Extended Abstract Track

Does equivariance matter at scale?

Editors: List of editors' names

Abstract

Given large data sets and sufficient compute, is it beneficial to design neural architectures for the structure and symmetries of a problem, or is it more efficient to learn them from data? We study empirically how equivariant and non-equivariant networks scale with compute and training samples. Focusing on a benchmark problem of rigid-body interactions and general-purpose transformer architectures, we perform a series of experiments, varying the model size, training steps, and dataset size. We find evidence for three conclusions. First, equivariance improves data efficiency, but training non-equivariant models with data augmentation closes this gap. Second, scaling with compute follows a power law, with equivariant models outperforming non-equivariant ones at each tested compute budget. Finally, the optimal allocation of a compute budget onto model size and training duration differs between equivariant and non-equivariant models.

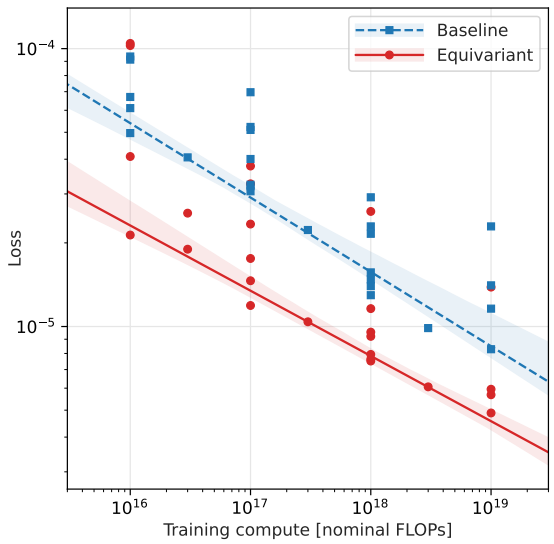

Figure 1: **Scaling with compute**. The dots show the training compute budget and test loss in our experiments. The lines indicate the best possible performance for each training compute budget according to the scaling laws we find. The performance of both non-equivariant (– –) and equivariant (——) transformers scales as a power law with compute, and the equivariant model outperforms the non-equivariant model by a similar factor at all tested compute budgets.

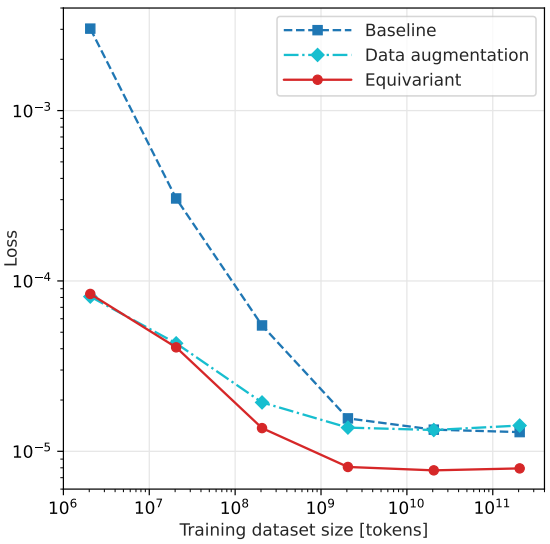

Figure 2: **Scaling with training data**. We show the performance of the non-equivariant transformer (– –), non-equivariant transformer trained with data augmentation (—·), and equivariant transformer (——) as a function of the number of unique tokens in the training dataset. Equivariance improves data efficiency compared to the baseline, but data augmentation closes this gap when the number of epochs is large.

1. Questions

In a time of big data and abundant compute, are there any benefits to strong inductive biases—in particular, network architectures tailored to the properties of a problem? Concretely, consider problems with known symmetries: should one take them into account through equivariances (Bronstein et al., 2021), or is it better to let the network learn them from data?

A common intuition is that strong inductive biases bring the biggest benefits when little training data is available, and that symmetry properties can just as well be learned from data given sufficient samples and compute. Recently, high-profile models of protein folding (Abramson et al., 2024) and conformer generation (Wang et al., 2023) have received considerable attention for their choice of non-equivariant architectures for geometric problems.

At the same time, there is reason to expect that equivariance is still beneficial in the large-data limit. Learning means successively narrowing down a hypothesis class based on evidence, and scaling laws can be explained from this perspective (Bahri et al., 2021). Whereas non-equivariant methods start from the space of virtually all functions, equivariant models start from the subspace of all functions that abide by the symmetries of the problem. The learning process may benefit from that by focusing solely on further refining this smaller hypothesis class, narrowing down the solution space further than without equivariances.

Until the theory of scaling laws is fully understood, the effects of equivariance on scaling is an empirical question, and in this work we study it empirically. We focus on a benchmark problem of rigid-body physics and compare a standard transformer architecture (Vaswani et al., 2017) to an E(3)-equivariant transformer (Brehmer et al., 2023). In this setup, which we describe in Sec. 2, we ask three questions:

1. *How do equivariant and non-equivariant models scale as a function of the available data?* Does data augmentation affect this?
2. *How do equivariant and non-equivariant models scale as a function of training compute?* Does this scaling follow power laws? Are their coefficients affected by equivariance?
3. *Given a compute budget, how should one allocate it to the model size and the number of training iterations?* Is this trade-off different for equivariant and non-equivariant models?

We discuss our empirical results and sketch our answers to these questions in Sec. 3.

2. Methods

Benchmark problem We choose a rigid-body modelling problem as our benchmark. Extended, rigid objects are initialized at some position, orientation, and velocity; they then interact with each other under gravity and collisions. We task networks with predicting the position of all mesh vertices as a function of the positions at two previous time steps. This task is known to be challenging because collisions are difficult to detect, since they do not usually occur at or near vertices, and because the forces acting during collisions are nearly discontinuous (Bauza and Rodriguez, 2017; Pfrommer et al., 2021; Allen et al., 2022). We describe the dataset generation and problem specification in more detail in Appendix B.1.

Models In selecting architectures, our main objective is not to achieve state-of-the-art results on the particular rigid-body benchmark. That would lead us to highly problem-specific architectures (Allen et al., 2022; Rubanova et al., 2024). Instead, we study two

general-purpose architectures.

