Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed
Computing

Benchmarking The Algorithmic Reasoning Of Neural Models

Bachelor's Thesis

Peter, Simon

petersim@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Peter Belcak
Prof. Dr. Roger Wattenhofer

January 11, 2023

Abstract

The concept of groups is of central importance in the area of abstract algebra. In this thesis we introduce a new view on the learning of different tasks on small groups and lay down a set of benchmarks aimed at conceptual understanding by machine learning models. These tasks allow the assessment of a model's ability to abstract and challenge them to generalize relevant information. To further aid research in this area, we provide a large dataset of small groups and a collection of models, that can be used as a baseline for further machine learning models and implementations in this area.

Contents

Chapter 1

Introduction

Group Theory is a branch of mathematics, studying algebraic structures called groups. A group is a collection of elements, together with a binary operation having certain properties. Groups are of interest in many applications like modelling the symmetry of systems in physics and chemistry or encrypting messages in cryptography.

In this thesis, we apply machine learning techniques to a dataset of groups with order up to 100. The dataset includes a variety of properties, characteristics and relationships of groups, which we want to predict based on the group multiplication table.

The goal of a machine learning model is to identify generalizable structures in the training data, rather than memorizing a set of examples, to then make accurate predictions on unseen input data. We therefore consider extrapolative generalization performance as the relevant criterion for model performance.

Our contributions in this thesis are:

- The implementation of a large dataset of small groups, curated for successive use in tasks, which challenge the models to gain understanding of the concepts determining the data.
- Introduction of a variety of benchmarking tasks, designed to evaluate a model's comprehension of patterns in groups.
- A set of evaluation metrics adjusted to the above tasks to assess model performance and judge potential progress in this area and
- a collection of neural baseline models to facilitate seamless experimentation.

Chapter 2

Dataset

As part of this thesis, we introduce an SQL database representing a dataset of around one thousand different small groups as well as over ten thousand total "augmentations" of these groups. To get access to a variety of groups we used GAP [1], a system for computational discrete algebra. Since the scope of this thesis is limited to groups of order up to 100, we used the small groups library [2] provided by GAP.

Figure 2.1 shows a simplified diagram of the database. In total we introduce 10 different tables. The backbone of the database is the table "GroupInfo". Here we store the group multiplication table, a minimal generating set, as well as 14 binary properties and 7 characteristics for each group. Each group in the database is uniquely identified by the primary key ("Order_", "OrderID"), where "Order_" describes the size of the group and "OrderID" indexes the different non isomorphic groups of a particular size. Both are provided by the the small groups library. We also introduce a serial id, representing a total ordering of all analyzed groups.

The zuppos and conjugacy classes of a group are stored in two separate tables. Each entry identifies a group via the foreign key ("Order_", "OrderID") and stores one specific zuppos/conjugacy class of the referenced group.

To describe different relationships between groups, we introduce several additional tables. Entries for these tables consist of one pair of groups (both referenced by ("Order_", "OrderID")) satisfying the relationship. The tables we introduce are:

- Table "NormalSubgroups": This table stores the normal subgroups of a group in separate entries. Each row also contains two booleans, describing if said normal subgroup is minimal or maximal.
- Table "SubgroupUnitRelationships": This table stores 8 specific subgroups for each group.
- Tables "SylowSystem"/"ComplementSystem"/"HallSystem": For a solvable group these tables contain the sylow system/complement system/hall

