| Model       | 0.9893 (±0.0014)    | 0.6689 (±0.0049)   | 0.6734 (±0.0087)   | 0.6737 (±0.0087)  | 0.6737 (±0.0087) | 0.6737 (±0.0087) | 0.6737 (±0.0087) |
| mac         | 0.7722 (±0.0023)    | 0.6794 (±0.0019)   | 0.7024 (±0.0029)   | 0.7024 (±0.0029)  | 0.7024 (±0.0029) | 0.7024 (±0.0029) | 0.7024 (±0.0029) |
| bilstm       | 0.8065 (±0.0024)    | 0.6346 (±0.0017)   | 0.6504 (±0.0049)   | 0.6976 (±0.0049)  | 0.6976 (±0.0049) | 0.6976 (±0.0049) | 0.6976 (±0.0049) |
| bilstm_mean  | 0.8597 (±0.0026)    | 0.6559 (±0.0012)   | 0.747 (±0.0054)    | 0.747 (±0.0054)   | 0.747 (±0.0054)  | 0.747 (±0.0054)  | 0.747 (±0.0054)  |
| bilstm_attn  | 0.8431 (±0.0028)    | 0.6588 (±0.0017)   | 0.7615 (±0.0034)   | 0.7615 (±0.0034)  | 0.7615 (±0.0034) | 0.7615 (±0.0034) | 0.7615 (±0.0034) |
Table 17. Training on the supporting facts noise under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model | 2.2.2.3_trail_n-loss | 2.2.2.3_trail_n-acc | Mean_test_accuracy | 1.3_test_acc | 2.2_test_acc | 2.3_test_acc | 3.3_test_acc | 4.3_test_acc |
|-------|-----------------------|---------------------|-------------------|-------------|-------------|-------------|-------------|-------------|
| gcn   | 0.9902 (±0.0062)      | 0.9622 (±0.0065)   | 0.7486 (±0.0071)  | 0.4075 (±0.0099) | 1.0 (±0.0) | 0.9153 (±0.0047) | 0.9421 (±0.0104) | 0.921 (±0.0115) |
| gat   | 0.0134 (±0.0057)      | 0.9961 (±0.0011)   | 0.9258 (±0.0052)  | 0.60 (±0.0207) | 1.0 (±0.0) | 0.9675 (±0.0112) | 0.767 (±0.0251) | 0.914 (±0.0427) |
| gcn   | 0.0295 (±0.0162)      | 0.9911 (±0.0067)   | 0.7037 (±0.0056)  | 0.207 (±0.0713) | 1.0 (±0.0) | 0.9058 (±0.0048) | 0.5105 (±0.0285) | 0.895 (±0.0035) |
| agnn  | 0.0106 (±0.0035)      | 0.9962 (±0.0011)   | 0.896 (±0.0461)   | 0.977 (±0.0169) | 1.0 (±0.0) | 0.956 (±0.0083) | 0.613 (±0.0264) | 0.934 (±0.0126) |
| rgnn  | 0.0104 (±0.0022)      | 0.9661 (±0.0011)   | 0.8967 (±0.0473)  | 0.977 (±0.0169) | 1.0 (±0.0) | 0.9555 (±0.0028) | 0.613 (±0.0264) | 0.937 (±0.0093) |

- gcn: Graph Convolutional Network
- gat: Graph Attention Network
- gcn: Graph Convolutional Network
- agnn: Attention Graph Neural Network
- rgnn: Relational Graph Neural Network

- m: Standard Multi-head Attention
- mac: Multi-modal Attention Convolution
- bistm: Bi-attention Self-attention Transformer
- bistm_mean: Bi-attention Self-attention Transformer with Mean Pooling
- bistm_att: Bi-attention Self-attention Transformer with Attention Pooling
### E.3 Irrelevant Facts Noise

**Table 18.** Optimal hyperparameters for training on the irrelevant facts noise under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss  | Model | Max val acc  |
|---------------|-------|-------------|
