COMPOSITIONALITY AND LOGIC IN LANGUAGE

Maria Boritchev

January 5th, 2023

Institute of Mathematics of the Polish Academy of Science

COMPOSITIONALITY & MEANING

Sentence-level compositionality & usage

A unicorn drinks tea

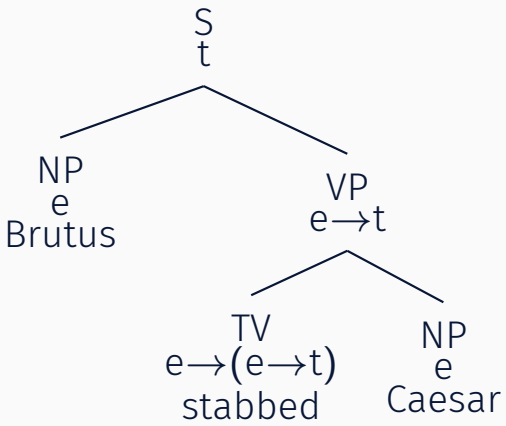
Sentence-level compositionality & usage?

Tea drinks a unicorn

Compositionality and meaning, a definition

"The meaning of an expression is a function of the meanings of its parts and of the way they are syntactically combined" [Partee, 1984]

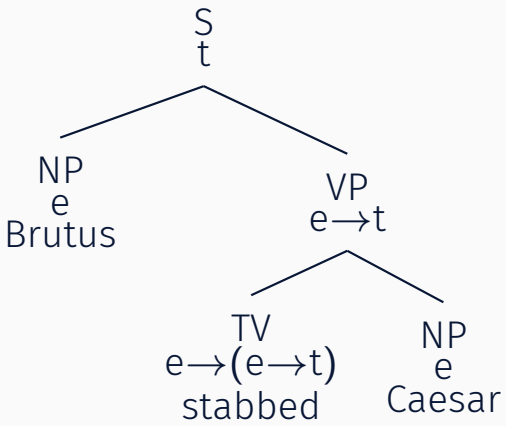
[Montague, 1973], compositional semantics



$$\begin{array}{ccc} [\![\text{Brutus}]\!] & : & \text{brutus}^{e} \\ [\![\text{stabbed}]\!] & : & \lambda o^{e} s^{e}.\text{stab}(s, o)^{t} \\ [\![\text{Caesar}]\!] & : & \text{caesar}^{e} \end{array}$$

$$[\![\text{Brutus stabbed Caesar}]\!] = ([\![\text{stabbed}]\!][\![\text{Caesar}]\!])[\![\text{Brutus}]\!]$$

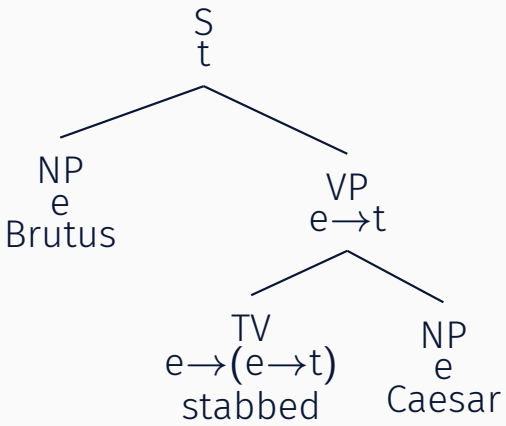
[Montague, 1973], compositional semantics



$$\begin{aligned} [\![\text{Brutus}]\!] &: \text{brutus}^{e} \\ [\![\text{stabbed}]\!] &: \lambda o^{e} s^{e}.\text{stab}(s, o)^{t} \\ [\![\text{Caesar}]\!] &: \text{caesar}^{e} \end{aligned}$$

$$\begin{aligned} [\![\text{Brutus stabbed Caesar}]\!] &= ([\![\text{stabbed}]\!][\![\text{Caesar}]\!])[\![\text{Brutus}]\!] \\ &= (\lambda os.\text{stab}(s, o)\ \text{caesar})\ \text{brutus} \end{aligned}$$

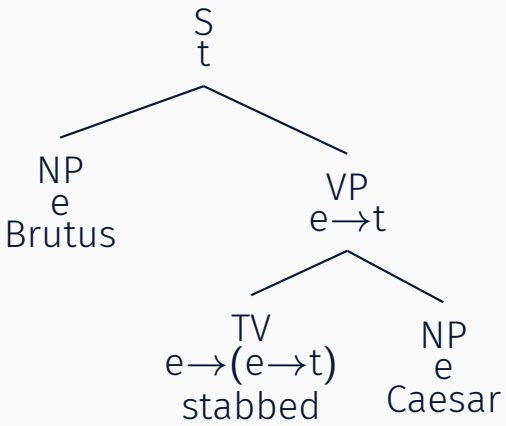
[Montague, 1973], compositional semantics



$$\begin{aligned} [\![\text{Brutus}]\!] &: \text{brutus}^{e} \\ [\![\text{stabbed}]\!] &: \lambda o^{e} s^{e}.\text{stab}(s, o)^{t} \\ [\![\text{Caesar}]\!] &: \text{caesar}^{e} \end{aligned}$$

$$\begin{aligned} [\![\text{Brutus stabbed Caesar}]\!] &= ([\![\text{stabbed}]\!][\![\text{Caesar}]\!])[\![\text{Brutus}]\!] \\ &= (\lambda os.\text{stab}(s, o)\ \text{caesar})\ \text{brutus} \\ &\to_{\beta} \lambda s.\text{stab}(s, \text{caesar})\ \text{brutus} \end{aligned}$$

[Montague, 1973], compositional semantics



$$\begin{aligned} [\![\text{Brutus}]\!] &: \text{brutus}^{e} \\ [\![\text{stabbed}]\!] &: \lambda o^{e} s^{e}.\text{stab}(s, o)^{t} \\ [\![\text{Caesar}]\!] &: \text{caesar}^{e} \end{aligned}$$

$$\begin{aligned} [\![\text{Brutus stabbed Caesar}]\!] &= ([\![\text{stabbed}]\!][\![\text{Caesar}]\!])[\![\text{Brutus}]\!] \\ &= (\lambda os.\text{stab}(s, o)\ \text{caesar})\ \text{brutus} \\ &\to_{\beta} \lambda s.\text{stab}(s, \text{caesar})\ \text{brutus} \\ &\to_{\beta} \text{stab}(\text{brutus}, \text{caesar}) \end{aligned}$$