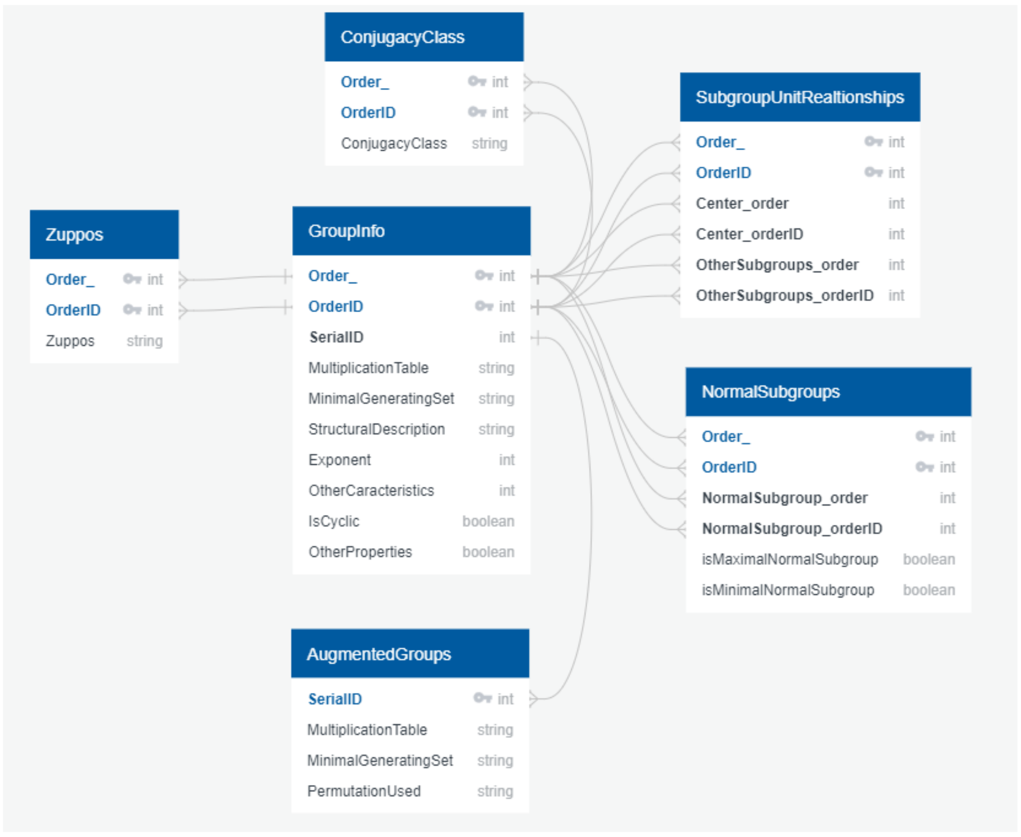

Figure 2.1: Simplified database diagram [3]. Entries between different tables can be linked via the primary key ("Order_", "OrderID").

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

2	3	4	1
3	4	1	2
4	1	2	3
1	2	3	4

Figure 2.2: Left: Multiplication table of group with "Order_" 4 and "OrderID" 1, identity element 1, minimal generating set [2]. Right: New table after applying the permutation $[4, 3, 2, 1]$, with new identity element 4 and new minimal generating set $\{3\}$.

system of a group and NULL otherwise.

2.1 Data Augmentation

All multiplication tables provided by GAP, have the following structure:

- The elements of a group of order n are mapped to the integers $1, \ldots, n$.
- The identity element is always mapped to integer 1
- and the k generating elements are mapped to the numbers $2, \ldots, k+1$

Due to this structure, predictions of the identity element and of the minimal generating set (section 3) would be trivial. We therefore introduce data augmentation as part of our dataset. Besides getting rid of the above structure, this also aims to further increase available training and testing data for future models.

For a group of order n, we consider a permutation γ on $\{1, 2, ..., n\}$ and translate the multiplication table accordingly:

$$\text{For a, b, c } \in \{1, ..., n\} : a * b = c \Rightarrow \gamma(a) * \gamma(b) = \gamma(c)$$

We also use the same permutation γ, to get the new minimal generating set of the augmented group. In figure 2.2 we show an example. For groups of order up to 3, we use all possible permutations, otherwise we choose 10 at random. In the database, these new multiplication tables and generating sets are stored in the table "AugmentedGroups".

Chapter 3

Experiments

Based on the dataset introduced in the previous section, we propose a set of benchmarking tasks and evaluation metrics to assess the ability of machine learning models to extrapolate important concepts.

In appendix A we provide further information on the group concepts, used in the following experiments.

3.1 Basic Experiments

We perform basic experiments on the dataset. These consist of

1. Finding the identity element of a group. The model takes as input the multiplication table of a group of order n and computes an integer between 1 and n.
2. Finding the inverse element of an element a. For a group of order n, the model takes as input the multiplication table as well as the integer index of a. As output it generates an integer between 1 and n.

For these classification tasks we use accuracy as our evaluation metric of choice.

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(y_i = \hat{y}_i),$$

where y are the target labels, $\hat{y}$ are the models prediction and N is the size of the testing set.

The dummy predictor for these tasks evaluated by accuracy, is a majority vote. It always predicts the most frequent label in the training set.

3.2 Property Classification Experiments

For these experiments, we predict all the binary properties of the groups in the dataset. That is a total of 14 properties, namely:

IsCyclic, IsElementaryAbelian, IsNilpotentGroup, IsPerfectGroup, IsSolvableGroup, IsPolycyclicGroup, IsSupersolvableGroup, IsMonomialGroup, IsSimpleGroup, IsAlmostSimpleGroup, IsFinitelyGeneratedGroup, IsSubsetLocallyFiniteGroup, IsPGroup and IsPowerfulPGroup.

As input we feed in the multiplication table and output 1 or 0, depending on whether or not the group fulfills this property. As evaluation metric we again use accuracy, with majority vote as the dummy predictor.

3.3 Prediction Of Characteristics

In this task, we predict specific integer-valued characteristics of the groups. The characteristics in question are:

Exponent, MinimalFaithfulPermutationDegree, PrimePGroup, PClassPGroup, RankPGroup, NrConjugacyClasses and CommutatorLength.