| ned | eed | hd | fi | hi | he | ep | ned | eed | hd | fi | hi | he | ep |
| 100 | 1500 | 100 | gcn | 100 | 1500 | 100 |
| 200 | 1500 | 2 | 100 | gat | 100 | 50 | 4 | 100 |
| 200 | 2000 | 100 | sgnn | 100 | 2000 | 100 |
| 100 | 512 | 100 | agnn | 500 | 32 | 100 |
| 200 | 50 | 100 | rgcn | 100 | 100 | 100 |
| 2000 | 2000 | 100 | graph_boe | 1000 | 1000 | 100 |
| 10 | 10 | 8 | 100 | graph_cnn | 10 | 10 | 8 | 100 |
| 10 | 10 | 5 | 100 | graph_cnnh | 10 | 10 | 5 | 100 |
| 2000 | 2000 | 1024 | 100 | graph_rnn | 2000 | 2000 | 64 | 100 |
| 1500 | 1500 | 64 | 100 | graph_lstm | 200 | 200 | 32 | 100 |
| 2000 | 2000 | 64 | 100 | graph_gru | 1500 | 1500 | 32 | 100 |
| 500 | 500 | 256 | 100 | graph_birnn | 200 | 200 | 32 | 100 |
| 200 | 200 | 64 | 100 | graph_bilstm | 1500 | 1500 | 32 | 100 |
| 2000 | 2000 | 64 | 100 | graph_bigru | 2000 | 2000 | 32 | 100 |
| 2000 | 2000 | 128 | 100 | graph_intra | 2000 | 2000 | 128 | 100 |
| 1500 | 1500 | 32 | 10 | 100 | graph_multistack | 1500 | 1500 | 32 | 10 | 100 |
| 2000 | 2000 | 1024 | 0 | 100 | graph_stack | 2000 | 2000 | 1024 | 0 | 100 |
Table 19. Training on the irrelevant facts noise under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

|                      | 3.2.3.3 train n loss | 3.2.3.3 train n acc | Mean test accuracy | 1.1 test acc | 3.2 test acc | 3.3 test acc | 4.3 test acc |
|----------------------|----------------------|---------------------|-------------------|--------------|--------------|--------------|--------------|
| gcn                  | 0.0416 (+0.0031)     | 0.7977 (+0.0128)    | 0.8489 (+0.0367)  | 0.1066 (+0.0443) | 0.482 (+0.1204) | 1.0 (+0.0) | 0.8556 (+0.0062) | 0.885 (+0.0) |
| gat                  | 0.0073 (+0.0015)     | 0.9959 (+0.0001)    | 0.9086 (+0.0357)  | 0.922 (+0.0063) | 0.971 (+0.009) | 1.0 (+0.0) | 0.973 (+0.0047) | 0.977 (+0.0277) |
| sgcnn                | 0.0111 (+0.0062)     | 0.9948 (+0.0003)    | 0.7752 (+0.0474)  | 0.812 (+0.1527) | 0.26 (+0.1402) | 1.0 (+0.0) | 0.873 (+0.0047) | 0.931 (+0.1709) |
| agnn                 | 0.0074 (+0.0014)     | 0.9961 (+0.0009)    | 0.8554 (+0.0128)  | 0.92 (+0.0632) | 0.512 (+0.009) | 1.0 (+0.0) | 0.86 (+0.0) | 0.985 (+0.0) |
| rgcnn                | 0.0065 (+0.0012)     | 0.9962 (+0.0008)    | 0.9232 (+0.0191)  | 0.922 (+0.0063) | 0.949 (+0.0093) | 1.0 (+0.0) | 0.86 (+0.0) | 0.985 (+0.0) |
| graph_boe            | 0.4176 (+0.0211)     | 0.7299 (+0.0096)    | 0.4892 (+0.0345)  | 0.084 (+0.0099) | 0.102 (+0.0060) | 1.0 (+0.0) | 0.275 (+0.019) | 0.705 (+0.0) |
