Transformers for Scattering Amplitudes

Garrett William Merz*
Data Science Institute
University of Wisconsin-Madison
garrett.merz@wisc.edu

Tianji Cai*
SLAC National Accelerator Laboratory
tianji@slac.stanford.edu

François Charton*
FAIR, Meta
fcharton@meta.com

Niklas Nolte
FAIR, Meta
nolte@meta.com

Matthias Wilhelm
Niels Bohr Institute
University of Copenhagen
matthias.wilhelm@nbi.ku.dk

Kyle Cranmer
Data Science Institute
University of Wisconsin-Madison
kyle.cranmer@wisc.edu

Lance Dixon
SLAC National Accelerator Laboratory
lance@slac.stanford.edu

Abstract

We pursue the use of Transformers to extend state-of-the-art results in theoretical particle physics. Specifically, we use Transformers to predict the integer coefficients of large mathematical expressions that represent scattering amplitudes in planar $\mathcal{N} = 4$ Yang-Mills theory, which is a quantum field theory closely related to the theory that describes Higgs boson production at the Large Hadron Collider. We first formulate the physics problem in a language-based representation that is amenable to Transformer architectures and standard training objectives. Then we show that the model can achieve high accuracy ($> 98\%$) on two tasks.

1 Introduction

Particle physics at the energy frontier is entering an exciting new era of high-precision experiments, ushered in by the high-luminosity upgrade of the Large Hadron Collider (LHC). Exploiting the full physics potential of this data will require substantial improvements in the precision of the theoretical predictions of the Standard Model (SM).

A key ingredient for these predictions are quantities called scattering amplitudes. The conventional way to compute scattering amplitudes is based on Feynman diagrams, which graphically organize a series of terms in a perturbative expansion. Higher-order terms in this expansion require Feynman diagrams with loops of virtual particles, whose unobserved momenta are latent variables that must be integrated over. Unfortunately, the number of Feynman diagrams grows factorially with the loop order; thus, this paradigm for organizing the calculation quickly becomes intractable. Currently, no LHC collider processes that involve quantum chromodynamics (QCD) are known at four-loop order, and only very few are known at three-loop order [2, 3, 13, 16, 15, 14, 26, 17, 18].

An alternate paradigm for the calculation, known as the *amplitude bootstrap* [10, 6, 5], has seen substantial successes in the simplest theoretical cousin of QCD, called *planar $\mathcal{N} = 4$ super-Yang-Mills theory* (SYM) [4]. As a theoretical laboratory or model system for QCD, SYM allows us to see much further into the perturbative expansion than QCD. For example, a class of SYM amplitudes was

*Equal contribution

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

recently computed to *eight* loops [11, 12] using the amplitude bootstrap approach. These amplitudes (referred to as three-gluon form factors $\mathcal{F}_{\texttt{3gFF}}$) are the SYM analog of the LHC Higgs production process $gg \to Hg$, which is known only to two loops [19].

The amplitude bootstrap leverages the rich, yet highly constrained, analytical structure of amplitudes that arises from the particular recurrent features of the Feynman integrals involved. The form of the answer is known and the finite-dimensional solution space can be constrained through a large system of linear relationships, so that a unique answer remains. Abstracted from the physics context, the problem has the flavor of an integer programming problem with a solution that is hard to find, but easy to check analytically.

In this paper, we investigate two approaches that use Transformers to help solve this problem. First, we train models to predict elements of the solution at a given loop order from elements of the same loop that have already been calculated. Using known linear relations, all model predictions can be readily verified. Since there is only one valid solution at each loop order, this problem is similar to low-rank matrix completion [22], where Transformers have been used in other works [25]. Second, the mathematical structure of the problem suggests that the solution at L loops may be a function of the integer coefficients from the solution at $L-1$ loops. We train Transformers to discover this implicit functional relation in order to facilitate the computation of solutions using the bootstrap method, and, potentially, to shed new light on the underlying structure of SYM.

2 Related work

Our work is most akin to [9], which uses Transformers to simplify polylogarithms—complicated mathematical expressions arising from high-loop order amplitude calculations. However, this approach uses Transformers to translate a given polylogarithmic expression into a shorter but mathematically equal form, and does not attempt our more challenging task of solving the "bootstrap", i.e. predicting which polylogarithms give the result of a particular scattering process.