NNS, REASONING, AND COMPOSITIONALITY

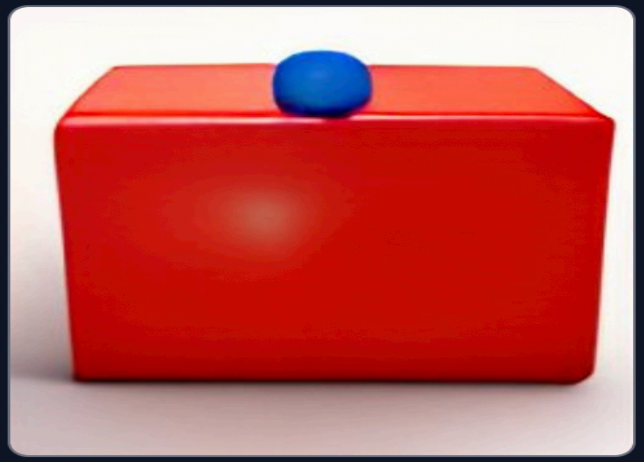
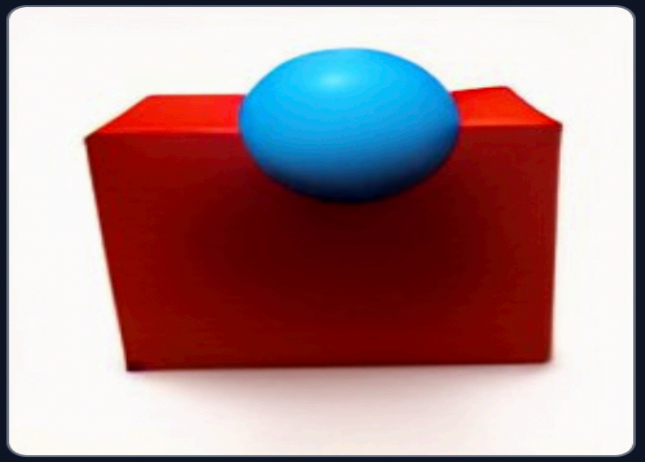
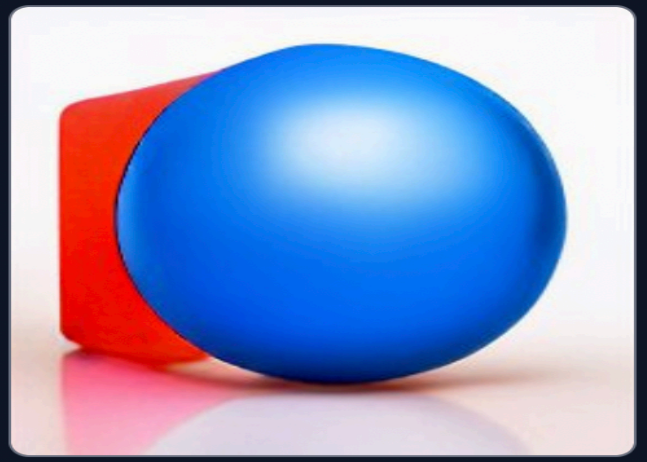
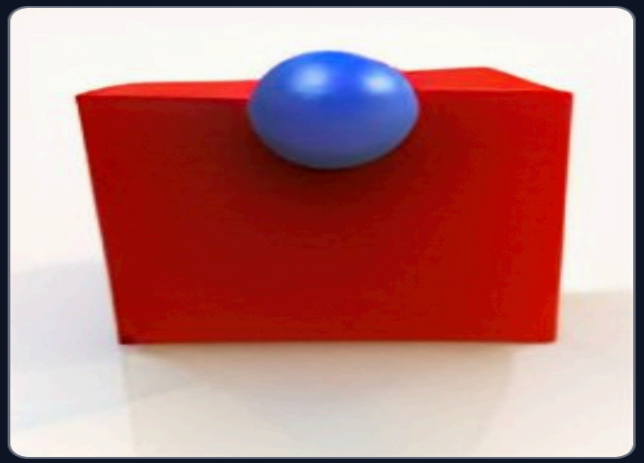
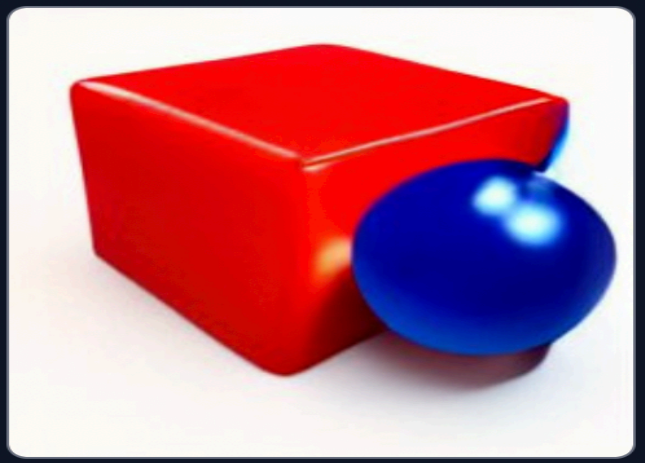
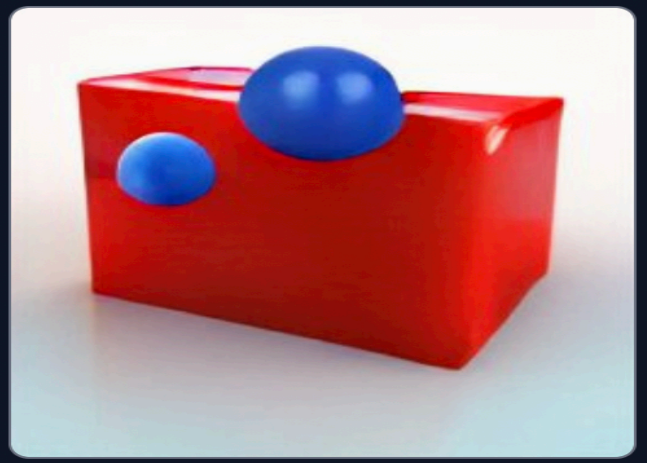
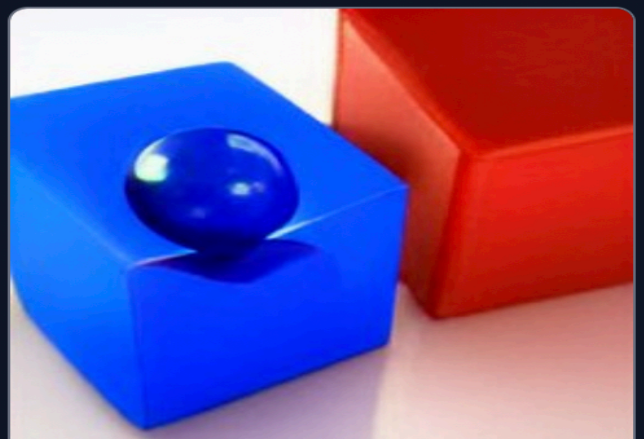
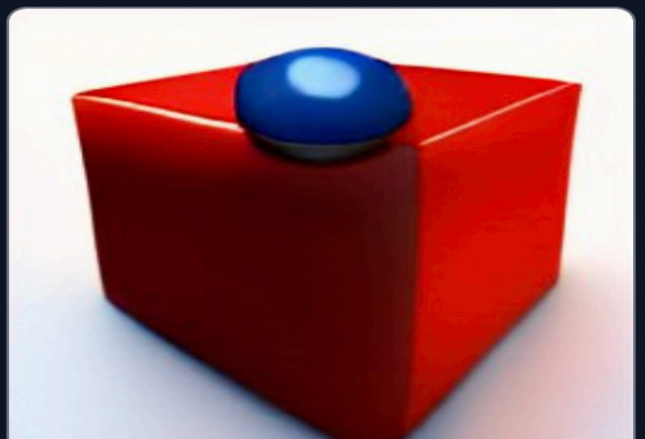
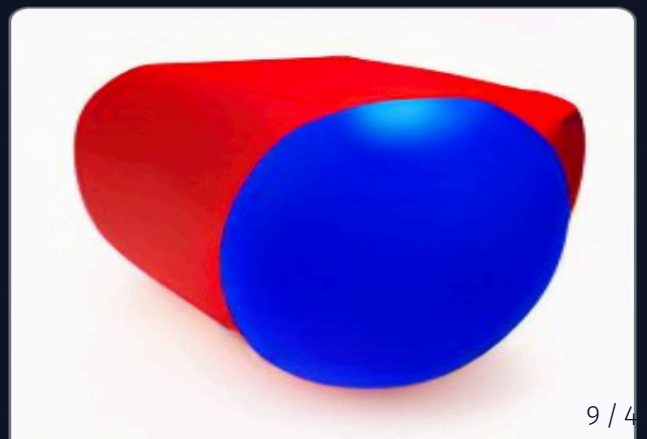
a unicorn drinking tea