| graph_cnn            | 0.9215 (+0.064)      | 0.5943 (+0.0393)    | 0.4645 (+0.0773)  | 0.169 (+0.1388) | 0.34 (+0.1047) | 1.0 (+0.0) | 0.336 (+0.0063) | 0.497 (+0.0269) |
| graph_cnnh           | 1.0745 (+0.0432)     | 0.5948 (+0.0189)    | 0.5143 (+0.0272)  | 0.348 (+0.1387) | 0.331 (+0.0875) | 1.0 (+0.0) | 0.2925 (+0.0096) | 0.599 (+0.2133) |
| graph_rnn            | 0.5633 (+0.1079)     | 0.8668 (+0.0274)    | 0.6766 (+0.0009)  | 0.182 (+0.1438) | 0.042 (+0.0076) | 1.0 (+0.0) | 0.882 (+0.0064) | 0.777 (+0.2609) |
| graph_lstm            | 0.0113 (+0.0122)    | 0.995 (+0.0003)    | 0.7667 (+0.0047)  | 0.646 (+0.1146) | 0.35 (+0.1178) | 1.0 (+0.0) | 0.8526 (+0.0037) | 0.985 (+0.0) |
| graph_gru            | 0.0371 (+0.0009)     | 0.996 (+0.0009)    | 0.7268 (+0.0064)  | 0.929 (+0.2286) | 0.29 (+0.1003) | 1.0 (+0.0) | 0.985 (+0.0069) | 0.91 (+0.1667) |
| graph_birnn          | 0.5555 (+0.1458)     | 0.9868 (+0.0164)    | 0.6134 (+0.2117)  | 0.12 (+0.1334) | 0.008 (+0.1208) | 1.0 (+0.0) | 0.871 (+0.0014) | 0.978 (+0.0221) |
| graph_bistm          | 0.0113 (+0.0125)     | 0.9554 (+0.0002)    | 0.7448 (+0.0059)  | 0.601 (+0.1273) | 0.328 (+0.1753) | 1.0 (+0.0) | 0.86 (+0.0) | 0.935 (+0.0) |
| graph_biquo          | 0.0207 (+0.0012)     | 0.959 (+0.0003)    | 0.7308 (+0.0299)  | 0.631 (+0.2114) | 0.228 (+0.1536) | 1.0 (+0.0) | 0.86 (+0.0) | 0.935 (+0.0) |
| graph_birn           | 0.0463 (+0.0056)     | 0.8949 (+0.0099)    | 0.6817 (+0.0718)  | 0.725 (+0.1765) | 0.08 (+0.1006) | 1.0 (+0.0) | 0.6836 (+0.0096) | 0.77 (+0.2701) |
| graph_multi          | 0.0102 (+0.0002)     | 0.9599 (+0.0001)    | 0.7202 (+0.0051)  | 0.360 (+0.197) | 0.46 (+0.0699) | 1.0 (+0.0) | 0.852 (+0.0033) | 0.927 (+0.1562) |
| graph_stack          | 0.1468 (+0.0013)     | 0.9416 (+0.0005)    | 0.7425 (+0.0057)  | 0.514 (+0.1364) | 0.5 (+0.1889) | 0.9971 (+0.0088) | 0.8644 (+0.0078) | 0.8367 (+0.2147) |
| m                    | 0.9073 (+0.0061)     | 0.6779 (+0.0186)    | 0.5905 (+0.0247)  | 0.6196 (+0.0088) | 0.5321 (+0.0573) | 0.7951 (+0.0131) | 0.551 (+0.0086) | 0.4545 (+0.0057) |
| mac                  | 0.8592 (+0.0311)     | 0.6037 (+0.0123)    | 0.5007 (+0.0072)  | 0.5545 (+0.0703) | 0.5545 (+0.0094) | 0.7715 (+0.1033) | 0.565 (+0.0028) | 0.45 (+0.0036) |
| bilist               | 0.8798 (+0.0382)     | 0.6345 (+0.0013)    | 0.5774 (+0.0447)  | 0.5195 (+0.0686) | 0.5877 (+0.0041) | 0.7995 (+0.0980) | 0.4971 (+0.0058) | 0.4733 (+0.0061) |
| bilist mean          | 0.8984 (+0.0402)     | 0.652 (+0.0153)     | 0.5629 (+0.0601)  | 0.463 (+0.0617) | 0.46 (+0.0648) | 0.8342 (+0.0497) | 0.55 (+0.0748) | 0.6075 (+0.2151) |