Other recent works that leverage deep learning to tackle *analytical* calculations in theoretical physics include, among others, [1] which uses a sequence-to-sequence model to symbolically compute the squared amplitude of a particle interaction, and [21] which invokes deep reinforcement learning to explore the landscape of string vacua.

Our methodology is also closely related to those using Transformers for *symbolic mathematical data*. [24] teaches Transformers to perform mathematical tasks such as solving differential equations. [8] extends this approach to recurrent sequences; such sequences connect terms in scattering amplitudes across loop orders. [7] uses a comparable approach to solve linear algebra tasks such as eigenvector decomposition and matrix inversion, which share many structural similarities with the existing amplitude bootstrap method.

3 Representing scattering amplitudes as language

Many amplitudes in SYM – including the aforementioned three-gluon form factor – can be expressed in terms of generalized polylogarithms of weight w. At L-loop order, the weight, i.e. the number of integrations, is $2L$. The analysis of such a function $\mathcal{F}^{(L)}$ typically involves another mathematical object known as the *symbol*, which encodes information about its derivatives [20],

$$\mathcal{S}[\mathcal{F}^{(L)}] = \sum_{l_{i_1},\ldots,l_{i_{2L}} \in \mathcal{L}_n} F^{l_{i_1},\ldots l_{i_{2L}}}\, l_{i_1} \otimes \cdots \otimes l_{i_{2L}}\,. \tag{1}$$

where $\mathcal{L}_n = \{l_1, \ldots, l_n\}$ is the *symbol alphabet* containing n *letters*, which are in turn functions of the particles' momenta, and $F^{l_{i_1},\ldots l_{i_{2L}}}$ is an $2L$-fold tensor of integers, many of which are zero. Thus, a candidate solution for the symbol at L-loop order can be represented by n^{2L} integers, forming a $2L$-dimensional sparse tensor, with each sequence of $2L$ letters a key or index into this tensor.

Symbols and their elements for the three-gluon form factor. The three-gluon form factor $F_{\texttt{3gFF}}$ has the simplest alphabet of amplitudes known to high loop orders, which makes it the prototype amplitude for the first application of machine learning methods. It has only six letters (i.e. $n = 6$):

$$\mathcal{L}_{\texttt{3gFF}} = \{a, b, c, d, e, f\}\,, \tag{2}$$

which are Lorentz-invariant functions of the gluons' four momenta, and exhibits a dihedral symmetry:

$$\text{cycle: } \{a,b,c,d,e,f\} \to \{b,c,a,e,f,d\} \text{ , and flip: } \{a,b,c,d,e,f\} \to \{b,a,c,e,d,f\} \quad (3)$$

To give a concrete example, the symbol of $\mathcal{F}^{(1)}_{\texttt{3gFF}}$ at 1 loop order contains only 6 non-vanishing terms,

$$\mathcal{S}[\mathcal{F}^{(1)}_{\texttt{3gFF}}] \quad = \quad (-2)\Big[b\otimes d + c\otimes e + a\otimes f + b\otimes f + c\otimes d + a\otimes e\Big]. \quad (4)$$

Here, the word `bd` (shorthanded for $b\otimes d$), a key into the tensor F^{l_1,l_2}, maps onto the value -2, while the word `ab` maps onto 0. In the general case, $\mathcal{S}[\mathcal{F}^{(L)}_{\texttt{3gFF}}]$ is a sum of 6^{2L} *elements*, pairs of keys, i.e. sequences of $2L$ letters, and integer coefficients. Most of the coefficients are zero (Table 1), and most zeroes (the 'trivial zeroes') are accounted for by two rules:

- **adjacency rule:** any key including one of the subsequence `ad`, `da`, `de` (or their dihedral images) has zero coefficient
- **prefix/suffix rule:** any key beginning with `d`, `e` or `f` or ending with `a`, `b` or `c` has zero coefficient

Loop	1	2	3	4	5	6	7	8
Total (6^{2L})	36	1,296	46,656	$1.7\cdot10^6$	$6.1\cdot10^7$	$2.2\cdot10^9$	$7.8\cdot10^{10}$	$2.8\cdot10^{12}$
W/O trivial zeros	6	102	1,830	32,838	589,254	$1.1\cdot10^7$	$1.9\cdot10^8$	$3.4\cdot10^9$
Total nonzero	6	12	636	11,208	263,880	$4.9\cdot10^6$	$9.3\cdot10^7$	$1.7\cdot10^9$

Table 1: Elements in symbol for loops 1 to 8.