The first is a (non-equivariant) transformer (Vaswani et al., 2017), which has become the de-facto standard across a wide range of machine learning tasks and is known for its good scaling behaviour. As E(3)-equivariant architecture, we use the Geometric Algebra Transformer (Brehmer et al., 2023), since it follows the transformer paradigm to a large extent and is broadly applicable to geometric problems. To the best of our knowledge, it is the only E(3)-equivariant architecture that—like the standard transformer—computes interactions between items through dot-product attention and can be used with efficient implementations like FlashAttention (Dao et al., 2022). We describe both models in Appendix B.2.

Scaling analysis We perform two series of experiments. First, we study the scaling with compute, in the (practically) infinite-data setting. We vary a training compute budget over three orders of magnitude. For each budget we consider, for both the baseline and the equivariant transformer, we perform multiple experiments, each with a different trade-off between model size N and training length D. Second, we study the scaling with training data, fixing the training compute budget, the model size, and the number of training tokens.

We then analyze the scaling with compute quantitatively by fitting scaling laws to the experiment results. Following Kaplan et al. (2020) and Hoffmann et al. (2022), we use a power-law ansatz for the loss L,

$$\hat{L}(N, D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + E\,, \tag{1}$$

where A, B, E, α, β are fit parameters. From this we can derive the optimal loss a model can achieve as a function of the training compute budget C as well as the optimal model size and training length as a function of C. We describe this analysis in Appendix B.3, including our procedures for choosing hyperparameters and estimating uncertainties.

3. Answers

Our empirical results, which can be found in Appendix C, provide evidence for the following three conclusions.

Equivariant transformers are more data-efficient, but data augmentation closes this gap. The first (and expected) benefit for the equivariant architecture is that it performs better than a non-equivariant architecture when only little training data is available, as we show in Fig. 2. However, we find that a non-equivariant model trained with data augmentation performs just as well as the equivariant architecture, at least when the number of epochs is sufficiently large.

This finding does not support the hypothesis that equivariance at each layer of the network yields better data efficiency than augmentation, which only makes the networks as a whole equivariant, and only on the training set.

The scaling with compute follows power laws, and equivariant models outperform non-equivariant ones at each tested compute budget. Both for non-equivariant and equivariant models, the test loss is well described by the power-law ansatz of Eq. (3), with parameters given in Tbl. 2 in Appendix C.

The best achievable test loss L^* for a given training compute budget of C FLOPs

therefore also follows a power law. We find

$$L^*_{\text{baseline}}(C) = \frac{1.03}{C^{0.268}} \quad \text{and} \quad L^*_{\text{equivariant}}(C) = \frac{0.132}{C^{0.235}}\,, \tag{2}$$

with exponents compatible with each other within the confidence intervals.

This shows a second (and perhaps less expected) benefit for the equivariant architecture: for any fixed compute budget, even in the infinite-data limit, it clearly outperforms the baseline method. As we show in Fig. 1, this benefit is approximately constant over the range of compute budgets we study. It is consistent with the conjecture that equivariant methods are more compute-efficient because their training can focus on a smaller hypothesis class: *not having to learn the symmetries from data saves FLOPs.*

Under the assumption that the implementations of equivariant and baseline architectures are similarly efficient and one can achieve the same FLOP throughput, this hints that researchers should choose equivariant architectures even in the large-data, large-compute regime. In practice, non-equivariant architectures may be easier to optimize for high FLOP throughput, in which case it remains to be seen which architecture is more efficient.

Equivariant and non-equivariant models require different trade-offs between model size and training duration. From the power-law scaling in model size and training length we also derive the optimal allocation of a given compute budget. In Fig. 5 in Appendix C we show that for a small FLOP budget, a compute-optimal equivariant transformer is substantially smaller than a compute-optimal baseline transformer. This gap decreases for larger compute budgets.

Limitations and open questions As much as we would like to, we do not conclusively settle the question we raised in the title of the paper. Our work is limited in several ways. First, we only analyze a single benchmark problem and two model families. We chose a task with a common symmetry group and general-purpose architectures that are frequently applied to a wide range of problems. We believe it is important to study to what extent our findings generalize to other problems or to other architectures, for instance those based on message-passing over graphs. Moreover, on the problem we do study, we do not set a new state of the art: we deliberately focus on general-purpose models, which do achieve the same level of performance on this particular task as highly problem-specific architectures (Allen et al., 2022).

Another limitation of our work is that our analysis measures compute with an idealized FLOP counting procedure, as is common practice (Hoffmann et al., 2022). As we discuss in Appendix B.3, this does not map one-to-one to real-world run time, at least not before further optimization.

Finally, we are only able to study training compute budgets of up to 10^{19} FLOPs per model—this does not come close to the approximately 10^{25} FLOPs that the currently largest language models are trained for (Dubey et al., 2024). We did not see power-law scaling break down in the range we studied, but we cannot make claims about the extrapolation beyond it.

We believe that the effects of strong inductive biases at scale is important for future progress in several fields of science and engineering, and we hope that our findings can encourage further investigations into this question.

References

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3, 2024. (Cited on pages 2 and 12)

Subutai Ahmad and Gerald Tesauro. Scaling and generalization in neural networks: a case study. *Advances in neural information processing systems*, 1, 1988. (Cited on page 11)

Ibrahim Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting vit in shape: Scaling laws for compute-optimal model design. *arXiv preprint arXiv:2305.13035*, 2023. (Cited on page 11)

Kelsey R Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks. *arXiv preprint arXiv:2212.03574*, 2022. (Cited on pages 2, 4, 12, and 13)

S-l Amari. Feature spaces which admit and detect invariant signal transformations. In *Proc. 4th Int. Joint Conf. Pattern Recognition*, pages 452–456, 1978. (Cited on page 11)

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv:1809.10853*, 2018. (Cited on page 13)

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021. (Cited on page 2)

Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Advances in Neural Information Processing Systems*, 35:11423–11436, 2022. (Cited on page 11)

Ilyes Batatia, Philipp Benner, Yuan Chiang, Alin M Elena, Dávid P Kovács, Janosh Riebesell, Xavier R Advincula, Mark Asta, William J Baldwin, Noam Bernstein, et al. A foundation model for atomistic materials chemistry. *arXiv preprint arXiv:2401.00096*, 2023. (Cited on page 11)

Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022. (Cited on page 11)

Maria Bauza and Alberto Rodriguez. A probabilistic data-driven model for planar pushing. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3008–3015. IEEE, 2017. (Cited on pages 2 and 12)