| bilist atten         | 0.8637 (+0.0541)     | 0.6767 (+0.019)     | 0.5582 (+0.0575)  | 0.474 (+0.0306) | 0.486 (+0.0406) | 0.7816 (+0.0645) | 0.492 (+0.0001) | 0.5625 (+0.2349) |
Table 20. Training on the irrelevant facts noise under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model   | 3.2.3 test acc | Mean_test accuracy | 1.3 test acc | 2.3 test acc | 3.2 test acc | 3.3 test acc | 4.3 test acc |
|---------|----------------|--------------------|--------------|--------------|--------------|--------------|--------------|
| gcn     | 0.6979 (±0.035) | 0.9771 (±0.0124)  | 0.8403 (±0.028) | 0.908 (±0.0454) | 0.454 (±0.0674) | 1.0 (±0.0) | 0.8545 (±0.0059) | 0.985 (±0.0) |
| gat     | 0.0146 (±0.0077) | 0.9665 (±0.0006)  | 0.8978 (±0.0515) | 0.924 (±0.0084) | 0.743 (±0.2093) | 1.0 (±0.0) | 0.891 (±0.0052) | 0.931 (±0.1708) |
| agcn    | 0.0217 (±0.0044) | 0.9661 (±0.0008)  | 0.7962 (±0.0291) | 0.882 (±0.075) | 0.232 (±0.1234) | 1.0 (±0.0) | 0.882 (±0.0464) | 0.985 (±0.0) |
| aqnn    | 0.0101 (±0.0028) | 0.9668 (±0.0006)  | 0.8204 (±0.0716) | 0.906 (±0.0089) | 0.626 (±0.0942) | 1.0 (±0.0) | 0.909 (±0.0052) | 0.661 (±0.0279) |
| rgnn    | 0.0107 (±0.0038) | 0.9663 (±0.0007)  | 0.8674 (±0.0408) | 0.92 (±0.0187) | 0.647 (±0.1687) | 1.0 (±0.0) | 0.893 (±0.0533) | 0.787 (±0.2277) |

| Model   | 3.2.3_test_acc | Mean_test_accuracy |
|---------|----------------|--------------------|
| graph_bne | 0.4296 (±0.0152) | 0.7399 (±0.0118)  |
| graph_cnn | 0.9454 (±0.0796) | 0.6059 (±0.0244)  |
| graph_cnnh | 1.1129 (±0.053)  | 0.5521 (±0.0123)  |
| graph_rnn | 0.0322 (±0.0092) | 0.9387 (±0.0074)  |
| graph_lstm | 0.0266 (±0.0029) | 0.9957 (±0.0028)  |
| graph_gru | 0.0128 (±0.0007) | 0.9963 (±0.0001)  |
| graph_bnn | 0.0534 (±0.1161) | 0.9298 (±0.0073)  |
| graph_bilstm | 0.0111 (±0.0024) | 0.9964 (±0.0001)  |
| graph_bignu | 0.0127 (±0.0051) | 0.9661 (±0.0017)  |
| graph_bistr | 0.0506 (±0.0212) | 0.987 (±0.0057)   |
| graph_multihead | 0.0115 (±0.0026) | 0.9964 (±0.0007)  |
| graph_stack | 0.1525 (±0.0181) | 0.9447 (±0.0005)  |

| Method | 3.2.3_mean | Mean_test_accuracy |
|--------|------------|--------------------|
| m      | 1.6908 (±0.2092) | 0.7558 (±0.0287)  |
| mac    | 1.6568 (±0.4004) | 0.7693 (±0.0324)  |
| bilstm | 2.5551 (±0.418)  | 0.7221 (±0.0235)  |
| bilstm_mean | 1.8141 (±0.1279) | 0.7467 (±0.0195)  |
| bilstm_attn | 1.8611 (±0.1537) | 0.7632 (±0.0149)  |
### E.4 Disconnected Facts Noise

**Table 21.** Optimal hyperparameters for training on the disconnected facts noise under minimum validation loss metric (left) and maximum validation accuracy metric (right).