In the previous classification tasks, results represented true/false or the label of a group element. In this case the result is an actual integer and we evaluate the experiment like a regression task. The model outputs float numbers as its prediction and we take the mean squared error as evaluation metric for model performance.

$$\mathrm{MSE}(y, \hat{y}) = \mathrm{mean}((y_1 - \hat{y}_1)^2, (y_2 - \hat{y}_2)^2, ..., (y_n, -\hat{y}_n)^2),$$

where y is the target value and $\hat{y}$ is the models prediction.

The dummy predictor for this task, evaluated by MSE, always predicts the mean of the training set values.

3.4 Relationship Classification

Here we predict, if two groups are in a specific relation to each other. As input the model takes the multiplication table of two groups G_1 and G_2 and predicts 1 or 0, depending on if they fulfill the relation. For example, whether or not G_1 is a normal subgroup of G_2.

The relationships we analyze are:

Center, DerivedSubgroup, FittingSubgroup, FrattiniSubgroup, PrefrattiniSubgroup, PerfectResiduum, Socle, SupersolvableResiduum, CharacteristicSubgroup, SylowSystem, ComplementSystem, HallSystem and NormalSubgroups.

We again use accuracy as the evaluation metric, with majority vote as the dummy predictor.

3.5 Generator Experiments

In this section we asses a model's ability to abstract generating sets. The small groups library provides one minimal generating set per group. The minimal generating set is not necessarily unique, however in this experiment we are interested in predicting the minimal generating set provided. We introduce two tasks:

1. As input the model takes the multiplication table of G and a set of elements S. We want to predict, if S is the minimal generating set of G.
2. Based on the multiplication table of G we want to predict the minimal generating set we store for G.

For the evaluation of the first task, we use binary accuracy. The dummy predictor is the majority vote based on the training set labels.

To analyze model performance in the second task, we introduce two metrics.

- % of times the prediction of the model is a generating set. So:
$$\text{MGS} \subseteq \text{Prediction}$$
- % of times the prediction of the model is the minimal generating set we have in our dataset:
$$\text{MGS} \equiv \text{Prediction}$$

We optimize for the second metric introduced. We suggest the following probabilistic dummy predictor: For each group G it predicts a uniformly chosen subset S of $G \setminus \{e\}$. We exclude the identity element e from G, because by definition it is never in a minimal generating set. The performance of the dummy predictor across a dataset D is then:

$$\text{Expected accuracy} = \frac{1}{|D|} \sum_{G \in D} \frac{1}{|\mathcal{P}(G \setminus \{e\})|}$$

Chapter 4

Baseline Model Performance

We run extensive experiments on the introduced benchmarks as a starting point for further research. In the scope of this thesis, we consider medium sized neural networks of different architectures. The experiments in this section highlight the feasibility of neural networks to learn generalizable information for the previously introduced tasks.

4.1 Models

To provide baselines for model performance on the defined tasks, we use 6 different neural models. Namely:

- A 1d convolutional neural network (1dconv),
- a 2d convolutional neural network (2dconv),
- a dense neural network (MLP),
- a 1d recurrent neural network (1drnn),
- a 2d recurrent neural network (2drnn) and
- a transformer neural network (transf).

We give details on their implementations and hyper-parameter selection in appendix B.

4.2 Setup

In this section we discuss the preprocessing performed for each task and some aspects of model training.

Since the multiplication tables vary in size, depending on the order of the group, we zero pad the multiplication tables up to 100×100 (100 is the highest order in the dataset). Figure 4.1 shows an example of our approach for a group of order 3. This preprocessing step results in a uniform input size of data for the neural networks.

$$\begin{array}{ccc} 3 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 1 \end{array} \quad \mapsto \quad \begin{array}{cccccc} 3 & 1 & 2 & 0 & ... & 0 \\ 1 & 2 & 3 & 0 & ... & 0 \\ 2 & 3 & 1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & 0 \end{array}$$

Figure 4.1: Zero padding of the multiplication table to uniform input size 100×100.

For the tasks of finding the identity (section 3.1), property classifications (section 3.2), prediction of characteristics (section 3.3) and prediction of the minimal generating set (section 3.5) we directly feed in the zero padded multiplication tables.

For finding the inverse (section 3.1), we add a row and a column to the multiplication table, where we specify the element a in question, as illustrated in figure 4.2.

$$\begin{array}{cccccc} 2 & 3 & 1 & 0 & ... & 0 \\ 3 & 1 & 2 & 0 & ... & 0 \\ 1 & 2 & 3 & 0 & ... & 0 \\ 0 & 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & 0 \\ 0 & 0 & 0 & 0 & ... & \mathbf{a} \end{array}$$

Figure 4.2: Preprocessing of the multiplication table for the task of finding the inverse of the element a.

For the relationship classification (section 3.4), we append the padded multiplication table of G_2 to the padded multiplication table of G_1. We therefore feed a 200×100 table to the network. For G_1 and G_2 we uniformly chose one corresponding augmented multiplication table.