4 Completing symbols by predicting coefficients from their keys

In this first set of experiments, we train sequence-to-sequence Transformers [27] to predict coefficients from their keys at a given loop order ($L = 5$ and 6). Models are trained on a fraction of the symbol and tasked to predict the rest. All tasks are framed as translation problems: input and output are encoded as sequences, and the model is trained to minimize the cross-entropy with the ground-truth solution. At the end of each epoch (300,000 training examples), the model is evaluated on a held-out test set.

For L loops, keys are encoded as sequences of $2L$ letters, e.g., `'a, a, b, d, d, c, e, e'`. Coefficients are encoded as sequences of digits in base 1000, e.g., 12334 as `'+, 123, 344'`. Models have between 1 and 4 layers, in the encoder and decoder, 512 dimensions and 8 attention heads. The optimizer is Adam, with a learning rate of 10^{-4}. All models are trained on a single NVIDIA V100 GPU, with 32 GB of memory.

The purpose of these experiments is to demonstrate the capability of the Transformer to "complete" a symbol: filling in the remaining elements once a part of the symbol has been constructed.

In a first series of experiments, we train 1-layer Transformers to **predict whether the coefficient of an element at loop 5 and 6 is zero**. Models are trained from balanced sets, with 50% non-zero and 50% zero elements (both trivial and non trivial), and tested on 10,000 elements (50% zero) not in the training set. At 5 loops, after being trained on 300,000 elements (57% of the symbol, one epoch), the model correctly predicts 99.96% elements in the test set (all test cases but 4). At 6 loops, the model predicts 99.91% of the test set after 300,000 elements (3% of the symbol), and 99.97% after 600,000 (6% of the symbol).

In a second set of experiments, we train 2-layer Transformers to **predict non-zero coefficients from their keys**. Coefficients are encoded as sequences of digits in base 1000. All models are tested on 100,000 elements not in the training set. At 5 loops, the model is trained on 163,880 elements (62% of the symbol). After one epoch, it correctly predicts 47% of the coefficients from the test set. Accuracy is 96% after 6 epochs, 99% after 15 and 99.9% after 58. Surprisingly, the absolute values of coefficients prove easier to learn than their sign (a separate token). After one epoch, 92% of absolute values are correctly predicted, but the signs are only predicted at about chance level (51%). After 3 epochs, 99% of absolute values are predicted correctly, but only 75% of signs.

At 6 loops, the model is trained on 1,000,000 elements (20% of the symbol), and tested on 100,000 elements not in the training set. After 188 epochs, the best model (of four) correctly predicts 98.9% of the coefficients in the test set. The learning curves in Figure 2 reveal two qualitative phases. During the first 20 epochs, the model learns the absolute values of coefficients. Then, accuracy saturates around 50%, while the model predicts the absolute values of coefficients with more than 95% accuracy, and their sign at chance level. From epoch 40 to 60 the model learns to predict the sign of coefficients. 95% of the coefficients in the test set are correctly predicted after 64 epochs, and 98% after 120. These results indicate that Transformers trained on a small fraction of the symbol can predict coefficients from their keys with very high accuracy.

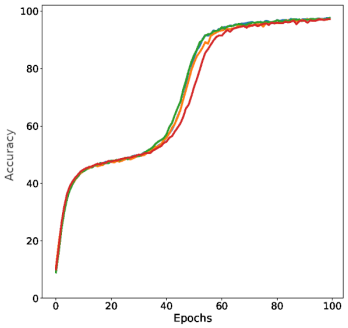

Figure 1: **Learning curves, 6 loops.** 4 model initializations.

