Xenia Ohmer *February 1, 2023*

Report: Summation game

Abstract

In this project, I implement and analyze a communication game in which the Sender neural network is given two integers as input, and the Receiver neural network must output their sum. For example, if the input to the Sender is [3,5], the Receiver output should be 8.

1 Setup

1.1 Summation Game

There are two agents, a sender and a receiver. One round of the summation game proceeds as follows:

1. Two randomly selected summands, $\{(s_1, s_2) \mid s_i \in \mathbb{N}_N\}$ with $\mathbb{N}_N = \{0, ..., N\}$, are passed to the sender.
2. The sender generates a single-symbol message $m \in V$, where $V = \{0, \ldots, |V| - 1\}$ is the vocabulary.
3. The receiver generates a sum $S \in \mathbb{N}_{2N}$.

1.2 Data set

The original data set consists of all $s_1 \times s_2$ combinations. These $|\mathbb{N}_N|^2$ samples were randomly split into a training and a test set at a ratio of 0.9 / 0.1. In addition, I increased the training set size by including each sample 50 times.

2 Architecture and optimization

Both the sender and the receiver agent are implemented as multi-layer perceptrons (MLPs). The sender processes the concatenated one-hot encodings of the input summands and generates a discrete message using Gumbel-Softmax (GS) (Jang, Gu, & Poole, 2017). The message is embedded and passed to the receiver, generating a probability distribution across possible sums. At validation and test time, the argmax values of the agents' output distributions are used.

I conducted a grid search, fixing $N = 20$ and $|V| = 200$, to determine working hyperparameters in a small-scale setup (see Appendix A). Sender and receiver MLP each have two hidden layers of dimensionality 128, and the symbol embedding layer has dimensionality 64. The GS temperature parameters is set to 2.0 at the beginning of the training and decays with rate 0.995 at every epoch, up to a fixed minimum of 1.35. For optimization, I use Adam with learning rate 0.001 and batch size 32.

3 Experiments

I trained the agents on different input spaces and different vocab sizes. In addition to using $N = 20$, I tested whether the model would generalize to larger input spaces, $N = 40$ and $N = 80$. For each N, I determined the minimal vocab size, required to communicate all possible sums with distinct symbols, $|V|_{min} = |\mathbb{N}_{2N}|$. I varied the vocab size from the minimal value to several times that value, $|V|_{min} \cdot \{1, 2, 4, 8\}$. For example, for $N = 20$ the minimal vocab size is $|V|_{min} = 41$ ($20 + 20 = 40$ plus zero sum) and the vocabulary sizes used for training were $|V| \in \{41, 82, 164, 328\}$. I conducted only one run for each combination of input size and vocab size, although it would be useful to calculate descriptive statistics across multiple runs.

In all experiments, the model was trained for up to 250 epochs, or until a training accuracy of 0.99 was reached. Training and test accuracies are reported based on discrete symbols, rather than the continuous relaxations used in the training process.

4 Results

4.1 Training performance

Figure 1.a shows accuracies (left) and mean absolute errors (MAE) between the predicted sum and the true sum (right) on the training set.[1] While the accuracies for the minimal vocab size lie at 0.85 ($N = 20$), 0.81 ($N = 40$), and 0.57 ($N = 80$), they increase to values > 0.96 for all N, if the vocabulary size is increased. As to be expected, the same patterns can be observed from the MAE between the predicted and the true output sums. For the minimal vocab size, the error lies at 0.93 ($N = 20$), 1.54 ($N = 40$), and 7.22 ($N = 80$), and decreases to < 0.35 for all N if vocabulary size is increased. Thus, if the number of symbols corresponds exactly to the number of distinct input sums, the communication channel capacity, $|C| = |\mathbb{N}_{2N}|$, is too small for the agents to converge. In that case, agents achieve higher accuracies for smaller input spaces. Increasing the channel capacity to $\geq 2 \cdot |\mathbb{N}_{2N}|$ is sufficient to achieve high accuracies for all tested values of N.

Encoding all distinct combinations of integers requires $|C| \geq |\mathbb{N}_N| \cdot |\mathbb{N}_N|$, while encoding distinct input sums requires only $|C| \geq |\mathbb{N}_{2N}|$. Therefore, successful communication at channel capacity $|C| = 2 \cdot |\mathbb{N}_{2N}| \ll |\mathbb{N}_N| \cdot |\mathbb{N}_N|$ suggests that the sender conveys information about the input sums rather than the input summands. The communication strategy will be investigated more closely in Section 4.3.

