Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing

Algorithm Learning on OEIS

Bachelor's Thesis

Flavio Schenker

flaviosc@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Peter Belcák, Ard Kastrati
Prof. Dr. Roger Wattenhofer

June 14, 2022

Acknowledgements

I thank my supervisors Peter Belcák and Ard Kastrati for their feedback, creative ideas, constant support and that they always took the time to guide me in the right directions. Furthermore, I would like to thank the Computer Engineering and Networks Laboratory institute for providing me with the computational resources this project required.

Abstract

Ordered lists of integers are of great importance in many fields of science. We look at these sequences from a broader perspective and lay down a set of benchmarking tasks that help machine learning models develop a conceptual understanding of the underlying rules governing such sequences. Additionally, we strengthen that view by introducing a rich set of different model architectures, both neural and classical. Our results showed that despite our models having some level of understanding, our neural models lack the ability to outperform our baseline models. Working towards the long term goal of creating sequence algorithm learning systems, this study gives a general and hierarchical overview of the implications of this broader perspective.

Contents

Chapter 1

Introduction

Integer sequences are often a good natural representation of fundamentally discrete abstractions. These sequences are present in many fields of science, for example describing the number of electrons in an atomic shell following Hund's rules or population growth, abiding the rules of discrete geometric growth.

If the reader were asked to continue the sequence 1,1,2,3,5,... it is certainly tempting to immediately answer 8, originating from the famous Fibonacci sequence $F_i = F_{i-1} + F_{i-2}$. But a more natural answer which arises from chemistry is 9. Representing the number of possible isomers in carbon alkanes.

How can we measure the quality of different suggestions? Especially if the answer does not come from a human but from a artificial intelligence system.

Understanding the structure and underlying rules of such sequences to its full extent is still hard for a machine learning model. Thus, as a long term goal, following the first stepping stones in the work of [1], we aim to develop an algorithm learning system that progresses in the direction of not only memorising a set of examples but rather identify straightforward universal abstractions that explains a given sequence.

To accomplish this goal, in chapter 3 we introduce a set of bench-marking-tasks that assesses the understanding of governing concepts behind the evolution of integer sequences. In sections 4.1 and 4.2 we propose to use several model architectures to give a baseline to our benchmarks. Furthermore, in section 4.3 we explain the methods and metrics applied to evaluate the quality of our models. We develop the dataset in chapter 2, on which they get trained on. For that, we use the famous OEIS-dataset [2] (Online Encyclopedia of Integer Sequences). Finally, we present our results in chapter 5 and in chapter 6 we explore potential future work on this topic.

1.1 Related Work

In [3] the OEIS-dataset was also used for a subset of our bench-marking-tasks. But in contrast to our work, the author utilized three different, but more sophisticated model architectures, whereas we try to give a more general view by applying a wide variety of individual models.

In 2016, the Kaggle data science community launched a competition [4], challenging people in creating a machine learning algorithm that is capable of guessing the next number in a given integer sequence. Likewise using the OEIS dataset. The results submitted to this challenge are in full conjunction to one of our bench-marking-tasks.

On the side of traditional architectures, [5] analyses how Dense-Neural-Networks learn to abstract. This view is in line with our goal of understanding the underlying rules of a given sequence.

1.2 Our contribution

Our contributions are:

- a wide variety of bench-marking-tasks designed to evaluate the model comprehension of conceptual patterns in integer sequences with a clearly established order of difficulty,
- a rich set of different baseline models, both classical and neural, to accomplish seamless experimentation and
- a collection of evaluation metrics tailored to the above tasks to asses model performance.

Chapter 2

Dataset

The dataset we worked with is a collection of two datasets. On the one hand, we have OEIS [2] (Online Encyclopedia of Integer Sequences) and on the other hand, FACT (Finitary Abstraction Comprehension Toolkit). In previous work [1], the OEIS dataset was studied and a broad set of classification methods were introduced. Our effort builds heavily on the results of this work. Described in Chapter 3, these labels are the building blocks of our evaluation data in our machine learning tasks.

2.1 OEIS dataset

OEIS [2] is a collection of noteworthy integer sequences. OEIS was founded in 1966 by Neil Sloane. At the time of writing the dataset contains over 340,000 entries. Each entry holds 18 fields, following the style sheet of [6]. A brief explanation of the 18 fields is given in Appendix B. The length of each sequence is of high variance, ranging from less than 10 entries up to over 200,000. We have processed it specifically for the use in machine learning in line with the license requirements.

2.2 FACT dataset

FACT introduces a dataset consisting of over 3.6 million synthetically generated integer sequences. The dataset was developed by Ard Kastrati and Peter Belcák in 2022. It is an extension to the OEIS dataset while abiding the structure and nature of the stem encyclopedia entries and providing carefully engineered automatic annotations wherever possible. Figure 2.1 gives an overview of the final dataset containing both OEIS and FACT entries.