We further note that, when trained to predict only the magnitude of the coefficient at 6 loops, the best model is able to do so at 97.7% accuracy after 50 epochs. However, when models are trained to predict only the sign of the nonzero coefficient, they exhibit random guessing behavior after 100 epochs. This suggests that learning the magnitude of the coefficient may be a prerequisite for learning the sign.

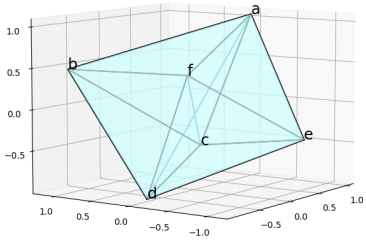

Figure 2: The **leading three PCA components of token embeddings** (77.7% of variance explained) for a 4-layer Transformer trained on 6-loop data for 100 epochs exhibit dihedral symmetry.

Finally, we **extract the learned embeddings of the input tokens** from the embedding layer of the Transformer and plot their leading three PCA components. Dihedral symmetry is clearly manifest in the embeddings – the "octahedron" structure that emerges obeys both cyclic and flip symmetries. In particular, when we restrict ourselves to the leading three PCA components, the angle between embeddings of "forbidden" letter pairs is almost back-to-back, $175^\circ \pm 2^\circ$, and the angles of all triangular faces are $60^\circ \pm 5^\circ$.

This property is not guaranteed – while it appears during training on 6-loop data, no comparable structure emerges when training the same model on 4-loop data, despite the fact that it quickly reaches >99% accuracy. This suggests that this property is likely a function of both model size and task complexity.

5 Predicting the next loop – the strike-two method

We now consider a different problem: direct computation of the L-loop symbol from the coefficients at $L-1$ loops. In practice, for any element E_L at L-loops, we want to discover a list of parent elements at $L-1$ loops, $\mathcal{L}(E_L)$, and a function f, such that the coefficient of E_L is $f(\mathcal{L}(E_L))$. The keys of elements at L loops are sequences of $2L$ letters. We define their *strike-two parents* as the $L(2L-1)$ elements from $L-1$ loops created by striking out two letters from the key (in order). For instance, the six strike-two parents of `aacf` are ~~`a`~~~~`a`~~`cf`=`cf`, ~~`a`~~`a`~~`c`~~`f`=`af`, ~~`a`~~`ac`~~`f`~~=`ac`, `a`~~`a`~~~~`c`~~`f`=`af`, `a`~~`ac`~~`f`~~=`ac and `aa`~~`c`~~`f`~~=`aa.

In these experiments, 4-layer Transformers are trained to **predict non-zero coefficients at** 6 **loops from the coefficients of their** 5**-loop parents**. Model inputs are sequences of 66 integers (encoded in base 1000), outputs are single coefficients, as before. From the symbol, we can create 4.9 million examples (5-loop parents and 6-loop coefficients), but this dataset includes a lot of duplicates. In fact, each example is duplicated 6.26 times on average, mostly because of the dihedral symmetry, but sometimes because unrelated elements have the same parents and coefficients. To avoid contamination between the training and test set, we restrict the data set to 783,500 unique pairs, split into 773,500 training and 10,000 test examples.

After 500 epochs, the model predicts 98.1% of test examples. Learning is fast: 90% accuracy is achieved after 20 epochs, and 95% after 80. In these experiments, the absolute values and signs of coefficients are learned simultaneously. These results suggest that there exist simple formulas for computing coefficients at 6 loops from their strike-two parents at 5 loops. We are not, so far, capable of recovering these formulas, but additional experiments shed light on some of their features (Table 2, see also Appendix A.2 for more detailed results).

First, we investigate whether we can **predict 6-loop coefficients from smaller sets of parents**. To study this, we only strike, in the 6-loop key, tokens that are no more than $k = 1, 2, 3, 5$ positions away from each other: i.e. for $k = 1$, we only strike adjacent tokens, for $k = 2$, for the 4-loop key bccbaeee, we allow bcc~~b~~a~~e~~ee and bccb~~a~~~~e~~ee but not bcc~~b~~ae~~e~~e or bc~~c~~baee~~e~~. This reduces the number of parents to $2kL - \frac{k(k+1)}{2}$, down to 11 for $k = 1$. Our models predict 98.3%, 98.4%, 98.1% and 94.3% of test examples, when $k = 5$, 3, 2 and 1, respectively. Predicting from 21 parents ($k = 2$), instead of 66, has no impact on model performance.