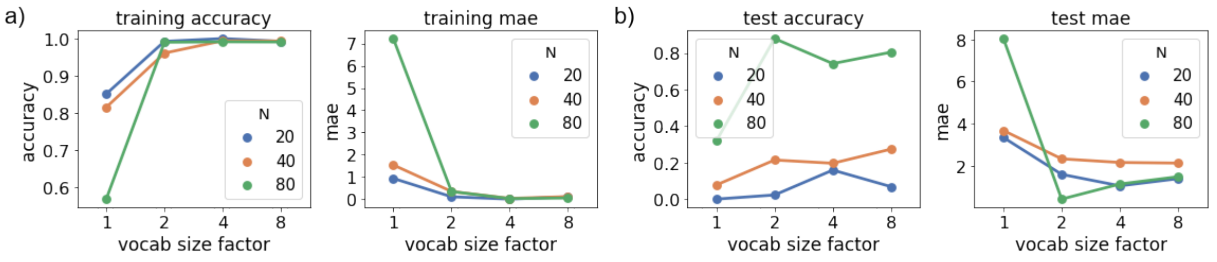


Figure 1: Accuracy and mean absolute error (MAE) on the training set (a) and the test set (b) for different input data ($N \in \{20, 40, 80\}$) and different vocab sizes ($|V|_{min} \cdot \{1, 2, 4, 8\}$).

4.2 Generalization

Figure 1.b shows accuracies (left) and MAEs (right) on the test set. For $N = 20$, test accuracies range between 0.00–0.16, for $N = 40$ between 0.07–0.21, and for $N = 80$ between 0.27–0.88. Thus, zero-shot generalization to novel combinations of input values largely depends on the size of the input space.

The MAEs between predicted and true sums reveal that the agents' generalization behavior is not random—even if test accuracies are low. For a random mapping between input sums and output sums, the MAEs would lie at ~ 11.81 ($N = 20$), ~ 23.62 ($N = 40$), and ~ 46.66 ($N = 80$). Even for the smallest vocabulary, the actual MAEs are far lower with 3.34 ($N = 20$), 3.67 ($N = 40$), and 8.04 ($N = 80$). So, also when test accuracy is low, the agents tend to map unfamiliar combinations of integers to a value that lies in the vicinity of the true sum.

4.3 Symbol use

I analyze symbol use for the different values of N, focusing on those vocabulary sizes that led to training rewards > 0.95, so factors 2, 4, and 8.

4.3.1 Numbers of symbols used

I first analyze whether larger vocab sizes also mean that more symbols are used. Table 1 shows the numbers of symbols that are used at least once to describe the training data, together with their proportion

[1]Plots of the training accuracies over time can be found in Appendix B.

of the total vocabulary size in parentheses. Indeed, for all N, more symbols are used if more symbols are available, without a trend towards an increase or decrease in proportion.

		$\|V\|_{min}$ factor		
		2	4	8
	20	58 (0.71)	86 (0.52)	149 (0.45)
N	40	114 (0.70)	132 (0.41)	383 (0.59)
	80	190 (0.59)	472 (0.73)	938 (0.73)

Table 1: Number and proportion (in parantheses) of symbols used for different N and different vocabulary sizes.

4.3.2 Correspondence between messages and inputs

It seems that there are two main communication strategies the agents could use. Either the sender encodes information about the input values and the receiver generates their sum, or the sender encodes the sum directly. I use entropy-based metrics to understand the correspondence between messages and inputs. The normalized mutual information $\mathcal{NI}(I, M)$ between inputs and messages can be used to quantify the degree of one-to-one correspondence.[2] For all simulations, there is a stronger correspondence between messages and inputs sums, with $\mathcal{NI}$ scores between 0.906–0.985, than between messages and input summands, with scores between 0.712–0.752.

To quantify the degree of polysemy and synonymy in communication, I rely on conditional entropies. The conditional entropy of inputs given messages, $\mathcal{H}(I \mid M)$, quantifies how much uncertainty about the input remains after learning the message and serves as a proxy for polysemy. The conditional entropy of messages given inputs, in turn, quantifies how much uncertainty about the message remains after knowing the input and serves as a proxy for synonymy.

Figure 2 shows the conditional entropy scores normalized by the respective marginal entropies. Based on these metrics, polysemy is close to zero with respect to input sums, but hovers around 0.4 with respect to input summands. Different sums need to be encoded by different symbols, otherwise performance suffers, but different combinations of integers resulting in the same sum can be encoded by the same symbol. Synonymy is generally rather low for input summands and input sums and increases with the vocabulary size. Hence, the increasing symbol use with increasing vocabulary size observed above can be explained by an increase in synonymous symbols.