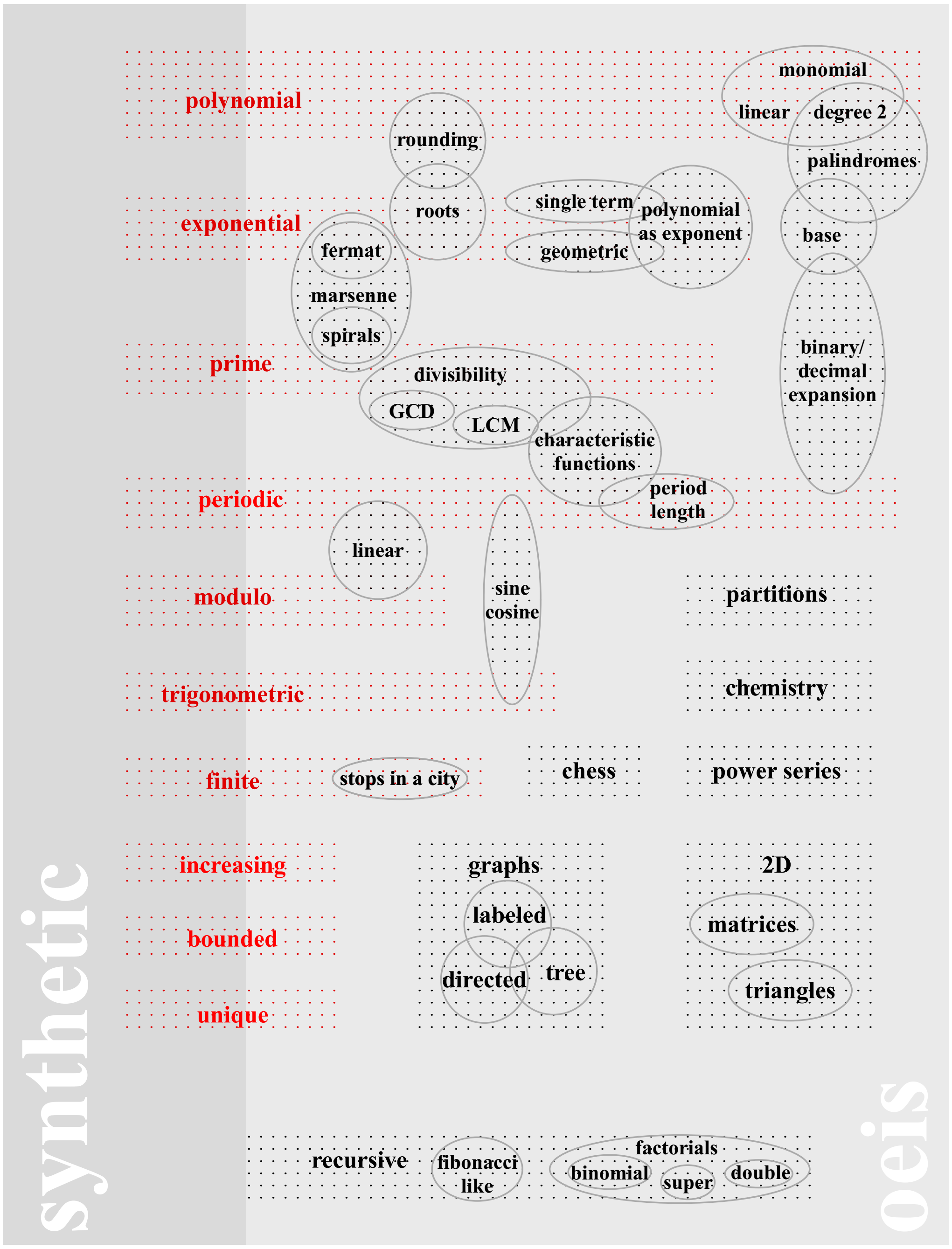

Figure 2.1: The categories in our dataset. It is composed of synthetic and OEIS entries. Each group in synthetic part consists of 500,000 sequences, whereas the size of the OEIS groups vary. Dotted regions represent the main categories identified in the dataset. Ellipses define the sub-categories from our processing step in OEIS. Red dots mark groups that are augmented with synthetic data (and used in our bench-marking setup).

2.3 Structure

2.3.1 Categories

Both datasets introduce 10 categories, namely polynomial, exponential, trigonometric, periodic, finite, modulo, prime, bounded, increasing and unique. Some categories are inclusive, whereas most of them are exclusive. Meaning that a sequence s can either be both a polynomial and increasing but not a polynomial and a exponential at the same time.

2.3.2 Composition and Preprocessing

Table 2.1 gives a detailed overview of the composition of the dataset. In total the OEIS contains 342,304 integer sequences of which we excluded 51,557 entries, which were either invalid or were too large and produced an integer64 overflow. Leaving us with a total of 290,747 OEIS entries. The FACT dataset on the other hand (From here on we will use the notation of synthetic sequences for the FACT dataset and oeis sequences for the OEIS dataset.) contains a total of 3,337,000 integer sequences. No integer64 overflow was produced. Next, to generate valid tensors we cut all sequences to a uniform length of 50 and excluded the shorter once. Leaving us with a total of 174,266 oeis and 3,075,240 synthetic sequences.

	Synthetic	OEIS
Polynomial	500,000	4,808
Exponential	319,402	912
Trigonometric	500,000	4,432
Periodic	338,000	1,789
Finite	418,838	0
Modulo	499,000	9,029
Prime	500,000	36,054
Bounded	1,164,939	41,882
Increasing	2,346,262	73,247
Unique	1,211,607	74,842
Total	3,075,240	174,266

Table 2.1: Dataset structure

2.3.3 Training, Evaluation and Testset

We split the synthetic dataset into three parts. The training-set, containing $\frac{4}{5}$ of all entries, the evaluation- and the test-set containing $\frac{1}{10}$ each. The training-set consists only of synthetic sequences, whereas the evaluation- and test-set were joined with $\frac{1}{2}$ of the oeis dataset respectively.

Chapter 3

Benchmark

With the dataset given, we established a set of 5 different bench-marking tasks. Namely Sequence Classification, Sequence Similarity, Next Sequence-Part Prediction, Sequence Continuation and Sequence Unmasking.

3.1 Task Hierarchy

The benchmark consists of tasks with an increasing order of difficulty over two dimensions: the tasks type and scope. Each task is performed over two different scopes: within and across categories. In within categories, we already know from which category the sequence originates and therefore the first scope is simpler then the second. Nevertheless, our baselines (cf. 4) are oblivious to this information and instead treat the category scope as if nothing was known about the data. Figure 3.1 displays a hierarchical view of the difficulty and scope of our tasks. In the following section we give a detailed explanation of the different types of tasks.

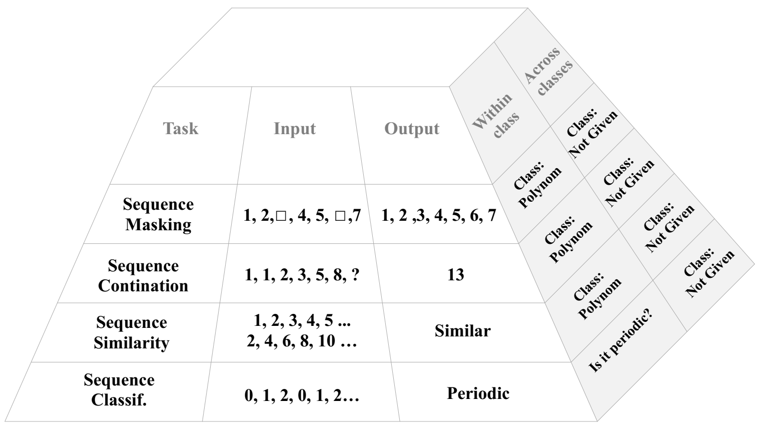

Figure 3.1: Our tasks ordered by the level of difficulty over two dimensions: type and scope.

3.2 Task Types

3.2.1 Sequence Classification

The first task in our benchmark is classifying a sequence to which category it belongs. Since our dataset contains non-exclusive categories we divide this task into two subtasks:

- One-vs-Rest (OvR): Identifying if a sequence belongs to a specific category or not. For each category we provide a balanced dataset of half the sequences who are actually belonging to the specific category and half the sequences sampled from the rest of the categories. Since this is a binary classification task, we use binary-accuracy as a evaluation metric.
- Multiclass classification: In this subtask we seek to predict for every sequence in the dataset to which categories it belongs. This poses the problem of an inbalanced dataset, especially in the OEIS-part. Therefore, we use the macro average F1-score as a metric.

$$\text{macro average F1-score} = \frac{2}{N}\sum_{i=0}^{N}\frac{precision \cdot recall}{precision + recall} \tag{3.1}$$

3.2.2 Sequence Similarity

The sequence similarity task aims to group together sequences that are similar in its type (e.g. agree on the same category) and properties (e.g. both are periodic and bounded). To achieve this, we create an embedding space $\mathcal{Z}$ and train our models to place the sequences in $\mathcal{Z}$ in a meaningful way.
Evaluation is done with the Recall@k score, where $k \cdot m$ candidates for a sequence s are proposed, by sampling k sequences from each of the m categories and then ordering all candidates by their euclidean distance to s in $\mathcal{Z}$. As a metric we finally use top-k-accuracy where we look at the top-k candidates whether one of them is in the same category of s or not.
Our choice of evaluation metrics is motivated in generalizing sequence classification.