Arash Behboodi, Gabriele Cesa, and Taco S Cohen. A pac-bayesian generalization bound for equivariant networks. *Advances in Neural Information Processing Systems*, 35:5654–5668, 2022. (Cited on page 11)

Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, pages 440–448. Springer, 2018. (Cited on page 11)

Alexander Bogatskiy, Timothy Hoffman, David W Miller, and Jan T Offermann. Pelican: permutation equivariant and lorentz invariant or covariant aggregator network for particle physics. *arXiv preprint arXiv:2211.00454*, 2022. (Cited on page 11)

Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S Albergo, Kyle Cranmer, Daniel C Hackett, and Phiala E Shanahan. Sampling using su (n) gauge equivariant flows. *Physical Review D*, 103(7):074504, 2021. (Cited on page 11)

Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J Bekkers, and Max Welling. Geometric and physical quantities improve E(3) equivariant message passing. In *International Conference on Learning Representations*, 2022. (Cited on pages 12 and 14)

Johann Brehmer, Pim de Haan, Sönke Behrends, and Taco Cohen. Geometric Algebra Transformer. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 37, 2023. (Cited on pages 2, 3, 12, 13, and 14)

Johann Brehmer, Joey Bose, Pim De Haan, and Taco S Cohen. Edgi: Equivariant diffusion for planning with embodied agents. *Advances in Neural Information Processing Systems*, 36, 2024. (Cited on page 11)

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021. (Cited on pages 2 and 11)

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pages 2990–2999. PMLR, 2016. (Cited on page 11)

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2024. (Cited on page 13)

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. (Cited on pages 3 and 12)

Pim de Haan, Taco Cohen, and Johann Brehmer. Euclidean, projective, conformal: Choosing a geometric algebra for equivariant transformers. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 27, 2024. URL https://arxiv.org/abs/2311.04744. (Cited on page 12)

Larissa de Ruijter and Gabriele Cesa. Equivariant amortized inference of poses for cryo-em. *arXiv preprint arXiv:2406.01630*, 2024. (Cited on page 11)

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. (Cited on pages 4 and 14)

Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. In *International conference on machine learning*, pages 2959–2969. PMLR, 2021. (Cited on page 11)

Mathis Gerdes, Pim de Haan, Corrado Rainone, Roberto Bondesan, and Miranda CN Cheng. Learning lattice quantum field theories with equivariant continuous flows. *SciPost Physics*, 15(6):238, 2023. (Cited on page 11)

Shiqi Gong, Qi Meng, Jue Zhang, Huilin Qu, Congqiao Li, Sitian Qian, Weitao Du, Zhi-Ming Ma, and Tie-Yan Liu. An efficient lorentz equivariant graph neural network for jet tagging. *Journal of High Energy Physics*, 2022(7):1–22, 2022. (Cited on page 11)

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022. (Cited on page 13)

Thomas Hehn, Markus Peschl, Tribhuvanesh Orekondy, Arash Behboodi, and Johann Brehmer. Probabilistic and differentiable wireless simulation with geometric transformers. *arXiv preprint arXiv:2406.14995*, 2024. (Cited on page 11)

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020. (Cited on page 11)

Jan Hermann, Zeno Schätzle, and Frank Noé. Deep-neural-network solution of the electronic schrödinger equation. *Nature Chemistry*, 12(10):891–897, 2020. (Cited on page 11)

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017. (Cited on page 11)

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal large language models. March 2022. (Cited on pages 3, 4, 11, 15, 16, and 17)

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer, 1992. (Cited on page 16)

Ilia Igashov, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia. Equivariant 3d-conditional

diffusion model for molecular linker design. *Nature Machine Intelligence*, pages 1–11, 2024. (Cited on page 11)

Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021. (Cited on page 11)

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. January 2020. (Cited on pages 3, 11, and 15)

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. *arXiv preprint arXiv:2206.11990*, 2022. (Cited on page 11)

Cong Liu, David Ruhe, Floor Eijkelboom, and Patrick Forré. Clifford group equivariant simplicial message passing networks. *arXiv preprint arXiv:2402.10011*, 2024a. (Cited on page 12)

Cong Liu, David Ruhe, and Patrick Forré. Multivector neurons: Better and faster o (n)-equivariant clifford graph neural networks. *arXiv preprint arXiv:2406.04052*, 2024b. (Cited on page 12)

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989. (Cited on page 16)

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks. *arXiv preprint arXiv:2005.00178*, 2020. (Cited on page 11)

Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis. Correspondence-free structure from motion. *International Journal of Computer Vision*, 75(3):311–327, 2007. (Cited on page 11)

Mirgahney Mohamed, Gabriele Cesa, Taco S Cohen, and Max Welling. A data and compute efficient design for limited-resources deep learning. *arXiv preprint arXiv:2004.09691*, 2020. (Cited on page 11)

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *arXiv preprint arXiv:2305.16264*, 2023. (Cited on pages 11 and 16)

Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *Nature Communications*, 14(1):579, 2023. (Cited on page 11)

Mircea Petrache and Shubhendu Trivedi. Approximation-generalization trade-offs under (approximate) group equivariance. *Advances in Neural Information Processing Systems*, 36, 2024. (Cited on page 11)

David Pfau, James S Spencer, Alexander GDG Matthews, and W Matthew C Foulkes. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical review research*, 2(3):033429, 2020. (Cited on page 11)

Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, pages 2279–2291. PMLR, 2021. (Cited on pages 2 and 12)

Shikai Qiu, Andres Potapczynski, Marc Finzi, Micah Goldblum, and Andrew Gordon Wilson. Compute better spent: Replacing dense layers with structured matrices. *arXiv preprint arXiv:2406.06248*, 2024. (Cited on page 11)

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019. (Cited on page 11)

Yulia Rubanova, Tatiana Lopez-Guevara, Kelsey R Allen, William F Whitney, Kimberly Stachenfeld, and Tobias Pfaff. Learning rigid-body simulators over implicit shapes for large-scale scenes and vision. *arXiv preprint arXiv:2405.14045*, 2024. (Cited on pages 2 and 13)

David Ruhe, Johannes Brandstetter, and Patrick Forré. Clifford group equivariant neural networks. In *Advances in Neural Information Processing Systems*, volume 37, 2023a. (Cited on pages 12 and 14)