	Accuracy	Magnitude accuracy	Sign accuracy
Strike-two, all parents	98.1	98.4	99.6
Strike-two, $k = 5$	98.3	98.6	99.7
Strike-two, $k = 3$	98.4	98.7	99.7
Strike-two, $k = 2$	98.1	98.3	99.5
Strike-two, $k = 1$	94.3	95.2	98.5
Randomly shuffled parents, all parents	95.2	99.1	96.3
Randomly shuffled parents, $k = 2$	93.5	98.1	95.0
Sorted parents, $k = 5$	93.9	95.4	97.9
Parent signs only	93.3	93.5	99.0
Parent magnitudes only	81.8	98.4	83.2

Table 2: **Global, magnitude and sign accuracy.** Best of four models, trained for about 500 epochs.

Second, we experiment with the order of parents. If parent coefficients (i.e. model inputs) are randomly shuffled, the model still achieves 95.2% accuracy (93.5% with 21 parents, i.e. $k = 2$). When parents are sorted in increasing order, it achieves 93.9% for $k = 5$. This suggests that the formula for computing 6 loops from 5 loops is mostly unaffected by a permutation of its variables.

Finally, we note that 6-loop coefficients can still be predicted when only the sign of their parents (-1, 0 or 1) is provided as input to the model. Such models achieve 93.3% accuracy, and correctly predict the sign in 99% of test cases. Models trained on the absolute values of the parents predict the magnitude of the coefficients with 98.4% accuracy, about the same level as models trained on full parent coefficients. On the other hand, the sign of the coefficient proves harder to learn without the sign of the parents. However, when these are shuffled or sorted in ascending order (i.e., all -1s, all zeroes, then all $+1$s), the model is totally unable to learn- in other words, we can drastically reduce information about the values of the strikeout parents or their ordering and still recover the full coefficient, but we cannot do both simultaneously.

We note that this behavior may be an artifact of the way the strikeout experiment is constructed. Because certain letter adjacency conditions imply zeroes, an ordered list of zero and nonzero parents implicitly encodes some information about the original letter structure of the word. From previous experiments, we know that this information can be used to reconstruct the coefficient. In future strikeout experiments, we may wish to take this effect into account in order to better model the relationships between words across loop orders.

6 Discussion / Future work

We have shown that Transformer models are able to predict the coefficients of scattering amplitudes from both "words" and information from their "strike-two parents". Precisely which structures are being learned, however, remains a mystery. In future work, we intend to explore this further, including by evaluating the many linear relations employed in the bootstrap method (in order to determine whether the model learns an algorithm similar to the ones that are currently used by humans), and by training on multiple loops simultaneously (in order to determine whether or not the learned structures are loop-agnostic). While we are hopeful that our approach will be able to augment the bootstrap procedure, one limitation is that, in order to train our models, we currently require a partial ground-truth solution at the loop order we wish to predict.

We see ample opportunities for these models to aid in scientific inquiry in ways that span what Krenn, et al. [23] characterized as *computational microscopes* and *resources of inspiration*, and hope that they may, eventually, come to serve as *agents of understanding*.

Acknowledgments and Disclosure of Funding

The work was supported in part by the U.S. Department of Energy (DOE) under Award No. DE-FOA-0002705, KA/OR55/22 (AIHEP). LD and TC are additionally supported by the U.S. Department of Energy Award No. DE-AC02-76SF00515. MW was supported by the research grant 00025445 from Villum Fonden.

References

[1] A. Alnuqaydan, S. Gleyzer, and H. Prosper. SYMBA: symbolic computation of squared amplitudes in high energy physics with machine learning. *Machine Learning: Science and Technology*, 4(1):015007, jan 2023.

[2] C. Anastasiou, C. Duhr, F. Dulat, E. Furlan, T. Gehrmann, F. Herzog, A. Lazopoulos, and B. Mistlberger. High precision determination of the gluon fusion Higgs boson cross-section at the LHC, 2016.