3.2.3 Next Sequence Part Prediction

Given two sub-sequences s_1 and s_2, the goal of the next sequence-part prediction task (NSPP) is to determine if the sub-sequence s_2 is a valid continuation of s_1 or not. On each category we take a balanced dataset, whereas across all, we take members from all categories with its counterparts. This is a binary classification task and thus performance is measured in binary-accuracy. This task is strictly more difficult than the similarity task, since the model not only has to understand the key properties of a given sequence but also has to possess the ability to differentiate between feasible and unfeasible combinations.

3.2.4 Sequence Continuation

The forth task in our difficulty hierarchy is to predict the next entries in a given sequence s. This task goes beyond making simple binary decisions but to make active suggestions, and as such this task demands a better understanding of the underlying rules governing sequences, than in Next Sequence-Part-Prediction. We divide this task into two subtasks:

- Single-shot: In single-shot continuation, we predict only a single candidate as the next entry. This is done by regression and as such we use the mean-squared-logarithmic error (MSLE) as a metric.
- Multi-shot: Here we provide more than a single candidate by using the same embedding space $\mathcal{Z}$ from the similarity task, but with a different measurement for comparison. Given to sequences s_1 and s_2 we are interested in the fraction of agreement on the first entries of these two sequences (e.g. [1,2,3,4],[1,2,4,8], agreement is $\frac{2}{4}$).
  For evaluation we use the top-k-root-mean-squared error - the root-mean-squared error (RMSE) across the top k candidates. Given k predictions of a model $\{\hat{y}_i\}_{i\in\{1,\dots,k\}}$ and ground truth y, we define the top-k-RMSE as $\min_{i\in\{1,\dots,k\}} \mathrm{RMSE}(y, \hat{y}_i)$. In other words, given all generated predictions we report the RMSE of the prediction closest to the ground truth.

3.2.5 Sequence Element Unmasking

At the peak of our complexity hierarchy is the sequence element unmasking task. Sequence continuation can be viewed as a special case of unmasking, where only the last elements are masked. Here we utilize the same methods as in the sequence continuation multi-shot approach but not limiting ourselves to the last candidates. Hence we also mask entries all over a sequence s. As an evaluation metric we use the top-k-RMSE defined above.

Chapter 4

Baseline model performance

To provide a baseline on our tasks, we employ a total of 24 different models across our benchmarks, namely four neural models, 9 classical classifiers, and 11 standard regressors. The following sections describe the architecture of these models.

4.1 Neural Models

4.1.1 Dense Neural Network (DNN)

The densely connected neural network is composed of three hidden layers with 64, 32 and 16 neurons each, and an input layer with 50 neurons, corresponding to the first 50 integers of the sequence. Each hidden neuron is activated with the rectified function (ReLU). The last layer is activated with the sigmoid function in case of classification tasks, and linearly activated in case of regression. In multi-class-classification, 10 output neurons with sigmoid activation (one for each main class) are used. In terms of hyperparameter optimization, we focused on the depth layout and kernel regularization. Our grid search contained the following discrete ranges for each parameter: L1 $\in \big[0.005, 0.01, 0.02\big]$, L2 $\in \big[0.0001, 0.001, 0.01\big]$, depth-layout $\in \big[(64, 32, 16, 8, 4), (64, 32, 16), (16, 16, 16)\big]$

4.1.2 Recurrent Neural Network (RNN)

The architecture of our recurrent model consists of three Long Short-Term Memory [7] layers, each with 64, 32 and 16 units respectively. The first two layers produce the whole sequence, whereas the last layer only yields the last output. The units of the last layer are fed into a dense-layer with a sigmoidal or linear activation. No dropout was used for RNNs. The hyperparameter-grid-search included: L1 $\in \big[0, 0.001, 0.01\big]$, L2 $\in \big[0, 0.001, 0.01\big]$, Dropout $\in \big[0, 0.1\big]$. Figure 4.1 gives a graphical overview of the hyperparameter optimization process achieved with the Weights & Biases software [8] (wandb).

4.1.3 Convolutional Neural Network (CNN)

Here we used a simple stack containing a convolutional layer with kernel size 2, 10 filters and a pooling-layer of size 2, with unit strides. The network was composed of three of these stacks, terminating in a dense layer with the same activation as above. The hyperparameter-tuning focused on filter and kernel size: filters $\in \left[1, 5, 10\right]$, kernel size $\in \left[2, 4, 6\right]$, stack depth $\in \left[2, 3\right]$. The graphical overview can be found in Figure 4.2.

4.1.4 Transformer

We followed the transformer architecture of [9] with only the normalisation layers excluded. We use six transformer blocks for the encoder and three for the decoder. Each multi-headed attention unit consists of 20 attention heads and the output dimension of the embedding and feed-forward layers is 12. In Figure 4.3 we focused on the number of attention heads as well as the embedding dimension in our grid search: heads $\in \left[5, 10, 20, 40\right]$, embedding dimension $\in \left[3, 6, 12, 24\right]$.

4.2 Classical Models

Table 4.1 summarizes all standard classifiers and regressors we utilized. Each standard model is implemented with the scikit-learn [10] library for python. For each model we applied its default parameters. We use the classical models exclusively in the sequence continuation single-shot and sequence classification tasks to give a comparison benchmark to the neural models.

Classifiers		Regressors	
KNNC	k-Nearest Neighbors	KNNR	k-Nearest Neighbors
GNBC	Gaussian naive Bayes	LIR	Linear Regressor
LSVC	Linear Sup. Vector Machine	RIR	Ridge Regressor
DTC	Decision Tree	LAR	Lasso Regressor
RFC	Random Forest	ENR	Elastic Net
GBC	Gradient Boosting	DTR	Decision Tree
ABC	AdaBoost	RFR	Random Forest
XGBC	XG Boost	GBR	Gradient Boosting
DYC	Dummy Classifier	ABR	AdaBoost
		XGBR	XGBoost
		DYR	Dummy Regressor