Draw

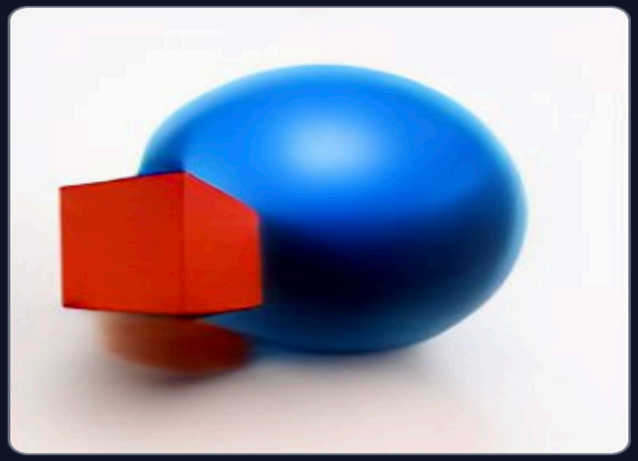
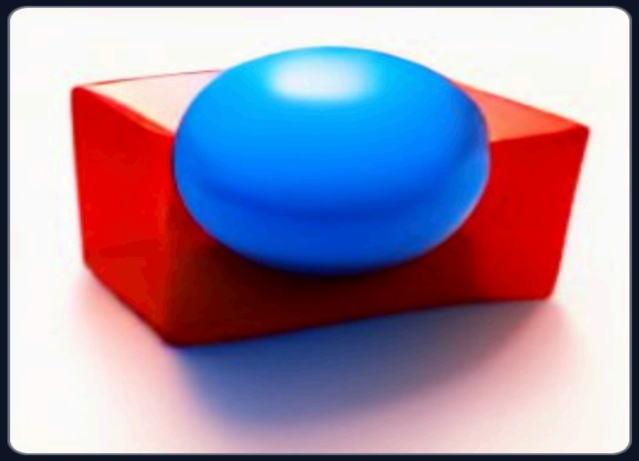
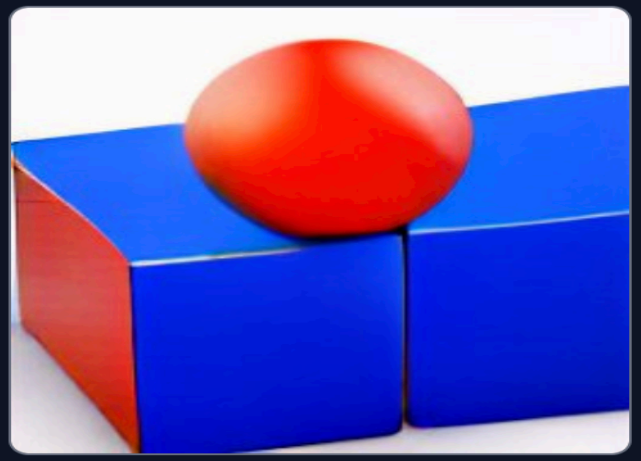
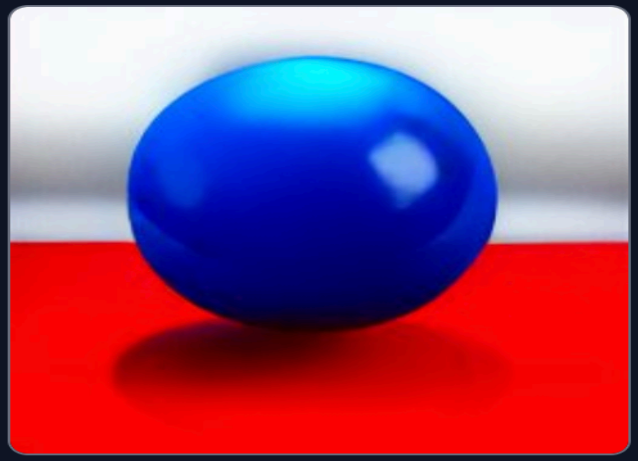
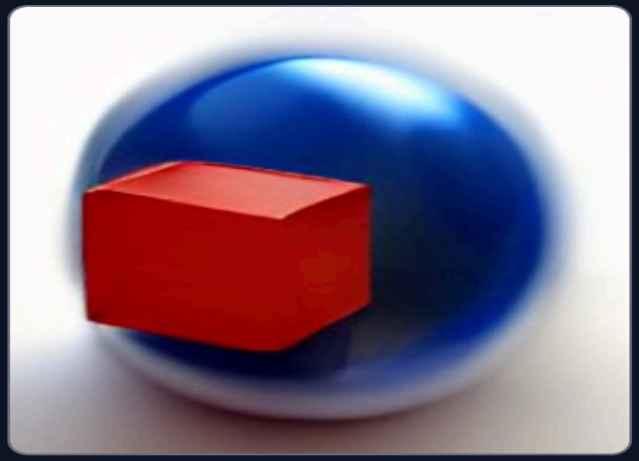
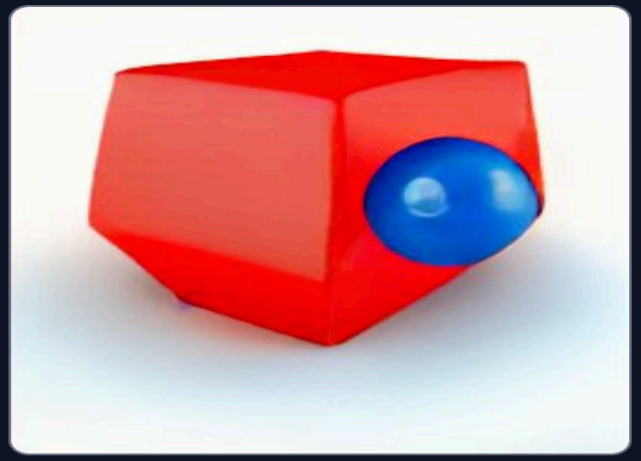
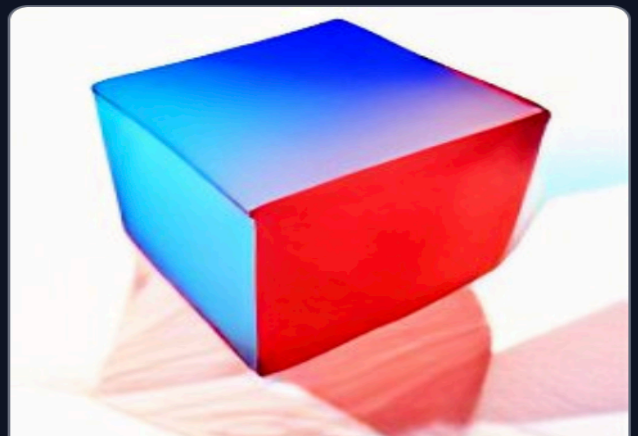
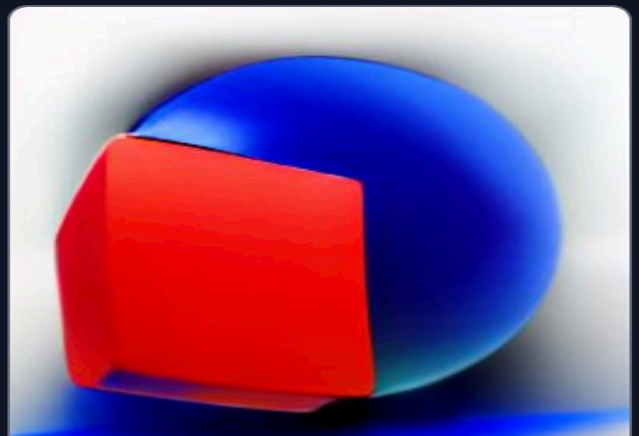
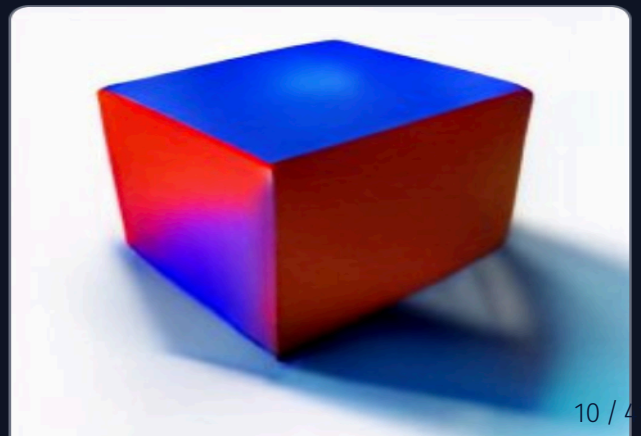
tea drinking a unicorn

Draw

blue ball behind a red cube

red cube behind a blue ball

Neural Networks and compositionality

In the past years, investigation of capacity of (neural) language models to use/produce/deduce compositional rules/sentences.

Neural Networks and compositionality

In the past years, investigation of capacity of (neural) language models to use/produce/deduce compositional rules/sentences.

- **Tree-structured composition in neural networks without tree-structured architectures**, [Bowman et al., 2015]
- **Diagnostic classifiers revealing how neural networks process hierarchical structure**, [Veldhoen et al., 2016]
- **Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization**, [Mul and Zuidema, 2019]

Neural Networks and compositionality

- **Analysing mathematical reasoning abilities of neural models**, [Saxton et al., 2019]
- **Compositionality decomposed: How do neural networks generalise?**, [Hupkes et al., 2020]
- **LogicInference: A New Dataset for Teaching Logical Inference to seq2seq Models**, [Ontanon et al., 2022]
- **Unit Testing for Concepts in Neural Networks**, [Lovering and Pavlick, 2022]

S. Bowman, S. R., Manning, C. D., & Potts, C. (2015). **Tree-structured composition in neural networks without tree-structured architectures.** arXiv preprint arXiv:1506.04834.

- ➔ Test whether neural sequence models such as LSTMs are able to discover and implicitly use recursive compositional structures (with explicit cues).
- ➔ Results: tree models perform significantly better than LSTMs on such tasks.
- ➔ Classification task: what type of logical relation (out of 7 possible) between pairs of sentences.

Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2019). **Analysing mathematical reasoning abilities of neural models.** arXiv preprint arXiv:1904.01557.

- ➔ Task suite of mathematical problems (arithmetic, algebra, probability, calculus) in English, question in English, short answer: only number, yes, no.
- ➔ Designed to test the capacity of NNs (Transformers and Recurrent architectures) to compose mathematical concepts and generalize mathematical operations.
- ➔ Performance varies greatly from one type of problem to another.

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). **Compositionality decomposed: How do neural networks generalise?** Journal of Artificial Intelligence Research, 67, 757-795.

➔ Set of 5 tests for NNs capacity to compositionally generalize:

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). **Compositionality decomposed: How do neural networks generalise?** Journal of Artificial Intelligence Research, 67, 757-795.

➔ Set of 5 tests for NNs capacity to compositionally generalize:

(1) can they successfully recombine known parts and rules?

(2) can they extend their predictions beyond the length they've seen in training?

(3) do they compose in a local or in a global way?

(4) are their predictions robust to synonym substitution?

(5) do they favor rules or exceptions during training?

Hupkes, D., Dankers, V., Mul, M., & Bruni, E. (2020). **Compositionality decomposed: How do neural networks generalise?** Journal of Artificial Intelligence Research, 67, 757-795.

➔ Set of 5 tests for NNs capacity to compositionally generalize:

(1) can they successfully recombine known parts and rules?

(2) can they extend their predictions beyond the length they've seen in training?

(3) do they compose in a local or in a global way?

(4) are their predictions robust to synonym substitution?

(5) do they favor rules or exceptions during training?

➔ Instantiation of the 5 tests on an artifical dataset: PCFG SET, and apply them to a recurrent NN, a convolution-based NN and a transformer.

In our work [Guzmán et al., 2022]

We would like to investigate the capacity of neural networks (NNs) to learn compositional structures/rules.

- ➔ In a minimal setting: **simple corpus**, where all composition comes from the structure of proofs.
- ➔ Testing different aspects of compositionality.

Compositionality, simple corpus, and NNs

Given that a model can derive:

Premisses: "A $\rightarrow$ B" "B $\rightarrow$ C"

Conclusion: "A $\rightarrow$ C"

Then it should derive:

Premisses: "A $\rightarrow$ B" "B $\rightarrow$ C" "C $\rightarrow$ D"

Conclusion: "A $\rightarrow$ D"

Compositionality, simple corpus, and NNs

Given that a model can derive:

Premisses: “A $\rightarrow$ B” “B $\rightarrow$ C”

Conclusion: “A $\rightarrow$ C”

Then it should derive:

Premisses: “X $\rightarrow$ Y” “Y $\rightarrow$ Z”

Conclusion: “X $\rightarrow$ Z”

A SIMPLE CORPUS

Corpus description

Constants: $\mathcal{C} = \{X_1 \dots X_n\}$

Formulas: "$X_i \to X_j$"

Derivation rule:

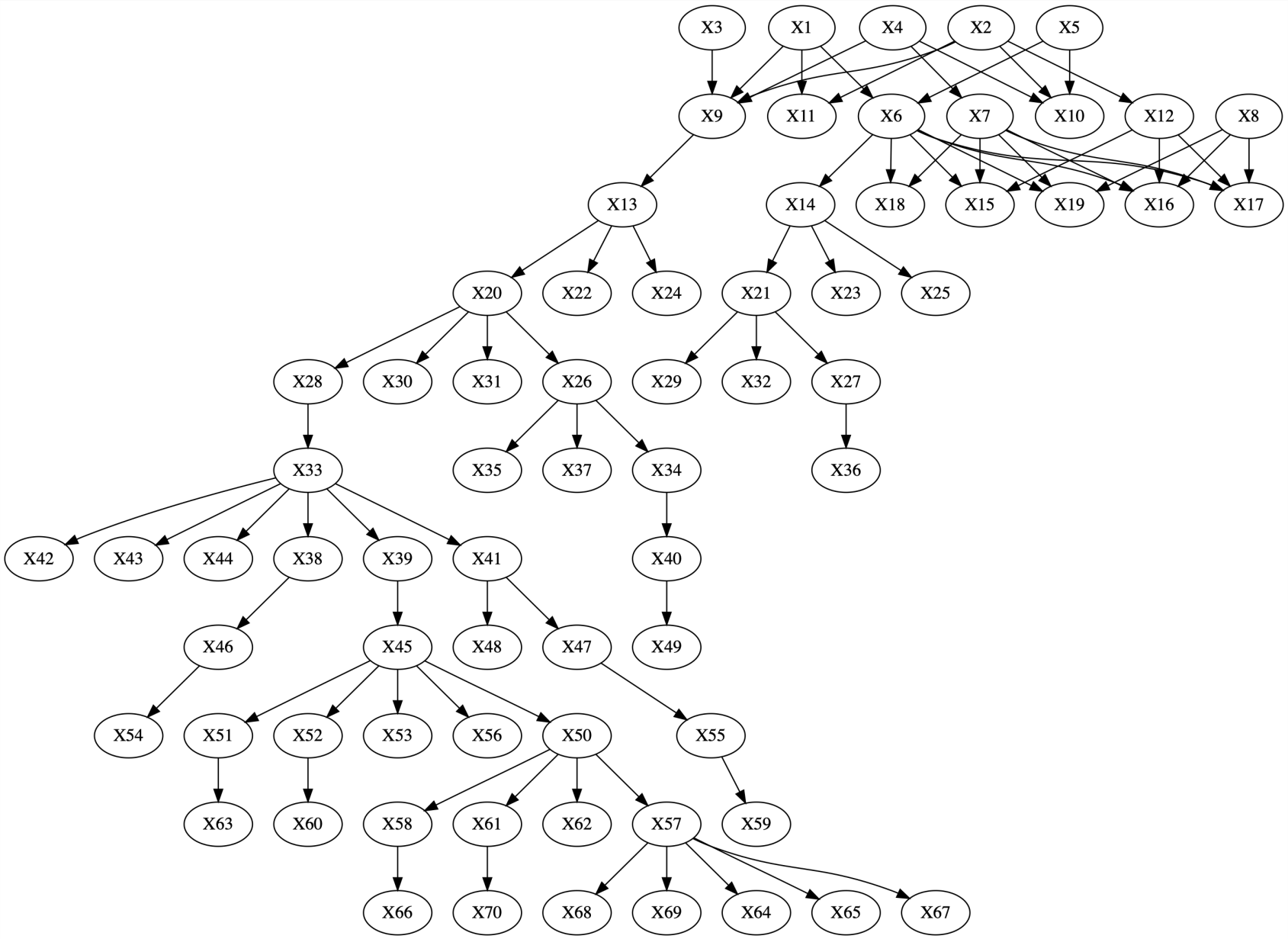
$$\frac{X_i \to X_j \qquad X_j \to X_k}{X_i \to X_k}$$

Example

$$\{X_1 \to X_2,\ X_2 \to X_3,\ X_3 \to X_4,\ X_4 \to X_5\} \vdash X_1 \to X_5$$

$$\dfrac{\dfrac{\dfrac{(1)\ X_1 \to X_2 \qquad (2)\ X_2 \to X_3}{X_1 \to X_3} \qquad (3)\ X_3 \to X_4}{X_1 \to X_4} \qquad (4)\ X_4 \to X_5}{X_1 \to X_5}$$

Knowledge Base (KB)



Knowledge Base (KB)

Parameters of the $\mathcal{KB}$: # constants $= 70$, # premises $= 82$, # hypotheses $= 4,830$, # valid hypotheses $= 531$.

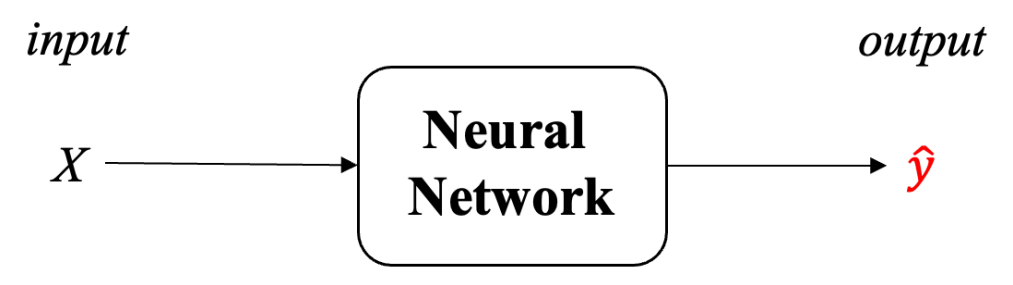
Length of proof	# of proofs
0	4,299
1	82
2	75
3	63
4	60
5	51
6	54
7	41
8	42
9	35
10	28

Connectionnist component

Given a knowledge base $\mathcal{KB}$ and a hypothesis $\mathbf{h}$, our connectionist component is a neural network that selects the (most likely) set of formulas from $\mathcal{KB}$ that are necessary to derive $\mathbf{h}$.

- ➔ Architectures: Multilayer perceptron (MLP), Recurrent neural network (RNN)

Connectionnist component



input: X is a vector that encodes a set $P = \{p_1, p_2, \ldots, p_n\}$ of all premisses and a hypothesis h

output: $\hat{y}$ is a vector of size n such that for each $i \in [1..n]$

$$\hat{y}_i = \begin{cases} 1 & \text{if } p_i \in \pi \\ 0 & \text{otherwise} \end{cases}$$

Where $\pi \subset P$ and $\pi \vdash h$, and π is necessary to prove h.

Example

1. $X = [\text{“}X_1 \rightarrow X_3\text{”} \text{“}X_3 \rightarrow X_6\text{”} \text{“}X_6 \rightarrow X_4\text{”} \text{“}X_2 \rightarrow X_7\text{”} \text{“}X_2 \rightarrow X_4\text{”} \text{“}X_1 \rightarrow X_4\text{”}]$

   $\hat{y} = [1 \quad 1 \quad 1 \quad 0 \quad 0]$

Example

1. $X = [\text{“}X_1 \rightarrow X_3\text{”} \text{“}X_3 \rightarrow X_6\text{”} \text{“}X_6 \rightarrow X_4\text{”} \text{“}X_2 \rightarrow X_7\text{”} \text{“}X_2 \rightarrow X_4\text{”} \text{“}X_1 \rightarrow X_4\text{”}]$

   $\hat{y} = [1 \quad 1 \quad 1 \quad 0 \quad 0]$

2. $X = [\text{“}X_1 \rightarrow X_3\text{”} \text{“}X_3 \rightarrow X_6\text{”} \text{“}X_6 \rightarrow X_4\text{”} \text{“}X_2 \rightarrow X_7\text{”} \text{“}X_2 \rightarrow X_4\text{”} \text{“}X_1 \rightarrow X_7\text{”}]$

   $\hat{y} = [0 \quad 0 \quad 0 \quad 0 \quad 0]$

EXPERIMENTS AND RESULTS

Initial experiment – data distribution

Length of proofs	Total data	Train data	Test data
0	4299	3224	1075
1	82	62	20
2	75	56	19
3	63	47	16
4	60	45	15
5	51	38	13
6	54	40	14
7	41	31	10
8	42	32	10
9	35	26	9
10	28	21	7

Table: 75/25% split of the $\mathcal{KB}$, stratified by length of proofs

Initial experiment – accuracies

Length of proofs	MLP test	RNN test
0	99.91%	99.81%
1	40.00%	50.00%
2	57.89%	63.16%
3	68.75%	75.00%
4	93.33%	93.33%
5	100.00%	100.00%
6	100.00%	100.00%
7	100.00%	100.00%
8	100.00%	90.00%
9	100.00%	100.00%
10	100.00%	100.00%

Initial experiment – accuracies

|proofs|	MLP test	RNN test	hypothesis	# of sub-proofs	# of formulas
0	99.91%	99.81%	3224	0	0
1	40.00%	50.00%	62	62	62
2	57.89%	63.16%	56	138	194
3	68.75%	75.00%	47	202	362
4	93.33%	93.33%	45	331	695
5	100.00%	100.00%	38	416	1017
6	100.00%	100.00%	40	640	1756
7	100.00%	100.00%	31	645	2002
8	100.00%	90.00%	32	839	2869
9	100.00%	100.00%	26	845	3178
10	100.00%	100.00%	21	827	3384

Hamming distances

Let y be the expected output vector for our NN, $\hat{y}$ the actual output vector:

$$y = [1\ 1\ 0\ 0\ 1]$$
$$\hat{y} = [0.68\ 0.98\ 0.33\ 0.12\ 0.46]$$
$$\hat{y} \simeq [1\ 1\ 0\ 0\ 0]$$