David Ruhe, Jayesh K Gupta, Steven de Keninck, Max Welling, and Johannes Brandstetter. Geometric clifford algebra networks. In *International Conference on Machine Learning*, 2023b. (Cited on pages 12 and 14)

Akiyoshi Sannai, Masaaki Imaizumi, and Makoto Kawano. Improved generalization bounds of group invariant/equivariant deep networks via quotient feature spaces. In *Uncertainty in artificial intelligence*, pages 771–780. PMLR, 2021. (Cited on page 11)

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. (Cited on page 13)

Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel Rodrigues. Generalization error of invariant classifiers. In *Artificial Intelligence and Statistics*, pages 1094–1103. PMLR, 2017. (Cited on page 11)

Jonas Spinner, Victor Bresó, Pim de Haan, Tilman Plehn, Jesse Thaler, and Johann Brehmer. Lorentz-equivariant geometric algebra transformers for high-energy physics. 2024. (Cited on pages 11 and 12)

Julian Suk, Pim de Haan, Phillip Lippe, Christoph Brune, and Jelmer M Wolterink. Mesh neural networks for se (3)-equivariant hemodynamics estimation on the artery wall. *Computers in Biology and Medicine*, 173:108328, 2024. (Cited on page 11)

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf. (Cited on page 13)

Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Won Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Q Tran, Dani Yogatama, and Donald Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? July 2022. (Cited on page 11)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *NeurIPS*, 2017. (Cited on pages 2, 3, 11, 12, and 13)

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pages 210–218. Springer, 2018. (Cited on page 11)

Dian Wang, Mingxi Jia, Xupeng Zhu, Robin Walters, and Robert Platt. On-robot learning with equivariant models. *arXiv preprint arXiv:2203.04923*, 2022a. (Cited on page 11)

Dian Wang, Robin Walters, and Robert Platt. SO(2)-equivariant reinforcement learning. *arXiv preprint arXiv:2203.04439*, 2022b. (Cited on page 11)

Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant q learning in spatial action spaces. In *Conference on Robot Learning*, pages 1713–1723. PMLR, 2022c. (Cited on page 11)

Yuyang Wang, Ahmed AA Elhag, Navdeep Jaitly, Joshua M Susskind, and Miguel Ángel Bautista. Generating molecular conformer fields. *arXiv preprint arXiv:2311.17932*, 2023. (Cited on pages 2 and 12)

Marysia Winkels and Taco S Cohen. 3d G-CNNs for pulmonary nodule detection. *arXiv preprint arXiv:1804.04656*, 2018. (Cited on page 11)

Jim Winkens, Jasper Linmans, Bastiaan S Veeling, Taco S Cohen, and Max Welling. Improved semantic segmentation for histopathology using rotation equivariant convolutional networks. 2018. (Cited on page 11)

Jeffrey Wood and John Shawe-Taylor. Representation theory and invariant neural networks. *Discrete applied mathematics*, 69(1-2):33–60, 1996. (Cited on page 11)

Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, et al. Mattergen: a generative model for inorganic materials design. *arXiv preprint arXiv:2312.03687*, 2023. (Cited on page 11)

Maksim Zhdanov, David Ruhe, Maurice Weiler, Ana Lucic, Johannes Brandstetter, and Patrick Forré. Clifford-steerable convolutional neural networks. *arXiv preprint arXiv:2402.14730*, 2024. (Cited on page 12)

Appendix A. Related work

Neural scaling laws The scaling of neural network performance as a function of model size or training steps has been studied extensively (Ahmad and Tesauro, 1988; Hestness et al., 2017; Rosenfeld et al., 2019; Henighan et al., 2020). Kaplan et al. (2020) first observed that the test loss of autoregressive language models follows a power law over many orders of magnitude. Hoffmann et al. (2022) improved the methodology further and found impactful scaling laws, to the extent that many language models compare their hyperparameter choices to the "Chinchilla-optimal" configuration. In our quantitative analysis of compute scaling, we largely follow their approach.

Several works have extended scaling laws from model size and training steps to other dimensions: Muennighoff et al. (2023) studied the effect of the training dataset size, which we also discuss, Alabdulmohsin et al. (2023) analyzed scaling of different architecture hyperparameters separately, and Jones (2021) investigated the scaling with problem complexity.

Scaling laws and inductive biases There has been comparatively little research into the relation between inductive biases and scaling behaviour, perhaps because the transformer architecture (Vaswani et al., 2017) is so established in language modelling. Tay et al. (2022) compared the scaling behaviour of different architectures. Recently, Qiu et al. (2024) investigated how structured linear transformations in transformers affect scaling laws. The authors conclude that imposing structure in them can improve the scaling behaviour. Our work differs from both of these studies through its focus on symmetric problems and equivariant architectures.

Geometric deep learning Geometric deep learning (Bronstein et al., 2021) is a paradigm for machine learning in which network architectures are designed to reflect geometric properties of the problem. One of its core ideas is that of equivariance to symmetry groups (Amari, 1978; Wood and Shawe-Taylor, 1996; Makadia et al., 2007; Cohen and Welling, 2016): roughly, a network f is said to be equivariant to a symmetry group G if $f(g{\cdot}x) = g{\cdot}f(x)$ for all elements $g \in G$ and all inputs x, where $\cdot$ is the group action. This means that when you transform the inputs into an equivariant network, its outputs transform consistently.

An equivariant network thus does not have to learn the symmetry structure from data, like a non-equivariant network does. This has been found to improve performance, data efficiency, and robustness to out-of-domain generalization in fields as diverse as quantum mechanics and quantum field theory (Pfau et al., 2020; Hermann et al., 2020; Boyda et al., 2021; Gerdes et al., 2023), molecular force fields (Batatia et al., 2022; Batzner et al., 2022; Liao and Smidt, 2022; Musaelian et al., 2023; Batatia et al., 2023), generative models of molecules (Zeni et al., 2023; Igashov et al., 2024), particle physics (Bogatskiy et al., 2022; Gong et al., 2022; Spinner et al., 2024), biological and medical imaging (Veeling et al., 2018; Bekkers et al., 2018; Winkels and Cohen, 2018; Winkens et al., 2018; Mohamed et al., 2020; de Ruijter and Cesa, 2024; Suk et al., 2024), wireless communication (Hehn et al., 2024), and robotics (Wang et al., 2022a,b,c; Brehmer et al., 2024). The potential of equivariance to improve generalization has also been shown theoretically (Sokolic et al., 2017; Lyle et al., 2020; Elesedy and Zaidi, 2021; Sannai et al., 2021; Behboodi et al., 2022; Petrache and Trivedi, 2024).