For the prediction of whether or not a set S is the minimal generating set of G, we again add a row and a column to the multiplication table of G, where we specify the elements of S in the first entries of the added row. Figure 4.3 shows the introduced approach.

For the tasks relationship classification (section 3.4) and minimal generating set classification (section 3.5), the dataset does not contain negative examples. So for each positive example in these tasks we construct one reasonable negative example. Since we are interested in subgroup relations we require that that the negative example satisfies $\text{Order}(G_1) \leq \text{Order}(G_2)$. For the classification of a

$$\begin{array}{ccccccc}
4 & 3 & 2 & 1 & 0 & ... & 0 \\
3 & 4 & 1 & 2 & 0 & ... & 0 \\
2 & 1 & 4 & 3 & 0 & ... & 0 \\
1 & 2 & 3 & 4 & 0 & ... & 0 \\
0 & 0 & 0 & 0 & 0 & ... & 0 \\
... & ... & ... & ... & ... & ... & 0 \\
\mathbf{1} & \mathbf{3} & 0 & 0 & 0 & ... & 0
\end{array}$$

Figure 4.3: Preprocessing for the task of classifying a minimal generating set with $S = \{1, 3\}$

minimal generating set, for each positive example we add a non generating set of the same size.

If a network architecture takes a 1 dimensional input, we flatten the previously introduced input representations row-wise.

Training is performed on the augmented dataset. We use a standard learning algorithm. For regularization, dropout and early stopping are introduced. We use 20% of the data for testing, 8% for early stopping and the rest for training. Optimization is performed using ADAM with batching. During training we optimize for the following loss functions: Binary cross entropy loss for binary classification, cross entropy loss for multi-classification and mean squared error for regression.

4.3 Results

An overview of the results can be found in the tables 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6.

Task:	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
find ID	0.1056	**0.9987**	0.9344	0.618	0.6362	0.1056	0.9944
find Inverse	0.024	0.0206	0.0191	0.0209	**0.0424**	0.023	0.0231

Table 4.1: Overview of the results of task 3.1. **Emphasis** marks the best performing model. The evaluation metric is accuracy, where higher is better.

4.4 Discussion

Analyzing our results, we observe good model performance for various binary classification tasks as well as finding the identity element. Often the 2d convolution network performs best among the chosen architectures and manages to outperform the dummy prediction by some margin. For certain tasks we also

Propertie:	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
IsCyclic	0.8974	0.9166	**0.9357**	0.9074	0.0.9161	0.9205	0.9
IsElementary Abelian	0.9648	0.9865	**0.9891**	0.9735	0.9739	0.9657	0.9661
IsNilpotent Group	0.5706	0.8005	**0.827**	0.7979	0.7792	0.7501	0.7866
IsPerfectGroup	0.9991	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*
IsSolvable Group	0.9991	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*
IsPolycyclic Group	0.9991	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*	0.9991*
IsSupersolvable Group	0.9392	0.9392*	0.9392*	0.9392*	0.9392*	0.9392*	0.9379
IsMonomial Group	0.9782	0.9782*	0.9782*	0.9782*	0.9782*	0.9782*	**0.98**
IsSimpleGroup	0.9709	0.9974	**0.9983**	0.9896	0.9939	0.9722	0.9744
IsAlmostSimple Group	0.9709	0.9948	**0.9983**	0.9896	0.9939	0.9722	0.9744
IsFinitelyGeneratedGroup	1	1	1	1	1	1	1
IsSubsetLocally FiniteGroup	1	1	1	1	1	1	1
IsPGroup	0.621	**0.9999**	0.9987	0.9861	0.9952	0.8853	0.9835
IsPowerful PGroup	0.9092	0.94	**0.9513**	0.9287	0.9418	0.9100	0.9187

Table 4.2: Overview for the results of task 3.2. Entries marked with * correspond to dummy predictions. **Emphasis** marks the best performing model. The evaluation metric is binary accuracy, where higher is better.

observe a good performance of the 1d convolution, MLP, 1d recurrent and the transformer architectures.

Generally the 2d recurrent network under-performs in comparison. This network is by design very slow and memory intensive, which we suggest is the bottleneck for this model.

The results suggest that neural models are able to learn extrapolative useful information based on multiplication tables. In the following we analyze some results in more detail.

Characteristic:	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
Exponent	457.87	171.83	**162.40**	180.72	173.96	339.24	192.10
MinimalFaithful Permutation-Degree	699.48	568.90	**545.94**	577.85	588.14	649.76	593.862
PrimePGroup	183.42	0.16	**0.15**	1.0	0.31	87.10	0.33
PClassPGroup	0.79	0.60	**0.58**	0.62	0.58	0.77	0.63
RankPGroup	0.88	0.57	0.58	**0.56**	0.60	0.87	0.62
NrConjugacy Classes	371.61	268.34	**61.68**	279.83	326.07	356.16	296.86
Commutator Length	0.00087	0.00417	0.00092	**0.00088**	0.00428	0.00108	0.00410

Table 4.3: Overview for the results of task 3.3. **Emphasis** marks the best performing model. The evaluation metric is mean squared loss, where lower is better.