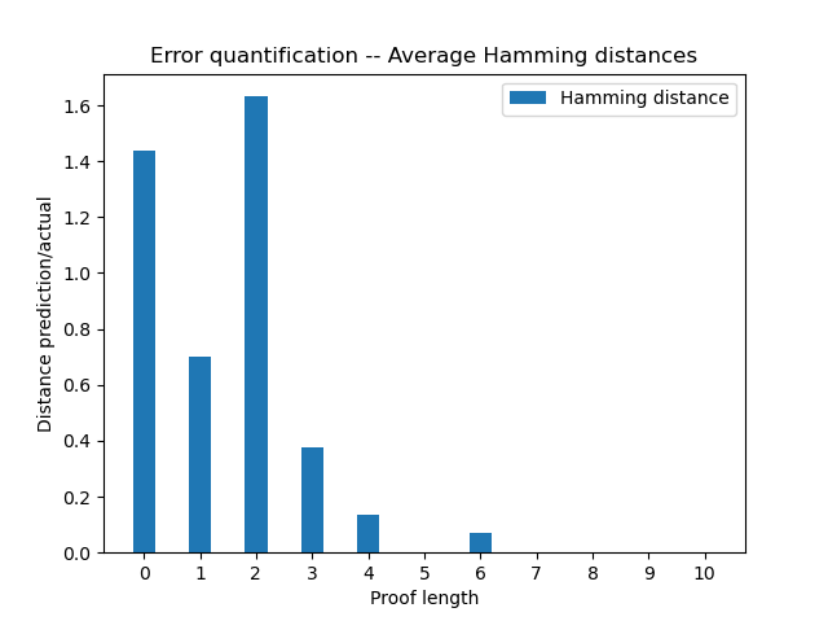
Hamming distances

Let y be the expected output vector for our NN, $\hat{y}$ the actual output vector:

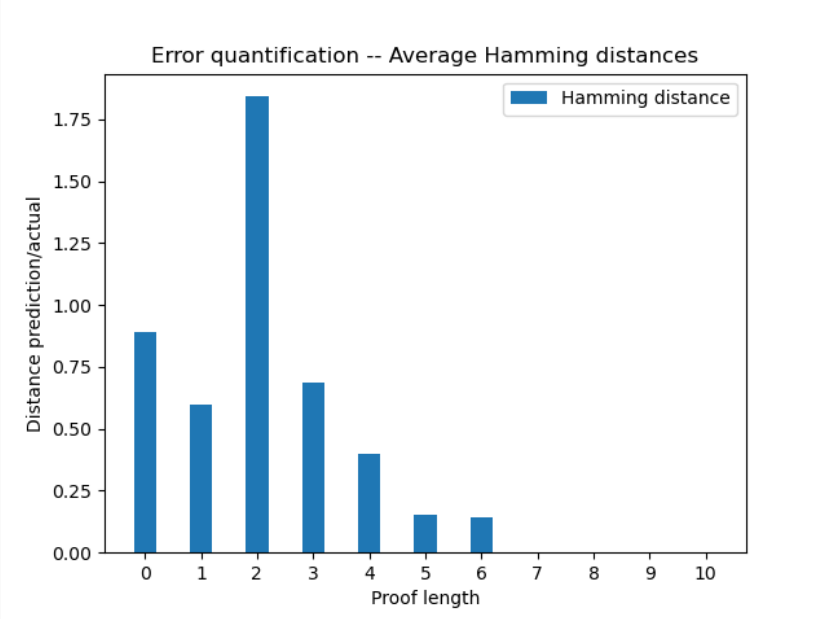
$$y = [1\ 1\ 0\ 0\ 1]$$
$$\hat{y} = [0.68\ 0.98\ 0.33\ 0.12\ 0.46]$$
$$\hat{y} \simeq [1\ 1\ 0\ 0\ 0]$$

$$H(y, \hat{y}) = 1$$

Hamming distances – Initial experiment



(a) MLP



(b) RNN

Figure: Average Hamming distances between the expected output and the actual output for the initial experiment

Compositionality tests

We explored compositionality tests in the context of our simple corpus.

- ➔ Variations in the number of premisses needed to prove the conclusion:
  - Train on short proofs, test on long proofs
  - Train on long proofs, test on short proofs
- ➔ Permutations in the order of the constants

Variations in the number of premisses

The model is trained on proofs from length n_1 to n_2 ($n_1 < n_2$). Then, how does it perform when confronted to valid proofs, including proofs of length $n < n_1$ or $n > n_2$?

Variations in the number of premisses – results

Train data	MLP test	RNN test
0-9	100.0%	100.0%
0-8	92.06%	93.65%
0-7	73.33%	82.86%
0-6	45.89%	65.07%
0-5	17.0%	46.0%

(a) Unseen longer proofs

Train data	MLP test	RNN test
1-10	44.1%	48.08%
2-10	40.95%	41.61%
3-10	37.5%	40.13%
4-10	7.37%	11.06%
5-10	1.81%	2.16%

(b) Unseen shorter proofs

Variations in the number of premisses – results

Train data	MLP test	RNN test
0-9	100.0%	100.0%
0-8	92.06%	93.65%
0-7	73.33%	82.86%
0-6	45.89%	65.07%
0-5	17.0%	46.0%

(a) Unseen longer proofs

Train data	MLP test	RNN test
1-10	44.1%	48.08%
2-10	40.95%	41.61%
3-10	37.5%	40.13%
4-10	7.37%	11.06%
5-10	1.81%	2.16%

(b) Unseen shorter proofs

➔ The less data is used for train, the less the model learns.

Variations in the number of premisses – results

Train data	MLP test	RNN test
0-9	100.0%	100.0%
0-8	92.06%	93.65%
0-7	73.33%	82.86%
0-6	45.89%	65.07%
0-5	17.0%	46.0%

(a) Unseen longer proofs

Train data	MLP test	RNN test
1-10	44.1%	48.08%
2-10	40.95%	41.61%
3-10	37.5%	40.13%
4-10	7.37%	11.06%
5-10	1.81%	2.16%

(b) Unseen shorter proofs

- ➔ The less data is used for train, the less the model learns.
- ➔ The RNN has better accuracy than the MLP.

Variations in the number of premisses – results

Train data	MLP test	RNN test
0-9	100.0%	100.0%
0-8	92.06%	93.65%
0-7	73.33%	82.86%
0-6	45.89%	65.07%
0-5	17.0%	46.0%

(a) Unseen longer proofs

Train data	MLP test	RNN test
1-10	44.1%	48.08%
2-10	40.95%	41.61%
3-10	37.5%	40.13%
4-10	7.37%	11.06%
5-10	1.81%	2.16%

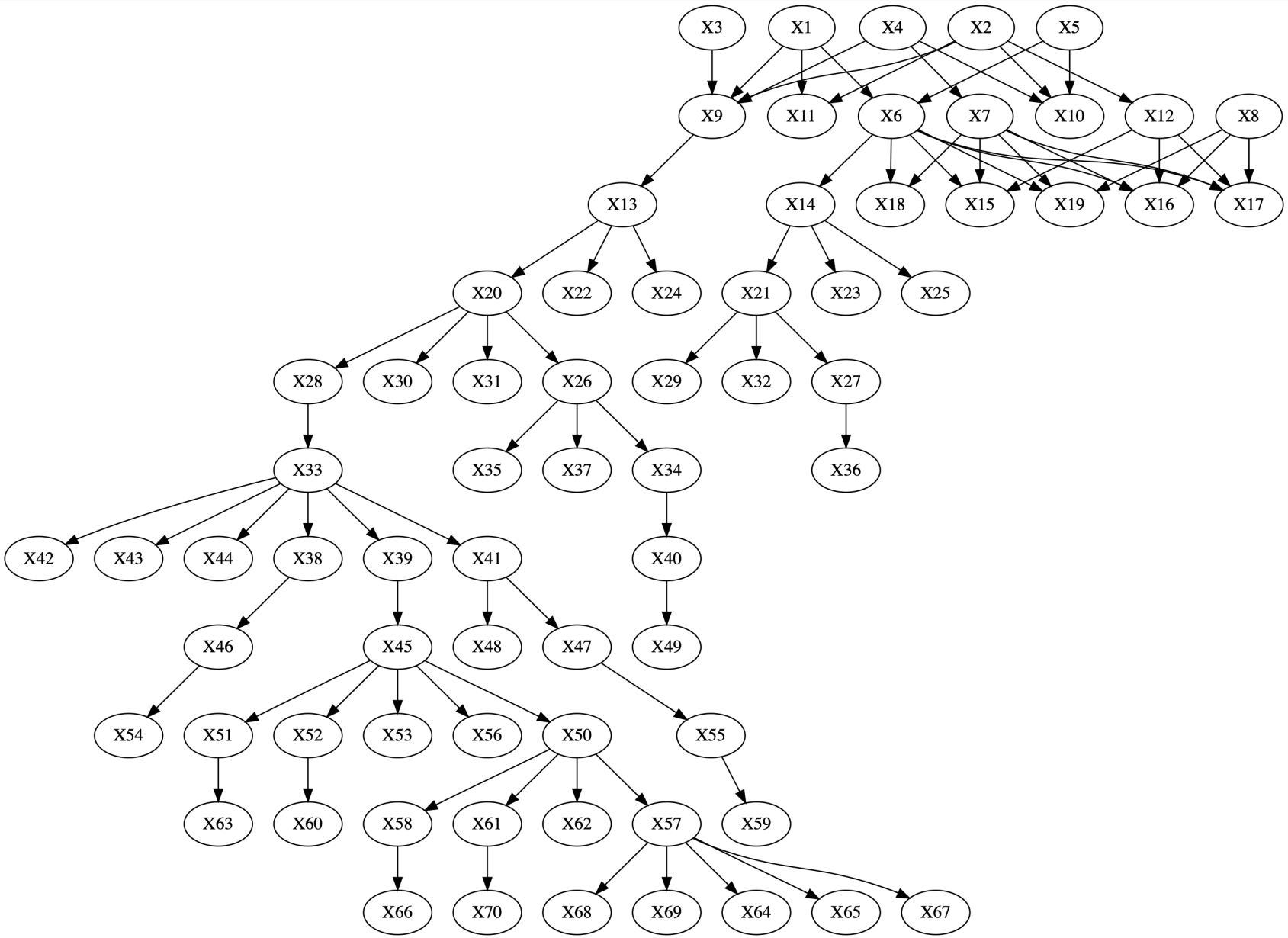
(b) Unseen shorter proofs

- ➔ The less data is used for train, the less the model learns.
- ➔ The RNN has better accuracy than the MLP.

