Scalable and interpretable quantum natural language processing: an implementation on trapped ions

Tiffany Duneau^{1,2}, **Saskia Bruhn**^{1}***, **Gabriel Matos**^{1}***, **Tuomas Laakkonen**^{1},
Katerina Saiti^{3}, **Anna Pearson**^{1}***, **Konstantinos Meichanetzidis**^{1}, **Bob Coecke**^{1}

^{1}Quantinuum, ^{2}University of Oxford, ^{3}Leiden University

{`tiffany.duneau, gabriel.matos, anna.pearson,`
`k.mei, bob.coecke`}`@quantinuum.com`,
`a.saiti@umail.leidenuniv.nl`,
`saskiabruhn.pub@gmail.com`,
`tsrl@mit.edu`

Abstract

We present a compositional implementation of natural language processing tasks on a quantum computer using the QDisCoCirc model. QDisCoCirc is a model that allows for both *compositional generalisation* - the ability to generalise outside the training distribution by learning compositional rules underpinning the entire data distribution - and *compositional interpretability* - making sense of how the model works by inspecting its modular components in isolation and the processes through which they are combined. We consider the task of question-answering for which we handcraft a toy dataset. The model components are trained on classical computers at small scales, then composed to generate larger test instances, which are evaluated on Quantinuum's H1-1 trapped-ion quantum processor. We inspect the trained models by comparing them to manually-constructed perfect compositional models, and identify where and why our model learned compositional behaviours. As an initial baseline comparison, we considered small-scale Transformer and LSTM models, as well as GPT-4, none of which succeeded at compositional generalisation on this task.

1 Introduction

Artificial intelligence (AI) permeates a wide range of activity, from academia to industrial real-world applications, with natural language processing (NLP) taking centre stage. In parallel, quantum computing has seen a recent surge in development, with the advent of *noisy intermediate-scale quantum* (NISQ) processors [1]. The merging of these two fields has given rise to *quantum natural language processing* (QNLP). This work focuses on the novel *DisCoCirc* framework [2, 3], where sentences are represented by circuits that can be further composed into *text circuits*, specifically in the context of a quantum setup (QDisCoCirc). One important feature that distinguishes the 'DisCo-' line of work [2–8] from other contributions to QNLP [9, 10], and the broad field of quantum machine learning [11–13], is that it is explicitly *compositional*, which in turn provides a path towards *explainability* and *interpretability* [14]. Indeed, while the advancements of contemporary AI are impressive, when things go wrong, one typically does not understand why.

[*]Equal contribution

We present experimental results for the task of question answering with QDisCoCirc. This constitutes the first proof-of-concept implementation of *scalable compositional QNLP*. Our results in Sec. 3 show, with statistical significance, that the accuracy of our models does not decay when tested solely on instances larger than those used in training: the model *compositionally generalises*. Moreover, the compositional structure allows for inspection of the trained model's internals. Recent work [14] explores how compositionality can be used in constructing interpretable and explainable AI models, and defined a class of *compositionally interpretable* models, which includes DisCoCirc. While the precise definition is given in terms of category theory, we can summarise it informally: a model is *compositionally interpretable* if we can assign a human friendly meaning (e.g. in natural language), to the components of a model, whilst also understanding how these components fit together. In Sec. 3, we demonstrate this by giving a full account of why our models work, as well as an analysis of when they *don't*. Importantly, our compositional approach provides a setup in which quantum circuit components may be pre-trained via classical simulation. We hence avoid the trainability challenges posed by exponentially vanishing gradients in conventional quantum machine learning (QML) [15] - a quantum computer is necessary only at test time to evaluate larger instances. To demonstrate compositional generalisation on a real device beyond instance sizes that we have simulated classically, we use Quantinuum's H1-1 trapped-ion quantum processor [16].

2 Setup

Model QDisCoCirc is a quantum model for text, constructed in the DisCoCirc framework by giving quantum semantics to text diagrams (read top to bottom). These are composed of *states*, *effects*, and *boxes*, understood as inputs, output *tests*, and *processes* that transform inputs into outputs [17, 18]. A sentence can be parsed into a text diagram, where each word is assigned a state or a box according to its part of speech [19]. Nouns are mapped to states, and verbs are mapped to boxes. Each state corresponds to a *noun wire* (a line in the diagram connecting to the state). In this work, diagrams are constructed specifically for the chosen task. A small example is given in Fig. 1, and more details in App. A. Sentences are composed sequentially according to the reading order of the text. A QDisCoCirc model is instantiated via a *semantic functor*, transforming text diagrams into quantum circuits, described in App. A.2. We chose the task of question answering as it can be represented natively in DisCoCirc via use of the primitive operation of *text similarity*, as presented in [20]. This works as follows: a text circuit states the facts, and the question is posed as an effect representing it as an affirmative statement. The effect is applied to the noun wires relevant to the question, and the irrelevant nouns wires are discarded. We show the structure of a text diagram for the question answering task in App. A.

```
Bob walks west.
Alice walks north.
John walks south.
Alice turns right.
John turns left.
Bob follows Alice.
Bob goes in the opposite
direction of John.

Is Bob going in the same
direction as Alice?
```

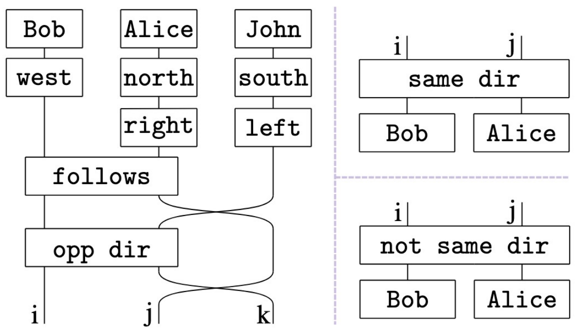
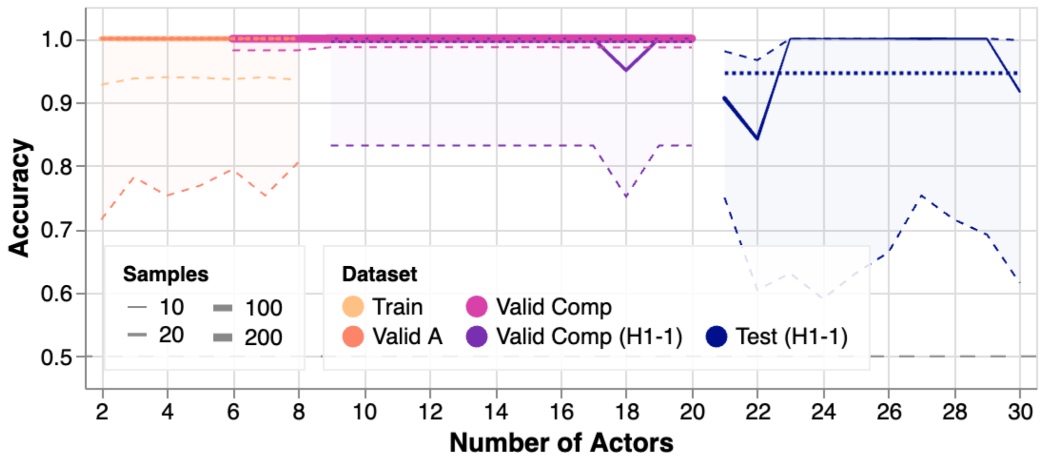


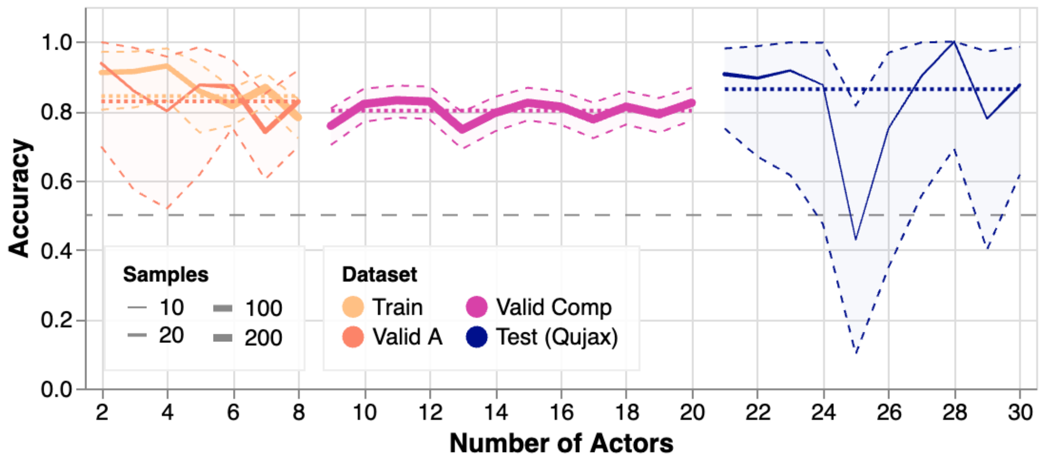
Figure 1: A story as a DisCoCirc text-circuit, and question as a pair of affirmative statements.

Dataset & Methodology We generated two custom small-scale toy datasets, the *two-directional* and the *four-directional* datasets, in order to perform our proof-of-concept experiments in a controlled setup. In the first dataset, the texts describe *actors* who initially walk in one of two cardinal directions; in the second, the actors walk in one of four cardinal directions. Throughout the text, each actor can perform actions, described by transitive and intransitive verbs that result in their directions changing. We generated texts with varying numbers of actors and sentences. We refer to the number of actors in a text by *text width* (or *number of actors*), and the number of sentences in a text by *text depth*. For each text, we ask the binary question of whether two actors are going in the same direction. A detailed characterisation of the datasets is given in App. C. We first use our parser to obtain diagrams for the stories we generated, then apply the semantic functor of App. A.2 to obtain quantum circuits. We convert these into tensor networks for evaluation. We train in batches, and select the final model with the highest validation accuracy. For more details, including on the specific packages used and cross-validation results, see App. D.

3 Results



(a) Two-directional model compositional generalisation



(b) Four-directional model compositional generalisation

Figure 2: Compositional generalisation results.

Compositional generalisation Having trained QDisCoCirc models for the two- and four-directional question answering datasets, we measure the degree to which the compositional structure of the model allows it to generalise to instances outside the training data distribution. Specifically, we quantify *productivity*, i.e. whether a model trained on small examples can use the learned rules to correctly classify larger problem instances [21]. The number of actors involved in the text was chosen as a metric of size, to easily quantify the worst-case resources required (App. I.1), and is positively correlated with the text depth and resulting quantum circuit depth (App. C). For both datasets, the *Train* set included up to a maximum of 8 actors (see App. C.2 for details on how the data was split between training, validation and test sets) evaluated using exact tensor contractions. We also used exact tensor contractions to obtain the *compositional validation* (*Valid Comp*) accuracy for instances featuring up to 20 actors. Pushing beyond 20 actors, we use Quantinuum's H1-1 to evaluate accuracy on *Test (H1-1)* for instances up to 30 actors from the two-directional dataset (App. E specifies how the data was adapted for H1-1), and used shot-based noisy simulation (as described in App. I.2.1) to obtain the results for the four-directional *Test (Qujax)* dataset.

With statistical significance, the accuracy of the model on the *Valid Comp* set does not decay for the two- and four-directional datasets, as shown in Figs. 2(a) and 2(b). We plot the model's accuracy per number of actors, with thickness reflecting the number of samples present. The error regions capture the 95% Clopper-Pearson confidence interval [22] for the mean, accounting for the number of samples available. Note that the *Valid Comp* dataset contained significantly more samples than either *Train* or *Valid A*, leading to a tighter confidence interval. The dashed grey line is the random guessing baseline. The average performance per dataset split is shown as a dotted line. This trend continues for the two-directional *Test (H1-1)* sub-dataset in Fig. 2(a), evaluated on Quantinuum's H1-1 using 50 shots per circuit. The four-directional *Test (Qujax)* sub-dataset in Fig. 2(b) was simulated with noise using `qujax` with 1000 shots per circuit. Although the accuracy and confidence intervals dip below the random guessing baseline for some text widths, this can be attributed to the low number of sampled data points available. Additionally, for 25 actors, approximately half of the samples selected were amongst those we expect the model to solve incorrectly *a priori* (App. H), which is a much higher proportion than in the rest of the dataset. We argue that this is experimental evidence that compositionality enables generalisation, as to perform inference we need only compose pre-trained components. In other words, we have provided experimental evidence that the quantum word circuits have learned effective compositional representations. For comparison, we trained both Transformer and LSTM models on the two- and four-directional datasets to explore their capabilities in compositional generalisation. The results are summarised in Table 1 with details documented in App. G. These models exhibited no signs of compositional generalization, performing on par with random guesses in stories with increased text widths. This poor performance may be attributed to the constraints inherent in drawing direct comparisons with QDisCoCirc.

Table 1: Model performance on the two- and four-directional datasets, including classical baselines.

Model	**Two-directional**				**Four-directional**			
	Train	*Valid A*	*Valid Comp*	*Test*	*Train*	*Valid A*	*Valid Comp*	*Test*
QDisCoCirc	**1.00**	**1.00**	**1.00**	**0.95**	**0.84**	**0.83**	**0.80**	**0.86**
LSTM	0.68	0.54	0.50	0.52	0.73	0.62	0.50	0.54
Transformer	0.66	0.58	0.50	0.61	0.66	0.62	0.65	0.51
GPT-4	0.50			-	0.49			-

Compositional interpretability The built-in compositional structure of QDisCoCirc allows us to interpret its behaviour, both by direct inspection of the components, which allows us to piece together how the model works, as well as by investigating how interventions at the dataset level affect the model's responses. Summarizing the results in App. H, we find that the two-directional model approximates an ideal model (described in Sec. H.4 Fig. 42) in which the state of each qubit tracks the direction faced by a given actor. It is then no surprise that it compositionally generalised so well, however the approximate nature of the model leaves room for error. Indeed, we can construct some failure cases where the error in each operation accumulates past a threshold, causing the model to answer incorrectly. Despite this, the model exhibits self-correcting behaviour, such that for an average story the error is likely to be reset before reaching the threshold.

Although the four-directional model fails to reach a perfect accuracy, it nevertheless exhibits compositional generalisation. In fact, the four-directional model is expected to struggle, as it must encode 2 bits of information (which of the four directions each actor is currently facing) into a single qubit. The ideal four-directional model (described in App. H Fig. 43) requires two qubits per actor. By inspecting the internal states, we see that the model actually learnt a two-directional approximation to the four-directional problem, by internally pairing directions. Thus, with high probability, the model is able to correctly answer stories where the actors face directions it distinguishes, and conversely is almost always wrong when the actors end up facing directions it has paired.

The above analysis allows us to characterise the expected behaviour of our models on unseen distributions, both providing extra evidence of successful compositional generalisation, as well as exposing situations in which we expect the model to fail. That the models fail in largely *predictable* ways is a feature of their compositional structure. Although this does not preclude failures arising from complex interactions that are difficult or intractable to predict, it is noteworthy that such an analysis can be made at all.

4 Discussion

In this work, we presented the first experimental implementation of the QDisCoCirc model, as described in [20], for the NLP task of question answering over two toy datasets we constructed. The compositional nature of the model enables compositional generalisation, a property that mainstream neural architectures struggle with, or achieve only with significant tuning and curriculum design [23]. Additionally, our setup provides a promising route for scalable quantum machine learning by avoiding trainability issues due to vanishing gradients : by training components classically and using them to compose larger instances which are evaluated on a quantum computer only for inference. Furthermore, we construct the model such that its semantic encoding follows the compositional structure of the input text, enabling interpretability. Given a trained model, we can understand how the model performs a task by inspecting the quantum word embeddings. When the model showed excellent performance, we were able to identify how it solved the task, and verified that these matched our intuitions. Conversely, for a sub-optimal model, we were able to identify how it learnt to compromise and take advantage of biases in the data to outperform random guessing.

Limitations As a proof-of-concept work, certain simplifying assumptions were made. More detail on how we simplified the dataset itself, as well as the *semantic rewrites* built into the model is documented in App. C and C.3. This encouraged the model to learn a representation that would compositionally generalise by reducing the parameter search space. Secondly, our choice to implement a quantum model and train via exact classical simulation imposes an upper bound on the size of the training samples we can use. Consequently, a quantum computer may be required to evaluate our model on larger instances. In App. I we quantify how the classical resources required to evaluate our model scale with the size of a story, and characterise the effect of noise on the model's accuracy in App. I.2. As discussed in [20], alternative *classical* DisCoCirc models could be proposed to avoid this. However, DisCoCirc, via DisCoCat, is derived from a grammar (argued to be necessary to fully capture the semantics of language [24, 25]) that enforces a tensor product structure. Using a quantum model such as QDisCoCirc then captures the most expressive setup with such a structure that remains efficient to evaluate at test time. Further work to quantify this dependency is ongoing.

Future Work We aim to train a QDisCoCirc model on larger-scale real-world data, making use of the forthcoming text-to-diagram DisCoCirc parser in `lambeq` [8], and fewer dataset simplifications. While we only considered productivity here, other aspects of compositionality could be explored (App. B), as well as adapting other NLP tasks to our framework. Further, we could explore ways of pre-training DisCoCirc word embeddings for general use, e.g. for in-task fine-tuning. The DisCoCirc framework invites quantum semantics beyond the expressive ansaetze used here (which lead to hard-to-simulate instances). There are *easy-to-simulate* quantum circuit families that would enable large-scale experimentation, e.g. Clifford circuits, which are however hard to optimise due to their combinatorial nature, and free fermions (i.e. matchgates) which come equipped with optimization-friendly continuous parameters [26]. Beyond a quantum implementation, we may also consider using neural networks in place of quantum circuits, for a fully classical model.

Acknowledgments and Disclosure of Funding

We thank Richie Yeung, Boldizsar Poor, Benjamin Rodatz, Jonathon Liu, and Razin Shaikh for support with DisCoPy and discussions on the formulation of text as process diagrams. We thank Nikhil Khatri for help with with training the classical baselines. We also thank the TKET and Hardware teams at Quantinuum for their support in compiling, submitting, and monitoring jobs to the H1-1 machine. Finally, we thank Harry Buhrman for discussions on information theory, and Ilyas Khan for supporting this long-term project.

References

[1] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[2] B. Coecke. The mathematics of text structure, 2019. arXiv:1904.03478.

[3] V. Wang-Mascianica, J. Liu, and B. Coecke. Distilling text into circuits. *arXiv preprint arXiv:2301.10595*, 2023.

[4] W. Zeng and B. Coecke. Quantum algorithms for compositional natural language processing. *Electronic Proceedings in Theoretical Computer Science*, 221, 2016. arXiv:1608.01406.

[5] B. Coecke, G. de Felice, K. Meichanetzidis, and A. Toumi. Foundations for near-term quantum natural language processing, 2020. arXiv preprint arXiv:2012.03755.

[6] Konstantinos Meichanetzidis, Alexis Toumi, Giovanni de Felice, and Bob Coecke. Grammar-aware sentence classification on quantum computers. *Quantum Machine Intelligence*, 5(1), Feb 2023.

[7] Robin Lorenz, Anna Pearson, Konstantinos Meichanetzidis, Dimitri Kartsaklis, and Bob Coecke. QNLP in practice: Running compositional models of meaning on a quantum computer. *Journal of Artificial Intelligence Research*, 76:1305–1342, April 2023.

[8] Dimitri Kartsaklis, Ian Fan, Richie Yeung, Anna Pearson, Robin Lorenz, Alexis Toumi, Giovanni de Felice, Konstantinos Meichanetzidis, Stephen Clark, and Bob Coecke. lambeq: An efficient high-level python library for quantum NLP. *arXiv:2110.04236*, 2021.

[9] Dominic Widdows, Aaranya Alexander, Daiwei Zhu, Chase Zimmerman, and Arunava Majumder. Near-term advances in quantum natural language processing. *Annals of Mathematics and Artificial Intelligence*, April 2024.

[10] Dominic Widdows, Willie Aboumrad, Dohun Kim, Sayonee Ray, and Jonathan Mei. Natural language, A, and quantum computing in 2024: Research ingredients and directions in QNLP. *arXiv preprint arXiv:2403.19758*, 2024.

[11] Vedran Dunjko and Hans J Briegel. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Reports on Progress in Physics*, 81(7), June 2018.

[12] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4), November 2019.

[13] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, October 2014.

[14] Sean Tull, Robin Lorenz, Stephen Clark, Ilyas Khan, and Bob Coecke. Towards Compositional Interpretability for XAI. *arXiv:2406.17583*, 2024.

[15] Michael Ragone, Bojko N. Bakalov, Frédéric Sauvage, Alexander F. Kemper, Carlos Ortiz Marrero, Martin Larocca, and M. Cerezo. A unified theory of barren plateaus for deep parametrized quantum circuits. *arXiv:2309.09342*, 2023.

[16] Quantinuum system model H1 product data sheet. `https://assets.website-files.com/62b9d45fb3f64842a96c9686/654528cf9dd7824353bd9b56_QuantinuumH1ProductDataSheetv6.03OOct23.pdf`. Accessed: 2024-02-27.

[17] Bob Coecke. *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, chapter The Mathematics of Text Structure, pages 181–217. March 2021.

[18] Vincent Wang-Mascianica, Jonathon Liu, and Bob Coecke. Distilling text into circuits, 2023.

[19] Jonathon Liu, Razin A. Shaikh, Benjamin Rodatz, Richie Yeung, and Bob Coecke. A pipeline for discourse circuits from ccg. *arXiv:2311.17892*, 2023.

[20] Tuomas Laakkonen, Konstantinos Meichanetzidis, and Bob Coecke. Quantum algorithms for compositional text processing. (in preparation).

[21] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

[22] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[23] Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers can achieve length generalization but not robustly. *arXiv preprint arXiv:2402.09371*, 2024.

[24] E. Grefenstette and M. Sadrzadeh. Experimental support for a categorical compositional distributional model of meaning. In *The 2014 Conference on Empirical Methods on Natural Language Processing.*, pages 1394–1404, 2011. arXiv:1106.4058.

[25] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. In J. van Benthem, M. Moortgat, and W. Buszkowski, editors, *A Festschrift for Jim Lambek*, volume 36 of *Linguistic Analysis*, pages 345–384. 2010. arxiv:1003.4394.

[26] Gabriel Matos, Chris N. Self, Zlatko Papić, Konstantinos Meichanetzidis, and Henrik Dreyer. Characterization of variational quantum algorithms using free fermions. *Quantum*, 7:966, March 2023.

[27] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12), October 2019.

[28] Bob Coecke and Konstantinos Meichanetzidis. Meaning updating of density matrices. *arXiv preprint arXiv:2001.00862*, 2020.

[29] Giovanni de Felice, Alexis Toumi, and Bob Coecke. Discopy: Monoidal categories in python. *Electronic Proceedings in Theoretical Computer Science*, 333:183–197, February 2021.

[30] Chase Roberts, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer. Tensornetwork: A library for physics and machine learning. *arXiv:1905.01330*, 2019.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Ben Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. In *NeurIPS Systems for ML Workshop*, 2018.

[34] Ax - adaptive experimentation platform. `https://github.com/facebook/Ax`.

[35] Quantinuum H1-1. https://www.quantinuum.com/. February 13-March 1, 2024.

[36] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. TKET: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, 6(1):014003, November 2020.

[37] Matthew DeCross, Eli Chertkov, Megan Kohagen, and Michael Foss-Feig. Qubit-reuse compilation with mid-circuit measurement and reset. *Phys. Rev. X*, 13:041057, Dec 2023.

[38] OpenAI. GPT-4 technical report. *arXiv:2303.08774*, 2024. GPT-4 accessed: 25/04/2024.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2023.

[40] M. Abadi *et. al.* TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[41] Solving the Prerequisites: Improving Question Answering on the bAbI Dataset. `https://cs229.stanford.edu/proj2015/333_report.pdf`. Accessed: 2024-05-30.

[42] J. B. Altepeter, E. R. Jeffrey, M. Medic, and P. Kumar. Multiple-qubit quantum state visualization. In *2009 Conference on Lasers and Electro-Optics and 2009 Conference on Quantum electronics and Laser Science Conference*, pages 1–2, 2009.

[43] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC '23. ACM, June 2023.

[44] Enrico Fontana, Manuel S Rudolph, Ross Duncan, Ivan Rungger, and Cristina Cîrstoiu. Classical simulations of noisy variational quantum circuits. *arXiv preprint arXiv:2306.05400*, 2023.

[45] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, August 2019.

[46] Johnnie Gray and Stefanos Kourtis. Hyper-optimized tensor network contraction. *Quantum*, 5:410, March 2021.

[47] Daniel G. a. Smith and Johnnie Gray. opt_einsum - a python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, 2018.

[48] H1-1 device emulator specification. `https://assets.website-files.com/62b9d45fb3f64842a96c9686/654528e836f471097a408357_Quantinuum%20H1%20Emulator%20Product%20Data%20Sheet%20v6.6%2030Oct23.pdf`. Accessed: 2024-02-19.

[49] Samuel Duffield, Gabriel Matos, and Melf Johannsen. qujax: Simulating quantum circuits with JAX. *Journal of Open Source Software*, 8(89):5504, September 2023.

[50] Kyungjoo Noh, Liang Jiang, and Bill Fefferman. Efficient classical simulation of noisy random quantum circuits in one dimension. *Quantum*, 4:318, September 2020.

[51] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs. `http://github.com/google/jax`, 2018.

[52] Amit Sabne. XLA: Compiling machine learning for peak performance. *Google Res*, 2020.

Contents

A QDisCoCirc

As outlined in Sec. 2 text circuits are composed of *states*, *effects*, and *boxes*, which are to be understood as inputs, output *tests*, and *processes* that transform inputs into outputs [17, 18]. We also consider special boxes, such as the 'identity' and 'swap' boxes, as well as the special 'discard' effect (denoted by $\mathbb{D}$). These constitute the *generators* of text diagrams, and are displayed in Fig. 3.