Relation:	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
Center	0.5119	0.7476	**0.7595**	0.7167	0.7262	0.4881	0.6976
DerivedSubgroup	0.5119	**0.7071**	0.6857	0.6595	0.6905	0.4881	0.6952
FittingSubgroup	0.5119	**0.6619**	**0.6619**	0.6595	0.6429	0.4881	0.6524
FrattiniSubgroup	0.5119	0.7452	0.7357	0.7452	0.7476	0.4881	**0.75**
PrefrattiniSub-group	0.5155	0.7661	**0.7924**	0.7279	0.7804	0.4845	0.7661
PerfectResiduum	0.5119	0.9071	**0.9214**	0.8976	0.8929	0.4881	0.9167
Socle	0.5119	0.6619	0.6643	**0.6833**	0.6738	0.4881	**0.6833**
Supersolvable Residuum	0.5119	0.8905	**0.9048**	0.8048	0.8857	0.4881	0.9024
Characteristic Subgroup	0.4972	-	-	-	-	-	-
SylowSystem	0.5071	0.6261	**0.6615**	0.6204	0.6431	0.4929	0.6431
Complement System	0.5071	0.7238	**0.7252**	0.7082	0.6997	0.4929	0.7082
HallSystem	0.5192	0.7007	0.7000	**0.7014**	0.6993	0.4808	0.7000
NormalSubgroup	0.501	-	-	-	-	-	-

Table 4.4: Overview for the results of task 3.4. Entries marked with * correspond to dummy predictions. **Emphasis** marks the best performing model. The evaluation metric is binary accuracy, where higher is better.

4.4.1 Basic Experiments

For the task of finding the identity element (table 4.1), the 1d/2d convolution and the transformer networks outperform all other architectures. Finding the identity

Task:	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
Clf. MinGenSet	0.5067	0.7553	0.5067*	0.4941	**0.8062**	0.4899	0.6854

Table 4.5: Overview for the results of task 3.5.1. (Prediction on whether or not a set is a minimal generating set). **Emphasis** marks the best performing model. The evaluation metric is binary accuracy, where higher is better.

Metric	dummy	1dconv	2dconv	MLP	1drnn	2drnn	transf
%ofTimesIsGenSet	-	6.30%	6.35%	5.79%	6.26%	0%	**6.39%**
%ofTimesIsMinGenSet	0.0902%	5.60%	5.43%	4.43%	5.17%	0%	**5.82%**

Table 4.6: Overview for the results of task 3.5.2. **Emphasis** marks the best performing model. For both evaluation metrics higher is better.

element of a table is equivalent to finding a row or column which is completely sorted. Based on local sums in the multiplication table one can make a strong guess about this. This seems well suited for a one dimensional network structure. The results show that the 1d convolution performs best with an accuracy of over 99%. We suggest that therefore the model is partly able to reconstruct an algorithm or a shortcut for the evaluation of this property.

For the task of finding the inverse of an element, most network architectures could not outperform the dummy prediction. By definition, computation of an inverse, usually requires knowledge of the identity element, so this task seems conceptually more complex than finding the identity element. We suggest that through a different setup of this task, where one appends the identity element to the input, or a differente representation of a, one might achieve better performance, but further research needs to be done. The current results suggest, that the task, in this setup, is not well suited for neural networks.

4.4.2 Property Classification

Our results (table 4.2) show that neural models can predict binary properties of groups well. The best performance is usually achieved by a 2d convolutional architecture.

For the property IsPGroup the 1d/2d convolution and the 1d recurrent achieved an accuracy of over 99%. We reason that therefore these models were able to accurately reconstruct an algorithm for its evaluation or find a shortcut.

For the properties IsPerfectGroup, IsSolvableGroup and IsPolycyclicGroup, our models can not outperform a dummy prediction. The unbalanced nature of these properties is one explanation for this. We store two perfect groups in our dataset, namely the trivial group and the group $G^{60,5}$ with "Order_" 60 and "OrderID" 5. For IsSolvable and IsPolycyclic the only negative example is again

$G^{60,5}$.

An approach for more robust results for these properties would be to further consider groups of order higher than 100. Rebalancing may not help in this case, due to the low absolute number of positive/negative examples, but further research needs to be done.

4.4.3 Prediction Of Characteristics

For the prediction of characteristics (table 4.3), we have very good performances for PrimePGroup, especially the 2d convolution network achieving a mean squared error of 0.15. This suggests that neural networks are able to extract generalizeable information from multiplication tables, to then later make fairly accurate predictions on this characteristic. Note that the 2d recurrent network performs bad for this characteristic, in comparison to its performance for others.