Table 4.1: Standard Models

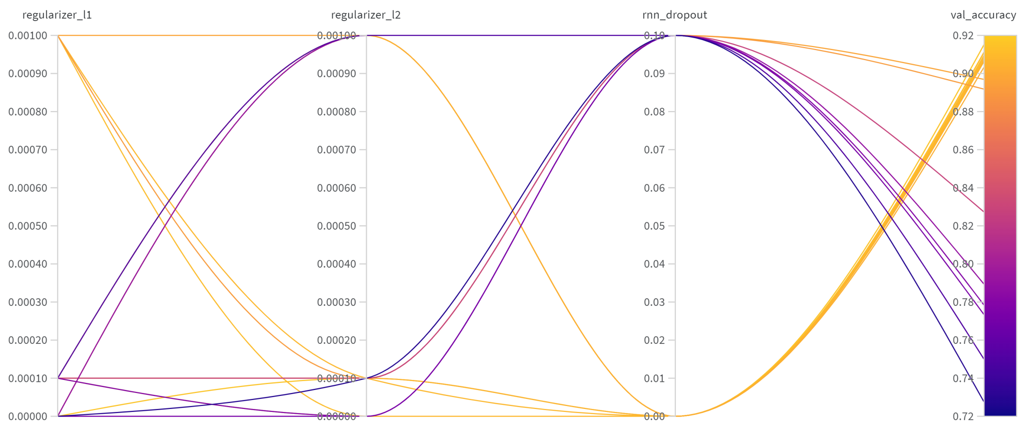

Figure 4.1: RNN hyperparameter optimization

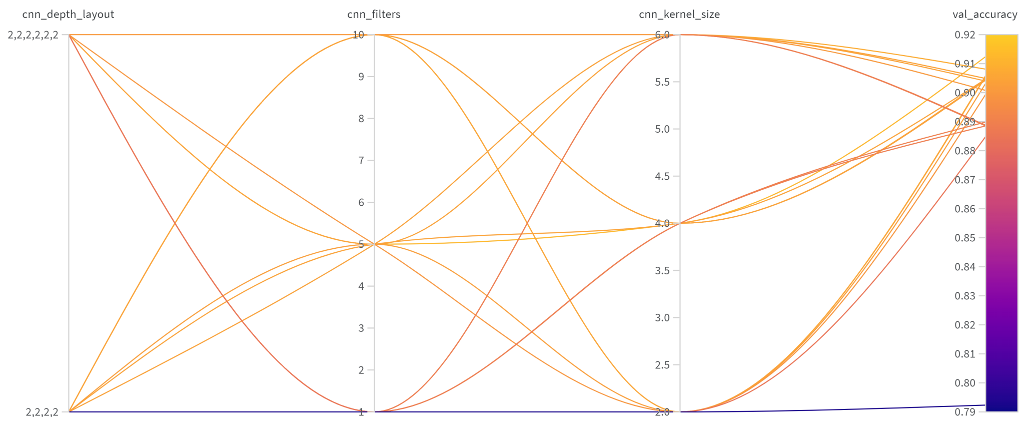

Figure 4.2: CNN hyperparameter optimization

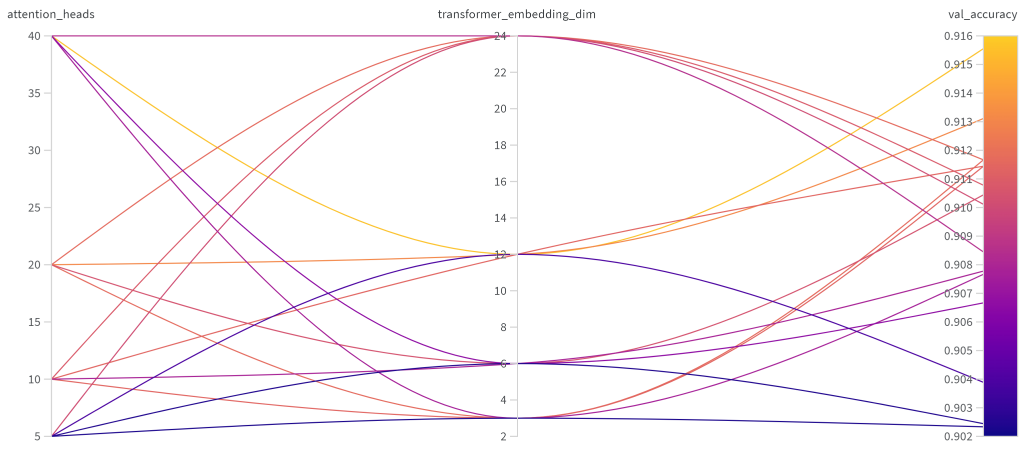

Figure 4.3: Transformer hyperparameter optimization

Max # of Epochs	20
Optimizer	Adam
DNN L1 regularizer	0.001
DNN L2 regularizer	0.0001
RNN L1 regularizer	0.001
RNN L2 regularizer	0.0001
RNN dropout probability	0
CNN filter count	10
margin distance	1

Table 4.2: Additional general training hypterparameters

4.3 Loss and Metrics

Table 4.2 lists all general and additional training parameters we finally choose, after the optimization described in the previous chapter.

4.3.1 Classification Loss

For our classification tasks, sequence classification and next sequence part prediction, we used binary-cross-entropy defined in equation 4.1 as our loss function.

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}) \tag{4.1}$$

Where N is the number of categories and y_i the ground-truth label of the i-th category and $\hat{y_i}$ the predicted label from our models.

As a evaluation metric we went with the root mean squared error (RMSE) in sequence unmasking and sequence continuation multi-shot, whereas mean squared logarithmic error (MSLE) was used in sequence continuation single-shot.

$$L_{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N}} \tag{4.2}$$

$$L_{MSLE} = \frac{\sum_{i=1}^{N}\left(\log(\hat{y_i}+1) - \log(y_i+1)\right)^2}{N} \tag{4.3}$$

4.3.2 Flexible Contrastive Loss

In some tasks we used contrastive learning to achieve a desired embedding space $\mathcal{Z}$, namely sequence similarity, sequence continuation multi-shot and sequence unmasking. There, we went for a contrastive loss described in the following formula:

$$\begin{aligned} d_a &= max\big(\alpha - d, 0\big)^2 \\ d_n &= d^2 \\ L &= (1-\lambda)d_a + \lambda d_n \end{aligned} \tag{4.4}$$

where d is the euclidean distance between two sequences in the embedding space and α is the margin-distance which penalises dissimilar pairs only if their distance d is inside its radius. The goal of this loss is to embed similar sequences near each other in terms of the euclidean distance and different ones further away. The parameter λ functions as a measurement of similarity. In each dynamic task we define this measurement differently. In sequence similarity λ is the indicator function between two classes. In sequence continuation, λ is the fraction between the first n similar numbers of two sequences and its total length, whereas in sequence unmasking λ is the fraction of masked entries in a sequence paired with its unmasked counterpart. With this approach we seek to build an embedding space that learns to differentiate different sequences, according to our predefined tasks.

4.4 Computational Resources and Code

Our program was entirely built in Python, wrapping around Keras [11] which itself wraps around TensorFlow [12]. Keras is an open-source neural-network library written in Python and founded by Francois Chollet. The entire code of this thesis can be found in the GitLab repository [13].
Training of all our models was done on the Slurm cluster from the Computer Engineering and Networks Laboratory (TIK) at ETH Zurich. Running mostly on a single Nvidia Titan Xp, GTX Titan X or GeForce RTX 2080 Ti depending on the resources available at the time of training. One experiment (One specific task, one specific model and one specific category) lasted on average between 45 minutes and 5 hours each, but never longer than 16 hours.

Chapter 5

Results

In Table 5.1 we provide a simplified preview of the performance within the category bounded on some of the baseline models. On the other hand, a complete and detailed listing of our results, evaluated within and across all categories, can be found in Appendix A.