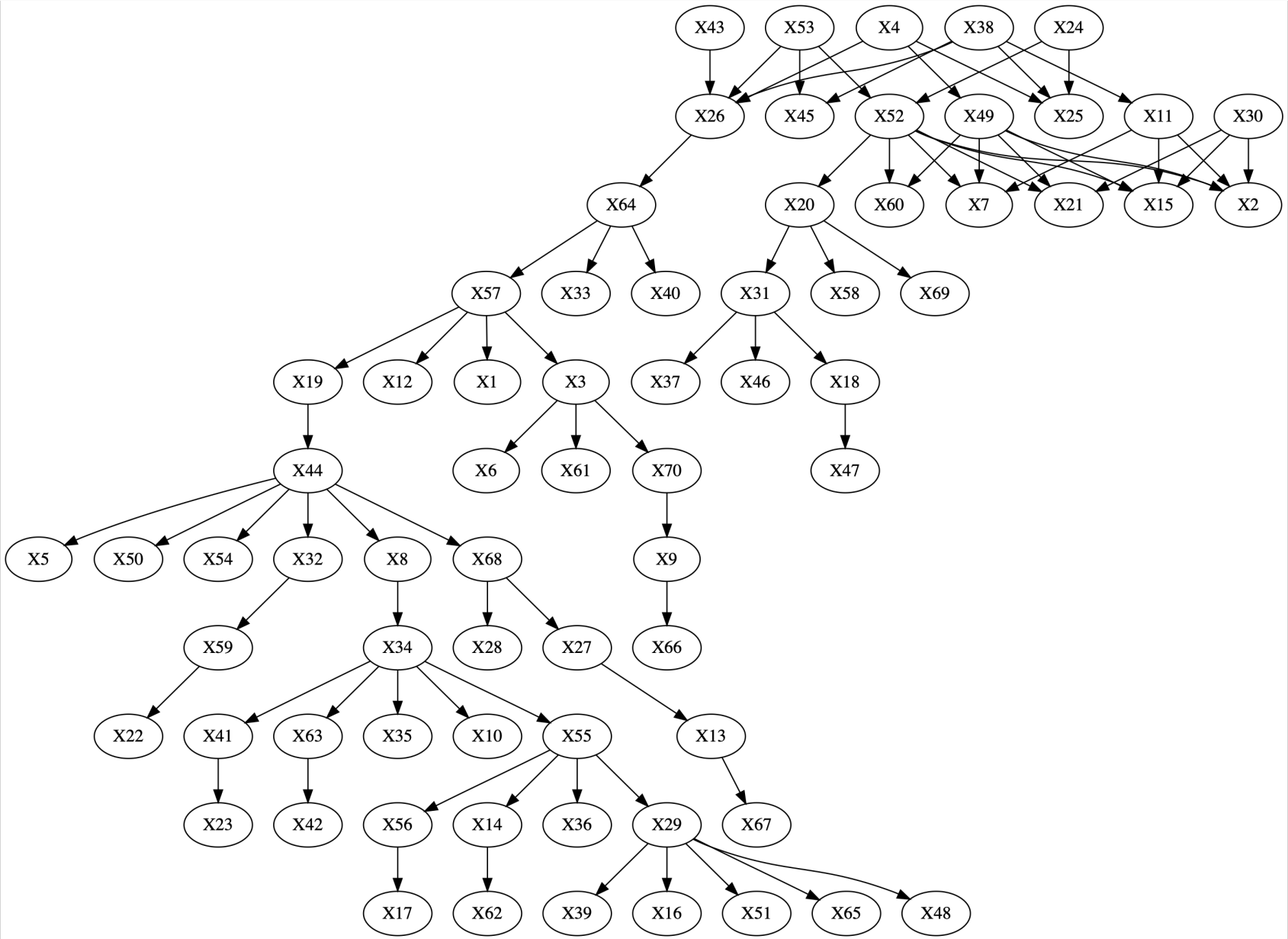
Permutations of constants

If the model has been trained on a given $\mathcal{KB}$, then a permutation function $\sigma : \mathcal{C} \to \mathcal{C}$ has been applied to $\mathcal{KB}$ to obtain $\mathcal{KB}'$, how will the model behave with inputs from $\mathcal{KB}'$?

Knowledge Base (KB)



Permuted Knowledge Base (KB')



Permutations of constants

Length of proofs	MLP test	RNN test
0	17.84%	66.43%
1	0.00%	0.00%
2	0.00%	1.33%
3	0.00%	0.00%
4	0.00%	0.00%
5	0.00%	0.00%
6	0.00%	0.00%
7	0.00%	0.00%
8	0.00%	0.00%
9	0.00%	0.00%
10	0.00%	0.00%

Permutations of constants

Length of proofs	MLP test	RNN test
0	17.84%	66.43%
1	0.00%	0.00%
2	0.00%	1.33%
3	0.00%	0.00%
4	0.00%	0.00%
5	0.00%	0.00%
6	0.00%	0.00%
7	0.00%	0.00%
8	0.00%	0.00%
9	0.00%	0.00%
10	0.00%	0.00%

➔ Next test: how does the model behave w.r.t permutation of constants once it has been exposed to permuted constants?

CONCLUSION

Conclusion & future work

NNs & compositionality:

- ➔ NNs pick up some **structure** from data: some amount of **generalization** in the variations in proof length compositionality tests, sub-proofs play a role in learning;
- ➔ Limited generalization: unseen length experiment, high sensitivity to the order of constants, $\gg$ overall structure of the $\mathcal{KB}$.

Conclusion & future work

NNs & compositionality:

- ➔ NNs pick up some **structure** from data: some amount of **generalization** in the variations in proof length compositionality tests, sub-proofs play a role in learning;
- ➔ Limited generalization: unseen length experiment, high sensitivity to the order of constants, $\gg$ overall structure of the $\mathcal{KB}$.

To be investigated next:

- ➔ what is the representation of the data that the NN builds in training?
- ➔ comparative studies through different dataset sizes, data encodings, other NN architectures.

References I


Bowman, S. R., Manning, C. D., and Potts, C. (2015). Tree-structured composition in neural networks without tree-structured architectures. arXiv preprint arXiv:1506.04834.


Guzmán, M. V., Boritchev, M., Szymanik, J., and Malicki, M. (2022). Compositionality in a simple corpus. In Journées Jointes des Groupements de RechercheLinguistique Informatique, Formelle et de Terrain(LIFT) et
Traitement Automatique des Langues(TAL), pages 55–63.


Hupkes, D., Dankers, V., Mul, M., and Bruni, E. (2020). Compositionality decomposed: how do neural networks generalise? Journal of Artificial Intelligence Research, 67:757–795.


Lovering, C. and Pavlick, E. (2022). Unit testing for concepts in neural networks. arXiv preprint arXiv:2208.10244.


Montague, R. (1973). The proper treatment of quantification in ordinary English. In Approaches to natural language, pages 221–242. Springer.

References II

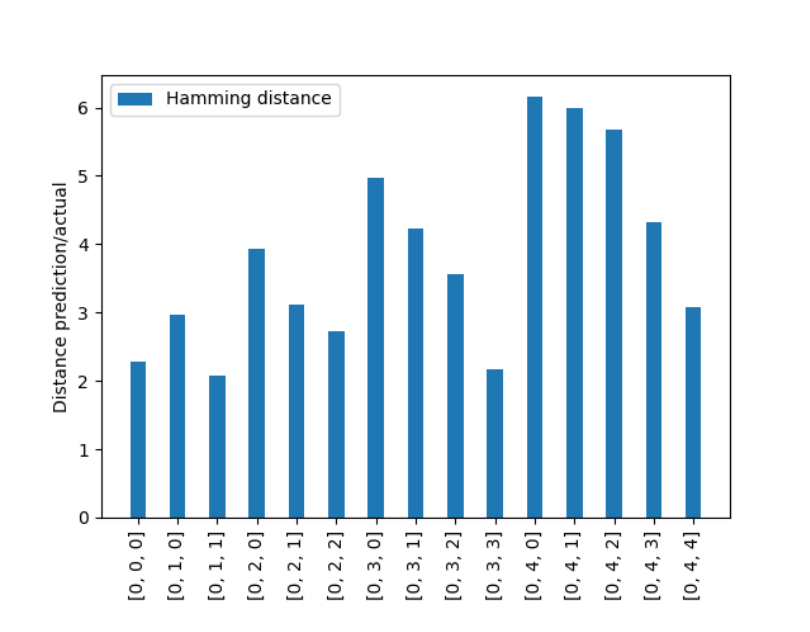

Mul, M. and Zuidema, W. (2019). Siamese recurrent networks learn first-order logic reasoning and exhibit zero-shot compositional generalization. arXiv preprint arXiv:1906.00180.


Ontanon, S., Ainslie, J., Cvicek, V., and Fisher, Z. (2022). Logicinference: A new datasaet for teaching logical inference to seq2seq models. In ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality.