The semantic functor implemented by a QDisCoCirc model is structure-preserving, as it is applied component-wise on the generators. The specific functor chosen in this work maps states to quantum states (which are prepared by unitary transformations from a fixed reference product state), and effects to measurements (whose values are the probabilities of specific outcomes). The identity and swap boxes are mapped to the usual identity and swap gates in quantum circuits, and the discard effect is mapped to the discard of a quantum system, i.e. a partial trace. Finally, boxes are mapped to quantum channels, which are realised by unitaries with ancilla qubits that get discarded. The action of the semantic functor to the generators is shown in Fig. 3.

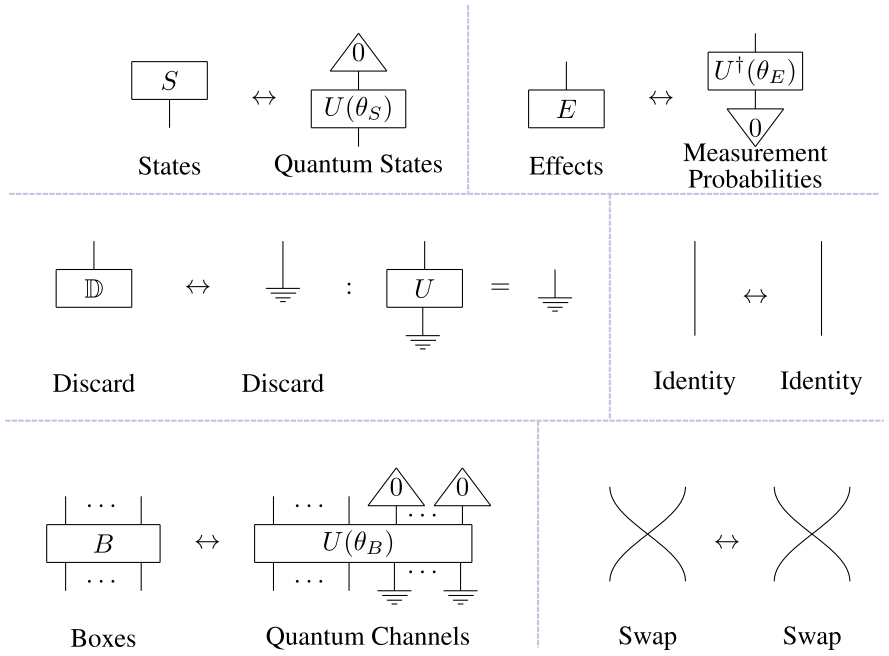


Figure 3: Left-hand side of each subdivision: text generators, with which any text diagram is composed. There are three generators considered in this work: states, which encode inputs to a process, effects, which act as tests on states, and boxes, which transform states. We also consider special boxes, such as the identity, (the trivial operation) and the swap (the permutation of wires), as well as the special discard effect (the deletion of a wire). Right-hand side of each subdivision: the action of the semantic functor that instantiates the QDisCoCirc model used in this work. Every wire is assigned n qubits. States are mapped to pure quantum states prepared by a unitary U from $|0\rangle^{\otimes n}$. Effects are mapped to measurement probabilities of specific outcomes (state unpreparations, hence the dagger). The discard effect is interpreted as discarding the corresponding qubits (partial trace). Boxes, in general, are mapped to quantum channels, realised via a unitary, with potentially some ancillae that get discarded. The special boxes of identity and swap are mapped to the equivalent identity and swap unitary operations.

A.1 Question answering

Using the semantic functor, we instantiate the question answering task as quantum circuits. These are then evaluated and subjected to classical post-processing, yielding the answer to the question. In this work, we focus on binary questions, which correspond to one effect for the positive answer and one effect for the negative answer. Evaluating the two resulting circuits, we obtain two scalars, which are passed into a softmax function, to obtain the answer to the question. We took this approach following the method set out in [20], but several alternatives exist. For instance, we could train an observable so that the measured value corresponds to the answer we wish to obtain, although this does not keep to the principle established in [20] of modelling state overlaps as the overlaps in meaning between text circuits. By choosing our method instead, we obtain a more transparently interpretable model. We could also have measured only one effect (for instance, the affirmative statement), but we found that by using the maximum of two effects the Hilbert space can be split more evenly, making the task easier to learn.

We show the structure of a text diagram for the question answering task in Fig. 4, while Fig. 5 depicts an example of what the model evaluates.

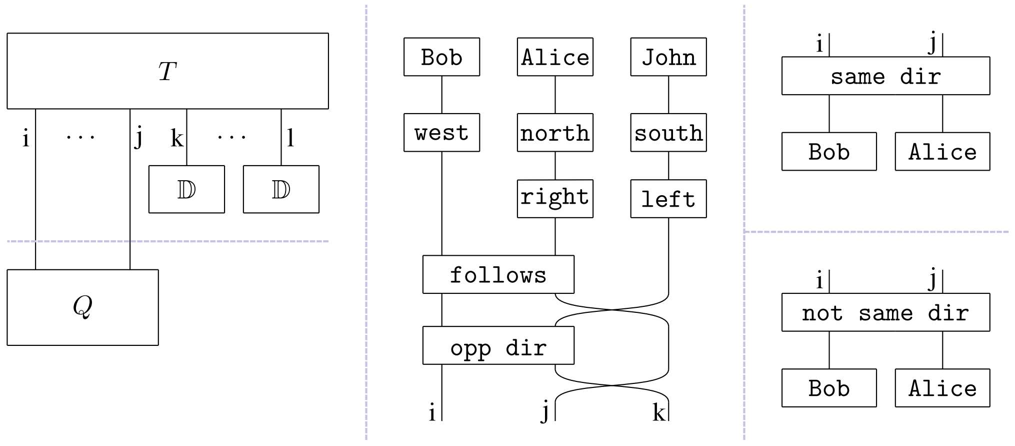


Figure 4: Left: Question answering as a process diagram in the DisCoCirc framework. A text-state T prepares the state of affairs. A question-effect Q, which is posed as an affirmative statement, tests the extent to which the statement holds using the primitive of text-similarity (state overlap), as introduced in [20]. Nouns irrelevant to the question are discarded. The horizontal dashed line is a guide to the eye and separates the text state from the question effect. Middle: An example text diagram from the 'following' datasets that we have constructed for the story: `Bob walks west. Alice walks north. John walks south. Alice turns right. John turns left. Bob follows Alice. Bob goes in the opposite direction of John.` Here, noun-states (`Bob`, `Alice`, `John`) are transformed by boxes representing intransitive verbs (`walks west`, `walks north`, `walks south`, `turns right`, `turns left`) and transitive verbs (`follows`, `goes in the opposite direction of`). The diagram prepares the composite text state. Right: The question effects (`goes in the same direction as`, `does not go in the same direction as`) test the degree to which `Bob` and `Alice` satisfy those statements.

A.2 Semantic functor

The semantic functor, as outlined in Sec. 2, is defined in terms of some hyperparameters, which we specify here. Each wire is assigned one qubit. Every unitary $U(\theta)$ is implemented using three layers of Circuit 4, which we show in Fig. 6. This parameterised quantum circuit is taken from Ref. [27], which studied a diverse set of ansaetze with regard to their entangling capabilities and their expressivity. Specifically for initialisations of single qubit states we used the *Euler paramterisation* which is defined as $|\psi\rangle = R_x(\theta_3)R_z(\theta_2)R_x(\theta_1)\,|0\rangle$.

In Fig. 7 we show in more detail the semantic functor as we have defined it for the specific vocabulary and question-answering task described in Sec. 2 and Sec. 2, including specifying which verbs are assigned pure unitaries and which are assigned channels with the addition of ancillae that are then discarded. The channels allow for meaning updating, in the spirit of [28]. In theory, any box in the text circuit could be assigned a channel while ensuring that only one of the text circuit and question circuit is mixed, which is a requirement for question answering as formulated here to be valid ([20], Section 3). This allows information to be discarded, which is beneficial, but since it requires ancilla qubits and more two-qubit gates, we want to avoid this wherever possible. As we expect that only the `follows` box would need to discard information, we assign all other boxes as unitaries. Note

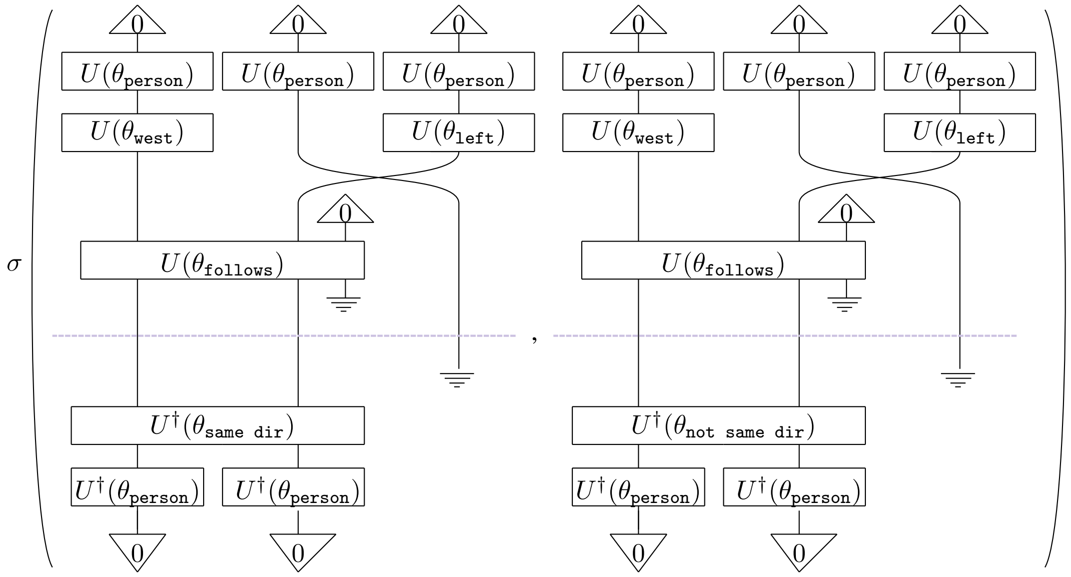

σ
$U(\theta_{\texttt{person}})$
$U(\theta_{\texttt{west}})$
$U(\theta_{\texttt{left}})$
$U(\theta_{\texttt{follows}})$
$U^\dagger(\theta_{\texttt{same dir}})$
$U^\dagger(\theta_{\texttt{not same dir}})$
$U^\dagger(\theta_{\texttt{person}})$
0

Figure 5: Our pipeline for question answering with QDisCoCirc. The effects for the positive and negative answer to the binary question are applied to the same quantum text state, using the primitive of text overlap (the dashed line separates the two texts whose overlap is being measured). Each diagram represents the probability of the all-zeros state being measured in the computational basis. Then, these two probabilities are passed to the softmax function (σ) to determine the answer to the question.

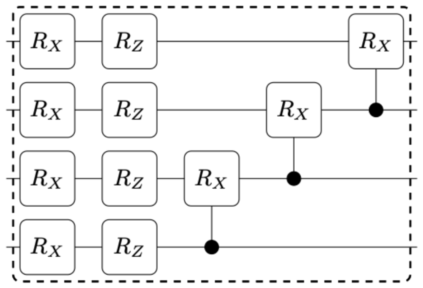


Figure 6: One layer of the ansatz used in this work, namely Circuit 4 from Ref. [27]. Specifically, here we show its 4-qubit version. This parameterised quantum circuit is used to implement all unitaries throughout this work.

that all actor names, such as `Alice` or `Bob`, in the stories, are replaced by the word `person` and share the same set of parameters $\theta_{\texttt{person}}$ among all circuits that prepare quantum noun-states. The specific actors are identified by their position in the circuit, i.e. their corresponding wire, by the fact that the tensor product is non-commutative.

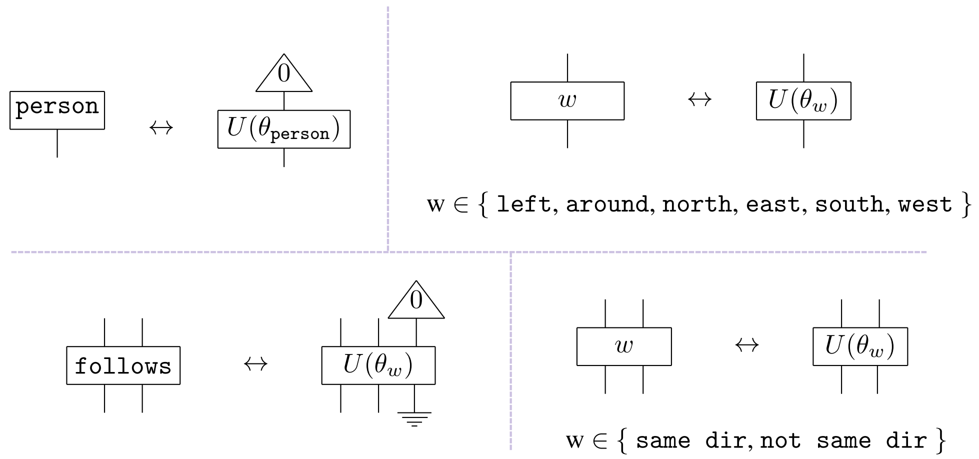


Figure 7: Semantic functor applied to the specific vocabulary used in this work. Each wire is assigned one qubit. All nouns are expressed as `person`. The word boxes (on the left) for `turns left`, `turns around`, `walks north`, `walks east`, `walks south`, and `walks west`, are assigned pure unitaries (on the right). The Box for `follows` gets an ancilla qubit which is then discarded. For each word w, the associated unitary $U(\theta_w)$ is controlled by a set of parameters that is unique to that word. `turns right` and `goes in the opposite direction of` are substituted according to the semantic rewrites in Fig. 8.

B Tests of compositionality

In this work, we focused on one test of compositionality, namely *productivity*, as defined in Ref. [21]: the ability of a model that is trained on small examples to use the learned rules to generalize to larger instances. Ref. [21] introduces four other measures of compositionality. They can also be adapted to the DisCoCirc framework. For all of them, we can provide quantitative measures in terms of differences in test and train accuracies, in a way that can be easily compared with other baseline models.

Systematicity is akin to the vanilla generalisation performance that most supervised learning setups measure by performing inference on a held-out test set of the same type of data as those that the model was trained on. In DisCoCirc, one would measure the effect of swaping word-boxes with other ones of the same shape (ie part of speech), and measure the effect this intervention has on the test accuracy. In other words, we test on instances created by the reshuffling of components that the model has already been trained on, where the reshuffling respects the grammatical structure.

Another property of compositionality is *substitutivity*, which tests to what effect equivalent instances that have different representation lead to the same prediction. In language, this is essentially analogous to paraphrase detection, and in DisCoCirc this can be quantified by measuring the effect of the application of axioms, such as the semantic rewrites of Fig. 8, to the data has on the model's prediction. Although we are replacing diagram segments as in the case of test systematicity, in the case of substitutivity, we do not expect a change in the final labels.

The property of *overgeneralisation* is an aspect of compositionality that reflects the degree to which the model is attempting to apply rules it has learned to situations where it should not. Even though this property shows a manner in which the model fails at a task, this quantifies that the model indeed does not just memorise the data but rather actually learns some underlying rules. To test this, we can compare the performance of models trained on datasets with increasing amounts of noise on a noiseless test dataset. Assuming that the noise added is random, and corrupts some of the labels in the training data, we can identify two different trends - overgeneralising and overfitting. Provided the noise is not so significant the dataset becomes effectively random, we can identify a model that overgeneralises as one who's test accuracy is higher than seen in training, as it learns the underlying compositional rules by ignoring the noise in the training set. Conversely, a model that overfits will achieve high accuracy in training by memorising samples, but will have worse accuracy on the test set as these memorised examples will no longer be relevant.

Finally, *localism* measures how local versus global the compositional structure is. DisCoCirc models are inherently local, however we can also test whether the semantics compose locally or globally using feature ablation, with the help of entanglement measures between the quantum systems (carried by the wires).

C Datasets

C.1 Generation

Here we specify in more detail how our synthetic datasets are generated.

Two-directional dataset: The actors can walk in two directions, north and south, and turn by $180°$. The set of possible actions, i.e. verbs, in this dataset, is:

$$\{\texttt{walks north},\ \texttt{walks south},\ \texttt{turns around},\\ \texttt{follows},\ \texttt{goes in the opposite direction of}\}.$$

Four-directional dataset: The actors can walk in all four cardinal directions, turn by $90°$ and $180°$. The set of possible actions in this dataset is:

$$\{\texttt{walks north},\ \texttt{walks south},\ \texttt{walks east},\ \texttt{walks west},\ \texttt{turns right},\\ \texttt{turns left},\ \texttt{turns around},\ \texttt{follows},\ \texttt{goes in the opposite direction of}\}.$$

The set of questions we use for both datasets is:

$$\{\texttt{goes in the same direction as},\ \texttt{does not go in the same direction as}\}.$$

Fig. 4 shows an example of a diagram from the four-directional dataset for an example story. For both these datasets, we generated sub-datasets with different story densities. The density of a story is related to the entanglement generated by the story, as interactions between actors (transitive verbs) are instantiated as entangling operations. We define the density of a story to be the number of two-actor interactions within the story divided by the number of sentences in the story and create the following five different subsets according to this definition.

simple, deeper: These datasets are generated by randomly applying actions to actors until a chosen number of sentences is reached. The simple dataset contains stories from 2 to 30 actors, and the deeper dataset contains stories from 6 to 30 actors. The stories in the deeper dataset also have more sentences than the stories in the simple dataset.

less dense, dense, superdense: To ensure high connectivity of the nouns in these datasets, we first generate fully connected stories, where each actor interacts with every other actor in the story exactly once. Then, we add some single-actor actions, shuffle the sentences, and cut the story off after a chosen number of sentences. The proportion of single-actor actions decreases going from less-dense to superdense. Like the deeper dataset, these datasets contain stories from 6 to 30 actors, and have the same number of sentences as the stories in the deeper dataset.

Table 2 below shows the number of data entries per dataset referenced in the paper.

C.1.1 Story densities

The density of a story is the number of two-actor interactions within the story divided by the number of sentences in the story. The density per dataset is shown in Table 3 below.

C.2 Dataset splits

Two-directional dataset: The model is trained on the simple dataset stories with up to 8 actors. 20% of the simple dataset with up to 8 actors is used as validation data; we refer to this set as *Valid A*.

Table 2: Number of data entries per dataset. For the training on the four-directional dataset all densities were used so the *Train* dataset size is 874 and *Valid A* is 218 which is 0.8/0.2 of the total respectively. Note that not all data entries for the datasets with 21-30 nouns (*Test*) were used due to device constraints. For details of which data entries were used from these datasets see App. C.2 and E.

Dataset	**Two-directional**			**Four-directional**		
(# entries)	*Train, Valid A*	*Valid Comp*	*Test*	*Train, Valid A*	*Valid Comp*	*Test*
simple	492	864	480	492	864	480
deeper	0	750	500	150	600	500
less-dense	0	750	500	150	600	500
dense	0	750	500	150	600	500
superdense	0	750	500	150	600	500

Table 3: Average densities of the two-directional and four-directional datasets.

Dataset	**Two-directional**		**Four-directional**	
(% density)	*Train, Valid A, Valid Comp*	*Test*	*Train, Valid A, Valid Comp*	*Test*
simple	26.5	24.8	26.6	25.0
deeper	48.2	54.3	48.0	54.3
less-dense	50.0	66.5	50.4	66.6
dense	58.2	71.6	58.1	71.6
superdense	68.7	77.5	68.7	77.5

The stories from all other (deeper to superdense) datasets with up to 8 actors, and the stories from all datasets (simple to superdense) with 9 to 20 actors, are used as a second compositionality validation dataset, referred to as *Valid Comp*. This set is used to pick the model with the best generalisation performance. Stories from all datasets with 21 to 30 actors, which compile down to 20 qubits or less with qubit reuse, form the test set. We simulate up to 20 actors and send a selection of circuits between 9 and 30 actors to Quantinuum's ion-trap quantum computer H1-1 (referred to as *Valid Comp (H1-1)* and *Test (H1-1)* depending on which dataset the instances were sampled from). See App. E for more details of the machine.

Four-directional dataset: The model is trained on all (simple to superdense) stories with up to 8 actors. 20% of the dataset with up to 8 actors is used as validation data. The stories from all datasets (simple to superdense) with 9 to 20 actors are used as a second compositionality validation dataset (*Valid Comp*), which is used to pick the model with the best generalisation performance. Stories from all datasets with 21 to 30 actors, which compile down to 20 qubits or less with qubit reuse, form the test set. We select a subset of these circuits to evaluate using shot-based noisy simulation (referred to as *Test (qujax)*).

Table 4 shows a detailed overview of the splits used in our experiments.

C.3 Semantic rewrites

To help the model learn a compositional solution, and reduce the number of parameters needed, we implemented *semantic rewrites* into the diagrams. These rewrites are depicted in Fig. 8, for both the two- and four-directional datasets. The semantic functor defines the word circuits $U(\theta_{\texttt{opp dir}})$

Table 4: Detailed splits used in our experiments. A visual representation is provided in Figs. 10, 12 and 14.

Dataset	**Two-directional**			**Four-directional**		
	Samples	Actors	Density	Samples	Actors	Density
Train	80%	2-8	simple	80%	2-8	all
Valid A	20%	2-8	simple	20%	2-8	all
Valid Comp	all	9-20	simple	all	9-20	all
	all	6-20	deeper-superdense			
Valid Comp (H1-1)	120	9-20	dense		-	
	120	9-20	superdense			
Test	all	21-30	all	all	21 - 30	all
Test (H1-1)	82	21-30	dense		-	
	47	21-30	superdense			
Test (qujax)		-		86	21-30	dense
				45	21-30	superdense

and $U(\theta_{\texttt{around}})$ in terms of $U(\theta_{\texttt{follows}})$ and $U(\theta_{\texttt{left}})$. Further, we impose `turns right` and `turns left` being inverse of each other by defining one circuit with respect to the other as $U(\theta_{\texttt{right}}) = U^{\dagger}(\theta_{\texttt{left}})$.

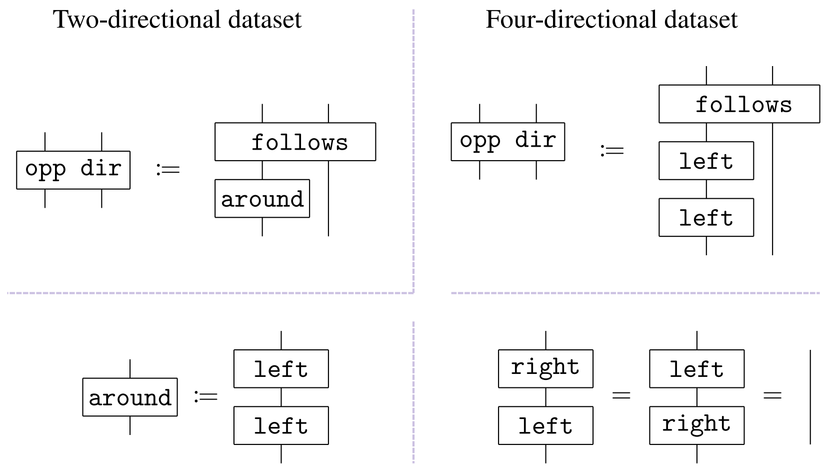


Figure 8: Hardcoded semantic rewrites for our model. For the two-directional dataset `goes in the opposite direction of` is implemented as a consecutive application of `follows` and `turns around` on the first actor. In the four-directional dataset we additionally replace `turns around` with two consecutive applications of `turns left`. Further, we enforce the consecutive application of `turns right` and `turns left` to be the identity.

D Training

D.1 Methodology

To obtain diagrams from our text-level stories we implemented a parser, based on earlier work [19] and a soon to be released tool for DisCoCirc on real world data in `lambeq` [8], using the `DisCoPy` [29] package. Our parser maps the individual words to their respective boxes and arranges them in the structure set by the DisCoCirc model. We then use the `lambeq` [8] package to apply the semantic functor explained in App. A.2 to the diagrams to generate the quantum circuits used in training.

The resulting quantum circuits are converted into tensor networks, which are evaluated exactly via tensor contraction using `Tensornetwork` [30] on Quantinuum's *duvel4* server (see App. I.1 for specifications). We use `PyTorch` [31] to track the parameters and the Adam optimizer [32] to train them. The hyper-parameters are tuned using `Ax` [33, 34] and documented below.

At each training epoch, the model is evaluated on the entire training dataset in batches. We save the parameters obtained at the end of each epoch and record the loss and validation accuracy. Every three epochs, starting from the first, we evaluate and record the accuracy of the model on the training dataset (we do not do this every epoch in order to keep model training times down). We select the model from the epoch with the best validation accuracy, tie-breaking if necessary with the closest logged training accuracy, then loss.

D.2 Hyperparameter tuning

The hyperparameters we considered and their final tuned values are summarised in Table 5 below. The tuning was conducted using the four-directional dataset. We did not do any further tuning on the two-directional dataset, as the model already achieved a high accuracy with the first hyperparameters we chose. The optimisation of the tuning was for the best *Valid A* accuracy, and the tuning was carried out using `Ax` [33, 34]. Each tuning run was run for 15 epochs. From these runs we selected the best hyperparameters and ran some further training runs for 200 epochs. To select our final model, we first picked 2 candidate models from the tuning runs with high *Valid A* accuracy. We then evaluated both on the *Valid Comp* dataset, and selected the model with the best generalisation performance.