Model	Dataset	Task					
		classification	next sequence	similarity	continuation s.	continuation m.	unmasking
		[binary-acc.]		*[top-5-acc.]*	*[MSLE]*	*[top-5-RMSE]*	
DNN	oeis	0.815	0.749	0.43	0.519	**0.250**	2.878
	synth	0.990	0.943	0.57	0.372	0.534	3.236
RNN	oeis	**0.860**	**0.833**	0.45	0.506	0.406	2.757
	synth	**0.998**	**0.978**	0.56	0.351	0.355	3.197
CNN	oeis	0.800	0.566	**0.59**	0.686	0.296	**2.260**
	synth	0.976	0.915	0.64	0.536	0.682	**2.834**
Transformer	oeis	0.825	0.752	0.50	**0.503**	0.284	2.717
	synth	0.993	0.946	**0.61**	**0.341**	**0.233**	3.045
k-Nearest Neighbors	oeis	0.793	–	–	0.669	–	–
	synth	0.994	–	–	0.433	–	–
Ada Boost	oeis	0.782	–	–	0.776	–	–
	synth	0.958	–	–	0.635	–	–
Decision Tree	oeis	0.820	–	–	0.618	–	–
	synth	0.995	–	–	0.368	–	–
Random Forest	oeis	0.843	–	–	0.619	–	–
	synth	**0.998**	–	–	0.368	–	–
Gradient Boosting	oeis	0.821	–	–	0.578	–	–
	synth	0.990	–	–	0.420	–	–

Table 5.1: Simplified preview. DNN, RNN, and CNN stand for dense neural network, recurrent neural network, and convolutional neural network. **Emphasis** and **emphasis** mark the best performing models in that task.

Chapter 6

Discussion and Future Work

Analyzing the results in Chapter 5 and in Appendix A, we can denote a clear winning model across most of the tasks. However, there are some tasks with a high variance concerning the individual performance. Especially in sequence classification, we notice strong performance of the random-forest-classifier in the categories exponential, trigonometric, modulo, prime and bounded evaluated on synthetic data, whereas the evaluation on the oeis dataset is scattered across multiple classifiers. On most of the non-exclusive categories we recognize the strong performance of the RNN, namely bounded, increasing, unique and across all categories in both oeis and synthetic. In the next sequence part prediction task, the RNN unsurprisingly dominates throughout all categories, as well as in some categories in the sequence continuation single-shot task. There, we witness a shift towards the Transformer on most of the categories except polynomial, exponential, trigonometric and periodic. The transformer also entirely dictates the sequence continuation multi-shot task, except for a few outliers. In sequence similarity the results are distributed over all neural models on the oeis evaluation, while the transformer leads on synthetic data. Surprisingly, in sequence unmasking, the CNN model has the upper hand against the other models.

In general we can see a strong tendency on all models to perform better on synthetic data than on the organic oeis. Overall, the models appear to show some level of understanding of the underlying rules governing the evaluated sequences. But there is still room for improvement, especially on multi-class classification, as well as sequence similarity. We also do not want to neglect the fact, that the neural models could not outperform the standard baseline models everywhere and that their performance is already quit strong in absolute terms.
More sophisticated models with more layers and a deeper architecture might improve neural model performance. The above results were all produced in a "static" mode of operation on which the data was provided to the model upfront. Perhaps a more natural occurring mode would be "dynamic", where the model is active in its learning, pooling for information until it is confident that it can provide an answer. But this requires a more intricate set of evaluation metrics. We believe that this setup deserves attention as it can provide valuable insights into model's reasoning.

Chapter 7

Conclusion

We were able to provide a broad set of bench-marking results across all our models. However, we saw that our neural models could understand some underlying rules in the evaluated sequences but did not have a serious advantage over our standard-models. Defining and creating more sophisticated architectures could certainly improve model performance but on the other hand increase training time by several magnitudes. It is therefore a trade-of between generalization (having a broad band of tasks and models) and good performance.

The focus on integer sequences allowed us to directly prioritize the problem of learning abstractions, without dealing with a large overhead of learning modality-specific representations. Hence, our work was by design general and hierarchically structured in contrast to the tasks that have appeared otherwise isolated in literature [3, 5].

It is our hope that our work will help attract attention to the challenges of designing models that perceive logical relationships, thus helping to facilitate future advancements on the frontiers of general artificial intelligence.

Bibliography

[1] E. Jampen, “Oeis sequence classification,” 2021, unpublished thesis.

[2] N. J. A. Sloane. The on-line encyclopedia of integer sequences. [Online]. Available: https://oeis.org/

[3] C. W. Wu, “Can machine learning identify interesting mathematics? an exploration using empirically observed laws,” *CoRR*, 2018.

[4] Kaggle. (2016) Integer sequence learning. [Online]. Available: https://www.kaggle.com/c/integer-sequence-learning

[5] R. Kozma, R. Ilin, and H. T. Siegelmann, “Evolution of abstraction across layers in deep learning neural networks,” *Procedia computer science*, vol. 144, pp. 203–213, 2018.

[6] N. J. A. Sloane. Style sheet for contributers. [Online]. Available: https://oeis.org/wiki/Style_Sheet

[7] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[8] L. Biewald, “Experiment tracking with weights and biases,” 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray,

C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[13] F. Schenker. (2022) Gitlab repository. [Online]. Available: https://gitlab.ethz.ch/disco-students/fs22/algorithm-learning-on-oeis

[14] N. J. A. Sloane, *A Handbook of Integer Sequences.* Academic Press, 1973.

[15] N. J. A. Sloane and S. Plouffe, *The Encyclopedia of Integer Sequences.* Academic Press, 1995.