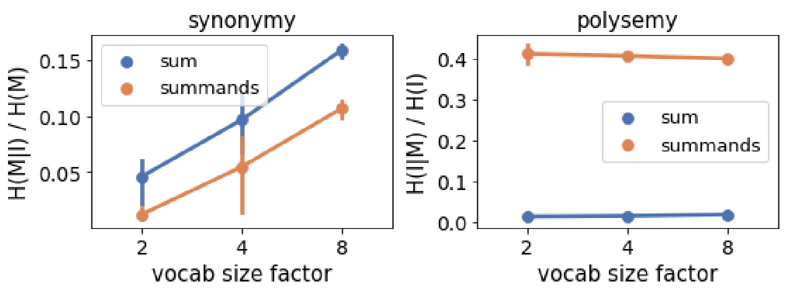


Figure 2: The normalized conditional entropy of messages given inputs (left) serves as a proxy for the degree of synonymy and the normalized conditional entropy of inputs given messages (right) serves as a proxy for the degree of polysemy. Results are averaged across $N \in \{20, 40, 80\}$, and error bars show bootstrapped 95% confidence intervals.

The conditional entropy scores take into account the co-occurrence frequencies for symbols and messages. If one rather cares about the existence of co-occurrences rather than their frequencies, one can also measure synonymy and polysemy by calculating the average number of symbols per input and the average number of inputs per symbols. Appendix C shows the results for this alternative analysis, Appendix D takes an example run, and visualizes in detail which and how many symbols and input sums co-occur.

[2]The mutual information is normalized by the arithmetic mean of the two marginal entropies, $H(I)$ and $H(M)$.

5 Discussion

5.1 Most important findings.

- The task is hard: neural networks for "discrete tasks" are always bad
- Generalization improves with input size

5.2 Caveats and possible improvements.

- upsampling of low frequency sums
- more runs per parameter setting would be good
- training is not super stable, it definitely fails for N=160
- hence 0.99 as early stopping; there might be better values for temperature
- or one could use a supervised - RL combination, which has been found to be more stable; or test RL

References

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. In *5th international conference on learning representations (ICLR).* Retrieved from `https://openreview.net/forum?id=rkE3y85ee`

A Hyperparameter search

I fixed certain hyperparameters to reduce the search space. All models were trained for $N = 20$, $|V| = 200$ using Adam optimizer with learning rate 0.001 and batch size 32. Training proceeded for a maximum of 150 epochs, with a fixed early stopping accuracy of 0.99 on the training set. The grid search was conducted across the following parameter values:

- number of MLP layers: $\{1, 2, 3\}$, so in the case of 1 layer the input is directly mapped onto the output.
- hidden layer dimension: $\{64, 128, 256\}$
- symbol embedding layer dimension: always half of the hidden layer dimension
- GS temperature: $\{1, 1.5, 2.0\}$
- GS temperature decay (applied after each epoch): $\{0.990, 0.995, 1.000\}$
- one-hot encoding of input summands: $\{\text{True}, \text{False}\}$

I selected the model with the highest training accuracy. Note, the minimum GS temperature was set to 0.75 for the grid search. The best model uses temperature 2.0 at decay rate 0.995 and reached accuracy > 0.99 after 77 epochs, so the temperature reached ~ 1.35. For the actual experiments, I extended the number of epochs to 250. To avoid potential instabilities at lower temperatures in later epochs, I therefore increased the minimal temperature to 1.35.

The grid search results can be found in the corresponding folder in the github repository:
`https://github.com/XeniaOhmer/summation_game/tree/main/grid_search`.