Table 5: Training hyperparameters used and the ranges we considered when tuning on the four-directional dataset. The learning rate and init param seed were 'range parameters' chosen from any value in the range specified, and the batch size was a 'choice parameter' chosen from a list of the values of 2 to the power of 0 through 8.

hyperparameter	**range considered**	**final value**
learning rate	0.0001 - 0.1	0.02840955
batch size	choice: 2^0, 2^1 - 2^8	256
init param seed	range: 0 - (2^{32}-1)	1151618203

For the two-directional dataset, we did not tune the hyperparameters. Table 6 summarises the values used.

D.3 Cross-validation

We perform 5-fold cross-validation using the hyperparameters chosen for the two-directional dataset, with 5 iterations per fold. Each model was trained for 10 epochs. For each fold, we pick the model with highest *Valid A* accuracy and evaluate it on the *Valid Comp* dataset (see App. C.2). The results are summarised in Fig. 9. All but one model managed to generalise perfectly, matching the performance of our final model. Note that some of the larger datapoints failed to evaluate. The total number of instances for each number of actors was 72 for simple and 50 for all other densities.

Table 6: Training hyperparameters used for the two-directional dataset. The initial parameters were recorded, but not explicitly seeded.

hyperparameter	**final value**
learning rate	0.005
batch size	1

We report the number evaluated for the averages, and when not all available datapoints were evaluated. The error bars show the 95% Clopper-Pearson confidence interval for the average accuracy of each split. From this we see that the generalisation capacity of the model is broadly independent of the initialisation, however further selection beyond *Valid A* is required to ensure that the final model generalises *compositionally*: in our case, although all models achieved 100% accuracy on the *Train* and *Valid A* datasets, we see that the last model failed to generalise as well on the *Valid Comp* dataset.

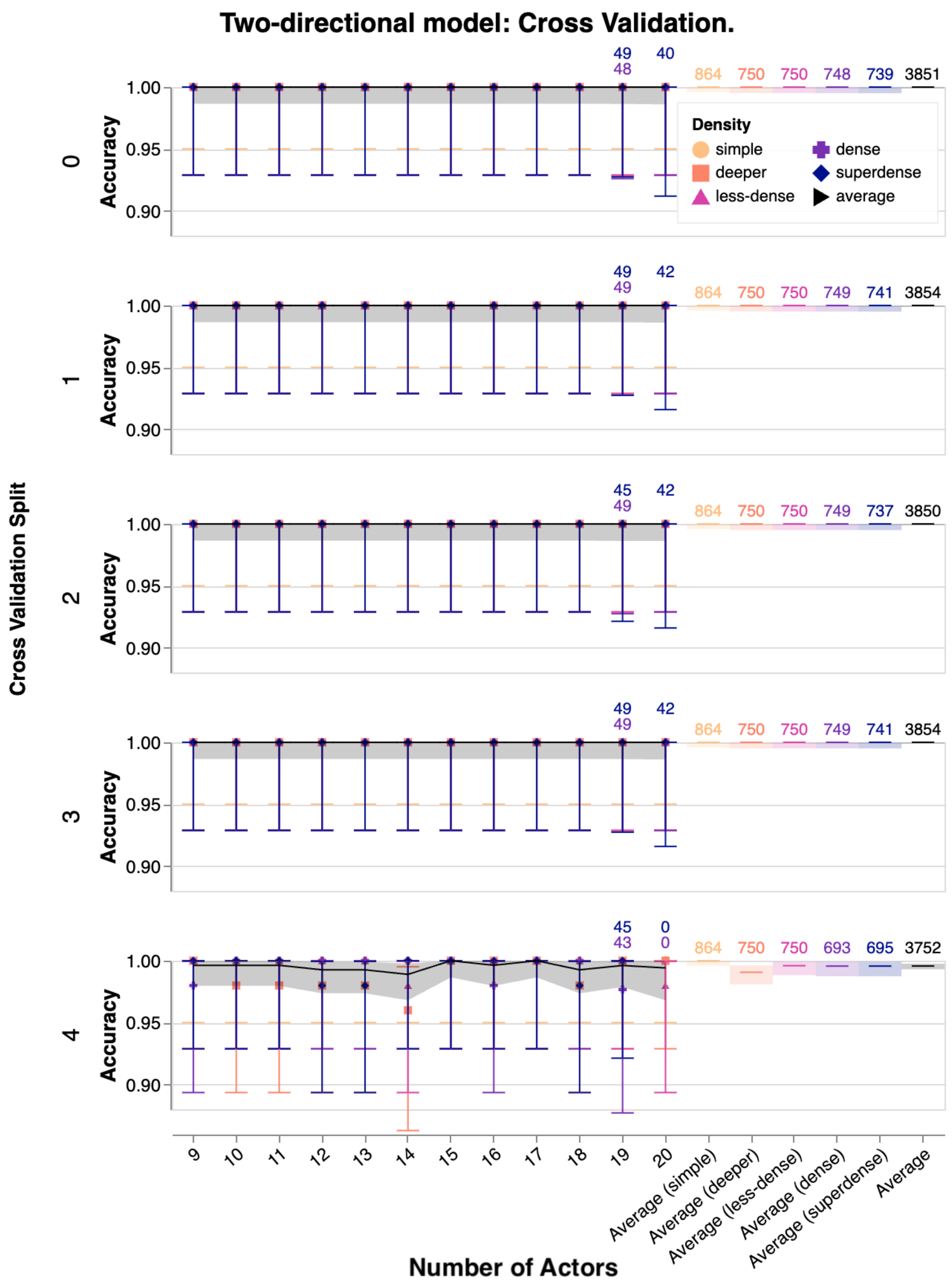


Figure 9: Cross-validated compositional generalisation for the two directions dataset on the *Valid Comp* split, shown per cross-validation split.

E Implementation on H1-1

We now present in more detail the experimental methodology and results from the execution of the question-answering task for the two-directional dataset on Quantinuum's H1-1 quantum computer [16, 35]. H1-1 is a state-of-the-art ion-trap quantum computer supporting 20 physical qubits, reaching 0.998 two-qubit gate fidelity and featuring any-to-any gate connectivity. This connectivity allows for any text circuit, which will in general be non-local, to be compiled without additional overhead. Further, the circuit ansatz we have chosen in Sec. 2 is hardware efficient, in the sense that its entangling gates can be efficiently expressed in terms of the native gate set of H1-1 (see App. E.1).

As discussed in App. C.2, the *Valid Comp (H1-1)* randomly sampled balanced subset from the *Valid Comp* dataset was selected to be evaluated on the H1-1 device. The *Test (H1-1)* set is formed from a further 82 data points sampled from the 'dense' *Test* set and 47 data points from the 'superdense' *Test* – these circuits were those that compiled down to 20 or fewer qubits with qubit reuse, and hence would fit on the device. Each circuit was repeated for a total of 50 shots. Note that each data point corresponds to two circuits: one for the positive question and another for the negative question. The circuits were first converted to `TKET` [36], before undergoing a qubit reuse compilation pass using the local greedy algorithm described in [37]. This pass attempts to reduce the number of qubits used in the circuit by identifying those which do not need to be kept for the full computation, and then reusing them where new qubits would need to be introduced. The result is a reduction in the number of qubits necessary to execute the circuit, in exchange for a higher number of measurement errors (which occur when resetting the qubits to be reused) and a deeper circuit.

The increase in the circuit depth resulting from qubit reuse is shown in Fig. 10 (all datasets) and Fig. 12 (subset sent to H1-1) of App. C. We exhibit the correlation between number of actors (text width) and text depth built into the data. Note that in Fig. 10, for densities other than simple, the dataset composition is almost exactly the same, and are thus overlapped on the display. The opacity of the points reflects the number of datapoints present. For each dataset, we show the distribution of two-qubit gate depth, post-compilation, with qubit reuse according to the number of actors. We plot the maximum and minimum gate depth as lines, marking the mean value with a point. The vertical lines represent the bootstrapped 95% confidence interval for the post-compilation mean, while the shaded area shows the extent of the pre-compilation gate depths for comparison. The resulting reduction in the number of qubits needed to execute the circuits is shown in Fig. 11 (all datasets) and Fig. 13 (subset sent to H1-1) of App. C. Circuits were compiled for Quantinuum's H1-1 ion-trap device, which has 20 qubits. We plot the average, with 95% confidence interval error bars, and shade the region covered by each dataset type.

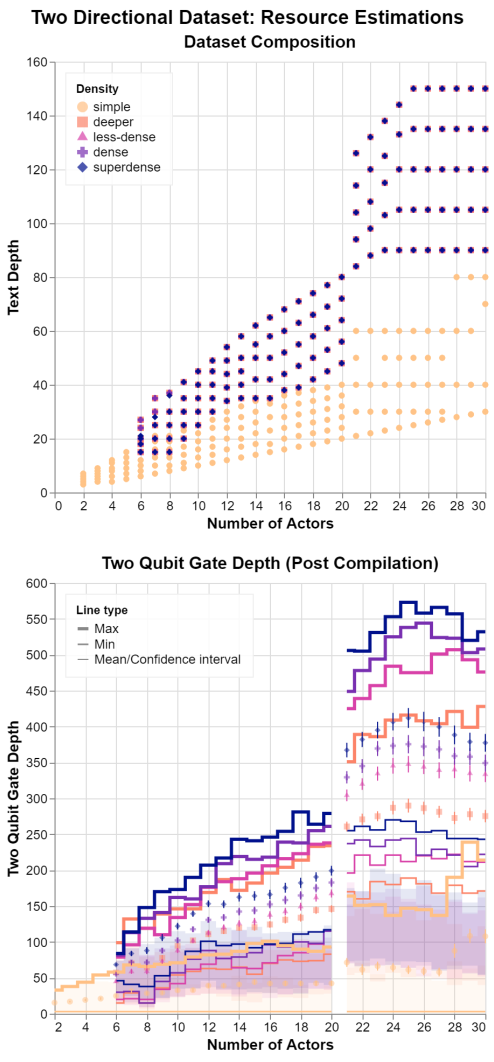
The results of evaluating the circuits on the H1-1 device are displayed in Fig. 2(a). As a result of the qubit reuse described above, we were able to compile circuits exceeding the 20 qubit limit of the H1-1 device down to this threshold, including some circuits with up to 30 actors. The largest of these circuits originally had 108 qubits. This large number of qubits arises mainly from the ancillas used for certain words, as each actor only takes one qubit each in our model. For a discussion on the use of ancilla qubits see App. A.2. As discussed in Sec. 3, we see that there is no significant drop in the model accuracy when the number of actors increases beyond what was used during training, confirming that the model successfully generalises to longer texts. This also shows that the H1-1 device noise levels do not corrupt this compositional generalisation for the text lengths considered. This agrees with the results in App. I.2, where we numerically simulate an evaluation of the model on the *Valid Comp* set using a noise model that emulates the native noise of the H1-1 device.

E.1 Conversion of ansatz to H1-1 native gates

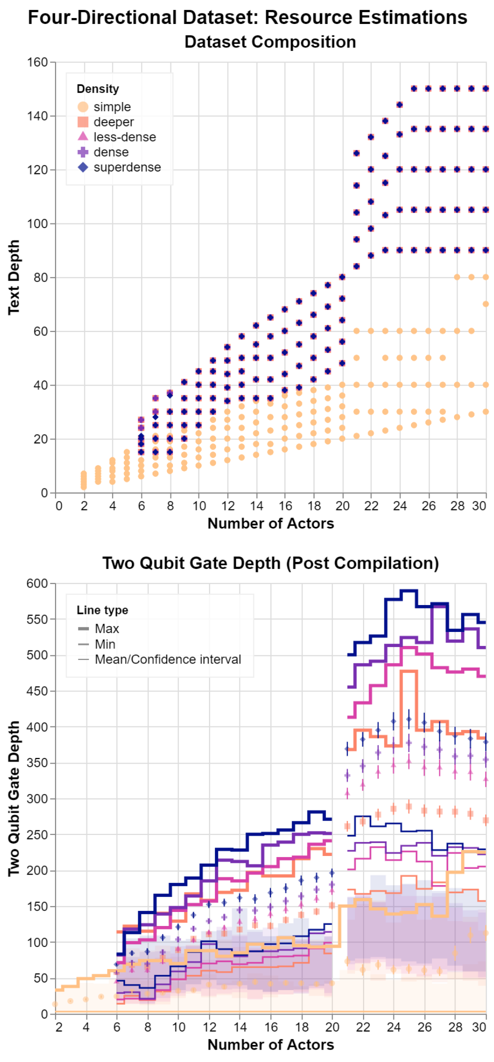
The ansatz employed (see Sec. 2) uses CR_X 2-qubit gates and R_Z 1-qubit gates. The R_Z gate is native to the H1-1 archtecture [16], while the CR_X gate can easily be converted to a R_{ZZ} gate. Indeed, given $i \neq j$ indexing qubits of the device, it is the case that

$$\begin{aligned} CR_X(\theta, i, j) &= H(j) CR_Z(\theta, i, j) H(j) \\ &= H(j) R_Z\left(-\frac{\theta}{2}, i\right) R_Z\left(-\frac{\theta}{2}, j\right) R_{ZZ}\left(\frac{\theta}{2}, i, j\right) H(j), \end{aligned}$$

where $H(j)$ are Hadamard gates acting on the jth qubit and $CR_X(\theta, i, j)$ is a controlled X rotation acting on the jth qubit controlled by the ith qubit (and analogously for CR_Z). In practice, this conversion between 2-qubit gates is not explicitly performed and is instead delegated to the `TKET` compiler [36].



(a) Two-directional dataset



(b) Four-directional dataset

Figure 10: Further dataset characterisation.

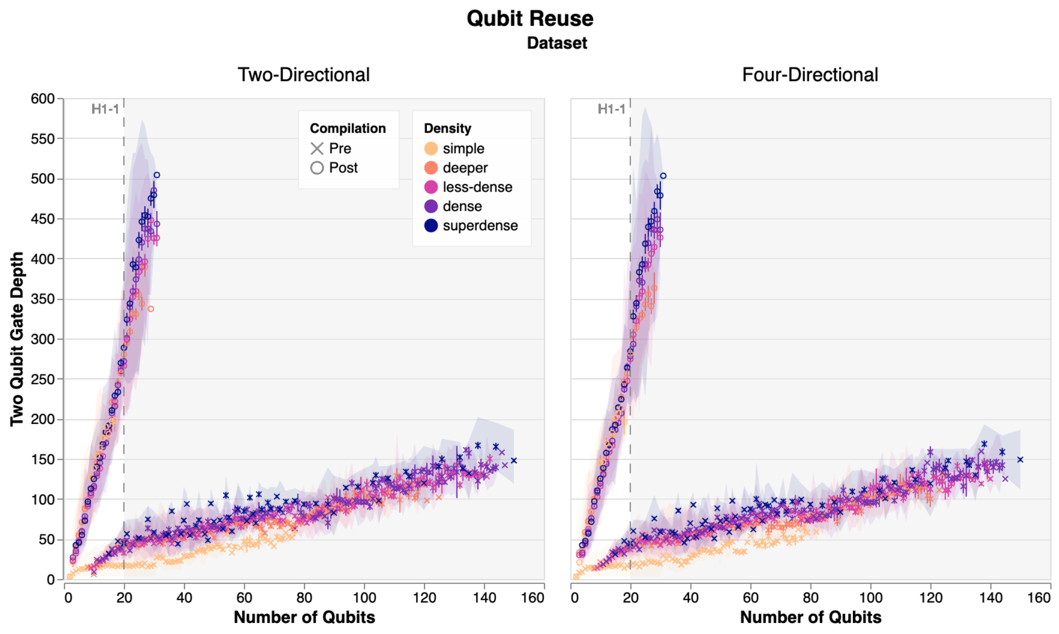


Figure 11: The effect of qubit reuse on the entire Following datasets.

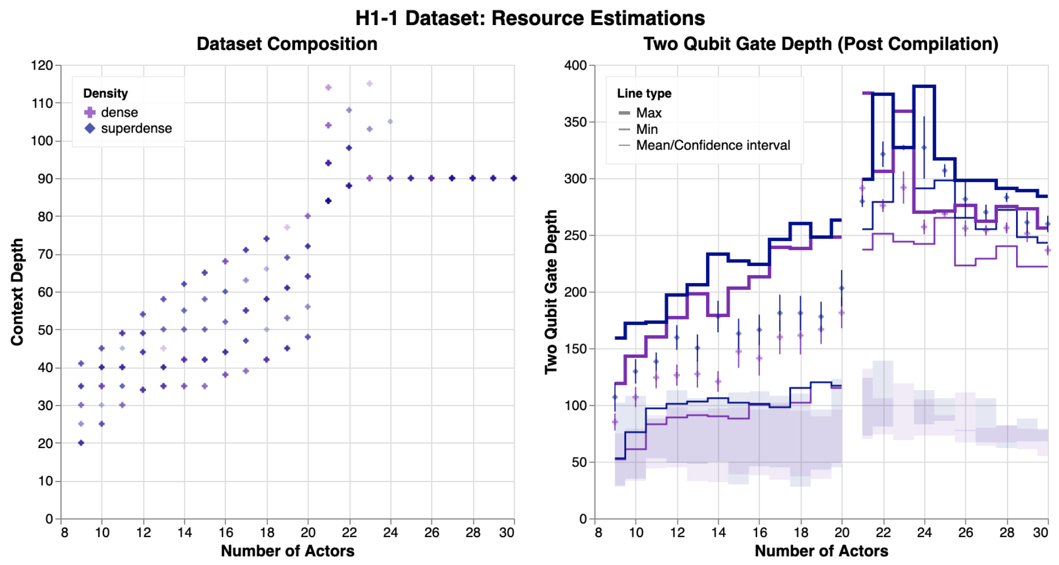


Figure 12: Dataset characterisation for the subset of the two-directional dataset sent to H1-1.

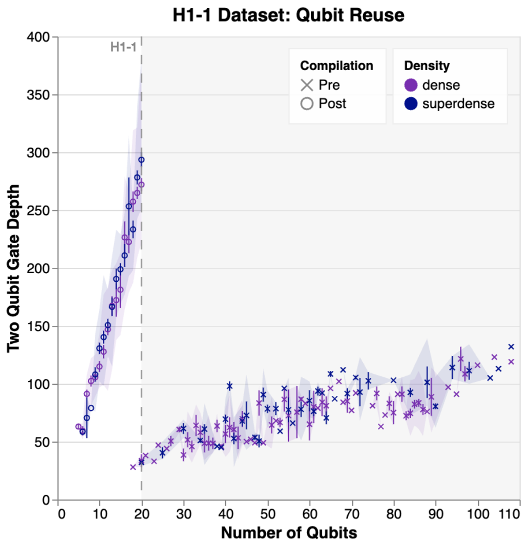


Figure 13: Effect of qubit reuse for the subset of the two-directional dataset sent to H1-1.

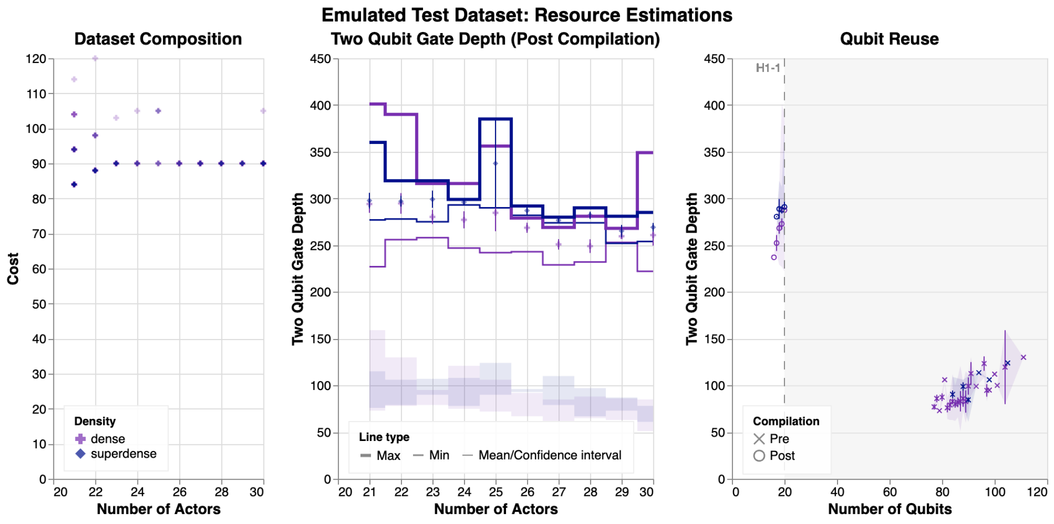


Figure 14: Dataset characterisation for the subset of the four-directional *Test (qujax)* dataset and effect of qubit reuse.

F Compositional generalisation

Here we give a more detailed breakdown of the compositional generalisation curves, looking at each density separately. Fig. 15 shows the two-directional model, whilst Fig. 16 displays the four-directional model. The 95% Clopper-Pearson confidence interval is shown as a shaded region; the variation in width reflects the number of datapoints that were evaluated. For the two-directional model, the training set (simple up to 8 nouns) is shown separately, and the model achieved 100% accuracy on all datasets. Note that for the deeper to superdense datasets, this means that the error bars are mostly overlapping.

F.1 H1-1 and *Test (qujax)* sub-datasets

Both the four- and two-directional datasets correlate the number of actors with text depth, however both the *Test (H1-1)* and *Test (qujax)* subsets that were actually evaluated were sampled from those that could be compiled down to 20 qubits. Fig. 12 visualises the subset of datapoints that were sent to H1-1, and Fig. 14 visualises the samples from the four-directional dataset. We see that above 20 actors, there is instead an anti-correlation between number of actors and text depth, though all instances had text depths above that seen in any of the *Valid Comp* instances. We visualise compositional generalisation in terms of the text depth directly in Fig. 17 and Fig. 18 for the samples sent to H1-1, and the four-directional *Test (qujax)* set. The error bars shown represent the 95% confidence interval for each depth, calculated using the exact Clopper-Pearson method, which is sensitive to the number of samples. Since the number of samples per depth is very uneven, we also plot confidence intervals for binned data, targeting around 30 samples per bin. The average per bin is drawn as a line covering the binned region, with a shaded confidence interval. The number of datapoints per bin is noted inside each binned region. Of particular note is that we had few samples for each depth, so the confidence intervals for the mean are very wide, however in the cases where more samples were present, the mean remains high, and above random guessing.

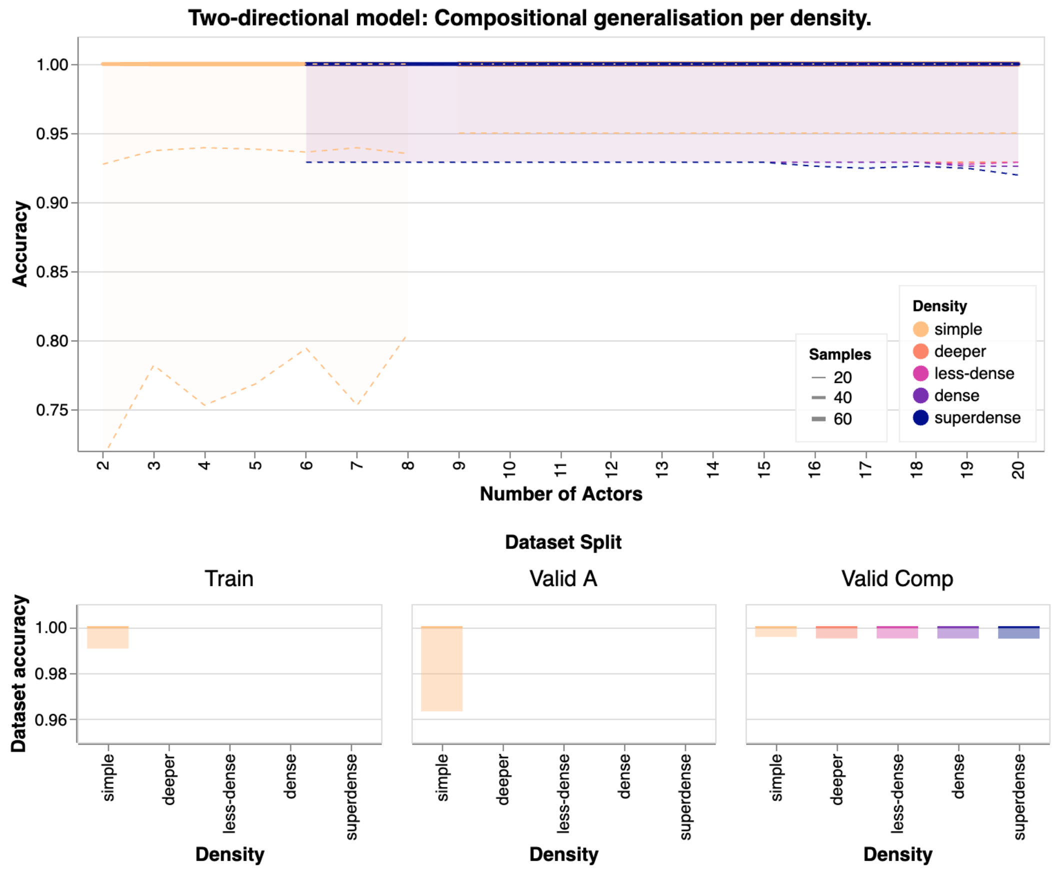


Figure 15: Compositional generalisation of the two-directional datasets per density, for the simulated datapoints only.