Appendix A

Baseline Model Performance

Model	Dataset	Metric	Scope									
			Within									
			polynomial	exponential	trigonometric	periodic	finite	modulo	prime	bounded	increasing	unique
Sequence Similarity			*[top-k-accuracy]*									
DNN	oeis	Top-1	**0.20**	**0.00**	0.08	0.16	n.a.	0.09	0.12	0.07	**0.09**	0.08
		Top-3	0.23	0.00	0.19	0.16	n.a.	0.23	0.21	0.29	**0.32**	0.26
		Top-5	0.44	0.00	0.27	0.33	n.a.	0.30	0.34	0.43	0.45	0.48
	synth	Top-1	0.12	0.13	0.19	0.10	0.09	0.11	0.32	0.10	0.07	0.09
		Top-3	0.27	0.34	0.30	0.25	0.31	0.23	0.54	0.38	0.27	0.33
		Top-5	0.43	0.46	0.44	0.44	0.40	**0.57**	0.69	0.57	**0.44**	0.48
RNN	oeis	Top-1	0.07	**0.00**	0.04	**0.20**	n.a.	**0.13**	0.14	0.06	0.06	0.11
		Top-3	0.25	0.00	0.27	**0.50**	n.a.	0.35	0.16	0.34	0.31	0.25
		Top-5	**0.53**	0.00	0.40	**0.50**	n.a.	0.41	0.34	0.45	0.44	0.42
	synth	Top-1	0.17	0.11	0.20	0.08	0.10	**0.13**	0.35	0.12	0.07	0.08
		Top-3	0.32	0.29	0.39	0.32	0.32	0.25	0.56	0.40	0.29	0.32
		Top-5	0.41	0.38	0.47	0.42	0.47	0.49	**0.70**	0.56	0.41	0.51
CNN	oeis	Top-1	0.15	**0.00**	0.06	0.00	n.a.	0.06	0.16	**0.14**	0.07	**0.15**
		Top-3	**0.34**	**0.12**	0.31	0.00	n.a.	**0.38**	**0.32**	**0.43**	0.26	0.36
		Top-5	0.34	**0.75**	0.41	0.40	n.a.	0.47	**0.46**	**0.59**	0.39	0.50
	synth	Top-1	0.09	**0.14**	0.16	0.07	0.13	0.11	0.32	**0.17**	**0.08**	**0.10**
		Top-3	0.37	**0.39**	0.37	**0.42**	0.29	**0.32**	**0.61**	0.49	0.30	**0.39**
		Top-5	0.50	0.41	0.60	0.48	0.47	0.51	0.66	0.64	**0.44**	0.60
Transformer	oeis	Top-1	0.03	**0.00**	**0.13**	0.00	n.a.	0.05	**0.18**	0.11	**0.09**	0.09
		Top-3	0.31	0.00	**0.39**	0.10	n.a.	0.33	0.28	0.42	0.29	**0.38**
		Top-5	0.48	0.66	**0.43**	0.20	n.a.	**0.49**	0.40	0.50	0.43	**0.55**
	synth	Top-1	**0.21**	0.13	**0.31**	**0.12**	**0.17**	0.10	**0.36**	0.13	0.07	**0.10**
		Top-3	**0.44**	0.32	**0.41**	0.41	**0.37**	**0.32**	0.56	**0.50**	**0.32**	0.35
		Top-5	**0.67**	**0.52**	**0.63**	**0.51**	**0.52**	0.55	0.65	**0.61**	**0.44**	**0.58**

Table A.1: The accuracy results for the sequence similarity task, evaluated both within categories and across the whole dataset. **Emphasis** and **emphasis** mark the best performing models for the OEIS and synthetic data, respectively.

Model	Dataset	Scope										
		Within										Across
		polynomial	exponential	trigonometric	periodic	finite	modulo	prime	bounded	increasing	unique	all classes
Sequence Classification						*[binary-accuracy]*						*[f1-score]*
DNN	oeis	0.634	0.575	0.621	0.451	n.a.	0.458	0.545	0.815	0.666	0.776	0.330
	synth	0.784	0.754	0.815	0.746	0.748	0.801	0.876	0.990	0.857	0.959	0.430
RNN	oeis	0.588	0.712	0.456	0.474	n.a.	0.489	0.586	**0.860**	**0.840**	**0.834**	**0.370**
	synth	0.790	0.788	0.828	**0.764**	**0.755**	0.825	0.907	**0.998**	**0.954**	**0.976**	**0.530**
CNN	oeis	0.569	0.550	0.619	0.470	n.a.	0.483	0.514	0.800	0.575	0.694	0.220
	synth	0.769	0.704	0.792	0.727	0.736	0.769	0.843	0.976	0.752	0.793	0.390
Transformer	oeis	0.599	0.672	0.524	0.464	n.a.	0.475	0.578	0.825	0.661	0.793	0.330
	synth	**0.791**	0.763	0.817	0.759	0.753	0.814	0.883	0.993	0.882	0.905	0.440
KNNC	oeis	0.650	0.615	0.545	0.489	n.a.	0.484	0.559	0.793	0.693	0.756	0.330
	synth	0.760	0.765	0.797	0.707	0.713	0.810	0.896	0.994	0.883	0.890	0.410
GNBC	oeis	0.626	0.276	0.560	0.464	n.a.	0.474	0.476	0.810	0.683	0.667	0.230
	synth	0.769	0.646	0.764	0.721	0.732	0.736	0.635	0.916	0.634	0.648	0.370
LSVC	oeis	**0.709**	0.377	0.646	0.410	n.a.	0.485	0.508	0.763	0.546	0.637	0.310
	synth	0.759	0.622	0.771	0.717	0.680	0.745	0.819	0.954	0.586	0.718	0.350
DTC	oeis	0.618	0.607	0.497	0.480	n.a.	0.483	0.558	0.820	0.624	0.727	0.360
	synth	0.722	0.754	0.807	0.690	0.677	0.812	0.887	0.995	0.918	0.949	0.490
RFC	oeis	0.595	0.680	0.507	0.493	n.a.	0.468	0.563	0.843	0.579	0.787	0.340
	synth	0.789	**0.791**	**0.837**	0.759	0.749	**0.830**	**0.908**	**0.998**	0.938	0.963	0.510
GBC	oeis	0.576	0.495	0.643	0.478	n.a.	0.470	0.548	0.821	0.623	0.788	0.270
	synth	0.785	0.746	0.804	0.758	0.751	0.803	0.872	0.990	0.835	0.860	0.400
ABC	oeis	0.617	0.392	**0.658**	0.470	n.a.	0.457	0.497	0.782	0.511	0.691	0.310
	synth	0.773	0.676	0.777	0.741	0.737	0.761	0.829	0.958	0.669	0.766	0.380
XGBC	oeis	0.603	0.670	0.499	0.480	n.a.	0.477	**0.595**	0.842	0.674	0.813	0.370
	synth	0.789	0.782	0.828	0.762	0.754	0.827	0.901	0.997	0.915	0.962	0.510
DYC	oeis	0.500	0.500	0.500	**0.500**	n.a.	**0.500**	0.500	0.500	0.500	0.500	0.500
	synth	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500
Next Sequence Part Prediction						*[binary-accuracy]*						
DNN	oeis	0.658	0.664	0.719	0.778	n.a.	0.760	0.744	0.749	0.753	0.726	0.733
	synth	0.941	0.927	0.940	0.943	0.939	0.918	0.914	0.943	0.924	0.936	0.943
RNN	oeis	**0.869**	**0.890**	**0.827**	**0.849**	n.a.	**0.855**	**0.860**	**0.833**	**0.876**	**0.889**	**0.869**
	synth	**0.988**	**0.973**	**0.979**	**0.987**	**0.988**	**0.972**	**0.955**	**0.978**	**0.982**	**0.976**	**0.984**
CNN	oeis	0.526	0.520	0.540	0.566	n.a.	0.548	0.535	0.566	0.547	0.539	0.551
	synth	0.893	0.912	0.922	0.903	0.895	0.898	0.890	0.915	0.885	0.901	0.900
Transformer	oeis	0.690	0.666	0.707	0.792	n.a.	0.759	0.747	0.752	0.744	0.737	0.736
	synth	0.947	0.930	0.945	0.947	0.949	0.927	0.919	0.946	0.926	0.943	0.938

Table A.2: The results for the classification and next sequence part prediction task, evaluated both within categories and across the whole dataset. DNN, RNN, and CNN stand for dense neural network, recurrent neural network, and convolutional neural network. **Emphasis** and **emphasis** mark the best performing models for the OEIS and synthetic data, respectively.