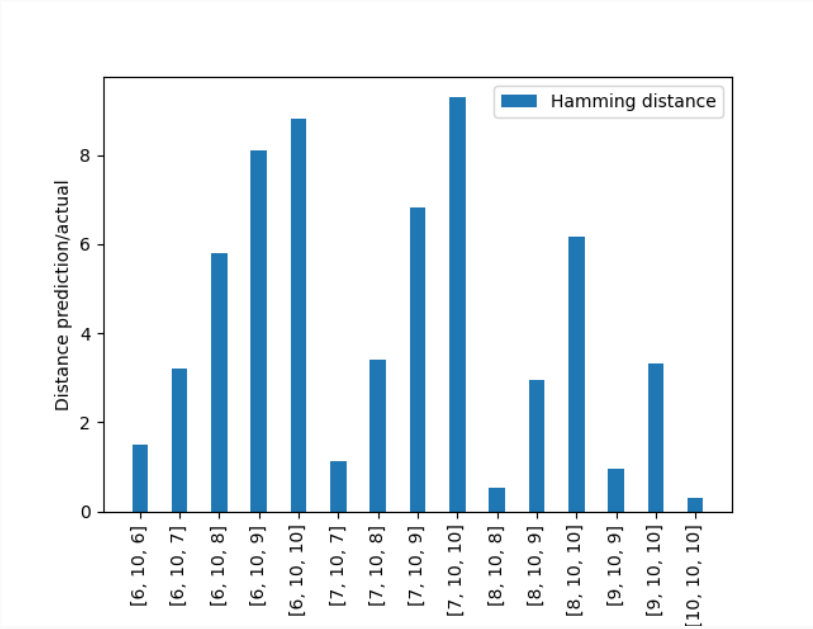

Partee, B. (1984). Compositionality. Varieties of formal semantics, 3:281–311.


Saxton, D., Grefenstette, E., Hill, F., and Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. arXiv preprint arXiv:1904.01557.


Veldhoen, S., Hupkes, D., and Zuidema, W. H. (2016). Diagnostic classifiers revealing how neural networks process hierarchical structure. In CoCo@ NIPS.

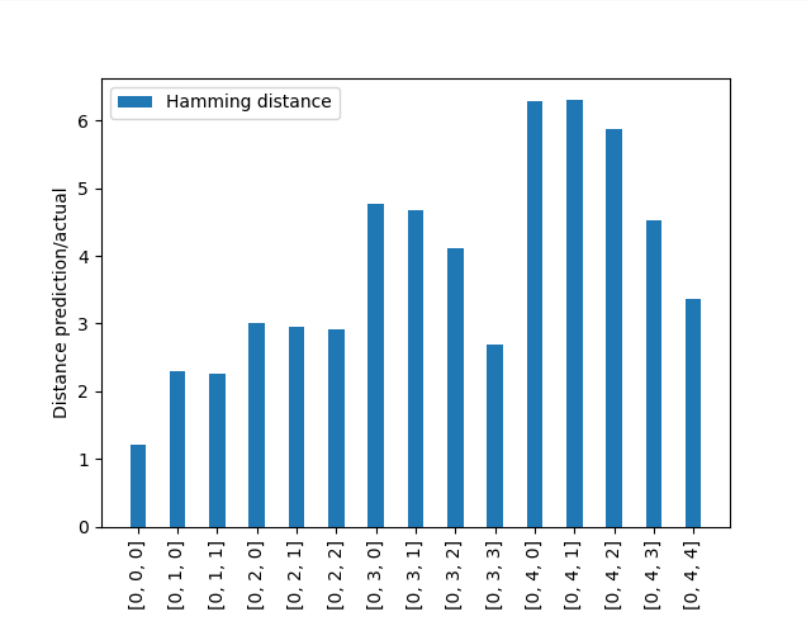


(a) Minimal length of unseen proof: 0

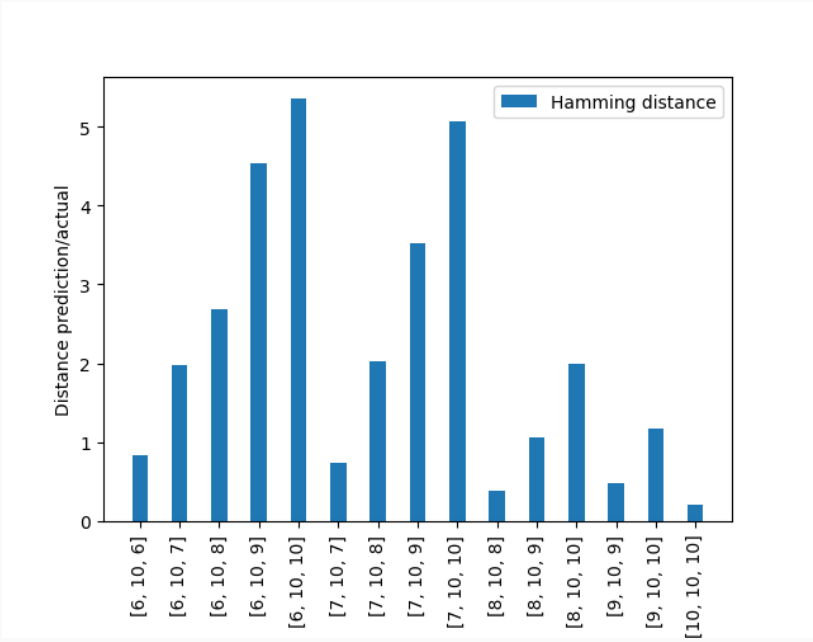


(b) Maximal length of unseen proof: 10

Figure: Average Hamming distances between the expected output and the actual output for the unseen length experiment, MLP



(a) Minimal length of unseen proof: 0



(b) Maximal length of unseen proof: 10

Figure: Average Hamming distances between the expected output and the actual output for the unseen length experiment, RNN

Data encoding

Each formula from input X is encoded in a vector of dimension 2n (where n is the the size of the set $\mathcal{C}$) as a one-hot fashion.

Example: let $\mathcal{C} = \{X_1, X_2, X_3, X_4, X_5\}$, then the formula $X_2 \rightarrow X_5$ is encoded as