| Min val loss | Model | Max val acc |
|--------------|-------|-------------|
| n_e | e_e | h_d | f_i | h_i | e_e | n_e | e_e | h_d | f_i | h_i | e_e |
| 50 | 100 | 100 | gcn | 100 | 100 | 100 |
| 10 | 1500 | 2 | 100 | gat | 500 | 50 | 7 | 100 |
| 100 | 128 | 100 | sgcgn | 100 | 128 | 100 |
| 200 | 512 | 100 | agnn | 10 | 256 | 100 |
| 100 | 1024 | 100 | rgcn | 200 | 1000 | 100 |
| 2000 | 2000 | 100 | graph_boe | 500 | 500 | 100 |
| 10 | 10 | 8 | 100 | graph_cnn | 10 | 10 | 8 | 100 |
| 10 | 10 | 7 | 100 | graph_cnnh | 10 | 10 | 7 | 100 |
| 1500 | 1500 | 512 | 100 | graph_rnn | 500 | 500 | 512 | 100 |
| 1000 | 1000 | 1024 | 100 | graph_lstm | 50 | 50 | 32 | 100 |
| 500 | 500 | 128 | 100 | graph_gru | 50 | 50 | 512 | 100 |
| 1500 | 1500 | 1024 | 100 | graph_birnn | 1500 | 1500 | 1024 | 100 |
| 100 | 100 | 32 | 100 | graph_bilstm | 50 | 50 | 256 | 100 |
| 50 | 50 | 128 | 100 | graph_bigru | 200 | 200 | 128 | 100 |
| 2000 | 2000 | 64 | 100 | graph_intra | 2000 | 2000 | 64 | 100 |
| 1000 | 1000 | 512 | 10 | 100 | graph_multthead | 1500 | 1500 | 512 | 10 | 100 |
| 2000 | 2000 | 1024 | 8 | 100 | graph_stack | 2000 | 2000 | 1024 | 8 | 100 |
Table 22. Training on the disconnected facts noise under minimum validation loss metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model   | 4.2.4.3 train n_loss | 4.2.4.3 train n_acc | Mean test accuracy | 1.3 test accuracy | 2.3 test accuracy | 3.3 test accuracy | 4.2 test acc | 4.3 test acc |
|---------|----------------------|---------------------|--------------------|------------------|------------------|------------------|--------------|--------------|
| gcn     | 0.0065 ± (0.0012)    | 0.9962 ± (0.0017)   | 0.8072 ± (0.0096)  | 0.797 ± (0.0136)  | 0.374 ± (0.0752)  | 0.9575 ± (0.0106) | 1.0 ± (0.0)  | 0.9073 ± (0.0225) |
| gat     | 0.0066 ± (0.0015)    | 0.9958 ± (0.0017)   | 0.802 ± (0.0091)   | 0.690 ± (0.0807)  | 0.689 ± (0.0297)  | 0.963 ± (0.0106)  | 1.0 ± (0.0)  | 0.902 ± (0.0169)  |
| sgcn    | 0.0065 ± (0.0014)    | 0.9961 ± (0.0014)   | 0.674 ± (0.0054)   | 0.875 ± (0.0057)  | 0.012 ± (0.0379)  | 0.588 ± (0.2104)  | 1.0 ± (0.0)  | 0.8867 ± (0.0)   |
| aqnn    | 0.0053 ± (0.0013)    | 0.9965 ± (0.0012)   | 0.833 ± (0.0204)   | 0.934 ± (0.0230)  | 0.405 ± (0.0932)  | 0.934 ± (0.0270)  | 1.0 ± (0.0)  | 0.8967 ± (0.0)   |
| rgnn    | 0.0061 ± (0.0012)    | 0.9964 ± (0.0013)   | 0.896 ± (0.0189)   | 0.937 ± (0.0096)  | 0.643 ± (0.0972)  | 0.9705 ± (0.0101) | 1.0 ± (0.0)  | 0.902 ± (0.0168)  |
| graph_conv | 0.4295 ± (0.0095)    | 0.276 ± (0.0018)    | 0.340 ± (0.0319)   | 0.232 ± (0.0885)  | 0.05 ± (0.0090)   | 0.139 ± (0.0307)  | 1.0 ± (0.0)  | 0.2903 ± (0.0329) |