At the same time, equivariant architectures are often more complex than non-equivariant architectures, which can make training more challenging. Some researchers believe that

equivariant architectures are more difficult to scale up, but to the best of our knowledge there has been little systematic study into this. However, recent impactful works on protein folding (Abramson et al., 2024) and conformer generation (Wang et al., 2023) found that equivariant architectures did not offer any benefits and opted for non-equivariant models and data augmentation instead.

One symmetry that is important in many scientific and industrial applications is the group E(3) of isometries of Euclidean space. It consists of translations, rotations, and reflections. This group is the focus of our investigation.

As an E(3)-equivariant architecture, we use the Geometric Algebra Transformer (GATr) (Brehmer et al., 2023). It stands out with two features. First, GATr uses multivectors from projective geometric algebra as representations, in addition to the usual real scalars used in most of machine learning. These multivectors are 16-dimensional objects that can represent various geometric primitives, including absolute positions in space, lines and planes, as well as translations and rotations. Geometric algebra representations power a number of network architectures that were recently proposed (Brandstetter et al., 2022; Ruhe et al., 2023b,a; Brehmer et al., 2023; de Haan et al., 2024; Spinner et al., 2024; Zhdanov et al., 2024; Liu et al., 2024a,b). Second, GATr is a transformer. It processes inputs in the form of a set of tokens. Pairwise interactions are not computed through local message passing, as in many other E(3)-equivariant architectures, but through an equivariant dot-product mechanism that is compatible with efficient implementations like FlashAttention (Dao et al., 2022). We choose GATr as the equivariant model for our scaling investigation because of this similarity to the baseline transformer.

On a side note, the second symmetry group that is relevant to the problem we study is that of permutations of the inputs. Both standard transformers (Vaswani et al., 2017) and GATr (Brehmer et al., 2023) are equivariant to it.

Appendix B. Problem setup

B.1. Benchmark problem

Desiderata For our empirical scaling study, we would like a benchmark task with a low floor and high ceiling: a small model trained on few samples should perform poorly, while a large model trained on many samples should score orders of magnitude better. To study data scaling, we need a large number of training samples. To study equivariance, we look for a geometric problem in which the symmetries and representations are known and exact.

Rigid-body modelling problem We choose a rigid-body modelling problem as our benchmark. Extended, rigid objects are initialized at some position, orientation, and velocity; they then interact with each other under gravity and collisions. Concretely, the inputs of the network consist of a set of triangular meshes for two time points $t = t_0, t_0 + \Delta t$, and the task is to predict all mesh vertices at time $t = t_0 + 2\Delta t$. As a loss function and evaluation metric, we use the mean squared error of the predicted mesh vertex positions.

This problem satisfies all desiderata for our study. Rigid-body interactions are known to be challenging to model: collisions are difficult to detect, since they do not usually occur at or near vertices; the forces acting during a collision are nearly discontinuous (Bauza and Rodriguez, 2017; Pfrommer et al., 2021; Allen et al., 2022). Synthetic data can be generated

cheaply with physics simulators. Finally, the physics of the process is clearly equivariant under E(3), provided that the direction of gravity is treated as a feature and rotated along with the scene.

Dataset We construct a dataset of rigid-body interactions following a proposal by Allen et al. (2022) (who, unfortunately, did not release their dataset or code). We use the Kubric simulator (Greff et al., 2022), which is based on the PyBullet physics engine (Coumans and Bai, 2016–2024). We recreate the MOVi-B dataset used by Allen et al. (2022) as best as we can, using parameters from their paper and private communication.

Our dataset consists of $4 \cdot 10^5$ trajectories, each consisting of 96 time steps. Each trajectory includes between 3 and 10 objects, each consisting of between 98 and 2160 mesh faces. The average number of total mesh faces in a scene is 5470.

B.2. Models

In selecting architectures, our main objective is not to achieve state-of-the-art results on the particular rigid-body benchmark problem we chose. That would lead us to highly problem-specific architectures (Allen et al., 2022; Rubanova et al., 2024). Instead, we aim for general-purpose architectures that are applicable to broad classes of problems.

Baseline architecture The transformer architecture (Vaswani et al., 2017) has become the de-facto standard across a wide range of machine learning tasks. It is versatile with respect to the input data, propagates gradients effectively, and scales well to large model sizes and input tokens. Most scaling studies have focused on transformers as well. We therefore use a standard pre-LayerNorm (Baevski and Auli, 2018) transformer with multi-query attention (Shazeer, 2019) as our non-equivariant baseline architecture.

We represent each mesh face as a token and the positions and velocities of vertices with random Fourier features (Tancik et al., 2020), which improved performance in initial tests.

Note that even this general-purpose architecture is not at all "free from inductive biases", in fact it is even equivariant with respect to one of the symmetries of our problem: that of permutations of the input tokens. In this respect, there is no difference between the two architectures, and we do not compare to any models that are not permutation-equivariant.

Equivariant architecture For the E(3)-equivariant architecture, we again look for broad applicability (at least within the class of E(3)-symmetric problems). In addition, we would like the architecture to be as structurally similar to the transformer, to isolate the effects of equivariance on scaling as well as possible. We therefore opt for the (to the best of our knowledge) only E(3)-equivariant architecture that is based on dot-product attention with unlimited receptive fields, and which also otherwise follows the transformer blueprint closely: the Geometric Algebra Transformer (GATr) (Brehmer et al., 2023).

Again, we represent each mesh face as a token. GATr uses geometric algebra representations in addition to the usual scalar channels, and we can represent the geometric properties of a mesh face in these geometric representations.

Hierarchical attention While we focus on general-purpose architectures, we find that both models benefit from two minor modifications to the transformer blueprint. First, we use a novel *hierarchical attention* mechanism, in which multiple attention heads use different

attention masks: half of the heads are restricted to attend only to mesh faces in the same object, while the other half attends to all tokens (mesh faces). This allows us to embed awareness of the mesh structure into the transformer architecture, while preserving the efficiency of dot-product attention.