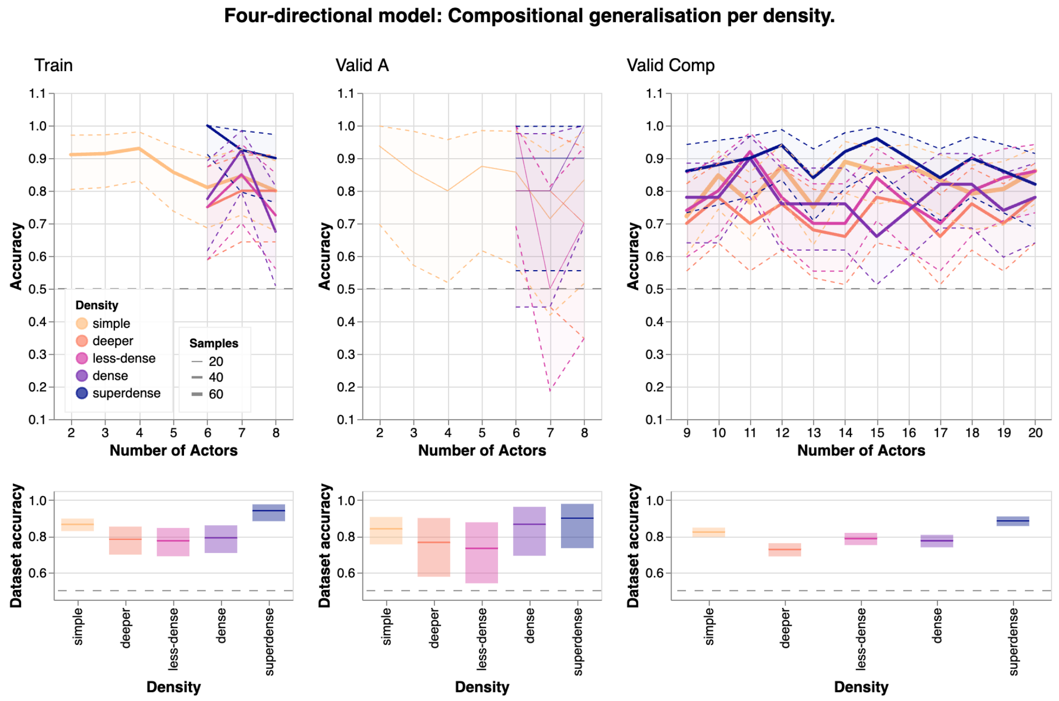


Figure 16: Compositional generalisation of the four-directional datasets per density.

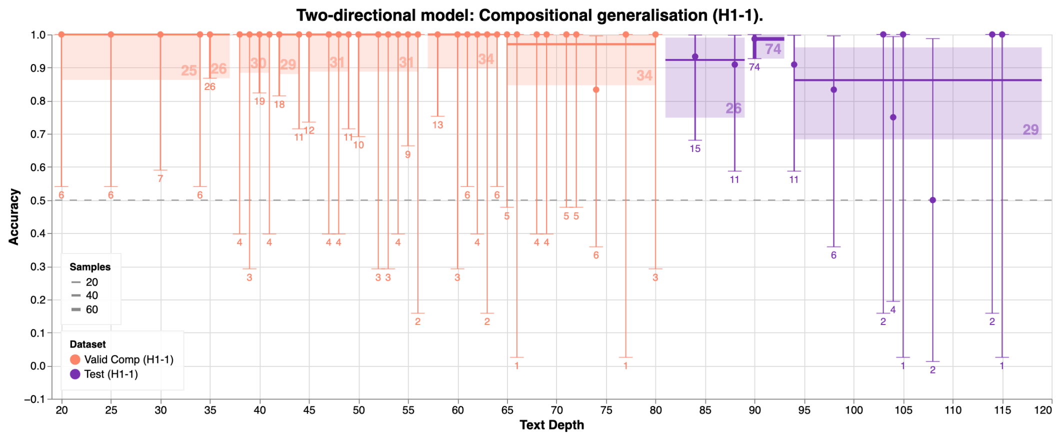


Figure 17: Compositional generalisation for the H1-1 samples per sentences in the text.

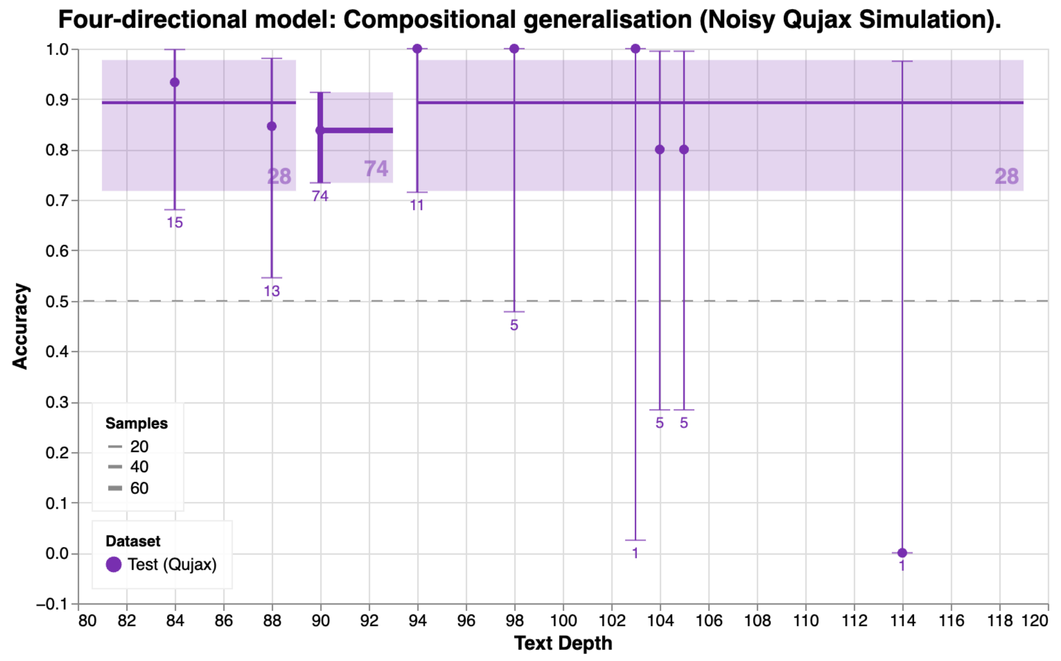


Figure 18: Compositional generalisation for the four-directional *Test (qujax)* samples, per number of sentences in the story.

G Classical baselines

We trained a transformer and an LSTM neural network to explore their generalisation capabilities across the two-directional and four-directional datasets. It is important to note that there is no fair direct comparison between the quantum and classical performances, due to inherent differences in their frameworks.

An implementation choice we made in training the QDisCoCirc model that could not be replicated in the classical baselines is replacing all names with `person` (see App. A.2). Instead, we opted for a uniform representation of all 30 names occurring in the *Train, Valid Comp and Test* datasets. We randomly replaced the names in the *Train* dataset with names from this distribution and used this alternative *Train* dataset for training the classical baselines to make sure the model learns representations for all names occurring. As a result, the two-directional vocabulary comprises approximately 62% names and the four-directional 58%, while the QDisCoCirc model has to learn only the word `person`.

Another crucial difference is the absence of hardcoded semantic rewrites, as opposed to the QDisCoCirc pipeline, see Fig. 8. However, we maintained the same *Train*/*Valid A* and *Valid Comp* splits as shown in Table 4. The two-directional models were trained only on the simple dataset, to mirror the training methodology from App. C.2.

Furthermore, we evaluated GPT-4 [38] and tracked its performance across all densities of both datasets. What follows is a description of the training methodology and a detailed evaluation of the performance of the classical models.

G.1 Transformer

We trained a custom made transformer neural network to explore its generalisation capabilities across the two-directional and four-directional datasets. We used `PyTorch` [31] to track the parameters and the Adam optimizer [32] to train them. The hyperparameters we considered and their final tuned values are summarised in App. G.1 below. We tuned the hyperparameters using `Ax` [33, 34], and aimed for the best *Valid A* accuracy.

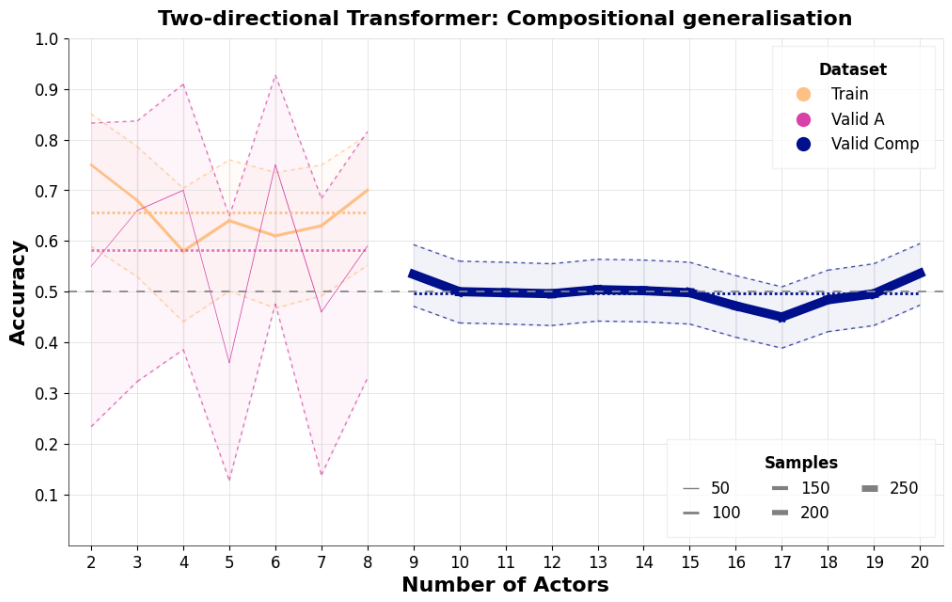
Table 7: Training hyperparameters used and the ranges we considered when tuning the transformer on the two-directional and four-directional datasets.

hyperparameter	**range/choice values**	**final value**	
		two-directional	**four-directional**
learning rate	0.0001 - 0.0003	0.00014	0.00017
batch size	32, 64, 128	32	32
dropout	0.3 - 0.75	0.413	0.565
hidden dimensions	512 - 768	569	565
model dimensions	256, 512, 768	256	512
number of layers	1 - 4	4	3
number of heads	2, 4, 8	2	4
seed	0 - 1000	446	215

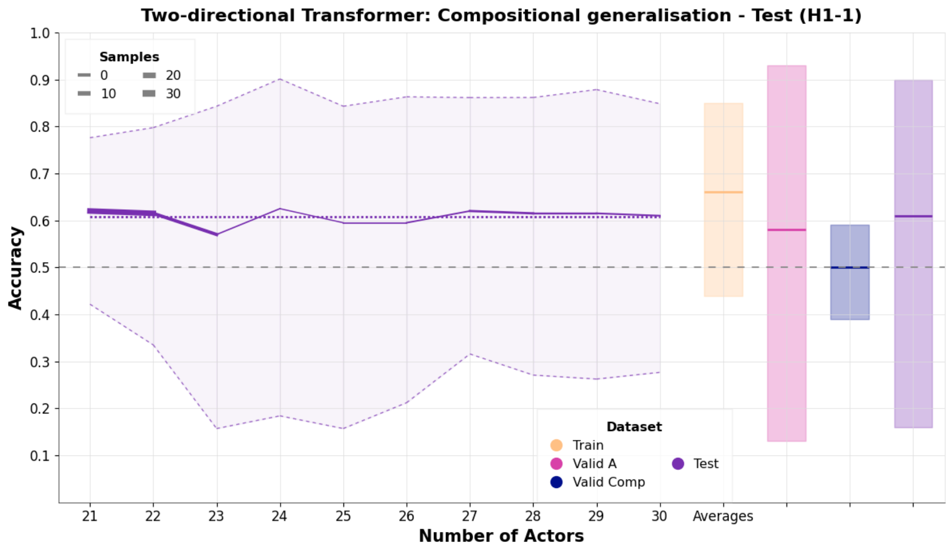
G.1.1 Transformer Architecture

Our implementation is based on the transformer architecture introduced in [39] and adapted to our needs as described below.

The input tokens are processed through an embedding layer, the dimensionality of which varies between the two-directional and four-directional models, as detailed in App. G.1. Positional encoding ensures that the sequential nature of the data is retained. The core of the model is the transformer encoder, consisting of several layers. The model dimensionality, attention heads, hidden layers, and dropout rate differ across the two-directional and four-directional models, as indicated in App. G.1. The final linear classification layer maps the encoder's high-dimensional output to the target classes.



(a) Performance on *Train*, *Valid A*, and *Valid Comp* datasets.



(b) Performance on the *Test (H1-1)* dataset.

Figure 19: Compositional generalisation of the two-directional transformer across different datasets.

G.1.2 Transformer Results: Two-Directional

For all plots, horizontal lines indicate average accuracy, and the dashed grey line shows the baseline accuracy. The shaded areas depict the 95% Clopper-Pearson confidence interval for the mean. Line thickness varies with dataset size (see Table 2). The model was trained only on the Simple dataset, mirroring the training methodology from App. C.2.

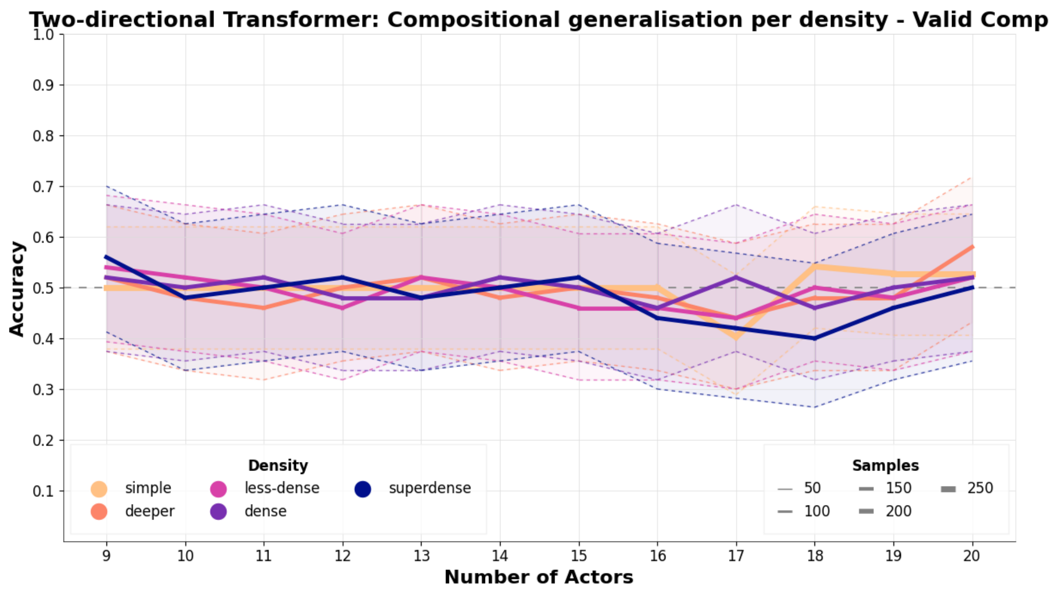


Figure 20: Compositional generalisation of the two-directional transformer, breakdown by density. The plot shows the *Valid Comp* results exclusively, since the *Train* and *Valid A* datasets contain only the Simple density and their results can be found in Fig. 19(a).

The *Train* accuracy of the two-directional transformer model (Fig. 19(a)) fluctuates around 66% across different text widths and never drops below 58%. The confidence intervals drop below 50% for text widths 4, 6 and 7.

The *Valid A* accuracy follows a similar trend (Fig. 19(a)) surpassing the *Train* accuracy on stories with 4 and 6 , but it is on average 8% lower. Partially due to the smaller number of data points in the *Valid A* dataset, the confidence interval drops well below 50%. When evaluated on stories from the *Valid Comp* dataset with a higher number of actors in the stories, the model's performance is about 50% across all text widths (Fig. 19(a)). We remind the reader that 'text width' refers to the number of actors in a story. In our task, this performance means random guessing, suggesting that the model is not able to generalise productively on text widths 9 to 20.

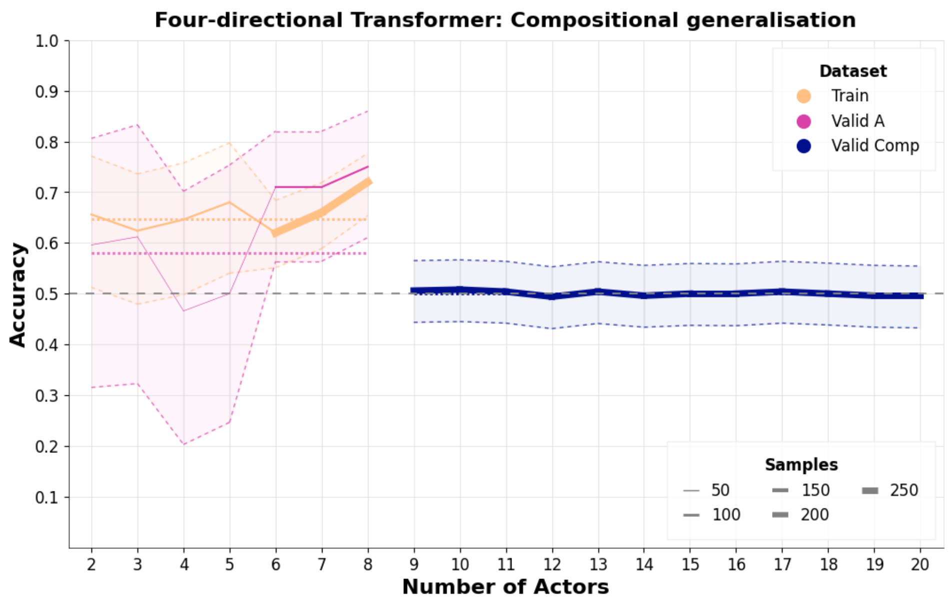
Figure 20 shows the *Valid Comp* accuracy split by the different story densities. It shows that the model's accuracy is consistently random across all story densities for varying text widths and no particular story density outperforms or pulls down the overall accuracy.

When evaluating the transformer on the *Test (H1-1)* dataset (Fig. 19(b)) with a text width up to 30, we observe that performance hovers around 61%, with a slight drop at 23 nouns. The confidence intervals fall significantly below 50%, reaching as low as 15%, partly due to the limited number of data points. In this scenario, the transformer appears to generalize well, performing better than on the *Valid Comp* dataset, and even the *Valid A* dataset (Fig. 19(b)).

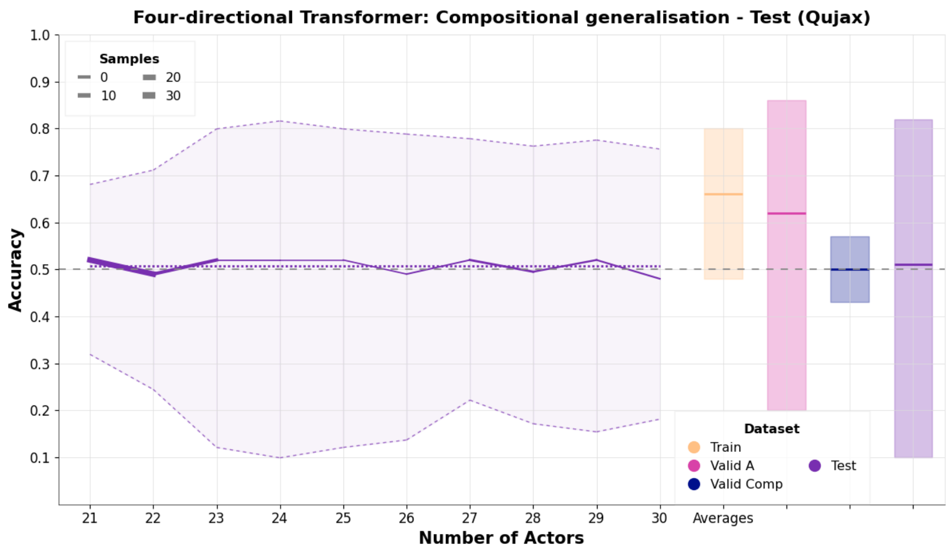
To delve deeper into this performance, we tested the model on the entire 'dense' and 'superdense' *Test* dataset with text width 21 to 30, rather than the selected subset of data points (which were the ones that compiled, with qubit reuse, down to 20 qubits or less due to hardware constraints) for the equivalent QDisCoCirc experiment. This performance was similar to that on the *Valid Comp* dataset, with accuracy hovering around 52%.

G.1.3 Transformer Results: Four-Directional

The *Train* accuracy of the four-directional transformer model (Fig. 21(a)) fluctuates around 66%. The *Valid A* accuracy deviates less from the *Train* accuracy for this model, by only about 4% on average. While the confidence intervals of the training accuracies only drop below 50% for text width 3, the confidence interval for *Valid A* is below 50% for most text widths, with the exception of 6, 7 and 8 nouns, largely due to the small number of datapoints. While there is an increase of the



(a) Performance on *Train*, *Valid A*, and *Valid Comp* datasets.



(b) Performance on the *Test (qujax)* dataset.

Figure 21: Compositional generalisation of the four-directional transformer across different datasets.

Valid A accuracy on 6, 7 and 8 nouns, the four-directional transformer does not generalise to larger text widths from the *Valid Comp* dataset and shows an accuracy of 50% (Fig. 21(a)).

When evaluating the transformer on the *Test (qujax)* dataset (Fig. 21(b)), we see that the performance hovers around 51%, with very slight fluctuations. The confidence intervals dip well below 50%, reaching 10%, due to the small number of datapoints. In the four-directional case, the model is not able to generalise.

Figure 22 shows the *Valid Comp* accuracy split by the different story densities. It shows that the model's accuracy is consistently random across all story densities for varying text widths and no particular story density outperforms or pulls down the overall accuracy.

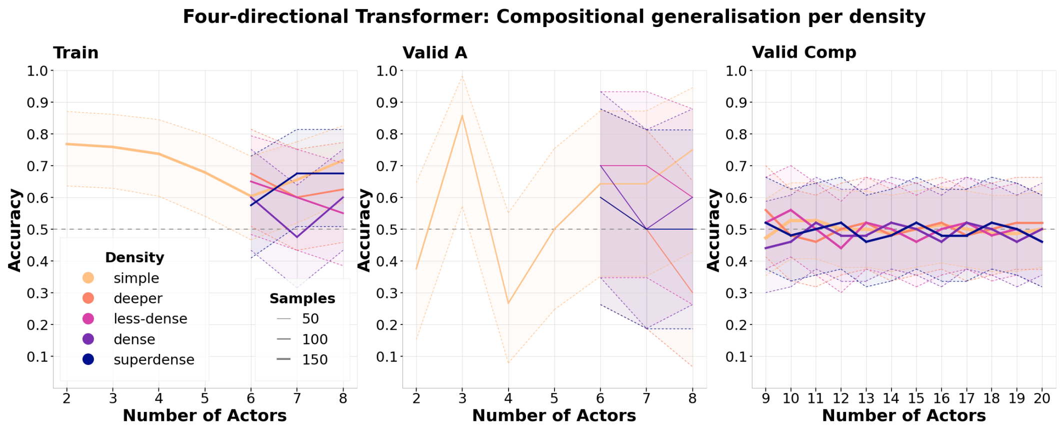


Figure 22: Compositional generalisation of the four-directional transformer, breakdown by density.

G.1.4 Discussion and future work

We compare the trained transformer and QDisCoCirc models. The two-directional QDisCoCirc model achieves 100% accuracy across the *Train*, *Valid A*, and *Valid Comp* datasets, while the four-directional version maintains accuracy above 80%. In contrast, both the two- and four-directional transformers consistently score below 80%, with performance on the *Valid Comp* dataset fluctuating around random guessing levels.

The sharp drop in accuracy from 8 to 9 nouns -from *Valid A* to *Valid Comp*- indicates that the transformers were unable to generalize to texts with different distributions of text widths and text depths. In contrast, the QDisCoCirc model consistently outperforms the transformers, excelling at compositional generalization. This holds true for the four-directional *Test (qujax)* dataset and even for the two-directional equivalent, when we consider the whole of the *Test* set and not just the restricted set chosen due to hardware constraints, as further detailed in the analysis of the results.

When comparing the two models, it is essential to acknowledge the differences in the training methodology of the two models mentioned at the beginning of this section. One reason for the model's poor performance might be that the characters' names comprise a large portion of the vocabulary. This causes the model to focus too much on specific character names and their behaviors, rather than understanding the underlying patterns of their interactions. In addition, our datasets are rather small compared to typical datasets used for training transformers. Thus, the limited *Train* data (Table 4) is another main reason for the transformer's poor performance.

For future work, we propose a more sophisticated and equitable experimental setup for the transformer models, while aligning closely with the QDisCoCirc training setup. This will involve modifications to the dataset: each data point corresponding to a story will be used to generate multiple samples. Each of these will maintain the original story structure but vary the names, aiming for a uniform representation across the augmented dataset. To compute the label for each original data point, all of the derived samples will be evaluated, and the final label will be calculated as the average across sub-stories. This intends to emulate the replacement of specific names with `person` in the QDisCoCirc setup.

Additionally, we could fix the pair of characters involved in the question (e.g., by fixing the question to always be 'Does Bob go in the same direction as Alice?'), in all stories and their sub-stories. This second modification will allow the model to know which characters are relevant from the start of the text, simplifying the task and potentially allowing the model to learn a more compositional representation.

G.2 LSTM

We trained an LSTM neural network to explore its generalisation capabilities across the two-directional and four-directional datasets. We used `Tensorflow` [40] to track the parameters and the Adam optimizer [32] to train them.

The hyperparameters we considered and their final tuned values are summarised in Table 8 below. We tuned for the best *Valid A* accuracy using `Ax` [33, 34].