| graph_conv | 0.6018 ± (0.0243)    | 0.291 ± (0.0592)    | 0.319 ± (0.0132)   | 0.236 ± (0.0757)  | 0.141 ± (0.0341)  | 0.6 ± (0.2500)    | 1.0 ± (0.0)  | 0.76 ± (0.0707)   |
| graph_conv | 0.613 ± (0.0475)     | 0.191 ± (0.0756)    | 0.248 ± (0.0113)   | 0.248 ± (0.0756)  | 0.1135 ± (0.0477) | 0.3 ± (0.2106)    | 1.0 ± (0.0)  | 0.2512 ± (0.0525) |
| graph_rnn | 0.938 ± (0.0106)     | 0.630 ± (0.0246)    | 0.130 ± (0.1081)   | 0.173 ± (0.1126)  | 0.173 ± (0.1063)  | 1.0 ± (0.0)       | 0.9867 ± (0.0) |
| graph_lstm | 0.933 ± (0.0053)     | 0.9036 ± (0.0077)   | 0.699 ± (0.0352)   | 0.350 ± (0.1594)  | 0.772 ± (0.1021)  | 0.960 ± (0.0097)  | 1.0 ± (0.0)  | 0.902 ± (0.0169)  |
| graph_gru | 0.954 ± (0.0075)     | 0.719 ± (0.0577)    | 0.485 ± (0.1968)   | 0.230 ± (0.1108)  | 0.20 ± (0.0756)   | 0.966 ± (0.0007)  | 1.0 ± (0.0)  | 0.9127 ± (0.0349) |
| graph_brnn | 0.959 ± (0.1343)     | 0.617 ± (0.0224)    | 0.767 ± (0.0699)   | 0.14 ± (0.1014)   | 0.267 ± (0.2387)  | 0.973 ± (0.0003)  | 1.0 ± (0.0)  | 0.902 ± (0.0169)  |
| graph_brnn | 0.937 ± (0.0057)     | 0.700 ± (0.0354)    | 0.421 ± (0.1503)   | 0.196 ± (0.2387)  | 0.973 ± (0.0003)  | 1.0 ± (0.0)       | 0.9103 ± (0.0207) |
| graph_brnn | 0.937 ± (0.0057)     | 0.700 ± (0.0354)    | 0.421 ± (0.1503)   | 0.196 ± (0.2387)  | 0.973 ± (0.0003)  | 1.0 ± (0.0)       | 0.9103 ± (0.0207) |
| graph_brnn | 0.937 ± (0.0057)     | 0.700 ± (0.0354)    | 0.421 ± (0.1503)   | 0.196 ± (0.2387)  | 0.973 ± (0.0003)  | 1.0 ± (0.0)       | 0.9103 ± (0.0207) |
| graph_stack | 0.385 ± (0.0079)     | 0.472 ± (0.0013)    | 0.643 ± (0.0646)   | 0.472 ± (0.064)   | 0.943 ± (0.0444)  | 0.897 ± (0.0377)  | 0.902 ± (0.0169) |
| m        | 1.0029 ± (0.0418)    | 0.530 ± (0.0209)    | 0.381 ± (0.0226)   | 0.461 ± (0.0962)  | 0.428 ± (0.0694)  | 0.486 ± (0.0576)  | 0.0319 ± (0.0524) | 0.4862 ± (0.0396) |
| mac      | 0.648 ± (0.0212)     | 0.346 ± (0.0403)    | 0.477 ± (0.0283)   | 0.494 ± (0.0438)  | 0.496 ± (0.0451)  | 0.76 ± (0.0569)   | 0.897 ± (0.0154) |
| bi lstm  | 0.607 ± (0.0118)     | 0.402 ± (0.0298)    | 0.426 ± (0.0521)   | 0.5 ± (0.0962)    | 0.511 ± (0.0936)  | 0.1 ± (0.0853)    | 0.489 ± (0.0446) |
| bi lstm_mean | 0.607 ± (0.0118)     | 0.402 ± (0.0298)    | 0.426 ± (0.0521)   | 0.5 ± (0.0962)    | 0.511 ± (0.0936)  | 0.1 ± (0.0853)    | 0.489 ± (0.0446) |
| bi lstm_atten | 0.607 ± (0.0118)     | 0.402 ± (0.0298)    | 0.426 ± (0.0521)   | 0.5 ± (0.0962)    | 0.511 ± (0.0936)  | 0.1 ± (0.0853)    | 0.489 ± (0.0446) |
Table 23. Training on the disconnected facts noise under maximum validation accuracy metric. The optimal values within the graph-based, the sequence-based and the text-based are in bold; the best performance among all models are in red.