Enforcing object rigidity Second, we enforce *object coherence and rigidity* when computing the outputs. The transformer first outputs a translation vector and a rotation quaternion for each mesh face. These are averaged over each object, resulting in a translation vector and a rotation for each rigid object. These E(3) operations are then applied to the input meshes, translating and rotating them as determined by the network. In preliminary experiments, enforcing object rigidity in this way improved performance substantially compared to directly predicting the positions or velocities of mesh vertices. We also experimented with outputting and exponentiating elements of the Lie algebra for each object, but found that that worked marginally worse.

Hyperparameters We tune the hyperparameters of both models manually. For both the baseline and equivariant transformer, we define a one-parameter family of hyperparameters, fixing the relation between the number of layers, attention heads, and channels to be linear. Our architectures are shown in Tbl. 1. Notably, we find that the equivariant transformer benefits from a more narrow architecture, which may be evidence of the expressivity of its multivector channels.

Hyperparameter	Baseline	Equiv.
Attention blocks	$2n$	$2n$
Scalar channels	$64n$	$4n$
MV channels	–	n
Attention heads	$2n$	$2n$
Scalars per key, query, value	64	8
MV per key, query, value	–	2
Hidden scalar channels in MLP	$128n$	$8n$
Hidden MV channels in MLP	–	$2n$

Table 1: Architecture hyperparameters as a function of a model size parameter n. The equivariant architecture is less wide, but part of their channels are 16-dimensional multivector (MV) channels, which can express a variety of geometric primitives (Brandstetter et al., 2022; Ruhe et al., 2023b; Brehmer et al., 2023; Ruhe et al., 2023a).

Optimization We train all models with the Adam optimizer, annealing the learning rate over the course of training from an initial value of $5 \cdot 10^{-4}$ on a cosine schedule. For experiments with small FLOP budgets of less than 10^{18} nominal FLOPs, we find that this learning rate can be too small. This is in line with other works that find larger learning rates beneficial for smaller compute budgets, for instance Dubey et al. (2024). We therefore repeat these experiments with a higher learning rate of 10^{-3} or even $2 \cdot 10^{-3}$, depending on the compute budget, and in the end report the better result. For simplicity, we use the same batch size of 64 samples (or on average $3.5 \cdot 10^5$ tokens) for all experiments, even though this does not maximize GPU utilization and thus FLOP throughput. Early stopping is used in all experiments.

B.3. Scaling-law analysis

Experiments We perform two series of experiments. First, we study the scaling with compute, in the (practically) infinite-data setting. We vary a training compute budget over three orders of magnitude, between 10^{16} and 10^{19} FLOPs. For each FLOP budget we consider, for both the baseline and the equivariant transformer, we perform multiple

experiments, each with a different trade-off between model size N and training length D. This requires understanding the relation between N, D, and the total training FLOPs; we discuss this in the following section.

Second, we study the scaling with training data, fixing the training compute budget, the model size, and the number of training tokens. For both models we choose settings that performed compute-optimally in the first series of experiments for a compute budget of 10^{18} nominal FLOPs. The number of unique samples in the dataset is varied over five orders of magnitude, from $2 \cdot 10^6$ tokens to $2 \cdot 10^{11}$. The lower end of this scan corresponds to training for $6 \cdot 10^5$ epochs, while every sample is seen only once on the upper end of this scan. For each of these settings, we train a baseline transformer, an equivariant transformer, and a baseline transformer trained with data augmentation, in which symmetry transformations are applied to the samples, independently for each epoch.

Counting FLOPs Setting up our experiments (see above) and analyzing the scaling with compute both require knowing the relation of the total number of training FLOPs $C(N, D)$ and the model size N as well as training tokens D. This relation will be different for the baseline and equivariant transformer.

Following Kaplan et al. (2020) and Hoffmann et al. (2022), we perform this FLOP counting in the limit of where the model parameters are much larger than the sequence length, which in turn is much larger than 1. In this case, the training compute is dominated by the linear maps. For both out models, we find

$$C(N, D) \approx \xi N D \,, \tag{3}$$

where ξ is an architecture-dependent constant.

For the baseline transformer, famously $\xi = 6$ (Kaplan et al., 2020). For the equivariant transformer, the value of ξ depends on the ratio of scalar and multivector channels: a model with only scalar channels would also have $\xi = 6$, while a pure-multivector model would has more weight sharing and thus a higher FLOPs-per-parameter ratio $\xi = 6 \cdot 16^2/9 \approx 171$. For the hyperparameters we use during our scaling study, we find $\xi \approx 61.2$.

Note that these *nominal FLOPs* do not necessarily correspond to the actual compute required to train the model. For one, the assumed hierarchy between the model parameters and the sequence length is not always satisfied. Second, our implementations of the models may not be able to fully utilize the GPUs. We observe this in particular for small models and for the implementation of the equivariant transformer, which features many small operations and faces CPU bottlenecks. The practical training loop adds overhead due to inter-GPU communication, data loading, logging, checkpoint saving, validating, and so on. In our experiments, two models with the same nominal FLOP count would differ by as much as an order of magnitude in real training duration.

So why do we still analyze models in terms of the nominal FLOPs? While they are an imperfect measure, they do not depend on the implementation and hardware environment, and we believe they are still the best predictor of the theoretically achievable compute cost after sufficient optimization and at scale.

Scaling-law ansatz We model the scaling with compute quantitatively by fitting a scaling law to all of our experiments. Following Kaplan et al. (2020), we model the test loss L as a power law in the model parameters N and the training duration D, measured in tokens, as

given in Eq. (3).

The parameter E represents the irreducible loss that even a perfect model achieves. Unlike in language or image modelling tasks, there is no clear reason to expect such an irreducible error of practically relevant size for the deterministic physics task we are using as a benchmark. We treat the choice of whether to include E as a fit parameter or fix it to zero as a hyperparameter and choose it through cross validation, as we will describe below.

For the scaling with the size of the training data set, we do not find a scaling law that convincingly describes our experiments. Our attempts at fitting (Muennighoff et al., 2023)'s data-constrained scaling law to our data did not result in a good fit. We therefore refrain from discussing the functional form for this direction of scaling, and will focus on scaling with compute for the remainder of this section.