Model	Dataset	Scope										
		Within										Across
		polynomial	exponential	trigonometric	periodic	finite	modulo	prime	bounded	increasing	unique	all classes
Sequence Continuation single		*[mean-squared-log error]*										
DNN	oeis	0.750	0.700	0.588	0.576	n.a.	0.567	0.617	0.519	0.600	0.614	0.597
	synth	0.496	0.408	0.345	0.485	0.489	0.398	0.379	0.372	0.477	0.452	0.430
RNN	oeis	**0.738**	**0.692**	**0.561**	**0.561**	n.a.	0.554	0.602	0.506	0.577	0.614	0.603
	synth	**0.470**	**0.375**	0.317	**0.466**	0.461	0.381	0.345	0.351	0.457	0.424	0.406
CNN	oeis	0.776	0.768	0.765	0.679	n.a.	0.727	0.758	0.686	0.730	0.737	0.733
	synth	0.586	0.550	0.498	0.585	0.581	0.557	0.623	0.536	0.599	0.612	0.579
Transformer	oeis	1.632	1.596	0.573	1.113	n.a.	**0.545**	**0.573**	**0.503**	**0.575**	**0.593**	**0.578**
	synth	2.051	1.420	**0.308**	1.978	**0.452**	**0.365**	**0.335**	**0.341**	**0.449**	**0.415**	**0.395**
KNNR	oeis	0.955	0.874	0.761	0.807	n.a.	0.730	0.796	0.669	0.783	0.832	0.808
	synth	0.575	0.451	0.373	0.560	0.551	0.459	0.401	0.433	0.564	0.513	0.486
LIR	oeis	0.872	0.784	0.723	0.880	n.a.	0.710	0.846	0.704	0.786	0.821	0.797
	synth	0.694	0.633	0.545	0.696	0.692	0.611	0.770	0.613	0.701	0.724	0.682
RIR	oeis	0.873	0.784	0.721	0.875	n.a.	0.713	0.846	0.703	0.786	0.822	0.797
	synth	0.692	0.632	0.546	0.695	0.692	0.613	0.769	0.613	0.701	0.725	0.682
LAR	oeis	0.910	0.750	0.727	1.012	n.a.	0.734	0.882	0.743	0.798	0.856	0.827
	synth	0.750	0.703	0.615	0.754	0.748	0.688	0.812	0.683	0.765	0.783	0.747
ENR	oeis	0.886	0.756	0.722	0.951	n.a.	0.723	0.862	0.724	0.793	0.840	0.814
	synth	0.722	0.672	0.583	0.727	0.723	0.656	0.794	0.651	0.734	0.754	0.716
DTR	oeis	0.868	0.801	0.693	0.741	n.a.	0.663	0.731	0.618	0.695	0.749	0.730
	synth	0.496	0.392	0.328	0.492	0.487	0.391	0.351	0.368	0.490	0.445	0.427
RFR	oeis	0.871	0.797	0.696	0.740	n.a.	0.666	0.730	0.619	0.696	0.748	0.730
	synth	0.496	0.393	0.325	0.492	0.488	0.396	0.348	0.368	0.491	0.446	0.427
GBR	oeis	0.857	0.789	0.622	0.694	n.a.	0.650	0.706	0.578	0.694	0.726	0.702
	synth	0.544	0.459	0.377	0.540	0.535	0.446	0.420	0.420	0.545	0.510	0.484
ABR	oeis	0.907	0.868	0.782	0.894	n.a.	0.754	0.837	0.776	0.796	0.878	0.842
	synth	0.635	0.621	0.542	0.667	0.659	0.594	0.652	0.635	0.658	0.672	0.662
XGBR	oeis	0.869	0.796	0.679	0.726	n.a.	0.651	0.707	0.607	0.688	0.735	0.719
	synth	0.499	0.400	0.330	0.497	0.490	0.400	0.360	0.372	0.496	0.452	0.433
DYR	oeis	0.972	0.770	0.821	1.202	n.a.	0.788	0.912	0.883	0.860	0.930	0.923
	synth	0.832	0.868	0.807	0.847	0.844	0.858	0.877	0.866	0.876	0.871	0.877

Table A.3: The MSLE results for the sequence continuation single-shot task, evaluated both within categories and across the whole dataset. **Emphasis** and **emphasis** mark the best performing models for the OEIS and synthetic data, respectively.