[3] C. Anastasiou, C. Duhr, F. Dulat, F. Herzog, and B. Mistlberger. Higgs boson gluon-fusion production in QCD at three loops. 2015.

[4] L. Brink, J. H. Schwarz, and J. Scherk. Supersymmetric Yang-Mills Theories. 1977.

[5] S. Caron-Huot, L. J. Dixon, F. Dulat, M. von Hippel, A. J. McLeod, and G. Papathanasiou. Six-gluon amplitudes in planar $\mathcal{N} = 4$ super-Yang-Mills theory at six and seven loops. 2019.

[6] S. Caron-Huot, L. J. Dixon, A. McLeod, and M. von Hippel. Bootstrapping a five-loop amplitude using Steinmann relations. 2016.

[7] F. Charton. Linear algebra with transformers, 2022.

[8] S. d'Ascoli, P.-A. Kamienny, G. Lample, and F. Charton. Deep symbolic regression for recurrent sequences, 2022.

[9] A. Dersy, M. D. Schwartz, and X. Zhang. Simplifying polylogarithms with machine learning, 2022.

[10] L. J. Dixon, J. M. Drummond, and J. M. Henn. Bootstrapping the three-loop hexagon. 2011.

[11] L. J. Dixon, A. J. McLeod, and M. Wilhelm. A three-point form factor through five loops. 2021.

[12] L. J. Dixon, Ömer Gürdoğan, A. J. McLeod, and M. Wilhelm. Bootstrapping a stress-tensor form factor through eight loops. 2022.

[13] F. A. Dreyer and A. Karlberg. Vector-boson fusion Higgs production at three loops in QCD. 2016.

[14] C. Duhr, F. Dulat, and B. Mistlberger. Charged current Drell-Yan production at N3LO. 2020.

[15] C. Duhr, F. Dulat, and B. Mistlberger. Drell-Yan cross section to third order in the strong coupling constant. 2020.

[16] C. Duhr, F. Dulat, and B. Mistlberger. Higgs boson production in bottom-quark fusion to third order in the strong coupling. 2020.

[17] C. Duhr and B. Mistlberger. Lepton-pair production at hadron colliders at N3LO in QCD. 2022.

[18] T. Gehrmann, P. Jakubčík, C. C. Mella, N. Syrrakos, and L. Tancredi. Planar three-loop QCD helicity amplitudes for V+jet production at hadron colliders, 2023.

[19] T. Gehrmann, M. Jaquier, E. W. N. Glover, and A. Koukoutsakis. Two-loop QCD corrections to the helicity amplitudes for $\mathrm{h} \to 3$ partons. 2012.

[20] A. B. Goncharov, M. Spradlin, C. Vergu, and A. Volovich. Classical Polylogarithms for Amplitudes and Wilson Loops. 2010.

[21] J. Halverson, B. Nelson, and F. Ruehle. Branes with brains: exploring string vacua with deep reinforcement learning. *Journal of High Energy Physics*, 2019(6), jun 2019.

[22] F. J. Király, L. Theran, R. Tomioka, and T. Uno. The algebraic combinatorial approach for low-rank matrix completion. 2012.

[23] M. Krenn, R. Pollice, S. Y. Guo, M. Aldeghi, A. Cervera-Lierta, P. Friederich, G. dos Passos Gomes, F. Häse, A. Jinich, A. Nigam, et al. On scientific understanding with artificial intelligence. 2022.

[24] G. Lample and F. Charton. Deep learning for symbolic mathematics, 2019.

[25] N. Lee, K. Sreenivasan, J. D. Lee, K. Lee, and D. Papailiopoulos. Teaching arithmetic to small transformers, 2023.

[26] B. Mistlberger. Higgs boson production at hadron colliders at N3LO in QCD. 2018.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. 2017.

A Additional results

A.1 Predicting coefficients from keys

In this section, we present additional experiments about models trained to predict the coefficients at a given loop order from their key.