Table 8: Training hyperparameters used and the ranges we considered when tuning the LSTM on the two-directional and four-directional datasets.

hyperparameter	**range/choice values**	**final value**	
		two-directional	**four-directional**
learning rate	0.00001 - 0.01	0.001	0.00008
batch size	32, 64, 128	128	64
dropout	0.35 - 0.65	0.39	0.38
l1 regularisation	1e-6 - 1e-4	5e-5	8e-5
l2 regularisation	1e-6 - 1e-4	3e-5	8e-5
hidden layers	20 - 150	74	134

G.2.1 LSTM Results: Two-Directional

As with the transformer, the LSTM model was trained only on the Simple dataset, mirroring the training methodology from App. C.2. Horizontal lines indicate average accuracy, and the dashed grey line shows the baseline accuracy. The shaded areas depict the 95% Clopper-Pearson confidence interval for the mean. Line thickness varies with dataset size (see Table 2).

The *Train* accuracy of the two-directional LSTM model (Fig. 23(a)) fluctuates around 68% across different text widths. The average validation accuracy *Valid A* is lower at 54%. The *Valid Comp* accuracy fluctuates around 50% for all story widths, which means random guessing for the presented task. It should be noted again that the particularly wide confidence intervals, especially for the *Valid A* curve, are partly due to the smaller number of data points available.

When evaluating the model on the *Test (H1-1)* dataset (Fig. 23(b)), we see that the performance hovers around 51%, with prominent fluctuations across text widths. The confidence intervals dip well below 50%, reaching 10%, due to the small number of datapoints. Considering that the accuracy is around random guessing, we conclude that the model is not able to generalise.

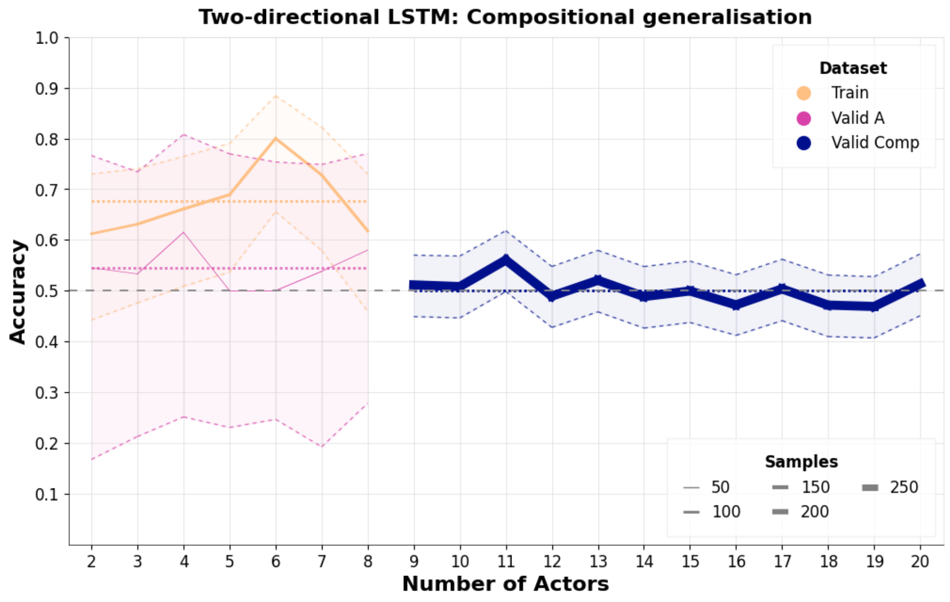
Fig. 24 shows the *Valid Comp* accuracy split by the different story densities. It shows that the model's accuracy is consistently random across all story densities for varying text widths and no particular story density outperforms or drags down the overall accuracy.

G.2.2 LSTM Results: Four-Directional

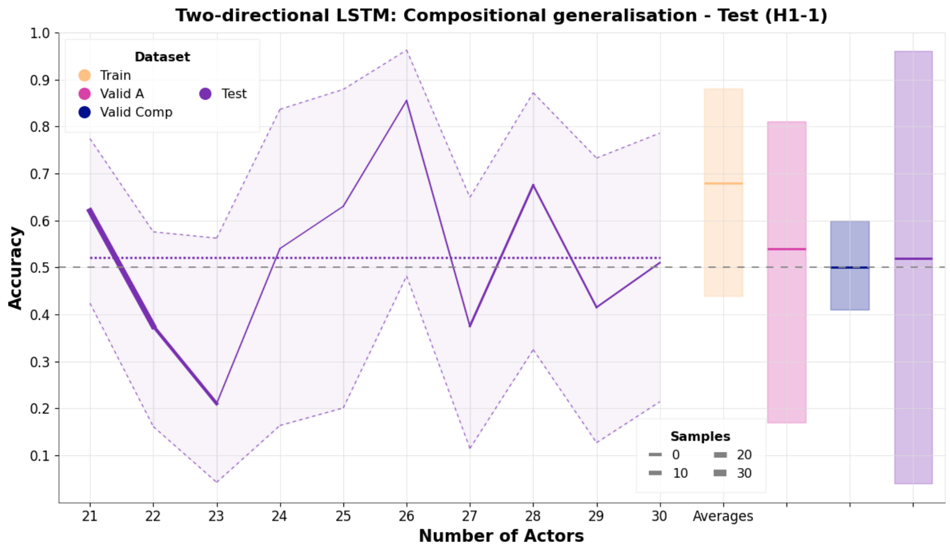
The *Train* accuracy of the four-directional LSTM model (Fig. 25(a)) fluctuates around 73% and is relatively stable across text widths. The *Valid A* accuracy is significantly lower than the *Train* accuracy, at 62% on average, suggesting substantial overfitting.

When evaluating the model on the *Test (qujax)* dataset (Fig. 25(b)), we see that the performance hovers around 54%, again with prominent fluctuations present. The confidence intervals dip well below 50%, reaching 7%, due to the small number of datapoints. Considering that the accuracy on the *Valid Comp* and *Test (qujax)* datasets is around random guessing, we conclude that the model is not able to generalise.

Just like in the two-directional case, Fig. 26 shows the *Valid Comp* accuracy split by the different story densities. It shows that the model's accuracy is consistently random across all story densities for varying text widths and no particular story density outperforms or drags down the overall accuracy.



(a) Performance on *Train*, *Valid A*, and *Valid Comp* datasets.



(b) Performance on the *Test (H1-1)* dataset.

Figure 23: Compositional generalisation of the two-directional LSTM across different datasets.

G.2.3 Discussion

In all cases the LSTM was trained on, the difference between the *Train* and *Valid A* accuracies was at least 10%, indicating overfitting. The LSTM consistently performs worse than QDisCoCirc and is unable to generalise to larger text width instances on either the *Valid Comp* or *Test* datasets. The LSTMs' bad generalisation performance mirrors this architecture's broader performance problems, as seen in tasks like the bAbI datasets [41].

Despite the inherent limitations of the LSTM, data augmentation as suggested in the transformer discussion App. G.1.4 might enhance the model's performance.

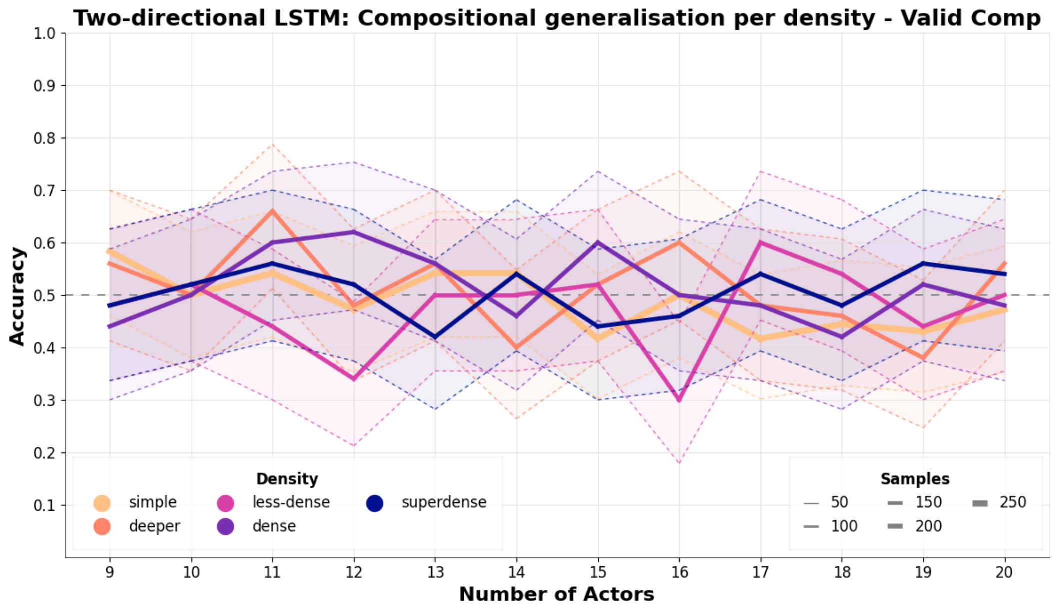
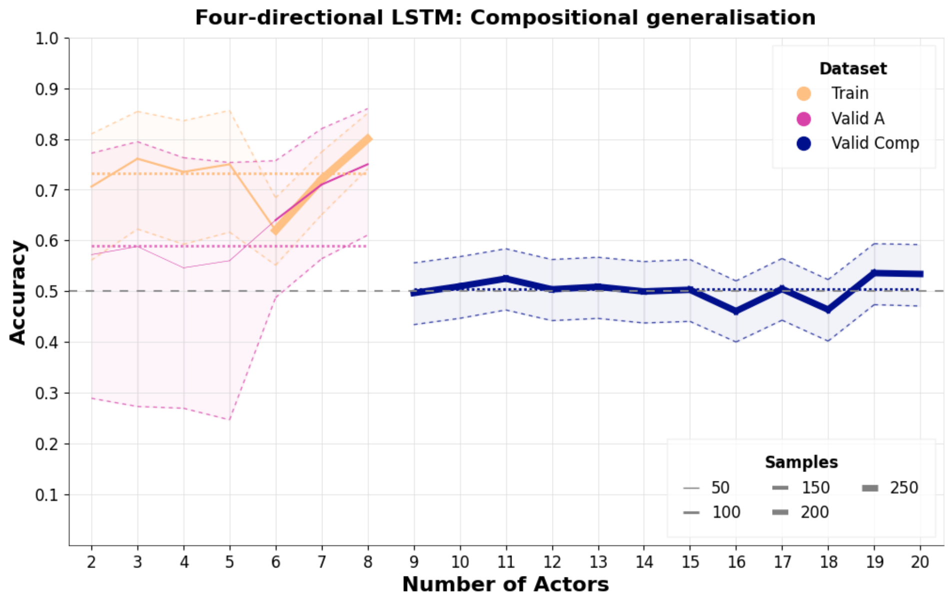
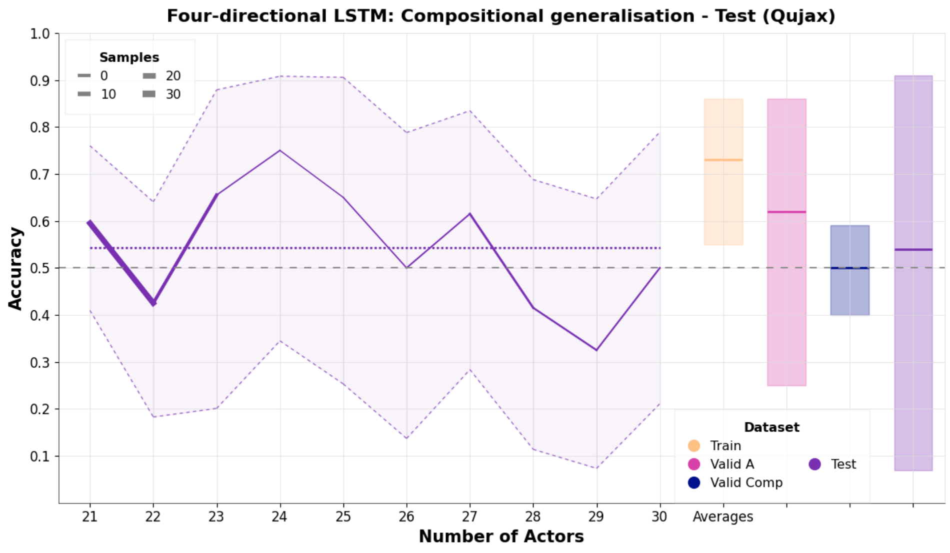


Figure 24: Compositional generalisation of the two-directional LSTM, breakdown by density.



(a) Performance on *Train*, *Valid A*, and *Valid Comp* datasets.



(b) Performance on the *Test (qujax)* dataset.

Figure 25: Compositional generalisation of the four-directional LSTM across different datasets.

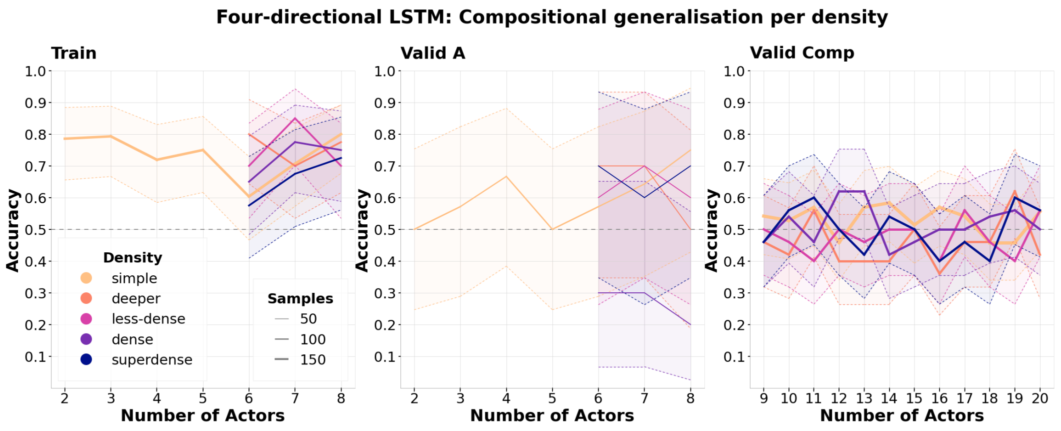


Figure 26: Compositional generalisation of the four-directional LSTM, breakdown by density.

G.3 GPT-4

Finally, we assessed GPT-4 using both the two-directional and four-directional datasets, on 25/04/2024. We tested GPT-4 on the *Train*, *Valid A* and *Valid Comp* datasets combined, consisting of 3756 entries in the two-directional and 4368 entries in the four-directional dataset. The stories were submitted in batches of 50. We compared GPT-4's responses to the correct labels and analyzed the outcomes, as illustrated in Figs. 27 and 29. Moreover, we conducted a detailed analysis of performance based on story density, as shown in Figs. 28 and 30.

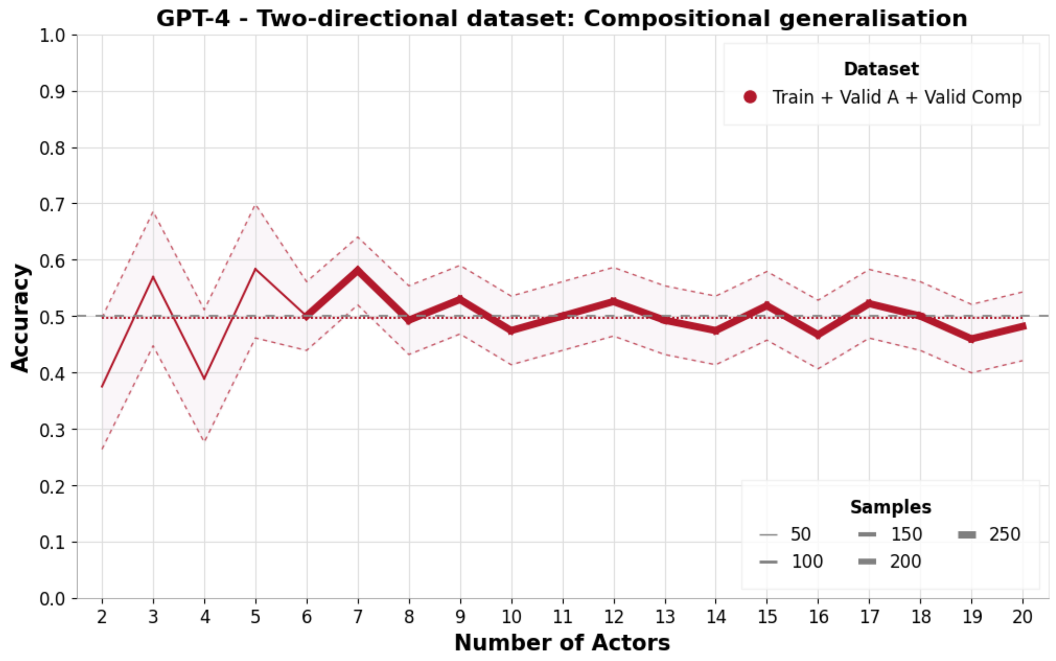


Figure 27: GPT-4 test accuracy for the two-directional dataset.

We evaluated GPT-4's performance on the two-directional and four-directional datasets across various story densities and widths.

The model's performance on both two- and four-directional datasets is on average close, or equal to 50%, across all text widths, meaning random guessing in the given task (Figs. 27 and 29). We restricted our evaluation on stories of up to 20 nouns, since the performance is consistently random, and we did not expect it to improve on larger text widths. A detailed breakdown of the performance by story density reveals a similar behaviour (Figs. 28 and 30). The predictions for the four-directional less-dense dataset were imbalanced and GPT-4 predicted negative labels for all stories. We did not observe such imbalances for any of the other two- and four-directional story density datasets.

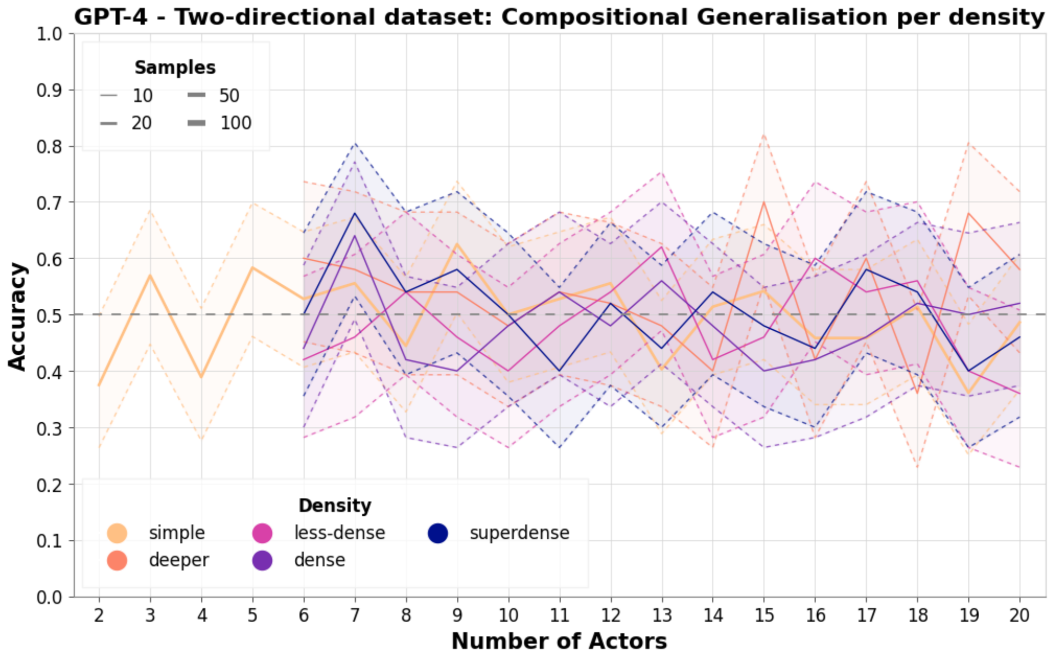


Figure 28: GPT-4 test accuracy for the two-directional dataset - breakdown by density.

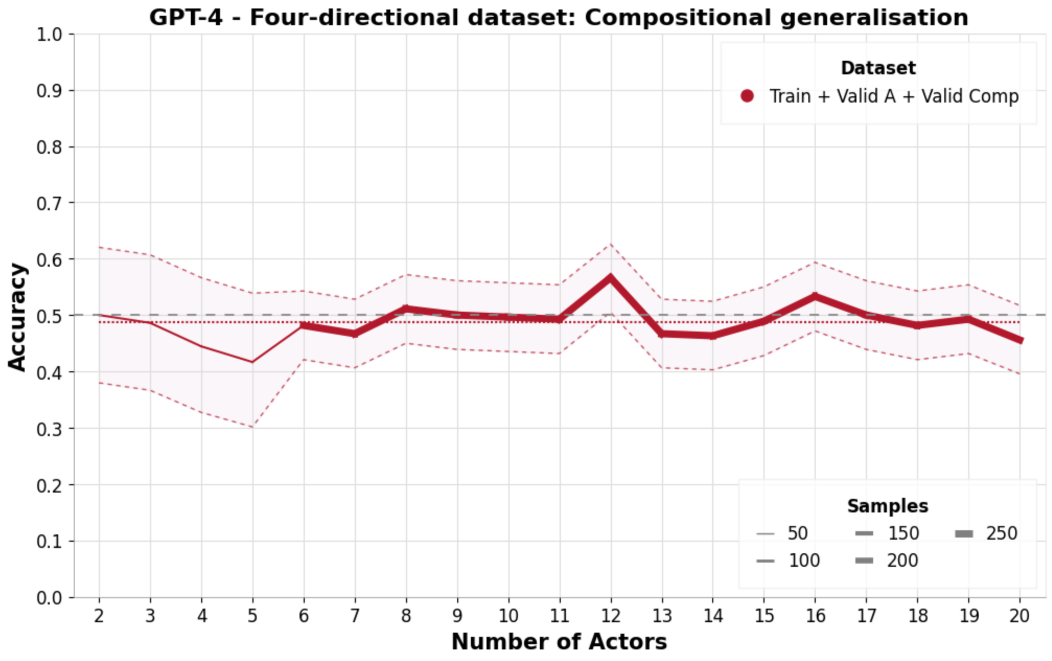


Figure 29: GPT-4 test accuracy for the four-directional dataset.

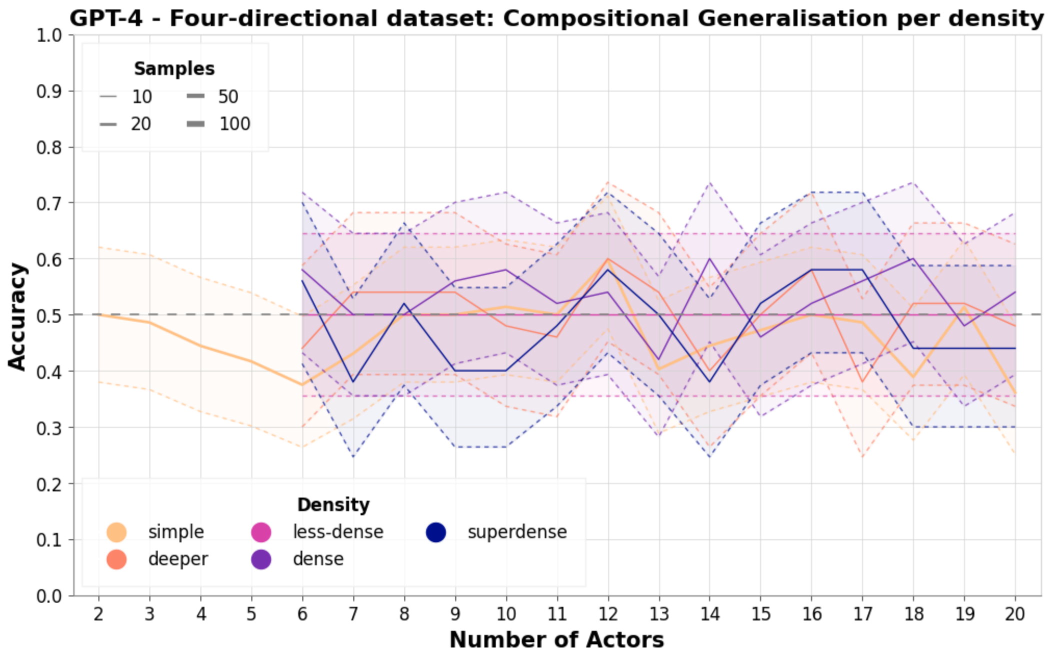


Figure 30: GPT-4 test accuracy for the four-directional dataset - breakdown by density.

H Compositional interpretability

The built-in compositional structure of QDisCoCirc allows us to interpret its behaviour. We begin by studying how each component in the quantum representation of the texts behave. Then, we use its inherent compositional structure to piece together how the model performs on our datasets. We confirm our analysis by examining how well the axioms we expect to hold are respected. In App. H.1, we observe that the two-directional model performs well due to behaving similarly to an ideal model we construct by hand in App. H. In App. H.2, we explain how the four-directional model does not perfectly correspond to this ideal model, but rather approximates it, performing well nonetheless. For it to correspond to an ideal model, the four-directional model would require two qubits per wire instead of our choice of one qubit per wire (made due to hardware constraints).