Scaling-law fit Following Hoffmann et al. (2022), we fit the scaling-law parameters (A, B, E, α, β) separately for each architecture by minimizing the Huber loss (Huber, 1992) between the predicted and observed log loss values,

$$\sum_{\text{experiments } i} \text{Huber}_\delta\Big(\log \hat{L}(N_i, D_i) - \log L_i\Big)\,. \tag{4}$$

Here δ is a hyperparameter, we choose it based on cross-validation, as we describe in a bit. We minimize this loss with the L-BFGS optimizer (Liu and Nocedal, 1989), starting from a grid of initializations.

Scaling-law hyperparameters The scaling-law fit depends on two hyperparameters: whether we include the offset E as a fit parameter and the value of δ. We determine both through leave-one-out cross-validation, performing scaling-law fits on all but one experiment and evaluating the error $|\log \hat{L}(N_i, D_i) - \log L_i|$ on the left-out experiment. In this way, we choose fixing $E = 0$ and $\delta = 0.001$, though the qualitative fit results are not sensitive to these choices.

Compute-optimal performance From a scaling law as in Eq. (1) and a FLOP function as in Eq. (3), we can derive the compute-optimal model size $N^*(C)$ and the compute-optimal training duration $D^*(C)$ as a function of the FLOP budget C as

$$N^*(C) = \frac{G}{\xi^a} C^a \quad \text{and} \quad D^*(C) = \frac{1}{G\,\xi^b} C^b\,, \tag{5}$$

where $G = (\frac{\alpha A}{\beta B})^{1/(\alpha+\beta)}$, $a = \beta/(\alpha+\beta)$, and $b = \alpha/(\alpha+\beta)$ (Hoffmann et al., 2022).

The optimal loss achievable for a given FLOP budget is then

$$L^*(C) = \hat{L}(N^*(C), D^*(C)) = E + \frac{F}{C^\gamma} \tag{6}$$

with $F = AG^{-\alpha}\xi^\gamma + BG^\beta\xi^\gamma$ and $\gamma = \frac{\alpha\beta}{\alpha+\beta}$.

Uncertainties No realistic scaling study directly measures the *optimal* model performance as a function of some parameters. Reasons for sub-optimality include the choice of hyperparameters, stochasticity in initialization and training, choosing a scaling-law ansatz that does not include the true functional form, and finite sampling of the space of model capacities

and training tokens. We estimate the effect of the latter with a nonparametric bootstrap, similar to Hoffmann et al. (2022). From 10^4 bootstraps, we construct 95 % confidence intervals on the scaling law coefficients as well as on any derived predictions, using the empirical (or basic) bootstrap method.

Appendix C. Results

C.1. Scaling with compute

We first focus on the limit of (essentially) infinite training data and study the model performance as a function of model size N and training tokens D.

Scaling law	Param.	Baseline			Equivariant		
		Central	Lower	Upper	Central	Lower	Upper
$\hat{L}(N,D) = A/N^\alpha + B/D^\beta$ (1)	A	**1.27**	0.443	4.69	**0.000272**	0.000155	0.000655
	B	**0.202**	0.0118	0.362	**516**	53.2	639
	α	**0.909**	0.825	1.02	**0.344**	0.287	0.424
	β	**0.379**	0.259	0.404	**0.739**	0.641	0.752
$N^*(C) \propto C^a$ (5)	a	**0.294**	0.219	0.308	**0.682**	0.608	0.716
$D^*(C) \propto C^b$ (5)	b	**0.706**	0.692	0.781	**0.318**	0.284	0.392
$L^*(C) = F/C^\gamma$ (6)	F	**1.03**	0.125	1.86	**0.132**	0.0476	0.64
	γ	**0.268**	0.213	0.283	**0.235**	0.209	0.272

Table 2: Scaling-law coefficients. In addition to the central values, we show the 95% confidence intervals from a nonparametric bootstrap.

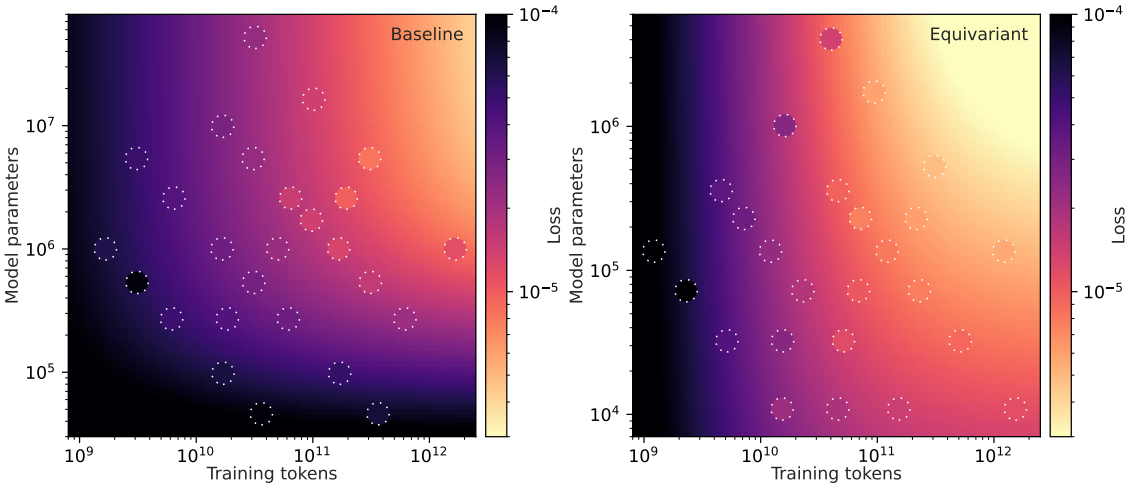

Figure 3: **Test loss** (dotted circles) **and scaling-law predictions** (background colour) **as a function of model size and training tokens**. Left: equivariant transformer. Right: non-equivariant transformer. In both cases, we observe good agreement of model performance and scaling-law fit.