All groups in the dataset have commutator length 1, except for the groups $G^{96,3}$ and $G^{96,203}$, where it is 2. We suggest that this tilt in the dataset is the reason we can not outperform a dummy predictor on this characteristic.

Note that for the characteristics PrimeClassPGroup and RankPGroup although the dummy predictor already performs well, the models outperform it. This suggests, that the model is able to learn generalizable information from the already limited and tilted dataset. Limited due to the fact that we only store these characteristics for 390 out of the 1048 groups in our dataset, because the others are not PGroups.

4.4.4 Relationships Classification

For the classification of relationships (table 4.4), we measured great performances across the board, with some accuracies over 90%. The 2d convolution again often outperforms the other models. We observed particularly good performances for the relations PerfectResiduum and SupersolvableResiduum.

In the dataset, the perfect residuum of a group is always the trivial group of order 1, except for the group of order 60 with "orderID" 5, which is the only perfect group in the dataset. For this group the perfect residuum is itself.

For the relation supersolvable residuum we again often store the trivial group in the dataset, but for the groups which are not super-solvable, around 7% of all groups in the dataset, we have non-trivial entries. This tilt in the dataset is one possible explanation for the high accuracy in these relations.

The 2d recurrent network is not able to learn useful information for this task.

We do not provide results for the relations characteristic/normal subgroups. Learning these relations was not feasible time-wise for our computational setup.

4.4.5 Generators

We see that for this task (table 4.5) the 1d recurrent network outperforms all other architectures. The 1d convolution as well as the transformer also seem to be able to learn useful information. With the fact that the 2d convolution is not able to outperform a dummy prediction, this suggest that this task is better suited for one dimensional network architectures.

For the second task (table 4.6), which is arguably more difficult than the first one (see appendix A), we still manage to outperform the dummy predictor by a good margin. We observe, that no network architecture seems to drastically outperform the others.

Chapter 5

Related Work

In this section we briefly review some relevant related work on group theory and general machine learning.

Focusing on group theory, we suggest [4] as reliable reference providing a concise mathematical introduction to the topic.

Machine learning methods are remarkably successful in the extraction of useful information for a wide range of application areas. In [5] explainable machine learning in the view of application in the natural sciences is reviewed. Further models baseline analysis has been done in [6], where the authors compared machine learning algorithms on image classification. In [7] a baseline discussion in the field of time series is introduced. In [8] the authors analyze machine learning models for their ability to abstract integer sequences.

A clear theoretical framework is introduced in [9], to explain the performance benefits of data augmentation, a similar strategy was used for our dataset. It proves, that adding data augmentation can lead to variance reduction.

In the paper [10], the authors study the generalization of neural networks on small datasets. Reasoning about data efficiency, memorization, generalization as well as speed of learning is performed. They introduce a phenomena called "grokking", which describes a sudden increase in validation accuracy from chance level toward perfect generalization even long after severely overfitting.

What is an appropriate feedback signal one can use, to make deliberate progress towards more intelligent and more human-like systems? This is one of the questions the authors of [11] answer. They propose a new formal definition of intelligence based on algorithmic information theory, describing intelligence as skill-acquisition efficiency and highlighting the concepts of scope, generalization difficulty, priors and experience. They further propose guidelines for what a general AI benchmark should look like. The abstraction and reasoning corpus (ARC) is introduced, which can be used to measure a human-like fluid intelligence and enable fair general intelligence comparison between AI systems and humans.

CHAPTER 6

Conclusion

Groups frequently arise in the areas of physics, chemistry and public key cryptography. Being able to make accurate predictions on their properties, characteristics and relationships, allows to optimize memory and time intensive computations.

The bench-marking tasks and baselines presented in this work aim to be used as comparison for future implementations of other models.

While the neural models we evaluated appear to show a level of understanding of generalizable patterns underlying groups, there is still significant room for improvement, especially in multi-class classification, where one wants to predict an element or a set of elements.

We suggest another interesting benchmarking task for future research: Given a partial multiplication table T of a uniquely specified group G (where by uniquely specified one means that T can not be filled into a valid multiplication table other than G's), we want to fill in all the missing entries. We reason that this task multi-classifies close to 10000 elements, which in our setup was infeasible, due to high training times for even small network sizes. Further research needs to be done, to see, if one can change the setup to achieve faster learning times.

The preprocessing decisions we performed during the setup are another aspect where further research is necessary. In particular the specification of the element whose inverse should be computed or the specification of the minimal generating set. Also further methods to achieve uniform input sizes could be explored.

It is our hope that our work will help attract attention to the challenges of designing models that are able to perceive abstract information, and help facilitate future advancements in the areas of group theory and artificial intelligence.

Bibliography

[1] https://www.gap-system.org/, accessed: 2022-12.

[2] https://www.gap-system.org/Packages/smallgrp.html, accessed: 2022-12.