Throughout this section, we make use of the following representations:

- Single-qubit states are visualised on a Bloch sphere in the standard way, where pure states lie on the surface of the sphere, and mixed states lie inside the sphere.
- Following [42], two-qubit states $\rho_{1,2}$ are visualised as coloured ellipsoid surfaces inside a pair of Bloch spheres in terms of partial projections. In this representation, each sphere corresponds to one of the qubits. The first Bloch sphere displays a surface defined by $\rho_{1,\psi} = (\mathbb{I} \otimes \langle\psi|)\rho_{1,2}(\mathbb{I} \otimes |\psi\rangle)$ for all pure states $|\psi\rangle$ that the second qubit can be in. Each state $\rho_{1,\psi}$ is coloured as per the reference colour of $|\psi\rangle$, shown in Fig. 31a. The second Bloch sphere similarly displays a surface describing the state $\rho_{\psi,2}$ of the second qubit given that the first qubit is in state $|\psi\rangle$.
- We visualise single-qubit gates as rotations on the Bloch sphere, plotting the trajectory of reference points $|\phi\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle\}$, under the action of the rotation U. The final state $U|\phi\rangle$ is coloured according to the colour of the initial state $|\phi\rangle$, as given in Fig. 31a.

H.1 Two-directional dataset

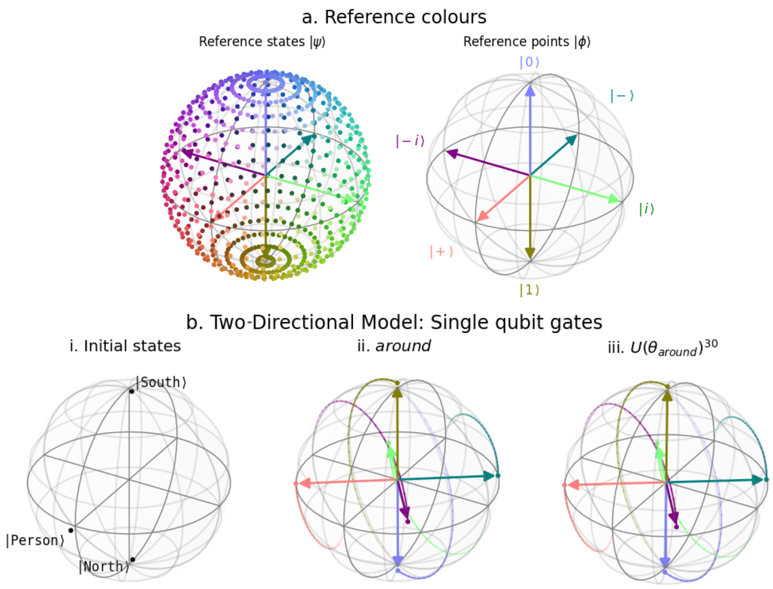


Figure 31: a. Colour mapping used for paired Bloch sphere pure states $|\psi\rangle$. The reference points $|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle$ are highlighted as vectors. b. Single qubit gate analysis for the two directions model: i. Single qubit initialisations, viewed as states. The two directions are aligned with the computational basis. ii. The effect of the single qubit rotation `turns around` on the reference points. iii. Applying `turns around` 30 times resembles applying it just once.

Initial States Since these only ever occur at the start of a text, they can be represented as states (rather than rotations). Note that the state labelled $|\texttt{North}\rangle$ represents the state obtained from the entire sentence `Person walks north`, and similarly for $|\texttt{South}\rangle$, as this is how it is always encountered in the text. These are depicted in Fig. 31b.i.

Single-qubit gates In Fig. 31b.ii, we plot the single-qubit rotation `turns around`. The gate is approximately a π rotation about the x-axis (up to a small phase). Since the directions $|\texttt{North}\rangle$ and $|\texttt{South}\rangle$ are approximately on the poles of the sphere, the expected axioms also hold approximately (see Fig. 32). Since the rotation is only *approximately* π, there is a small offset that can accumulate if `turns around` is applied successively - Fig. 31b.iii. shows the effect of applying `turns around` thirty times in a row, demonstrating that an extra π rotation has been accumulated: thirty applications resemble a single application, whilst we would expect that turning around an even number of times should be equivalent to not turning around at all. A perfect model (as described in Fig. 42) would not exhibit such behaviour, however the model has learnt to accommodate this error, as will be shown in the next section.

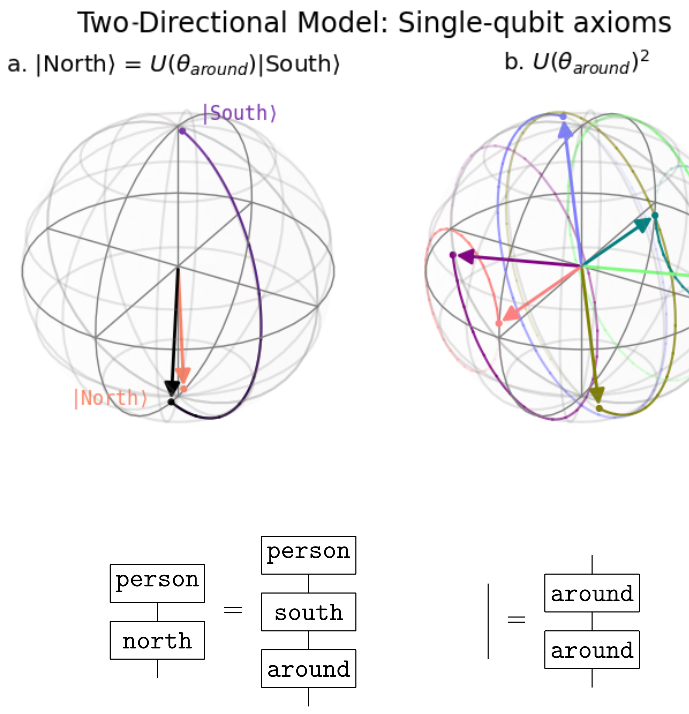

Two-Directional Model: Single-qubit axioms
a. |North⟩ = U(θ_around)|South⟩
b. U(θ_around)²
|South⟩
|North⟩
person
north
person
south
around
around
around

Figure 32: Single qubit axioms for the two-directional dataset. We verify that the two-directional model satisfies them approximately: a. The final states (highlighted as vectors) are very close to one another. Note that in this case the colours are to be taken as a visual aid only. b. We see that the reference points are almost mapped back to their original positions. The slight discrepancy is what leads to the peirodicity shown in Fig. 31b.iii.

Two-qubit gates As our ansatz contains some built-in semantic rewrites, there is only one two-qubit gate: `follows`. We consider its action on a set of states defined by the possible directions the two people involved might be initially facing. Since $|\texttt{North}\rangle$ and $|\texttt{South}\rangle$ are approximately orthogonal, this acts as a basis, and so investigating the action of `follows` on each of these initial states fully characterises the gate. Depicted in Fig. 33, the states considered encode texts of the form:

$$\texttt{Person}_1\ \texttt{walks}\ x.\quad \texttt{Person}_2\ \texttt{walks}\ y.\quad \texttt{Person}_1\ \texttt{follows}\ \texttt{Person}_2. \tag{1}$$

for $x, y \in \{\texttt{North}, \texttt{South}\}$. The gate behaves largely as a joint projection of the qubits onto the $|\texttt{North}\rangle$ - $|\texttt{South}\rangle$ axis, controlled by the state of qubit 2. The joint aspect of this projection entangles the qubits in such a way that some certainty as to the original directions faced is lost, however since the state of the $\texttt{Person}_1$ remains correlated with the new location of $\texttt{Person}_2$, and largely independent of the original direction $\texttt{Person}_1$ was facing, the `follows` gate seems to have correctly captured the notion of `following`.

We also visually confirm that the axiom

$$\texttt{Person}_1\ \texttt{follows}\ \texttt{Person}_2\texttt{.}\quad \texttt{Person}_1\ \texttt{follows}\ \texttt{Person}_2\texttt{.} = \texttt{Person}_1\ \texttt{follows}\ \texttt{Person}_2\texttt{.} \tag{2}$$

approximately holds, by comparing the visualisations for both gates. Applying following a second time induces a slight phase shift, but the state remains at the relevant pole with high probability.

We also note that the projective aspect of `follows` can correct for some accumulated errors from `turns around`. Although we would expect even powers of `turns around` to be approximately the identity, we find that $U^{30}(\theta_{\texttt{around}}) \approx U(\theta_{\texttt{around}})$, as demonstrated in Fig. 31b.iii. Provided `follows` is interspersed between `turns around` sufficiently frequently, the states will be projected back towards the pole they are closest to, thus clearing the error accumulated so far. Sufficiently frequently in this case means at most 15 `turns around` gates have been applied in succession, as from this point the probability of projecting to the correct state will dip below 50%.

Questions Both question effects are maximally entangled, approximately corresponding to two states from the Bell basis. Written in terms of our initial states, we have

$$|\texttt{same dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{North}\rangle \otimes |\texttt{North}\rangle + |\texttt{South}\rangle \otimes |\texttt{South}\rangle)$$

$$|\texttt{opp dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{South}\rangle \otimes |\texttt{North}\rangle + |\texttt{North}\rangle \otimes |\texttt{South}\rangle)$$

which intuitively captures the meaning of these questions (see Fig. 40 for visualisations).

H.2 Four-directional dataset

Initial states For four directions, we have two additional inital states, $|\texttt{East}\rangle$ and $|\texttt{West}\rangle$, shown in Fig. 34a. Like the two-directional model, this model places $|\texttt{North}\rangle$ and $|\texttt{South}\rangle$ at approximately orthogonal locations. $|\texttt{East}\rangle$ and $|\texttt{West}\rangle$ are similarly orthogonal, however these two 'bases' are not maximally distinct from each other: $|\texttt{North}\rangle$ is somewhat close to $|\texttt{West}\rangle$, and $|\texttt{South}\rangle$ and $|\texttt{East}\rangle$ are similar.

Single-qubit gates Though there are three single qubit gates in the text, the semantic rewrties built into the ansatz express each of these in terms of `turns left` only. `turns left` is approximately a $\pi/2$ rotation about the x-axis as shown in in Fig. 34(b), so we can derive that `turns around` is approximately a π rotation, and `turns right` a $3\pi/2$ rotation about the same axis. The derived turns are depicted in Fig. 36.

The single-qubit axioms hold less well in the four directions model, due to the fact that the directions are clustered into pairs. Fig. 34(c) shows that `turns left` has the expected period of (approximately) 4. The axioms concerning `turns around` mostly hold, but not the axiom establishing the quarter-turns between the initial directions, Fig. 35. The favouring of `turns around` can be explained by its featuring in the decomposition of `goes in the opposite direction`, as this will bias the dataset towards the occurrence of `turns around` compared to `turns left` or `turns right`. As such, the optimisation will tend to favour parameters where `turns around` interacts correctly since this will have a greater impact on the accuracy of the model.

Two-qubit gates We have only one two qubit gate to consider: `follows`. We additionally consider initial states including the two new directions. Like the two directions model, the four directions model projects the state of $\texttt{Person}_1$ onto a state matching $\texttt{Person}_2$ independent of the initial state of $\texttt{Person}_1$, this time with significantly less entanglement. Since such a projection can only occur along a single axis, we see an even clearer pairing of the directions, such that the model is effectively collapsing into a two-directions model over states we shall call `North-West` ($|\texttt{NW}\rangle$) and `South-East` ($|\texttt{SE}\rangle$).

Fig. 37a. visualises the following gate on a subset of the 16 initial states. We visually confirm the approximate idempotence of following in Fig. 37b. Finally, we exhibit an axiom the model fails to adequately capture in Fig. 39:

$$\texttt{P}_2\ \texttt{turns left.P}_1\ \texttt{follows P}_2\texttt{.} = \texttt{P}_1\ \texttt{follows P}_2\texttt{.P}_1\ \texttt{turns left.P}_2\ \texttt{turns left.} \quad (3)$$

This axiom captures the idea that turns can be 'copied through' `follows` when they act on the person being followed, however the model is unable to handle the quarter turn correctly.

Questions The question states are again very similar to the two directions model. We can express them in terms of the $\{|\texttt{NW}\rangle, |\texttt{SE}\rangle\}$ basis derived from the two-qubit `follows` projections (see App. H for visualisation in Fig. 40):

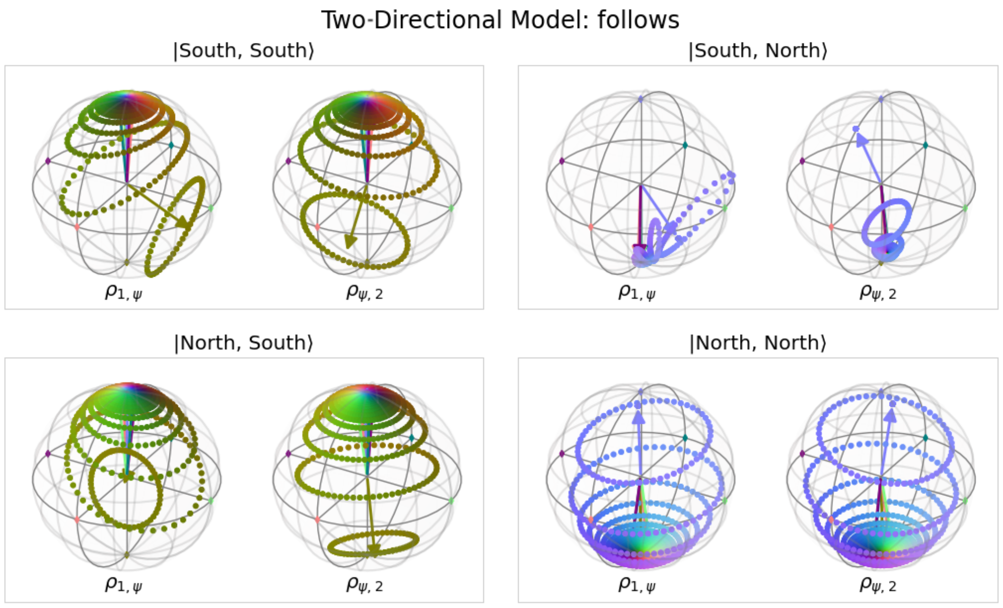
$$|\texttt{same dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{NW}\rangle \otimes |\texttt{NW}\rangle - |\texttt{SE}\rangle \otimes |\texttt{SE}\rangle)$$
$$|\texttt{opp dir}\rangle \approx \frac{1}{\sqrt{2}}(|\texttt{NW}\rangle \otimes |\texttt{SE}\rangle - e^{-i\pi/8}\,|\texttt{SE}\rangle \otimes |\texttt{NW}\rangle)$$

Once again, this shows the underlying two-directional nature of the model, in that it tests for correlation or anti-correlation along the $\{|\texttt{NW}\rangle, |\texttt{SE}\rangle\}$ basis.

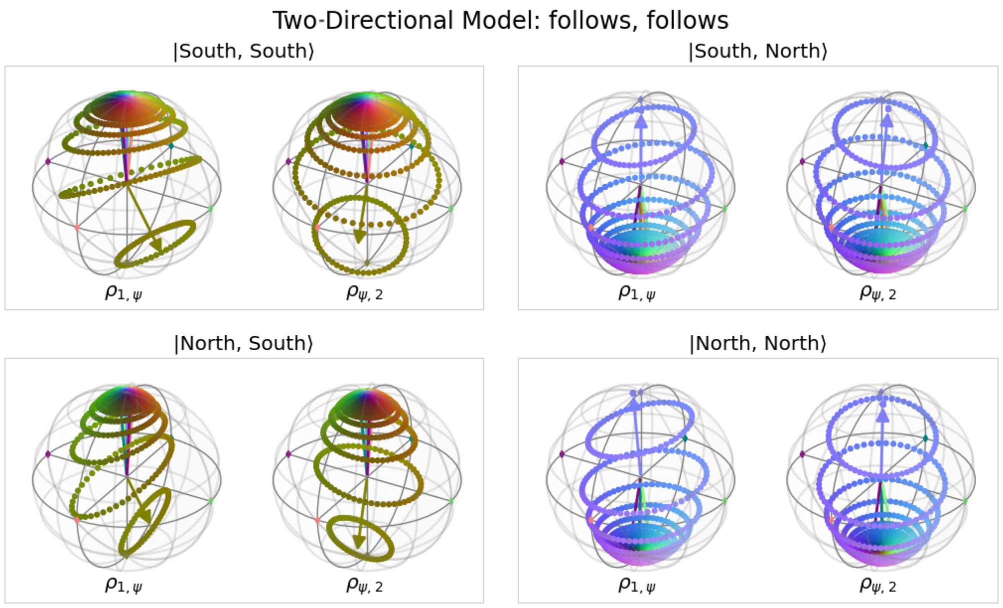
Discussion The analysis above helps us to identify and explain biases that the model exhibits. The bias can be seen to arise from a bias in the dataset - there are very few instances where the final directions faced by the model are at right angles to each other, so the model can afford to ignore these instances at relatively little cost. We can hence identify the instances that are 'difficult' for the model as those that depend on the correct resolution of a quarter turn at some point in the text. As these are a relative minority across the dataset, the model appears to perform better than it should.

It is in fact expected that the model should struggle to perform well, as it is effectively trying to encode 2 bits of information (which of the four directions each actor is currently facing) into a single qubit: indeed the equivalent deterministic model for the four directional dataset (described in Fig. 43) requires two qubits.

The fact that we were able to visualise the model in a modular way, and thus understand *why* it appeared to work, allowed us to identify a bias in our dataset. This was possible in part due to the relative simplicity of the model, as well as its compositional nature, which allowed us to consider only one and two qubit components of the model in isolation, and from this infer its action on arbitrary texts. In the two-directions case, this broadly supported the empirical evidence that the model had correctly learnt to generalise, whilst also exposing a failure case to be wary of. In contrast, for four directions we were able to identify that the model had *not* learnt to generalise in the way we had hoped, despite the otherwise promising empirical evidence.



(a)



(b)

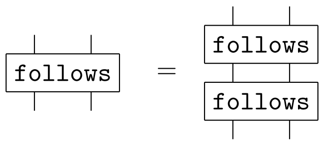


Figure 33: Visualising the two-qubit `follows` gate for the two-directional model to confirm axiom 2. The label $|x, y\rangle$ identifies the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. The expected joint final state is $|y, y\rangle$. Note that we mark the reference points with a coloured dot on the paired Bloch spheres as a visual aid; these are not to be considered part of the plotted ellipsoids. a. Single application of the `follows` gate. b. `follows` applied twice.

Figure 33 *(previous page)*: Reading the plot: a. Consider the paired Bloch sphere for $|\texttt{South, South}\rangle$ for a single application of `follows`. The left-most sphere shows that $\rho_{1,\psi} \approx |0\rangle \approx |\texttt{South}\rangle$ for $|\psi\rangle \not\approx |1\rangle \approx |\texttt{North}\rangle$, and vice-versa for $\rho_{\psi,2}$. This suggests that the pair of qubits is 'mostly in the $|00\rangle \approx |\texttt{South, South}\rangle$ state'. Although somewhat mixed, many of the points are close to the surface of the sphere, suggesting that the qubits are entangled (though not maximally so). This aligns with the interpretation that `follows` acts as an approximate projection of the first qubit onto the state of the second (as measured in the $\{|\texttt{North}\rangle, |\texttt{South}\rangle\}$ basis), with an additional entangling component that allows for $\rho_{1,\texttt{North}} \approx |\texttt{North}\rangle$ (whereas we would expect a perfect projection to give $\rho'_{1,\texttt{North}} \approx |\texttt{South}\rangle$). The paired state for $|\texttt{North, South}\rangle$ is very similar, whilst $|\texttt{North, North}\rangle$ is instead concentrated towards the $|11\rangle \approx |\texttt{North, North}\rangle$ state. For $|\texttt{South, North}\rangle$, we can see the state as being even more concentrated towards $|\texttt{North, North}\rangle$, to the point where the state is almost separable. b. The paired Bloch spheres describing the application of `follows` twice are visually quite similar to those describing `follows` once, however the colours are rotated slightly along the vertical axis, representing an induced phase shift. In the $|\texttt{South, North}\rangle$ case, there is a more significant difference, as the second application of `follows` results in a more entangled state, though in both cases the final state remains approximately $|\texttt{North, North}\rangle$. We thus conclude that axiom 2 holds at least approximately.

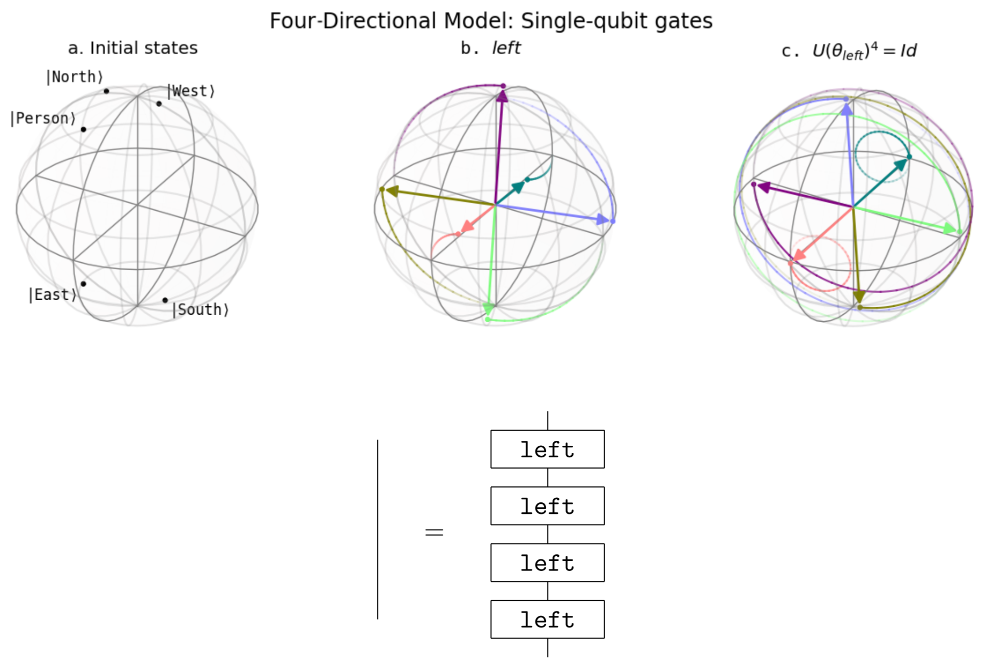


Figure 34: Visualising single qubit gates for the four-directional model: a. Single qubit initialisations. b. Single qubit rotation `turns left` - the other rotations are defined in terms of `turns left`, see App. H for an explicit visualisation. c. `turns left` four times is (approximately) the identity.

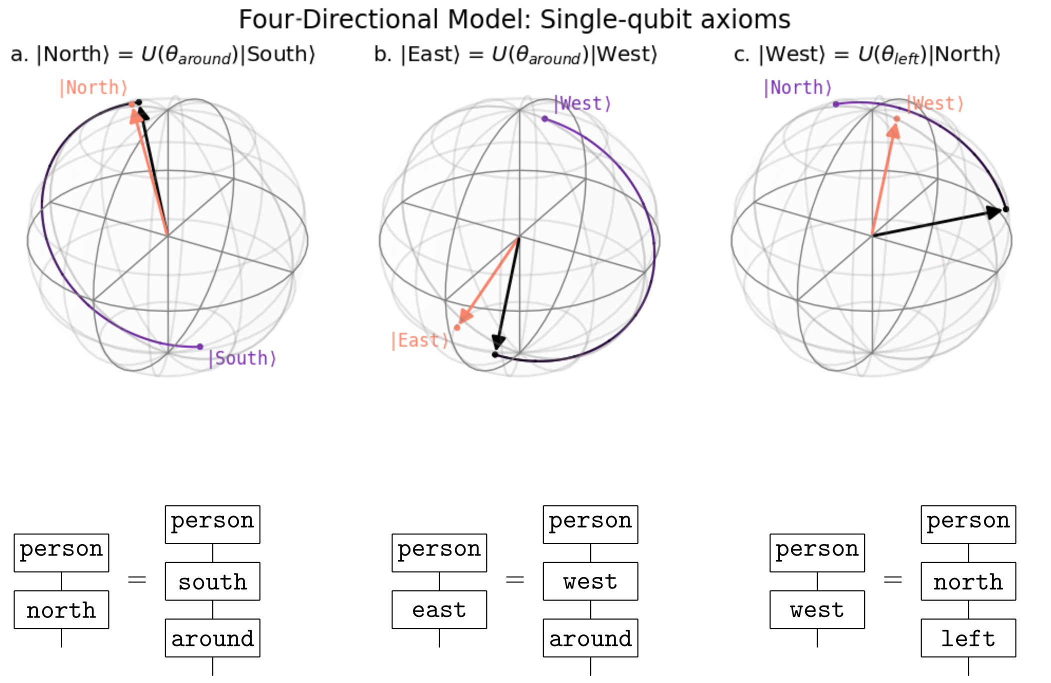

Four-Directional Model: Single-qubit axioms
a. |North⟩ = U(θ_around)|South⟩
b. |East⟩ = U(θ_around)|West⟩
c. |West⟩ = U(θ_left)|North⟩
|North⟩
|South⟩
|West⟩
|East⟩
|North⟩
|West⟩
person
north
=
person
south
around
person
east
=
person
west
around
person
west
=
person
north
left

Figure 35: Single qubit axioms for the four-directional dataset. a,b. the axioms relating |North⟩ to |South⟩, and |East⟩ to |West⟩ hold approximately. c. The model fails to satisfy the final axiom relating |North⟩ to |west⟩. Note that the initial states are coloured to as a visual aid only.