| Model | 4.2.4.3_train_n_loss | 4.2.4.3_train_n_acc | Mean_test_acc | 1.3_test_acc | 2.3_test_acc | 3.3_test_acc | 4.2_test_acc | 4.3_test_acc |
|-------|-----------------------|---------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| gcn   | 0.0291 (±0.0179)      | 0.9945 (±0.0046)    | 0.926 (±0.0038) | 0.9005 (±0.008) | 1.0 (±0.0)   | 0.902 (±0.0067) |
| gat   | 0.0088 (±0.00044)     | 0.968 (±0.00016)    | 0.926 (±0.00143) | 0.944 (±0.0149) | 1.0 (±0.0)   | 0.917 (±0.00268) |
| sgcn  | 0.0096 (±0.0041)      | 0.967 (±0.0135)     | 0.823 (±0.0103)  | 0.966 (±0.0145) | 1.0 (±0.0)   | 0.902 (±0.00169) |
| aqnn  | 0.174 (±0.0605)       | 0.968 (±0.0357)     | 0.926 (±0.02411) | 0.956 (±0.0195) | 1.0 (±0.0)   | 0.902 (±0.00857) |
| rcnn  | 0.0067 (±0.0042)      | 0.991 (±0.0006)     | 0.934 (±0.0126)  | 0.9095 (±0.0099) | 1.0 (±0.0)   | 0.917 (±0.00268) |

| Model | 4.2.4.3_test_n_loss | 4.2.4.3_test_n_acc | Mean_test_acc | 1.3_test_acc | 2.3_test_acc | 3.3_test_acc | 4.2_test_acc | 4.3_test_acc |
|-------|---------------------|---------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| graph boe | 0.0361 (±0.0026) | 0.742 (±0.0009) | 0.127 (±0.0124) | 0.145 (±0.0132) | 1.0 (±0.0) | 0.201 (±0.0028) |
| graph cnn | 0.9572 (±0.0065) | 0.515 (±0.0037) | 0.296 (±0.0017) | 0.341 (±0.0147) | 0.1 (±0.0) | 0.255 (±0.00547) |
| graph cnnh | 1.1399 (±0.1396) | 0.518 (±0.0368) | 0.296 (±0.0087) | 0.341 (±0.0173) | 1.0 (±0.0) | 0.255 (±0.00534) |
| graph rnn | 0.1150 (±0.3239) | 0.937 (±0.0099) | 0.127 (±0.01226) | 0.296 (±0.01652) | 1.0 (±0.0) | 0.899 (±0.00268) |
| graph lstm | 0.145 (±0.0557) | 0.947 (±0.00345) | 0.127 (±0.01648) | 0.296 (±0.0219) | 0.1 (±0.0) | 0.899 (±0.00268) |
| graph gru | 0.022 (±0.0101) | 0.94 (±0.0076) | 0.222 (±0.0121) | 0.35 (±0.01573) | 1.0 (±0.0) | 0.010 (±0.00299) |
| graph bimn | 0.0733 (±0.2012) | 0.931 (±0.0102) | 0.124 (±0.0060) | 0.117 (±0.0066) | 1.0 (±0.0) | 0.031 (±0.00231) |
| graph bistm | 0.0449 (±0.0062) | 0.996 (±0.0034) | 0.167 (±0.01087) | 0.4 (±0.1668) | 0.975 (±0.0075) | 1.0 (±0.0) | 0.005 (±0.00299) |
| graph bigru | 0.041 (±0.1068) | 0.94 (±0.0037) | 0.167 (±0.01251) | 0.189 (±0.017) | 0.9 (±0.0) | 0.097 (±0.00322) |
| graph intmn | 0.02 (±0.0597) | 0.959 (±0.0069) | 0.167 (±0.01251) | 0.189 (±0.017) | 0.9 (±0.0) | 0.097 (±0.00322) |
| head | 0.015 (±0.0015) | 0.94 (±0.0033) | 0.217 (±0.0107) | 0.469 (±0.0108) | 0.975 (±0.0075) | 1.0 (±0.0) | 0.005 (±0.00299) |
| graph stack | 0.1417 (±0.00129) | 0.96 (±0.0006) | 0.465 (±0.0034) | 0.472 (±0.00183) | 0.975 (±0.0075) | 1.0 (±0.0) | 0.005 (±0.00299) |

| m | 1.7181 (±0.2409) | 0.597 (±0.00621) | 0.553 (±0.0040) | 0.525 (±0.0081) | 0.393 (±0.002) | 0.467 (±0.002) |
| mac | 1.751 (±0.3328) | 0.739 (±0.0064) | 0.409 (±0.0082) | 0.484 (±0.0083) | 0.537 (±0.0081) | 0.606 (±0.00542) |
| bilstm | 2.507 (±0.300) | 0.717 (±0.0062) | 0.409 (±0.0067) | 0.484 (±0.0054) | 0.507 (±0.0067) | 0.490 (±0.00602) |
| bilstm mean | 2.0877 (±0.1541) | 0.702 (±0.009) | 0.351 (±0.002) | 0.447 (±0.00472) | 0.247 (±0.015) | 0.444 (±0.00898) | 0.402 (±0.00568) |
| bilstm atten | 1.9974 (±0.229) | 0.736 (±0.0018) | 0.352 (±0.0027) | 0.447 (±0.00343) | 0.314 (±0.0029) | 0.453 (±0.00499) | 0.401 (±0.00372) |