[3] https://www.quickdatabasediagrams.com/, accessed: 2022-12.

[4] C. B and S. H, "Introduction to group theory," *EPJ Web of Conferences*, vol. 22, 2012.

[5] R. Roscher, B. Bohn, M. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *IEEE Access*, vol. 8, pp. 42 200–42 216, 2020.

[6] E. Larsen, D. Noever, K. Macvittie, and J. Lilly, "Overhead-mnist : Machine learning baselines for image classification," *arXiv preprint*, vol. 7, 2021.

[7] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 1578–1585, 2017.

[8] P. Belcák, A. Kastrati, F. Schenker, and R. Wattenhofer, "Fact: Learning governing abstractions behind integer sequences," 2022.

[9] S. Chen, E. Dobriban, and J. H. Lee, "A group-theoretic framework for data augmentation," *The Journal of Machine Learning Research*, vol. 21, pp. 9885–9955, 2022.

[10] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, "Grokking: Generalization beyond overfitting on small algorithmic datasets," 2022.

[11] F. Chollet, "On the measure of intelligence," 2019.

[12] https://groupprops.subwiki.org/wiki, accessed: 2022-12.

[13] https://en.wikipedia.org/wiki/Group_(mathematics), accessed: 2023-01.

[14] https://wandb.ai/home, accessed: 2022-12.

[15] https://machinelearningmastery.com/early-stopping-to-avoid-\
overtraining-neural-network-models/, accessed: 2023-1.

[16] https://medium.com/the-dl/transformers-from-scratch-in-pytorch-8777e346ca51, accessed: 2022-12.

Appendix A

Definitions

This is a non-complete set of semi-formal definitions of the group concepts used in this thesis. The following websites [1] and [12] give a more complete overview. All the definitions are from [1], [13] and [12].

We first cover general group definitions

- The order of a group is defined by the number of elements in its set.
- The identity element id of a group is defined as:

  $$\forall a \in G : id * a = a * id = a$$

  .
- The inverse a^{-1} of an element a is defined as:

  $$a * a^{-1} = a^{-1} * a = id,$$

  where id is the identity element.
- The order of an element is defined as the smallest positive power of element resulting in the identity element.

We now provide definitions for all the binary properties in our dataset.

- IsCyclic: A cyclic group is generated by a single element.
- IsElementaryAbelian: This is an abelian group, in which every nontrivial element has order p, where p must be prime.
- IsNilpotentGroup: This is a group, that is intuitively "almost abelian". A formal definition is, that it has a central series of finite length.
- IsPerfectGroup: A group is said to be perfect, if it equals its own commutator subgroup. The commutator subgroup is the subgroup, generated by all the commutators.

- IsSolvableGroup: This is a group for which there exist a subnormal series whose factor groups (quotient groups) are all abelian.
- IsPolycyclicGroup: This is a group, which is solvable and that satisfies the maximal condition on subgroups (that is, every subgroup is finitely generated).
- IsSupersolvableGroup: A group is supersolvable, if it has an invariant normal series where all the factors are cyclic groups.
- IsMonominalGroup: A monomial group is a finite group, whose complex irreducible characters are all monomial, that is, induced from characters of degree 1.
- IsSimpleGroup: This is a group, whose only normal subgroups are the trivial group and the group itself.
- IsAlmostSimpleGroup: A group G is almost simple, if there exists a non-abelian simple group S, such that G is isomorphic to a subgroup of the automorphism group of S, that contains all inner automorphisms of S.
- IsFinitelyGeneratedGroup: This is a group G, that has some finite generating set S, so that every element of G can be written as the combination (under the group operation) of finitely many elements of S and of inverses of such elements.
- IsSubsetLocallyFiniteGroup: A group is called locally finite, if every finitely generated subgroup is finite. This property checks, whether the group U is a subset of a locally finite group.
- IsPGroup: Given a prime number p, a p-group is a group in which the order of every element is a power of p.
- IsPowerfulPGroup: A finite p-group G is said to be a powerful p-group, if the commutator subgroup $[G, G]$ is contained in G^p, if the prime p is odd, or if $[G, G]$ is contained in G^4 for $p = 2$.

Now we cover the characteristics.

- Exponent: That is the least common multiple of the orders of a groups elements.
- MinimalFaithfulPermuationDegree: For a finite group G, this is the least positive integer n, such that G is isomorphic to a subgroup of the symmetric group of degree n.
- PrimePGroup: For a p-group, this returns the prime integer p.

- PClassPGroup: This is the length of the lower p-central series of a p-group.
- RankPGroup: For a p-group, this is the rank, which is defined as the minimal size of a generating system of G.
- NrConjugacyClasses: This is the number of conjugacy classes of a group.
- CommutatorLength: This is the smallest possible n, such that each element in the derived subgroup, can be written as a product of (at most) n commutators.

Here we describe the relationships we analyzed.