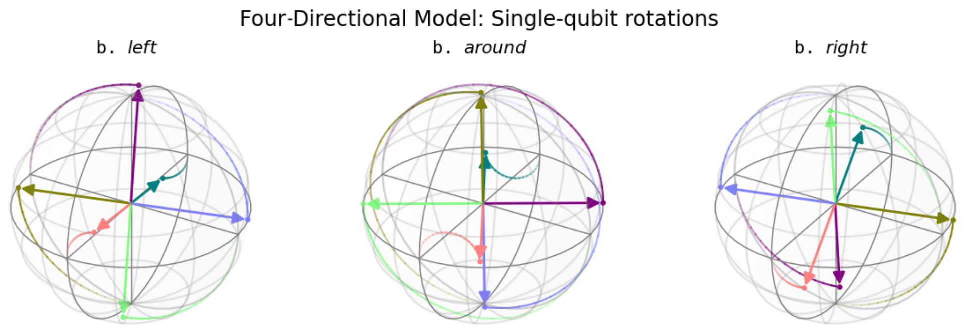
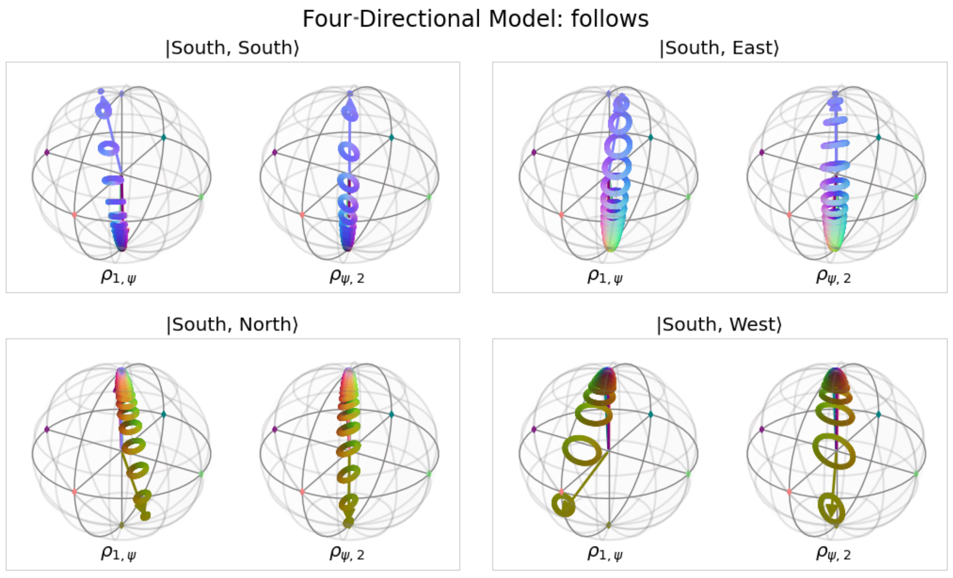
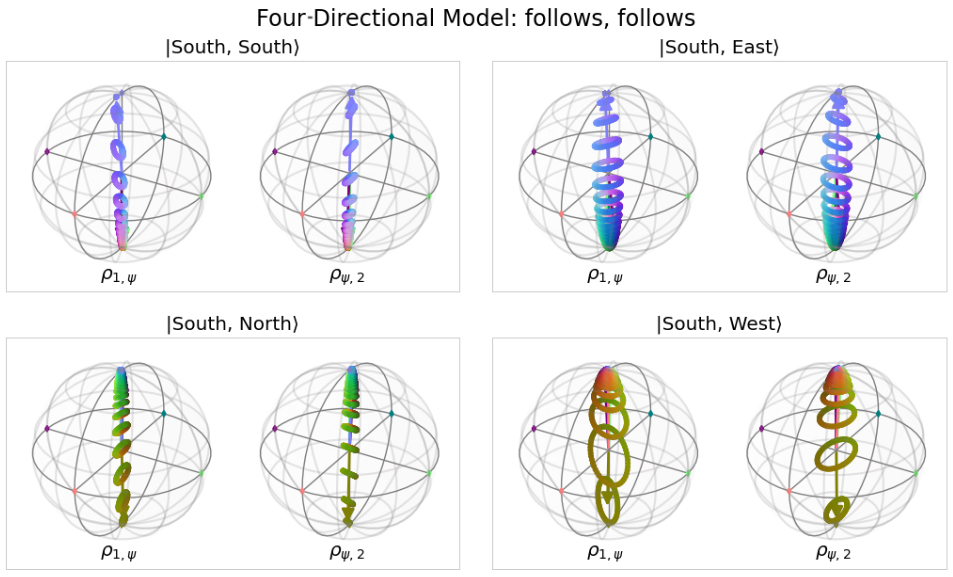


Figure 36: Four-directional single qubit rotations. `around` and `right` are defined in terms of `left`.



(a)



(b)

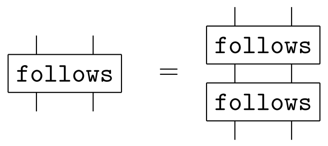


Figure 37: Visualising the effect of the two qubit `follows` gate of the four-directional model. For brevity, we consider its application to each of the possible initialisations for the second actor, fixing the first actor to always walk south. See Fig. 38 for the full plot. The label $|x, y\rangle$ identifies the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. The expected joint final state is $|y, y\rangle$. The paired Bloch spheres for the four directional model are very similar to the two directional model, except that the surfaces are much thinner. This suggests that the `follows` gate is acting much more like a projection, keeping the final states relatively un-entangled (though they remain classically correlated). We verify the (approximate) idempotence of the two qubit `follows` gate. Comparing the result of applying `follows` twice (b), to applying `follows` only once (a), we see that a phase shift is applied, though the shape of the states are very similar.

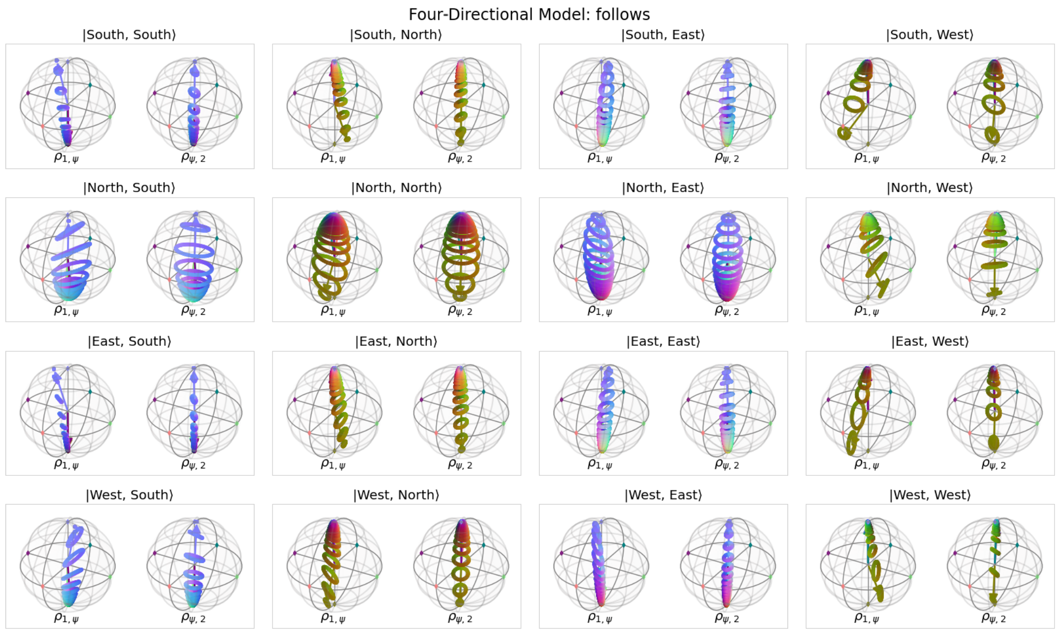
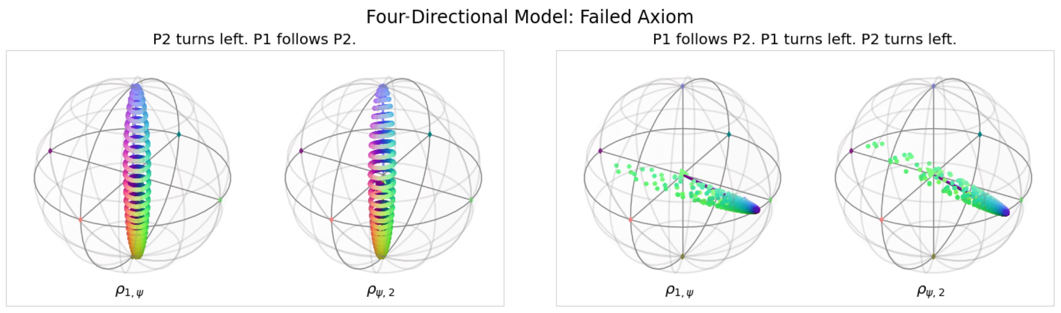


Figure 38: Full characterisation of the four direction following gate. Each paired Bloch sphere labelled $|x, y\rangle$ describes the state $U(\theta_{\texttt{follows}})\,|x, y\rangle$. We see the pairing of directions, as well as the independence of the final state on the initial state $|x\rangle$ of the first qubit.



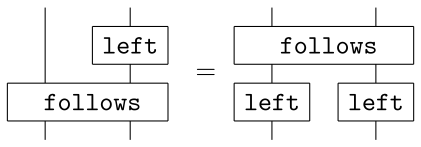
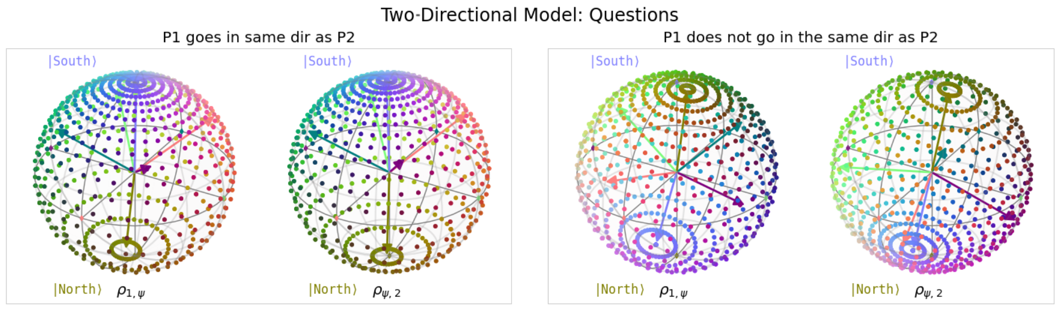
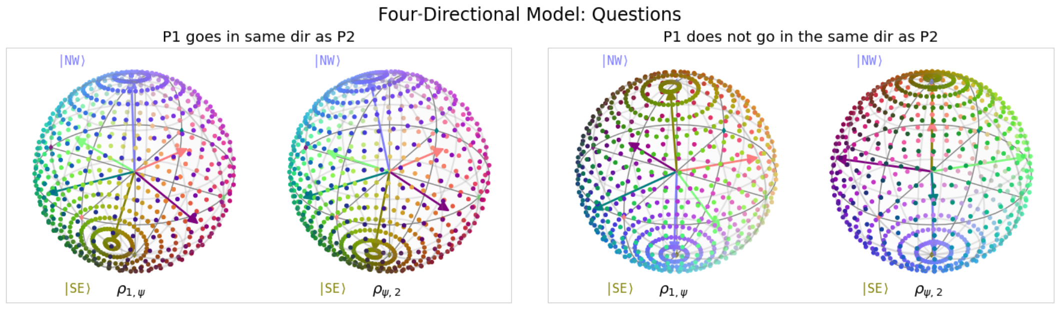


Figure 39: Demonstration that axiom 3 does not hold for this model. In both cases shown, the initial directions were $|\texttt{South, South}\rangle$. The resulting states are respectively approximately a classical ensemble $\frac{1}{2}(|00\rangle\langle 00| + |11\rangle\langle 11|)$, and the pure, separable state $|ii\rangle\langle ii|$. This reinforces the finding that `follows` involves a projection, and highlights the inherent difficulty faced by the four directional model in encoding two bits of information into a single qubit.



(a)



(b)

Figure 40: Visualising the questions for the two and four direction models. The main directions distinguished by the model are labelled and coloured according to the state identified. These are all maximally entangled pure states as both paired surfaces extend over their entire spheres. a. Two directional model i. `same dir`: we see here that $\rho_{1,\psi} \approx \rho_{\psi,2}$, such that states are correlated along $|\texttt{North, North}\rangle$ and $|\texttt{South, South}\rangle$ with no phase shift. We also see that the possible states $\rho_{1,\psi}, \rho_{\psi,2}$ are slightly concentrated towards the $|0\rangle$ state, which suggests the model can tolerate more error when the final state of the text *should* be $|\texttt{South, South}\rangle$, but has instead deviated towards the equator. ii. `not same dir`: Conversely, the negative question expresses an anti-correlation along $|\texttt{North}\rangle$ - $|\texttt{South}\rangle$, with a slight phase shift between the two surfaces. $\rho_{1,\psi}$ and $\rho_{\psi,2}$ also exhibit a slight skew towards $|\texttt{South}\rangle$ and $|\texttt{North}\rangle$ respectively, which would allow the model to tolerate more error when the final text state is expected to be $|\texttt{South, North}\rangle$. b. Four directional model: This model exhibits similar behaviour to the two-directional one, with similar, though less prominent, skews (recall that $|\texttt{NW}\rangle \approx |0\rangle$ for the four directional model), such that `same dir` favours $|\texttt{SW, SW}\rangle$ and `not same dir` favours $|\texttt{NE, SW}\rangle$.

H.3 Interventions

Another way to better understand our model's robustness and inner workings is to carry out interventions on the original stories and investigate how these affect the model's output. Therefore, we generated an alternative test dataset from a subset of the different density four-directional datasets. This subset contains the full simple and deeper datasets with up to 20 nouns, less dense with up to 14 nouns, dense with up to 12 nouns, and superdense with up to 10 nouns. We chose this reduced subset for efficient simulation. At the end of each story, we added an action from the set of actions {`turns right`, `turns left`, `turns around`, `follows`, `goes in the opposite direction of`} to one or both actors involved in the question to see how it would affect the model's output. The single-actor actions were always applied to the second actor in the question, while the two-actor actions were applied to both actors in the occurring order. For instance, if we consider a story in which `Bob goes in the same direction as Alice`, which is classified correctly by our model, we add the sentence `Alice turns left`, so now they are not facing the same direction anymore and check whether our model classifies this new data point correctly. The results are depicted in Fig. 41.

Discussion Interventions with `turns left` and `turns right` reduce the overall accuracy, which can be seen because the 'correct after intervention' line is lower than the 'correct before intervention' line and the 'misclassified after intervention' line is higher than the 'corrected after intervention' line. For interventions with `turns around`, the overall accuracy stays about the same. The 'correct before intervention' and 'correct after intervention' lines are quite similar because about the same amount of stories are 'corrected after intervention' as 'misclassified after intervention'. Interventions with `follows` and `goes in the opposite direction of` increase the accuracy to 100%.

This aligns with the insights gained from examining the Bloch spheres in App. H.2. The model faces difficulties in accurately classifying stories when the actors are positioned at 90 degrees relative to each other, likely a reflection of the bias within the dataset of having more stories with actors facing the same or opposite directions than being positioned at 90 degrees to each other. See App. H.5 for further details. Adding `turns left` and `turns right` interventions to stories, where characters previously faced opposite directions, results in them now being positioned at 90-degree angles to one another. Conversely, stories where characters originally faced each other at 90 degrees are adjusted to face the same or opposite directions. Given that the majority of our original dataset comprises stories with actors facing the same or opposite directions, the majority of stories in our intervention dataset will have actors facing each other at 90 degrees. Thus, the 'misclassified after intervention' line is higher than the 'corrected after intervention' line.

Interventions with `turns around` do not affect the angle at which people are facing each other. Thus, the overall accuracy stays about the same. From Fig. 31b iii) we see that the action `turns around` introduces some error. Depending on the story's course up to the point of the intervention this noise might be the cause for the 'corrected after intervention' and 'misclassified after intervention' stories.

The questions do not consider the absolute directions people are going but rather check the correlation of the two actors involved. The actions `follows` and `goes in the opposite direction of` set this correlation correctly as they project the state of the second actor onto the same or opposite state of the first actor regardless of the exact position of the first actor's state on the bloch sphere. Therefore interventions with `follows` and `goes in the opposite direction of` correct the labels of all stories and increase the accuracy to 100%. This also aligns with the finding that the model generally performs better on instances that require fewer inference steps. A more detailed analysis of the model's accuracy depending on the number of inference steps can be found in App. H.5.1.

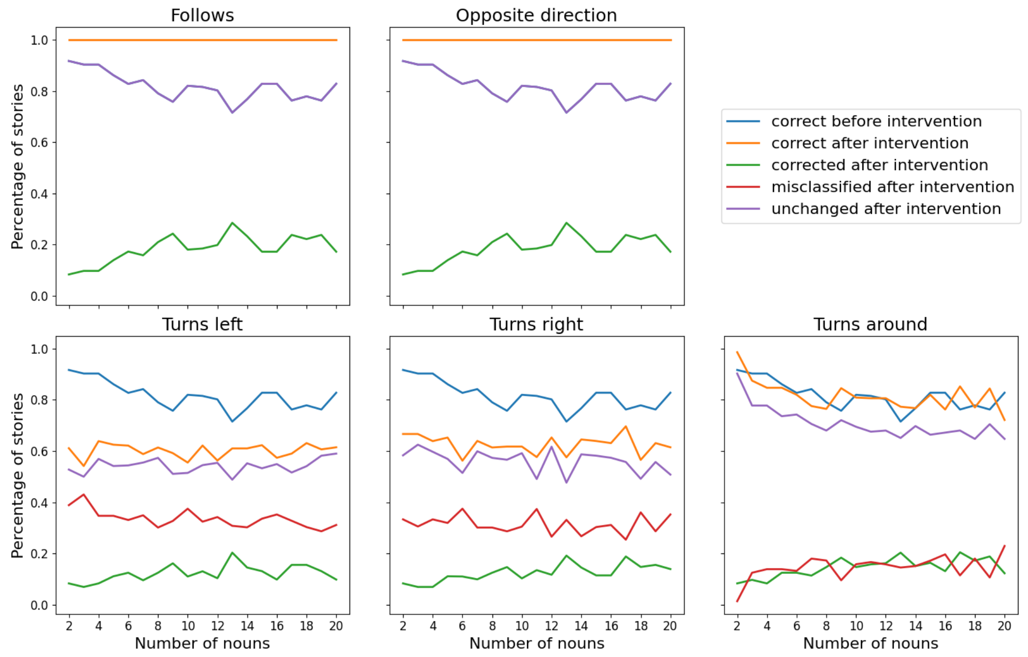


Figure 41: Effect of performing interventions at the end of stories. The lines for 'correct before intervention' and 'correct after intervention' indicate the total percentages of stories that were accurately classified without and with the intervention. The 'corrected after intervention' line shows the percentage of stories incorrectly classified without the intervention but corrected with the intervention. Conversely, the 'misclassified after intervention' line describes stories classified incorrectly after the intervention that were previously classified correctly. The 'unchanged after intervention' line shows the percentage of stories that maintained their classification accuracy regardless of the intervention, although this might still imply a change in the story's label. For instance, if in the original story, `Bob is not going in the same direction as Alice`, and the model accurately identified this, but after the intervention, `Bob is following Alice` and the model still classifies it accurately (now as `goes in the same direction`), this story would contribute to the 'unchanged after intervention' percentage because its classification remained accurate despite the label change.

H.4 Clifford models

Given the simplicity of the task, we are able to construct a perfect, deterministic, compositional model that solves the task by hand. The two-directional model is described in Fig. 42, while Fig. 43 describes the four-directional model. These models are constructed to perfectly implement the axioms required to solve the task. In the two-directional case, this provides a target for the trained model to strive towards, as well as a baseline for comparison.

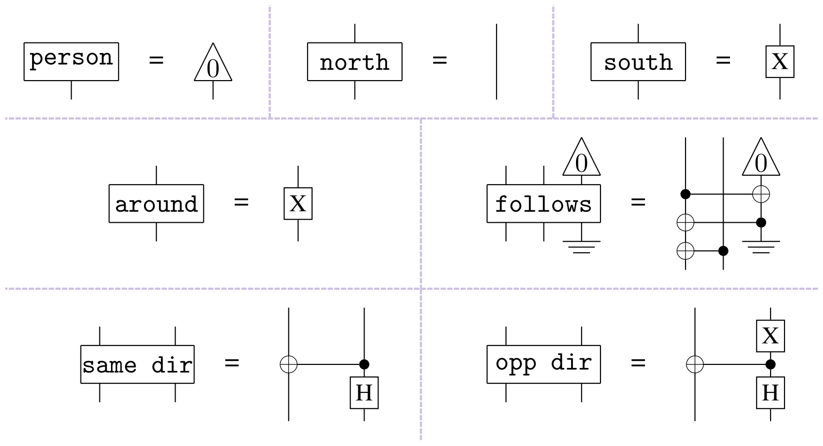


Figure 42: An implementation of a deterministic model for the two-directional dataset. The model associates $|\texttt{North}\rangle$ with $|0\rangle$ and $|\texttt{South}\rangle$ with $|1\rangle$. Turning around is implemented as a bit-flip. `Follows` discards the first qubit, then copies the second qubit along the computational basis, such that the state of the first qubit will match that of the second when projected onto the computational basis. Finally, the questions become bell states checking for correlation or anti-correlation along the computational basis.

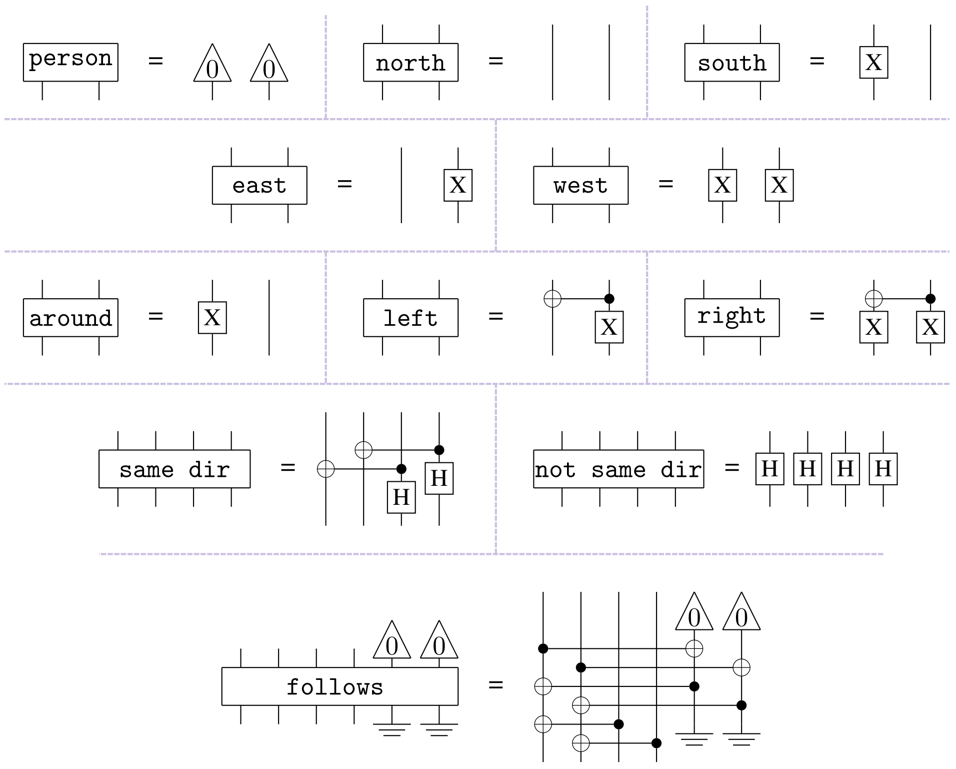


Figure 43: An implementation of a deterministic model for the four-directional dataset. Here, each of the cardinal directions is assigned a state of the computational basis: $|\texttt{North}\rangle = |0,0\rangle$, $|\texttt{South}\rangle = |1,0\rangle$, $|\texttt{East}\rangle = |0,1\rangle$, $|\texttt{West}\rangle = |1,1\rangle$. We can interpret these states as numbers, such that the rotations can be implemented as addition (modulo 4): `left` acts as $+1$, `around` as $+2$ and `right` as $-1 \equiv +3$. Like the two-directional model, `Follows` discards the wires corresponding to the first actor, then copies the state of the second noun along the computation basis. The positive question `same dir` acts as a bell state, so the overlap will be one when the states are the same and zero otherwise, whilst the overlap with `not same dir` will always be one half, and thus the maximum of the two checks if the states are the same or different.

H.5 Dataset bias

Fig. 44 shows the accuracy for different combinations of directions the actors involved in the final question face, contextualised by the number of such data points as the size of the point (also labelled). We can see that the model correctly classifies stories in which the actors are going in the same or opposite direction. Stories where the actors are positioned at 90 degrees to each other are only classified correctly for the combinations `north-east` and `south-west`. These are the cases where the directions are located on opposite poles of the Bloch sphere in Fig. 34 a. due to the clustering of directions. The figure also shows that there are only very few instances with people facing directions at 90 degrees to each other compared to the other combinations. The high amount of stories with people going in the same direction arises from the need to generate a balanced dataset in terms of labels. Other than that the proportion of different combinations of directions was not controlled in the data generation and the strong imbalance originates from favouring two-actor actions (which result in people facing the same or opposite directions) over single-actor actions (which can result in all possible combinations including 90 degrees angles between the actors) in order to increase story density. Therefore the denser the dataset, the fewer quarter-turns in the stories. Thus, the performance of the four-directional model can be seen to arise by exploiting this unintentional bias in our dataset.