Model	Dataset	Metric	Scope										
			Within										Across
			polynomial	exponential	trigonometric	periodic	finite	modulo	prime	bounded	increasing	unique	all classes
Sequence Continuation multi			*[top-k-root-mean-square error]*										
DNN	oeis	Top-1	2.037	1.662	3.542	1.392	n.a.	1.867	1.432	0.850	2.488	1.300	2.668
		Top-3	0.931	1.662	0.442	0.390	n.a.	0.785	0.587	0.367	0.900	0.912	0.599
		Top-5	0.855	1.662	0.361	**0.152**	n.a.	0.696	0.433	**0.250**	0.561	0.912	0.301
	synth	Top-1	5.847	3.261	5.616	2.790	2.695	1.597	1.896	3.951	2.153	2.302	2.144
		Top-3	0.870	0.927	0.365	1.192	1.483	0.475	1.246	0.534	1.130	1.900	1.690
		Top-5	0.752	0.549	0.365	0.766	1.483	0.448	0.944	0.534	0.825	0.520	1.690
RNN	oeis	Top-1	2.089	2.278	1.624	1.735	n.a.	1.326	1.831	1.264	1.473	1.450	1.367
		Top-3	1.038	1.025	0.677	0.661	n.a.	0.645	0.823	0.597	0.617	0.658	0.596
		Top-5	0.706	0.665	0.429	0.439	n.a.	0.435	0.585	0.406	0.427	0.457	0.383
	synth	Top-1	1.889	1.488	0.974	2.193	2.331	1.034	1.210	1.290	1.453	1.604	1.509
		Top-3	0.784	0.566	0.405	0.954	1.033	0.448	0.508	0.589	0.654	0.713	0.667
		Top-5	0.506	0.364	0.244	0.595	0.607	0.290	0.320	0.355	0.401	0.476	0.438
CNN	oeis	Top-1	3.117	2.943	1.957	1.715	n.a.	2.426	1.570	1.430	1.950	1.490	1.825
		Top-3	1.309	1.302	0.679	0.813	n.a.	1.046	0.702	0.501	0.579	1.003	0.807
		Top-5	0.631	0.646	0.585	0.521	n.a.	0.736	**0.361**	0.296	0.406	0.711	0.428
	synth	Top-1	2.475	1.662	1.399	3.533	2.338	1.848	7.013	2.288	2.333	2.033	2.348
		Top-3	1.060	0.695	1.115	1.031	1.151	0.880	1.007	1.152	0.826	0.871	1.105
		Top-5	0.664	0.440	0.836	0.777	0.697	0.647	0.648	0.682	0.463	0.613	0.643
Transformer	oeis	Top-1	**1.503**	**1.408**	**0.984**	**0.786**	n.a.	**0.977**	**1.161**	**0.663**	**1.022**	**0.935**	**0.847**
		Top-3	**0.746**	**0.748**	**0.398**	**0.382**	n.a.	**0.454**	**0.584**	**0.348**	**0.499**	**0.438**	**0.383**
		Top-5	**0.529**	**0.524**	**0.253**	0.287	n.a.	**0.285**	0.414	0.284	**0.376**	**0.288**	**0.267**
	synth	Top-1	**1.816**	**0.962**	**0.788**	**1.730**	**1.651**	**0.811**	**0.801**	**0.873**	**1.084**	**1.198**	**1.258**
		Top-3	**0.727**	**0.410**	**0.312**	**0.787**	**0.706**	**0.316**	**0.331**	**0.389**	**0.447**	**0.507**	**0.490**
		Top-5	**0.448**	**0.248**	**0.202**	**0.484**	**0.418**	**0.225**	**0.205**	**0.233**	**0.270**	**0.331**	**0.284**
Sequence Unmasking			*[top-k-root-mean-square-error]*										
DNN	oeis	Top-1	3.702	**3.529**	**3.460**	3.248	n.a.	3.451	**3.451**	3.274	**3.307**	3.315	3.384
		Top-3	3.305	3.163	2.936	2.917	n.a.	3.114	3.059	2.976	3.061	3.027	3.061
		Top-5	3.125	3.000	2.748	2.779	n.a.	2.972	2.880	2.878	2.925	2.903	2.918
	synth	Top-1	4.240	3.711	3.374	4.062	4.092	3.633	3.786	3.619	3.839	3.766	3.855
		Top-3	3.898	3.448	2.958	3.710	3.744	3.353	3.321	3.356	3.547	3.470	3.524
		Top-5	3.776	3.335	2.825	3.582	3.593	3.239	3.161	3.236	3.441	3.361	3.408
RNN	oeis	Top-1	3.789	3.720	3.548	3.314	n.a.	3.484	3.499	3.320	3.490	3.485	3.455
		Top-3	3.206	3.175	3.052	2.876	n.a.	3.060	2.998	2.908	3.109	3.083	3.091
		Top-5	3.010	2.990	2.847	2.688	n.a.	2.898	2.830	2.757	2.925	2.909	2.944
	synth	Top-1	4.141	3.765	3.368	4.094	4.115	3.535	3.775	3.696	3.674	3.913	3.855
		Top-3	3.663	3.463	3.082	3.639	3.663	3.254	3.314	3.346	3.328	3.515	3.511
		Top-5	3.473	3.291	2.961	3.472	3.507	3.140	3.162	3.197	3.208	3.370	3.379
CNN	oeis	Top-1	3.738	3.699	3.615	3.268	n.a.	**3.383**	3.521	**3.165**	3.453	**3.257**	**3.355**
		Top-3	**2.943**	**2.922**	**2.873**	**2.594**	n.a.	**2.683**	**2.785**	**2.510**	**2.727**	**2.643**	**2.690**
		Top-5	**2.689**	**2.631**	**2.577**	**2.370**	n.a.	**2.437**	**2.539**	**2.260**	**2.490**	**2.423**	**2.440**
	synth	Top-1	**3.906**	**3.584**	**3.223**	**3.791**	**3.886**	3.500	3.646	**3.531**	**3.627**	**3.702**	**3.611**
		Top-3	**3.179**	**3.100**	**2.811**	**3.128**	**3.168**	**2.968**	**3.112**	**3.008**	**2.988**	**3.122**	**3.033**
		Top-5	**2.931**	**2.799**	**2.642**	**2.891**	**2.898**	**2.784**	**2.890**	**2.834**	**2.747**	**2.899**	**2.812**
Transformer	oeis	Top-1	**3.635**	3.674	3.586	**3.245**	n.a.	3.465	3.556	3.378	3.456	3.606	3.524
		Top-3	3.042	3.045	2.951	2.798	n.a.	2.991	2.974	2.903	2.947	3.038	3.017
		Top-5	2.820	2.781	2.658	2.606	n.a.	2.808	2.717	2.717	2.756	2.816	2.811
	synth	Top-1	3.953	3.719	3.403	3.968	4.079	**3.478**	**3.638**	3.605	3.717	3.780	3.757
		Top-3	3.374	3.259	3.019	3.395	3.475	3.190	3.115	3.226	3.293	3.359	3.291
		Top-5	3.171	3.067	2.872	3.161	3.259	3.041	2.941	3.045	3.121	3.173	3.091

Table A.4: The top-k RMSE results for the sequence continuation multi-shot and unmasking tasks, evaluated both within categories and across the whole dataset. **Emphasis** and **emphasis** mark the best performing models for the OEIS and synthetic data, respectively.

Appendix B

OEIS field names

In this section we give a brief explanation of the 18 fields in the OEIS dataset. This is based on the style sheet provided by OEIS [6].

- **oeis_id**

  A unique 6 digit number preceded by an "A". For example *A005735*

- **identification**

  This refers to the ID the given sequence had in one of the books *A Handbook of Integer Sequences*[14] (M followed by a 4-digit number) or *The Encyclopedia of Integer Sequences*[15] (N followed by a 4-digit number).

- **value_list**

  A list of comma separated integers. The actual sequence of interest. Depending on the sequence in question the length of this list can vary by some orders of magnitude. For example $A058445$ contains only one element while the value list of $A175320$ has a length of 1 578 727.

- **name**

  A brief explanation of the sequence. Sometimes, when possible, this already contains an easy to use formula to generate the sequence. For example $A005843$ has the name *The nonnegative even numbers: $a(n) = 2n$.*

- **comments**

  Further general details and side-notes about the sequence that would make the name too long. Here we often can find alternative formulas to generate the sequence or different places in all of mathematics where this sequence pops up. For example one of the comments of sequence $A000045$, the Fibonacci Numbers, is *"Also the number of independent vertex sets and vertex covers in the (n-2)-path graph."*

- **detailed_references**

  References to journal papers and books that can not be linked in the "links" field.

- **links**

  References to material which can be accessed online.

- **formulas**

  Generating functions, closed formulas and other methods to calculate the sequence.

- **examples**

  Examples of how to find a term of the sequence and how to interpret its value.

- **maple_programs**

  Programs written in maple to generate elements of the sequence.

- **mathematica_programs**

  Programs written in mathematica to generate elements of the sequence.

- **other_programs**

  Programs written in programming languages other than maple or mathematica (for example python) to generate elements of the sequence.

- **cross_reference**

  References to other sequences in the dataset which are related in some way.

- **keywords**

  Keywords from a short set of possibilities. For example *nonn* is used for sequences that have no negative values currently in their respective *value_list* field.

- **offset_a**

  Index of the first element in the *value_list*. For example $A005843$, the aforementioned sequence of *"The nonnegative even numbers: a(n) = 2n."*, has an offset_a of 0 because the first element is calculated by $a(0) = 2 * 0$, i.e. the index is 0.

- **offset_b**

  Index of the first element that has an absolute value larger than 1.

- **author**

  Name of the original contributor and date of first contribution.

- **extensions_and_errors**

  Used to claim credit for additions to the entry that can't be properly acknowledged in other fields.