B Training accuracies

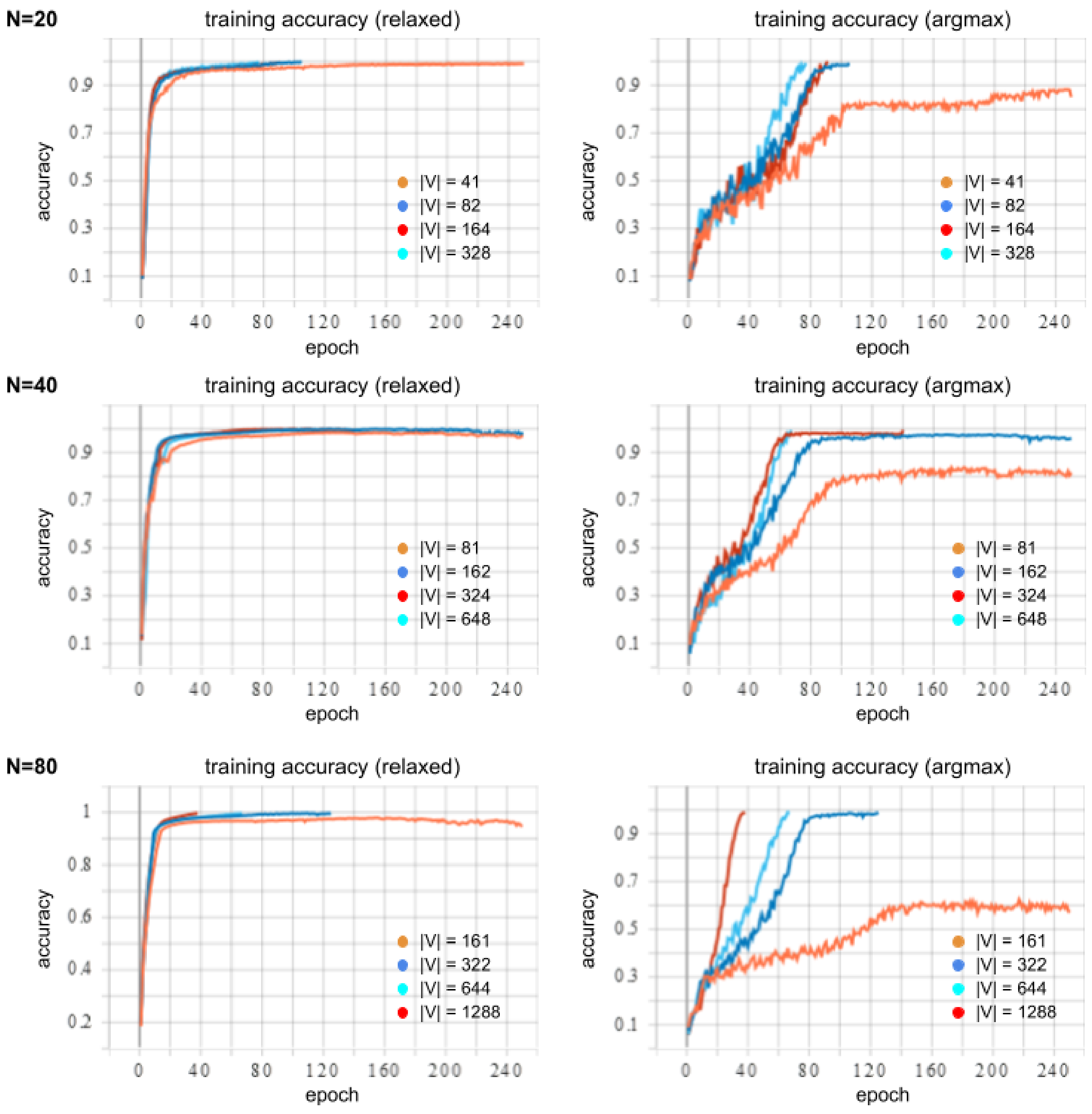


Figure 3: Training accuracies for $N \in \{20, 40, 80\}$ and all vocabulary sizes. The two columns show the training accuracies for the relaxed (left) and the true (right) categorical distribution over symbols. Note the reversed order of colors for $N = 80$.

C Synonymy and polysemy

Figure 4 shows the same information as Figure 2, using an alternative way to quantify synonymy and polysemy. Synonymy is measured as the average number of symbols co-occurring (at least once) with each input sum, and polysemy is measured as the average number of input sums co-occurring (at least once) with each symbol. While there are qualitative differences between the two figures, the main observations are the same. First, synonymy increases with the vocabulary size. If more symbols are available, more symbols are used. Second, polysemy is much higher with respect to input summands than with respect to input sums, which means that symbols are used to encode input sums and not input summands.

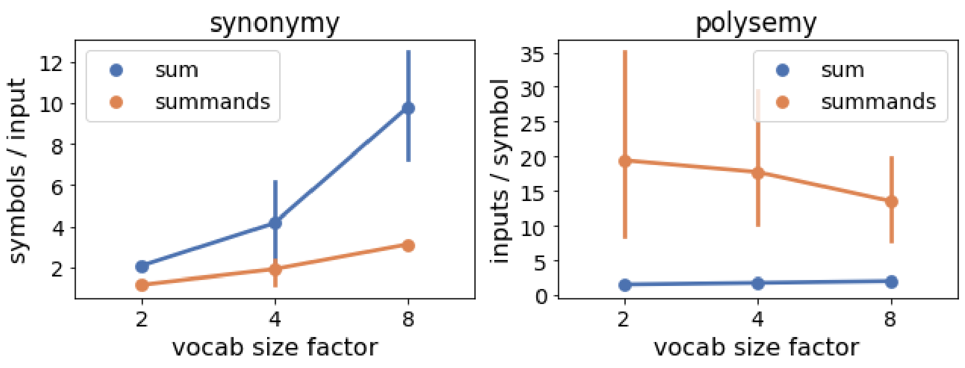


Figure 4: Synonymy and polysemy for different vocabulary sizes, averaged across $N \in \{20, 40, 80\}$. Synonymy is quantified as the average number of symbols that occur at least with the same input, and polysemy as the average number of inputs that occur at least once with the same symbol, evaluated on the training data. The metrics are evaluated for different input encodings: either the inputs are encoded by the two summands (orange) or by their sum (blue).