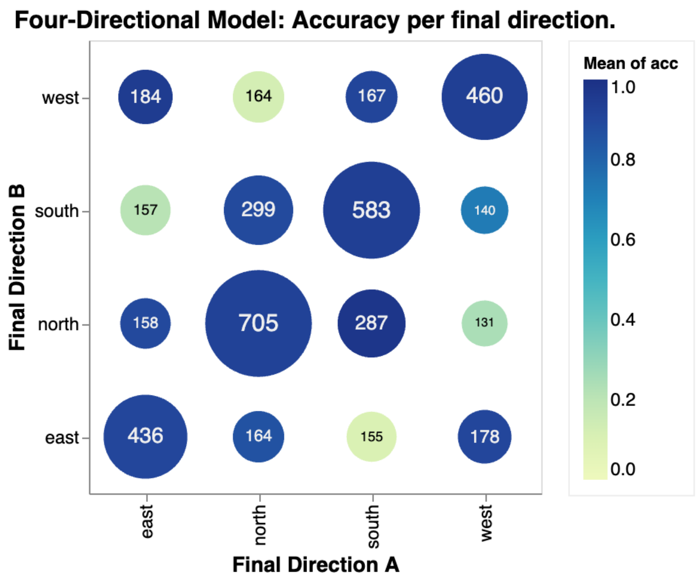


Figure 44: Comparing model accuracy against the final direction faced by each person.

H.5.1 Inference steps

Fig. 45 shows the correlation between accuracy and the number of sentences required to solve the task. The height of the bars corresponds to the number of datapoints in that region, which is also labelled. This metric intends to capture the task's expected 'difficulty'. This particular metric was not controlled during data generation, and favouring connectivity in our generation method resulted in most stories requiring fewer inference steps. We see that the model struggles more with the 'harder' instances: the accuracy tends to decrease as the inference steps are increased. From App. H.2 we know that `turns right`, `turns left` and `turns around` introduce an error that accumulates and results in an accuracy drop for stories that require more inference steps.

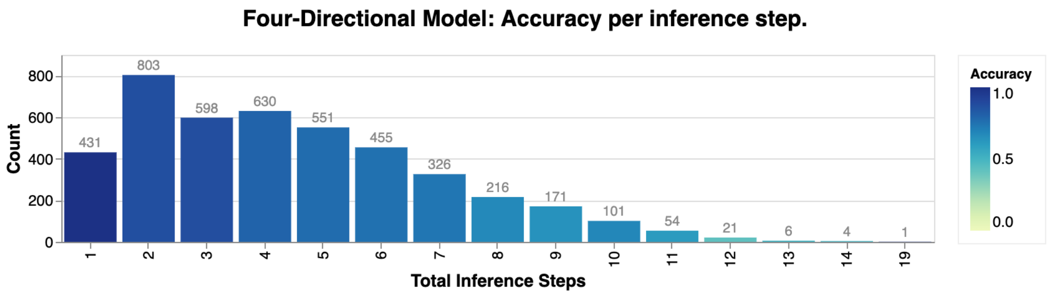


Figure 45: Model accuracy against the total inference steps required to solve the question.

I Resources

Having demonstrated that productivity provides a scaffold for constructing circuits that solve the question-answering task for the two datasets we have created, it is natural to investigate at which instance size a quantum computer is necessary for executing the circuits involved. This requires the quantification of the classical resources necessary to simulate the quantum circuits, as well as the quantification of the effect of noise in the circuits to the output of the model. This is because the presence of noise can be used to reduce the cost of classical simulation using approximate simulation methods [43, 44].

Here, we present a thorough study of the classical resources necessary for simulating the circuits involved in the question-answering task for the dataset we have introduced in this work. We consider the cost of exact tensor contraction for establishing a reasonable upper bound on the simulation cost that is less naive than state vector simulation. Further, we perform noisy simulations of the circuits using the simple noise model of a depolarising channel to quantify the effect of noise on the accuracy of the task at hand.

I.1 Tensor contraction cost estimates

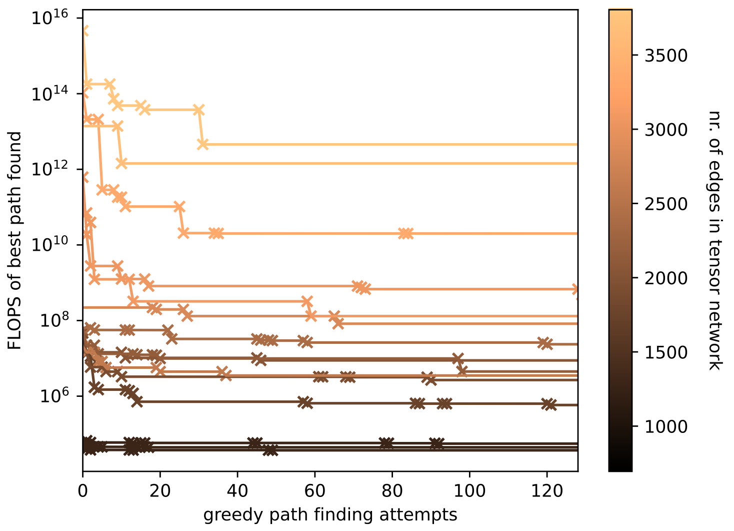


Figure 46: FLOPS required to perform a tensor contraction using the path that minimises this number plotted against the number of random greedy path optimiser runs performed so far.

We provide resource estimates for the exact simulation of our circuits using tensor networks, which are one of the most competitive methods for simulating quantum systems exactly [45, 46]. Throughout this work, tensor contraction paths were found by running a stochastic greedy optimizer in the `opt_einsum` Python package [47]. This optimiser works by building the contraction step-by-step,

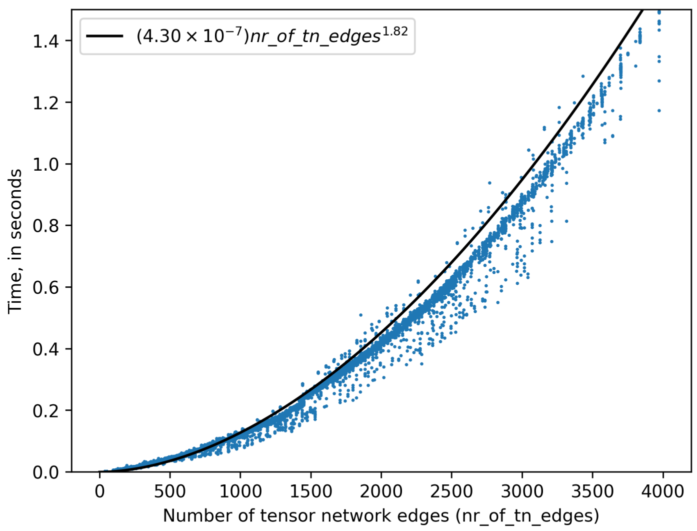


Figure 47: Time spent computing a single run of the random greedy path optimiser plotted against the number of edges in the tensor network.

at each stage choosing between the available contractions with a probability given by a Boltzmann distribution, i.e. with a probability proportional to

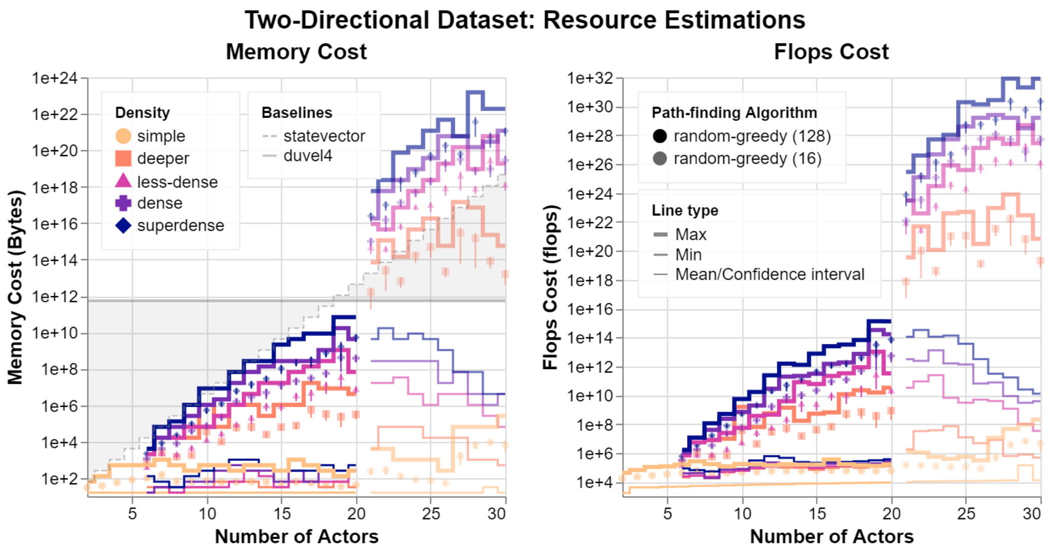
$$\exp\left(-\frac{\epsilon}{kT}\right) \tag{4}$$

where ϵ is determined by the number of FLOPS (floating-point operations per second) the contraction takes. This process repeats a set number of times N and the path with the lowest total number of FLOPS is returned. In Fig. 46, we plot the number of FLOPS of the best contraction path found against the number of times the random greedy optimiser has been run. We see a large initial drop in the number of FLOPS required as the number of runs increases, and then sporadic drops in this number, suggesting that there are diminishing returns in the time spent looking for the optimal path. Data is for 15 points out of "dense" class of the four-directions "following" dataset, sampled uniformly according to the total number of edges in the tensor network, which is illustrated using the colour of the plot lines. For the experiments in this paper, we found that 128 runs represented a good trade-off between time spent looking for an optimal path and improvement in the contraction path.

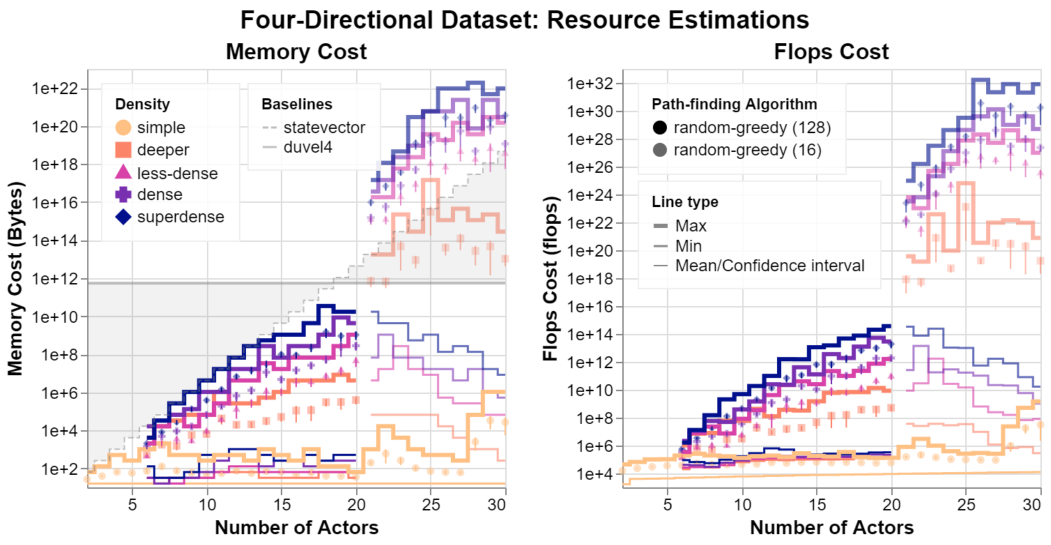
In Fig. 47 we plot the time taken to compute one run of the random greedy optimiser as a function of edges present in the tensor network for the four-directional "following" dataset. We see an approximately quadratic relationship between these quantities, for which we perform a fit with parameters given in the plot legend. Points correspond to the full four-directions "following" dataset, where only one of the positive and negative answer circuits is picked for each point (these circuits have the same tensor structure).

All simulations involving tensor contractions were run on our *duvel4* machine on 16-core hyper-threaded Intel® Xeon® Gold 5317 CPU and 512GB of RAM. While we experimented with using a GPU to perform the tensor contractions, we found no improvement over using a CPU for this purpose. This is because the tensor networks, having different shapes, can not be directly batched and executed in parallel on a GPU. While further processing could be employed to batch sub-components of the tensor network having the same shape, we leave this for future work. Fig. 48(a) displays the exact tensor contraction costs for the two-directional dataset. It shows an exponential scaling of the space and time costs for the exact classical simulation of the text circuits involved in our question-answering task. The same data for the four-directional dataset can be found in Fig. 48(b).

For each density, we plot the required memory and FLOPS (floating-point operations per second) for the exact evaluation of each data point as a full tensor contraction. In each case, the thickest line shows the most expensive contraction, while the thin line gives the lower bound. The mean for each number of actors is plotted as a point, with a vertical line showing the bootstrapped 95% confidence interval for the mean. Fewer iterations were used to compute the paths for data points with more



(a) Two-directional dataset.



(b) Four-directional dataset.

Figure 48: Resource estimates.

than 20 actors, to keep the time spent reasonable (see the rest of App. I.1 for further detail). We show two baselines for the memory cost to contextualise this data. To train and evaluate the models we used Quantinuum's *duvel4* server, which has 512GB of memory. The second baseline reflects performing a statevector-style simulation of the circuit. We assume the statevector requires at most $n + 1$ qubits, that is: one qubit per noun, n nouns, and a single reused ancilla qubit. The circuit must be evaluated as a doubled state (or density matrix) due to the presence of discards. The total memory cost is then $2^{2(n+1)}$.

I.2 The effects of noise

Finally, we quantify the effect of noise on the *Valid Comp* accuracy of the model, and examine how its influence on the model's predictions depends on the number of actors that the texts feature. To this end, we employ a noise model based on the emulator [48] for Quantinuum's H1-1 ion-trap quantum computer, which is applied after the circuits have been compiled down to the device's native gate

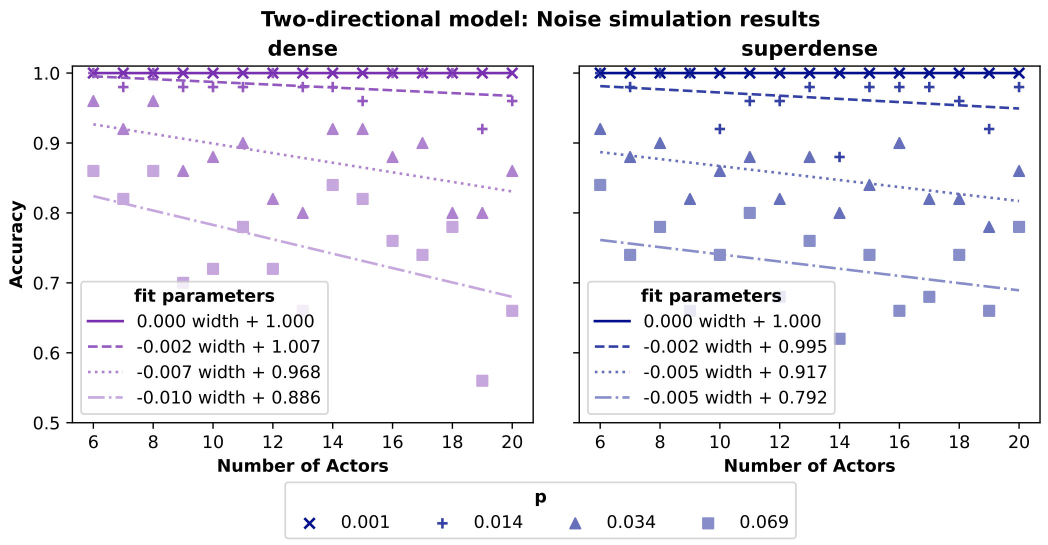


Figure 49: *Valid Comp* accuracy for the two directional dataset, stratified by number of actors, evaluated numerically using the depolarising noise levels listed in the figure legend, where p is as in Eq. (5).

set. We focus our noise investigations on the two-directional dataset, since this is the one we ran on the H1-1 device.

For simplicity, we only consider a symmetric depolarising 2-qubit gate noise, which affects the device's $R_{ZZ} = e^{-i\theta Z_j Z_k}$ $(j \neq k)$ gates with probability

$$P_{2q}(\theta) = \left(a \frac{|\theta|}{\pi}^{c} + b \right) p, \tag{5}$$

where $a = 1.651, b = 0.175, c = 1, p = sp_0, p_0 = 1.38 \times 10^{-3}$, and s is a scaling factor parameter that we vary to control the noise level. We noramalise the angles to be in the interval $[-\pi, \pi)$. Note that P_{2q} depends on the angle of rotation used, and that $p = p_0, s = 1$ corresponds to the noise level of the H1-1 device. The simulations were run using `qujax`'s [49] statevector simulator by following a Monte-Carlo sampling approach described in App. I.2.1.

In Fig. 49, we plot how the accuracy varies with increasing circuit depth for the different noise scaling factors, where we consider $s = 1, 10, 25, 50$. We see that when the noise level models that of the H1-1 device ($p \approx 0.001$), no drop in accuracy occurs. As the noise probability increases, we observe a decrease in the model's accuracy with the number of actors. This is because the number of actors in a text and the depth of the corresponding circuit are correlated (App. C Fig. 10), and the effect of noise on deeper circuits is more pronounced. For larger noise levels, there is an approximately linear decay of the accuracy; this decay is expected to plateau at 0.5, indicating the model's performance has degraded to effectively random guessing. Note that the two-qubit gate noise present in H1-1 is not enough to cause the model's accuracy to decrease for the number of actors considered in this noise study, i.e. up to 20 actors. This shows a direct correspondence between noise present in the device and the maximum admissible text depth, which indicates that classical simulation techniques that leverage robustness to noise do not scale as the length of the text increases. Simulations were run using a Monte-Carlo sampling approach as described in App. I.2.1.

It is interesting to investigate the noise threshold below which the model's accuracy is not correlated with text size for a given task and dataset. Classical simulation techniques that leverage the presence of noise (such as those in e.g. [50]) would then aim to inject as much noise as possible to reduce the classical resources for simulating the circuits, while not exceeding the threshold. It is beyond the scope of this work to determine for which text sizes the classical resources outweigh the quantum resources for achieving the same test accuracy. Note that resources can be measured either in the number of primitive operations, or even time-to-solution, as the time per operation differs vastly between quantum and classical computers as they stand today.

I.2.1 Noise simulation details

The depolarising channel on two qubits $\{i, j\}$ acting on a mixed state σ can be written as

$$D(\sigma) = (1-p)\sigma + p\mathrm{Tr}_{\{i,j\}}(\sigma) \otimes \frac{I_2}{4}, \tag{6}$$

$$= \sum_{P \in \mathcal{P}^2} \alpha_P P \sigma P, \tag{7}$$

$$\alpha_P = \begin{cases} 1 - \frac{15}{16}p, & \text{if } P = I \otimes I \\ \frac{p}{16}, & \text{otherwise} \end{cases} \tag{8}$$

where $\frac{I_2}{4}$ is the fully mixed state on the qubits, $\mathrm{Tr}_{\{i,j\}}$ is the partial trace over the subsystem defined by the qubits, and p is the probability of noise occurring.

We simulate noise using a sampling approach as follows: suppose we wish to apply a 2-qubit gate U, but our system is affected by a depolarising noise channel. Suppose also that we are interested in computing the expectation value $\mathrm{tr}(\sigma O)$ of some operator O. Assuming a pure initial state $\sigma_0 = |\psi_0\rangle\langle\psi_0|$, this expectation can be expressed as

$$\alpha_{I\otimes I}\langle U|\psi_0\rangle\rangle_O + \sum_{P \in \mathcal{P}^2 \setminus I\otimes I} \alpha_P \langle P|\psi_0\rangle\rangle_O \tag{9}$$

The above equality means that we can sample expectation values of several pure state simulations according to the probabilites α_P to arrive at the expectation value of the operator O under the noise model, which would otherwise require mixed state simulation. Note that, in practice, we will have multiple noisy gates in our circuit. To apply this technique, every time we encounter a noisy gate, we sample and apply a gate according to the probabilities α_P. This technique is often called *quantum trajectories* or *Monte Carlo (wavefunction) simulation*. It avoids the memory required to simulate mixed states using density matrices (which is double that of pure state simulation using statevectors) in exchange for an approximate result that depends on the number of samples taken (but which is unbiased, i.e. will converge to the exact value as the number of samples increases). We use qujax [49], a Python library which leverages Google's JAX [51] platform, to just-in-time compile the statevector simulations using the XLA [52] framework. This greatly speeds up simulation, and allows us to batch circuit execution, which can be done over both the number of samples and the noise strengths used. These simulations were run on our *duvel1* machine, on a NVIDIA A100 80GB GPU.

I.3 Project costs

Finally, we document the approximate upper bound of resource costs incurred throughout this project. Table 9 displays the resources used to process the datasets - this includes generating the texts, then compiling them through our pipeline to obtain quantum circuits. Path optimisation for the training samples was also performed as described above.

The training resources are summarised in Table 10. The difference in training times per epoch between datasets is largely due to the different composition of the respective *Train* datasets: the two-directional dataset only includes simple datapoints whereas the four-directional dataset also includes denser examples (deeper - superdense). Some variation can also be attributed to whether other processes were running on the machines at the same time - in some cases, the device ran out of memory and SWAP memory was used instead, leading to disproportionately long running times. The most extreme outliers were not counted when computing the typical per-model costs, however do feature in the total cost.

We provide an upper bound only for the execution time for the Test datasets as these timings were not monitored explicitly. Finally, we present the combined time classical compute time spent directly building up to the results presented in this paper. These figures include training of candidate models, cross-validation runs, hyperparameter tuning runs, and finally evaluation on the *Valid Comp* datasets. The time spend on the four-directional dataset is an order of magnitude larger than that spent on the two-directional largely due to the hyperparameter tuning. Note that extra compute was also used during preliminary experiments not reported on here. Our experiments were run across a series of devices as per Table 11.

Finally, we provide the exact time taken to evaluate our model on a Quantum computer. On average, entries from the *Test (H1-1)* sub-dataset took 5.09 minutes to execute both question circuits with 50 shots each, while the *Valid Comp (H1-1)* entries took about half the time at 2.80 minutes per datapoint on average. In total, 22.2 hours of compute were required to evaluate both datasets. We contrast this to the time spent collecting equivalent data for the four directional *Test (Qujax)* dataset, which was about three times slower.

Table 10 also provides estimates for the resources used when training the baseline LSTM and transformer models. As the hyperparameter tuning was performed separately, the final architectures for the two-directional and four-directional models lead to differing runtimes per epoch.

Table 9: Approximate compute resources used for preparing the datasets.

Task	**Time**	**Device**
Dataset compilation	≈ 7.5 hours	*duvel3*
Path optimisation	≈ 7 days	*duvel2*

Table 10: Approximate compute resources used for training models.

Task	**Time**		**Device**
	2 Directions	4 Directions	
LSTM training (per model and epoch)	≈ 2.6 secs	≈ 4.8 secs	*duvel4*
Transformer training (per model and epoch)	≈ 5.6 secs	≈ 3.9 secs	*duvel4*
Baselines evaluation *Valid Comp, Test* (per model)	≤ 4 min	≤ 4 min	*duvel4*
Baselines training & evaluation (total)[*]	≤ 36 min	≤ 37 min	*duvel4*
QDisCoCirc training (per model and epoch)	$3 - 7$ min	$10 - 20$ min	*duvel2,4*
QDisCoCirc evaluation *Valid Comp* (per model)	$8.5 - 12$ hours	$20 - 30$ hours	*duvel4*
QDisCoCirc training & evaluation (total)[**]	4.4 days	51 days	*duvel2,4*
QDisCoCirc evaluation *Test (Qujax)*		≈ 1.5 days	*duvel1*
QDisCoCirc evaluation *Valid Comp (H1-1)*	11.2 hours		H1-1 [16]
QDisCoCirc evaluation *Test (H1-1)*	11.0 hours		H1-1 [16]

[*]Includes only runs relevant to the final models.
[**]Includes hyperparameter tuning and runs of models that were not eventually selected. Note that some runs occurred in parallel but were timed separately.

Table 11: Classical device specifications.

Name	**Primary resource**	**Memory**
Duvel1	**GPU:** NVIDIA A100	80GB
Duvel2	**CPU:** 12-core hyper-threaded Intel® Xeon® W-2265	125GB
Duvel3	**CPU:** 32-core hyper-threaded Intel® Xeon® Gold 6326	252GB
Duvel4	**CPU:** 16-core hyper-threaded Intel® Xeon® Gold 5317	512GB

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract lists the main results of our paper, as does the introduction in the second paragraph. The main limitations are highlighted (eg the toy nature of the dataset), and the choice to use a quantum model.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a paragraph highlighting some of the most important limitations in Sec. 4. Limitations of comparing classical baselines to our model are acknowledged briefly in Sec. 3, and App. G in more detail. Detailed analysis of the resources required for the model can be found in App. I.1.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This work does not involve theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed description of our dataset, model and setup in App. A, C and D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code used to generate our results is not currently open source. We can provide a showcase package of our trained models upon request, which exposes our trained models.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: This is summarised in Sec. 2. The exact data splits used are documented in App. C.2 while hyperparameters and training packages can be found in App. D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The error bars in all figures are described (in the majority of cases, we report a 95% CI for the mean accuracy of a single model, calculated via the Clopper-Pearson method [22]).

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide a detailed analysis of the resources required to train our model classically (App. I.1), and provide estimates of the actual compute time in App. I.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No human subjects were involved in this research. Datasets used have no bearing on any real entities, and are entirely synthetic. We do not forsee any negative societal impact from this research.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The model presented in this work is a proof-of-concept for toy data. As the model is not yet applied in any real-world applications or at scale, we consider the societal impact of the work largely negligible in favour of its theoretical impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our model does not pose a risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Relevant code packages used are cited appropriately.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide a mini web application that showcases the models reported on in this paper, along with setup instructions. Instructions for interacting with the model via the app are provided at the default page.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing or human subjects were involved in this research.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.