Quads and octuples. As the loop number increases, the number of elements in the symbol becomes very large. The 6-loop symbol has about 5 million elements, the 7-loop symbol has 93 million, and the 8-loop symbol 1.7 billion. A compact representation of the symbol can be built by noticing that there exist many linear relations between the coefficients of terms with different suffices. Up to cyclic symmetries ((a,b,c) and (d,e,f)), there are 8 possible "quads", which can be represented by the following suffices: dddd, bbbd, bdbd, bbdd, dbdd, fbdd, dbbd and cddd. In quad representation, we add 8 new letters to represent these 8 endings, and represent all elements in the loop by their $2L-4$ first letters, and their quad letter. There are 391,570 quads in the 6-loop symbol (vs 5 million elements), and 7.3 million quads in the 7-loop symbol (vs 93 million). An even more compact representation can be created by considering the last 8 letters in elements. There are 279 possible octuples, a number that can be reduced to 93 by factoring out cyclic symmetry. There are 16,971 octuples in the 6-loop symbol, 312,463 for 7 loops and 5.6 million for 8 loops.

Learning from quad and octuple data is more challenging for our models, because these compact representations eliminate most of the obvious symmetries in the three-gluon form factor. Yet, models trained on quad data at 6 loops achieve 99% accuracy (98% after 123 epochs), and models for 7 loops achieve 99.1% after 300 epochs, and 96.7% after 200 epochs, when trained from 4 million quads only (55% of the symbol). For octuples, 8-loop symbols are predicted with 95.2% accuracy, after 865 epochs. The training curves still have a two-step shape, where the magnitude is learned first, then the sign.

A.2 Predicting the next loop

Table 3 presents the overall, magnitude and sign accuracy of models trained for up to 700 epochs, for different variations of the strike-two method. Two new sets of experiments are presented. In the sorted unique experiments, all parents are sorted, and duplicates are removed. This proves harder to train, with the best models achieving 83.4% accuracy, after 350 epochs. In the zero/non-zero experiments, all non-zero parent coefficients are encoded as "1".

	Distance	Best Epoch	Train Set Size	Accuracy	Magnitude Accuracy	Sign Accuracy
Strike-two parents	Full	455	772,500	98.1	98.4	99.6
	5	524	769,060	98.3	98.6	99.7
	3	601	753,352	98.4	98.7	99.7
	2	681	703,869	98.1	98.3	99.5
	1	646	591,510	94.3	95.2	98.5
Shuffled parents	Full	407	4,906,466	95.2	99.1	96.3
	5	376	4,906,466	94.7	98.8	95.8
	3	408	4,906,436	95.1	99.0	96.1
	2	433	4,906,249	93.5	98.1	95.0
	1	442	4,882,510	91.1	92.2	96.5
Sorted parents	Full	389	591,864	91.5	93.8	96.6
	5	432	717,534	93.9	95.4	97.9
	3	514	702,363	93.1	94.6	97.5
	2	453	657,863	90.7	92.4	96.5
	1	459	536,588	76.8	79.3	90.5
Sorted unique parents	Full	329	476,932	79.0	84.0	92.1
	5	355	538,325	83.4	87.4	93.6
	3	349	487,813	79.7	84.1	91.7
	2	287	436,012	73.6	78.5	89.4
	1	314	355,147	57.2	61.9	80.9
Zero / non-zero	Full	304	497,112	40.7	60.1	61.7
	5	229	467,871	35.0	53.8	59.4
	3	93	415,230	22.9	39.1	54.9
	2	18	344,831	10.1	20.1	50.3
	1	1	131,812	0.8	1.4	50.4
Parent signs only	Full	404	748,088	93.3	93.5	99.0
	5	330	739,479	92.4	92.5	99.0
	3	294	711,450	88.2	88.4	98.2
	2	331	653,368	73.3	73.6	94.8
	1	22	468,339	5.8	6.5	64.6
Parent magnitudes only	Full	395	751,675	81.8	98.4	83.2
	5	445	747,187	81.2	98.4	82.4
	3	452	726,554	79.7	98.3	80.9
	2	611	672,561	77.8	97.9	79.4
	1	509	527,843	64.1	93.6	67.7

Table 3: **Global, magnitude and sign accuracy for strike-two method.** We show the best of four models, using 4-layer Transformers with 512 dimensions, 8 attention heads, and the Adam optimizer with a learning rate of 10^{-4}. All models where trained for up to 700 epochs, and we indicate the epoch at which the indicated best accuracy was first achieved.