D Example: Mapping between symbols and input sums

Figure 5 visualizes the co-occurrences between symbols and input sums for $N = 20$ and $|V| = 82$. The barplots show the number of symbols that co-occur with each input sum (top) and the number of input sums that co-occur with each symbol (bottom). The average number of symbols per sum is 2.07 and the average number of sums per symbol 1.47.

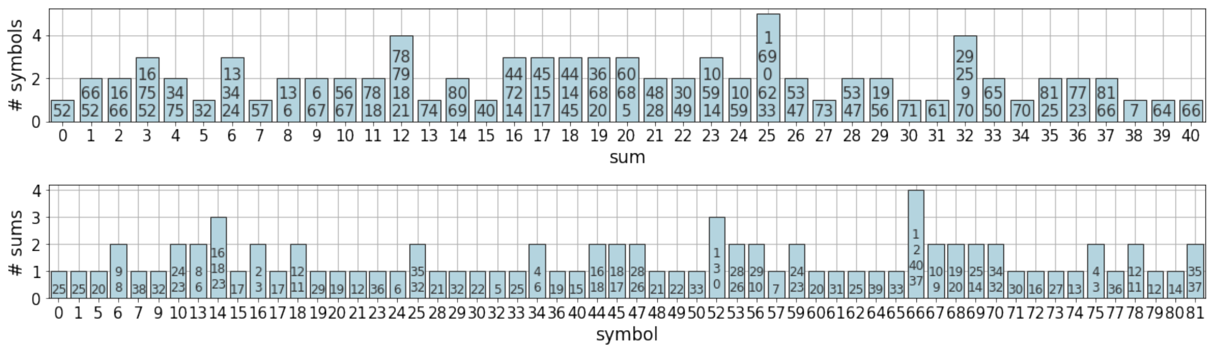
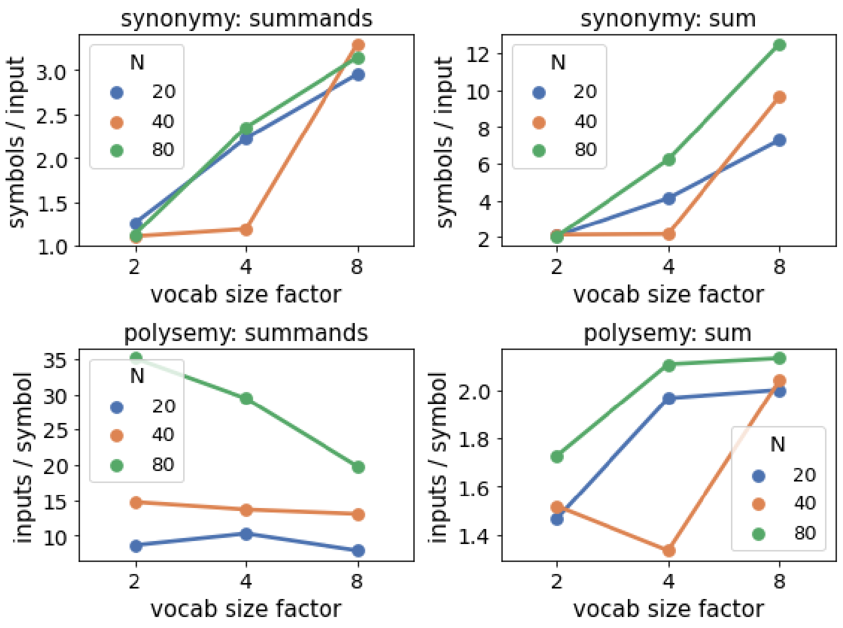


Figure 5: (Top) Synonymy: Barplot of the number of symbols that co-occur at least once with each input sum. The symbols that co-occur with each sum are used as bar labels. (Bottom) Polysemy: Barplot of the number of sums that co-occur at least once with each symbol. The sums that co-occur with each symbol are used as bar labels.


synonymy: summands
synonymy: sum
polysemy: summands
polysemy: sum
symbols / input
inputs / symbol
vocab size factor
N
20
40
80