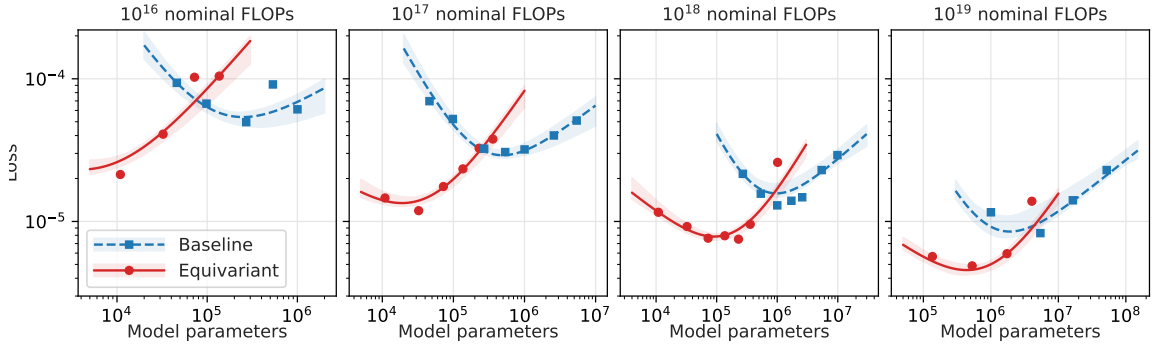

Figure 4: **Model performance at different training compute budgets** (panels) as a function of the model size. We show our experiments (dots) and the predictions of our scaling-law fit (lines). The scaling-law fit describes the measurements well.

Scaling laws We fit the scaling law of Eq. (1) with $E = 0$ to these experiments. For the baseline transformer, we find coefficients

$$\hat{L}(N, D) = \frac{1.27}{N^{0.909}} + \frac{0.202}{D^{0.379}} , \tag{7}$$

The equivariant model yields

$$\hat{L}(N, D) = \frac{2.7 \cdot 10^{-4}}{N^{0.344}} + \frac{516}{D^{0.739}} . \tag{8}$$

Confidence intervals are provided in Tbl. 2.

These two models look quite differently, which will implications for the optimal allocation. We will get to that later.

Fit quality First, we show how well these fitted scaling laws describe the data. In Fig. 3, we show all experiments in the near-infinite-training-data setting. In Fig. 4, we focus on four budgets and show all experiments performed for either. In both sets of figures we compare the observed loss values to the predictions from the scaling laws.

The scaling laws capture the relation between model parameters, training compute, and loss well. There are no glaring deviations, although the power law underestimates the loss for the largest equivariant models and for one baseline outlier.

Scaling with compute Next, we analyze the model performance and its scaling with compute. Rather than focusing on the scaling with model size and training tokens separately, we first study the optimal model performance as a function of the training compute budget, as given by Eq. (6). We show the empirical compute-loss measurements and the derived optimal compute-loss relationship in Fig. 1.

For any given compute budget, the equivariant transformer outperforms the baseline. Its loss is better by an approximate factor of 2, independently of the compute budget. This translates to a scaling law with the same exponent (within the uncertainties we measure), but a smaller prefactor.

Optimal allocation of compute From the scaling laws we can also derive the optimal allocation of a given computational budget to the parameter count and training duration, see Eq. (5). We show our results for both models in Fig. 5.

We find that a compute-optimal equivariant transformer has less parameters than a compute-optimal baseline transformer. This is expected because the equivariant transformer performs more compute per parameter.

Perhaps more surprising is that the optimal trade-off depends on the compute in a different way for the two models. We find that for a regular transformer, one should scale training tokens more steeply than model size. For the equivariant model, we find the opposite trend: one should put additional compute more in the model size than the training tokens. The compute-optimal model sizes thus become more similar for larger compute budgets.

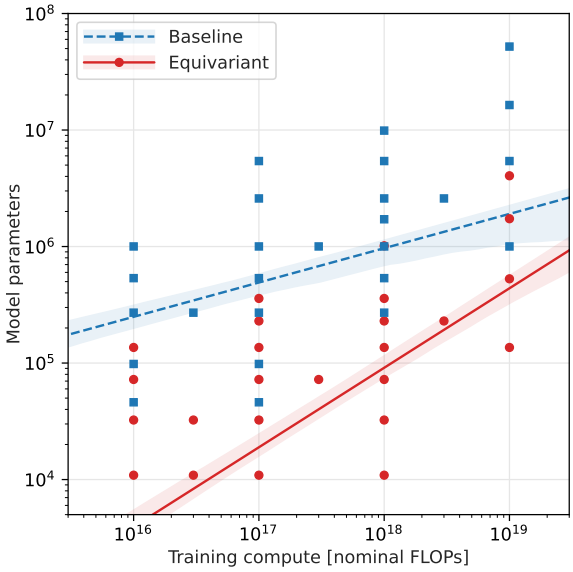

Figure 5: **Optimal parameter allocation**. We show the compute-optimal model size as a function of the training compute budget for the equivariant transformer (red line) and the non-equivariant transformer (blue line). The dots indicate for which compute budgets and model sizes we ran experiments. The equivariant architecture requires smaller models to achieve a compute-optimal performance, but this gap closes for bigger compute budgets.

The reason for these different trade-offs is not obvious. One possible reason is that the baseline transformer, the more mature architecture, may have a better initialization scheme and thus require less training steps to reach a good performance.

Another possible explanation is linked to the internals of the equivariant transformer architecture. The GATr model we use can express certain primitives very efficiently: the free movement or the gravitational acceleration of rigid bodies can be represented with few multivector channels, largely thanks to the geometric product operation that is integrated into the architecture. This explains why the architecture can achieve a good performance with very few parameters. However, lowering the loss further requires precise collision detection and modelling. These need substantially more computational operations and a substantial amount of scalar channels, similar to the non-equivariant transformer. This could explain why at a larger compute budget, a model size closer to that of the baseline transformer is compute-optimal.

C.2. Scaling with data

Next, we turn to the scaling with training data for a fixed training compute budget. In Fig. 2 we show the test loss as a function of the number of unnique training tokens. We compare baseline and equivariant transformers, each using a compute-optimal model size and training tokens for a training compute budget of 10^{18} nominal FLOPs.

The right end of these curves corresponds to the infinite-data limit considered in the previous section. Here we again see that the equivariant transformer has a performance benefit over the baseline when using the same compute budget. Moving to smaller training sets, this gap widens substantially, confirming the expectation that equivariance improves data efficiency.

In Fig. 2 we also show results for a baseline transformer model trained with data augmentation. As expected, data augmentation does not make a difference when training for a single epoch. However, it drastically improves the performance in the small-data regime: when training for thousands of epochs, data augmentation makes a baseline transformer as data-efficient as an equivariant model.