- Center: The center of a group G, is the set of elements that commute with every element of G.
- DerivedSubgroup (CommutatorSubgroup): The derived subgroup or commutator subgroup of a group is defined as the subgroup, generated by all commutators of the group.
- FittingSubgroup: The unique largest normal nilpotent subgroup of a group.
- FrattiniSubgroup: This is the intersection of all maximal subgroups.
- PrefrattiniSubgroup: A subgroup P is a prefrattini subgroup of G, if P covers each frattini chief factor of G and if for each maximal subgroup of G there exists a conjugate maximal subgroup, which contains P.
- PerfectResiduum: This is the smallest normal subgroup, that has a solvable factor group.
- Socle: This is the subgroup generated by all the minimal normal subgroups of G.
- SupersolvableResiduum: This is the smallest normal subgroup N, such that the factor group G/N is supersolvable.
- CharacteristicSubgroup: This is a subgroup that is mapped to itself by every automorphism of the parent group.
- SylowSystem: A sylow system of a group G is a set of sylow subgroups of G, such that every pair of subgroups from this set commutes as subgroups. Sylow systems exist only for solvable groups.
- ComplementSystem: This is a set of hall p-subgroups of G, where p runs through the subsets of prime factors of $|G|$ that omit exactly one prime. Every pair of subgroups from this set commutes as subgroups. Complement systems exist only for solvable groups.

- HallSystem: This is a set containing one hall P-subgroup for each set P of prime divisors of the order of G. Hall systems exist only for solvable groups.
- NormalSubgroups: A normal subgroup of G is a subgroup, which is invariant under conjugation by members of G.

The definitions concerning generators:

- A subset of G is a generating set $\Leftrightarrow$ each element of G can be expressed by the elements of the subset, with respect to the groups binary operation and also inversion.
- Furthermore a generating set is called minimal $\Leftrightarrow$ it's the smallest generating set possible.

Appendix B

Baseline Models

To provide the baselines for model performance on the tasks introduced, we use a total of 6 different neural models. The following sections describe the architecture of these models. All the hyper-parameters were optimized, using random search provided by weights and biases [14]. For each model the same hyperparameters were used across all the tasks. For some tasks (for example Relationship Classification 3.4) the input layer of some models got adjusted slightly, to match the new input size.

For all the calculations we used early stopping with a patience of 40 epochs (expect for the 2d-RNN, where we used 8 epochs for the patience). This ensures optimal fitting on unseen data inputs and prevents over-fitting on training examples [15].

B.1 1d Convolution Neural Network

We used a total of three 1d convolution layers, with 8, 16 and 24 channels. We used a kernel of size 4. After each convolution layer we use a pooling layer of size 4, with unit strides. The network terminates in a dense-net with two hidden layers, consisting of 240 and 120 neurons each, connected via a dropout. All the layers are activated using 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, 100 output neurons with sigmoid activation (one for each element) are used.

The following hyper-parameters were used: Learning rate = 0.0001, dropout = 0.5 and batch-size = 12

B.2 2d Convolution Neural Network

Here we used a stack of three 2d convolution layers, with 20, 40 and 50 channels. We used a kernel of size 5×5. A pooling layer of size 2×2, with unit strides, is used after each convolution. The networks terminates in the same dense-net as for the 1d convolution network.

The hyperparameters were selected as for the 1d convolution network.

B.3 Dense Neural Network

The densely connected neural network is composed of two hidden layers with 2048 and 512 neurons each. Each hidden neuron is activated with the rectified function (ReLU). After each layer we apply a dropout. 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, 100 output neurons with sigmoid activation (one for each element) are used.

In terms of hyper-parameters, we used the following: Learning rate = 0.0001, dropout = 0.05 and batch-size = 12.

B.4 1d Recurrent Neural Network

The architecture of this model consists of three RNN layers, with a hidden state of size 512 and a sequence length of 10. We used the ReLu activation function. The units of the last layer are fed into a dense-layer with a sigmoidal or linear activation. No dropout was used in this model.

As learning rate we used 0.001, with a batch-size of 12.

B.5 2d Recurrent Neural Network

Here we used one RNN layer, with hidden state of size 12 and sequence length 1. The rest of the network is as for the 1d recurrent network.

For hyper-parameters we used a learning rate of 0.0001 and a batch-size of 12. No dropout was used.

B.6 Transformer Network

We followed the transformer encoder architecture of [16]. Using 2 layers, 8 attention heads and 256 neurons. We used a sequence length of 100. The network terminates with one fully connected layer, consisting of 512 neurons, connected via dropout. This layer is activated by the rectified function (ReLU). After that we apply the sigmoid function in case of classification, and linearly activated in case of regression. In multi-class-classification, 100 output neurons with sigmoid activation (one for each element) are used.

For hyper-parameters we used a learning rate of 0.0001, a batch-size of 16 and a dropout of 0.1