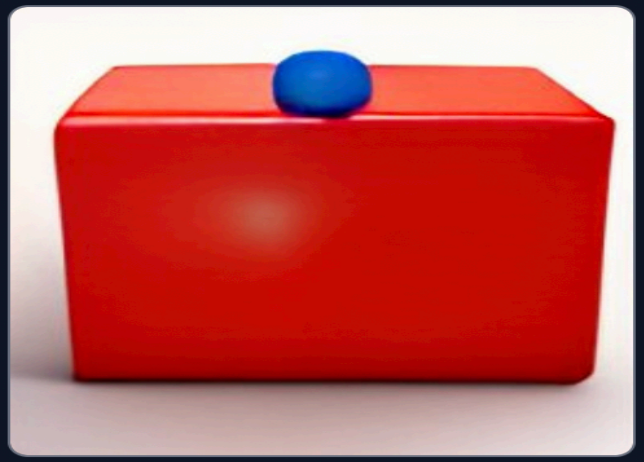
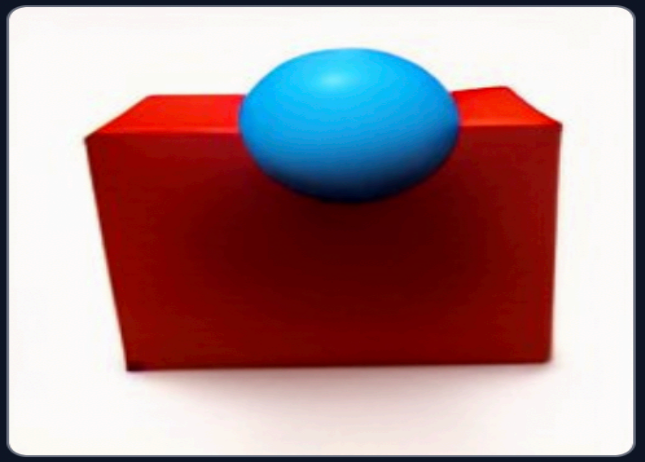
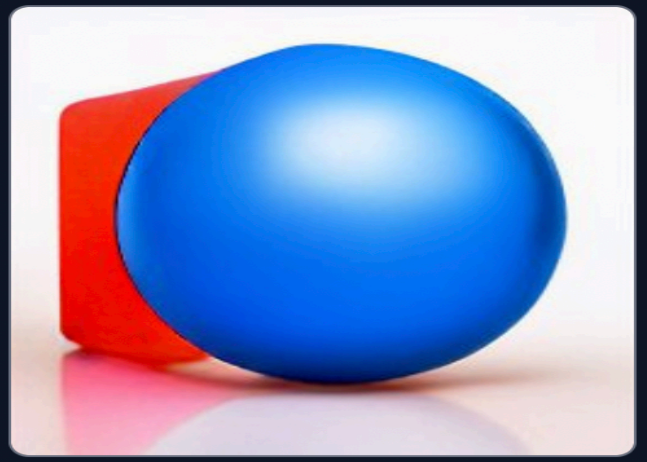
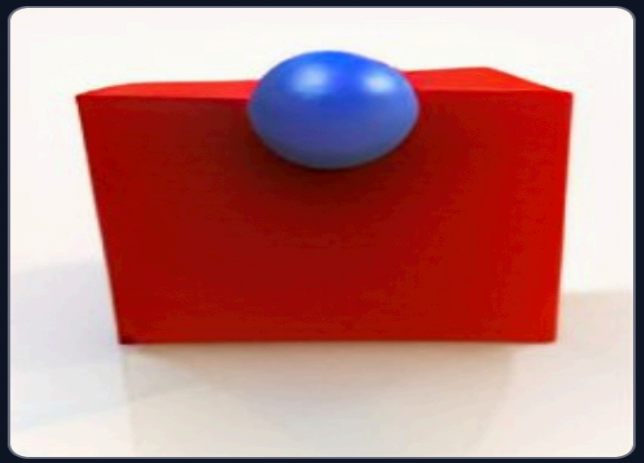
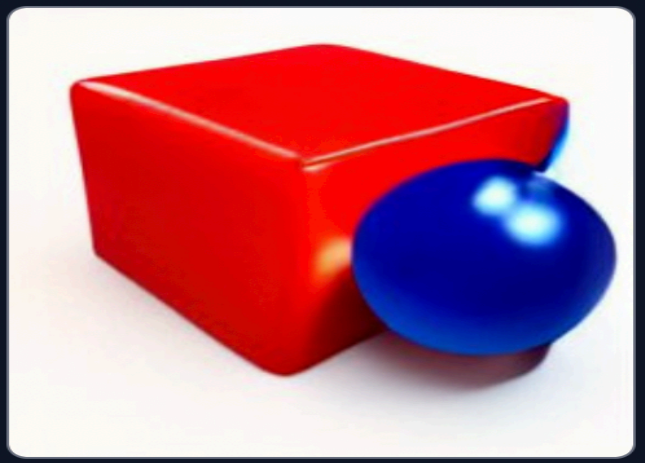
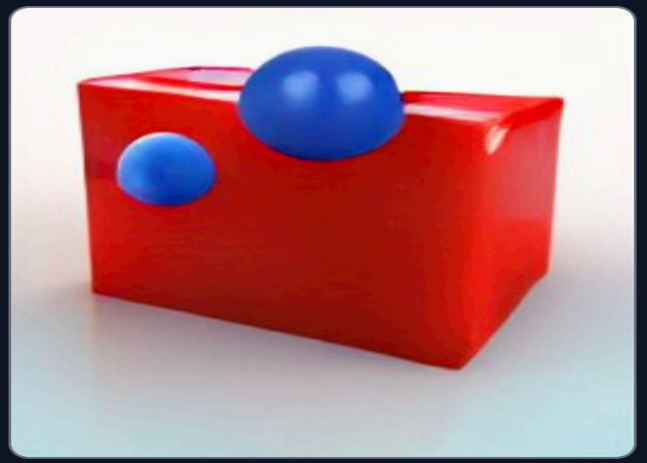
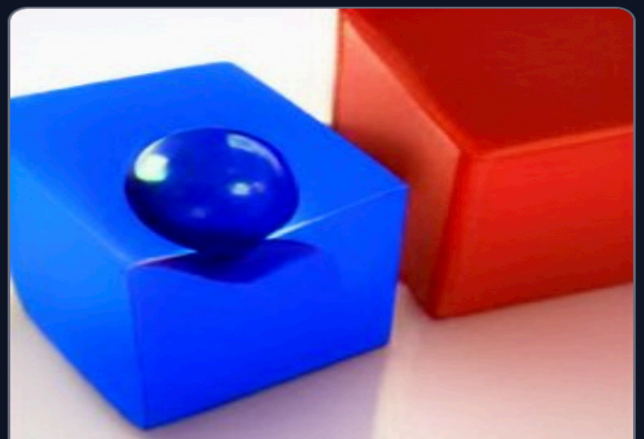
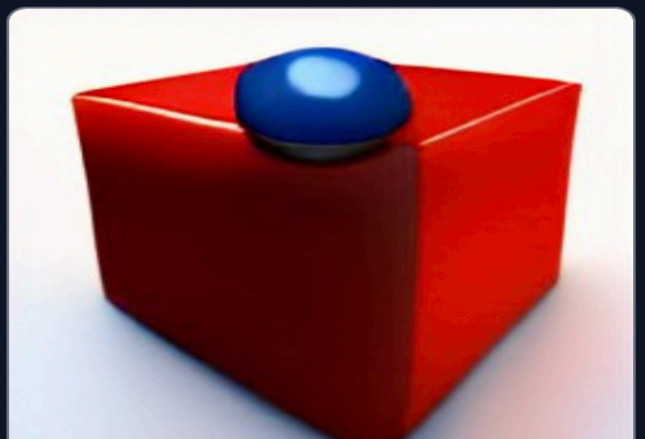
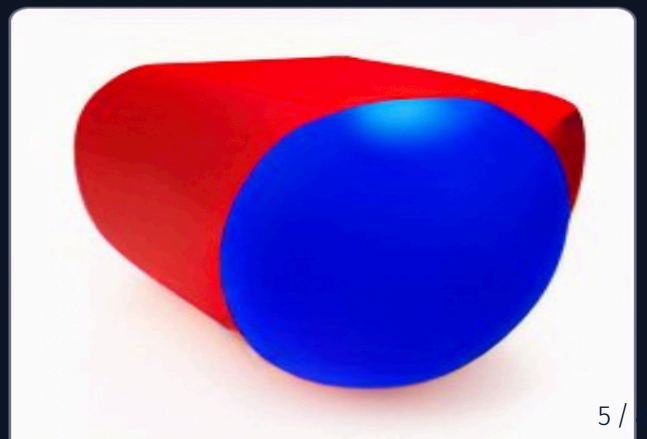
$$[0\ 1\ 0\ 0\ 0 \mid 0\ 0\ 0\ 0\ 1]$$

where the first n digits represent X_2 and the constant X_5 is encoded within the last n bits from the vector.

Neural networks setup

Architecture	MLP	RNN
Optimization algorithm	Adamax algorithm, default learning rate 0.001	same
# hidden layers	1	2
# neurons in each layer	2500	200
Function in hidden layers	hyperbolic tangent function (tanh)	tanh
Function in the output layer	sigmoid function	sigmoid function
# epochs	between 200 and 300 epochs	same
batch size	20	20

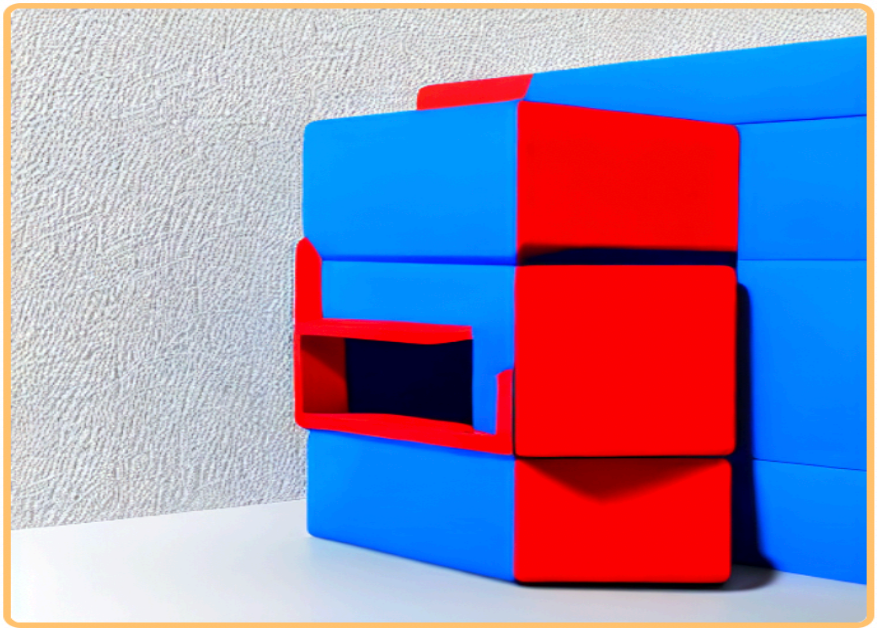
blue ball behind a red cube

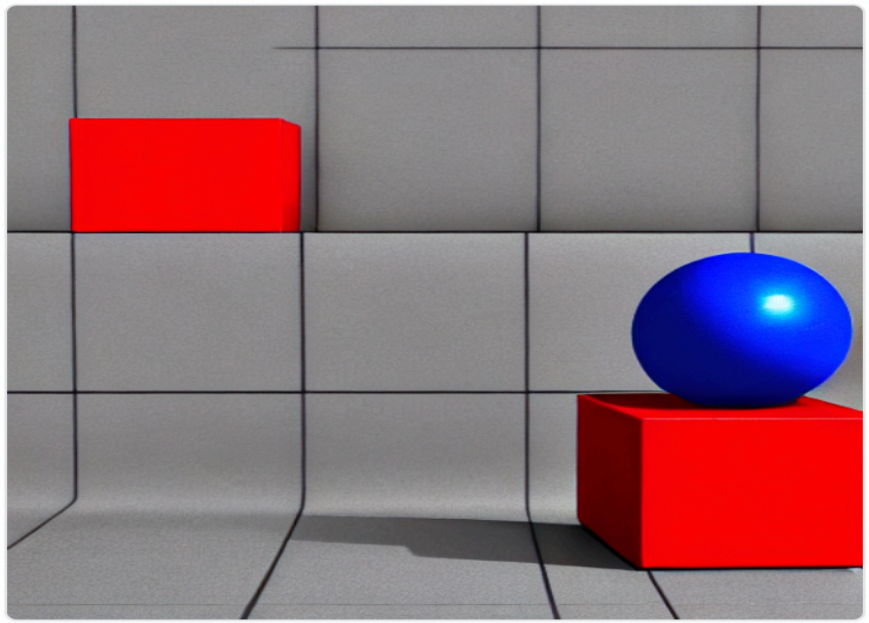










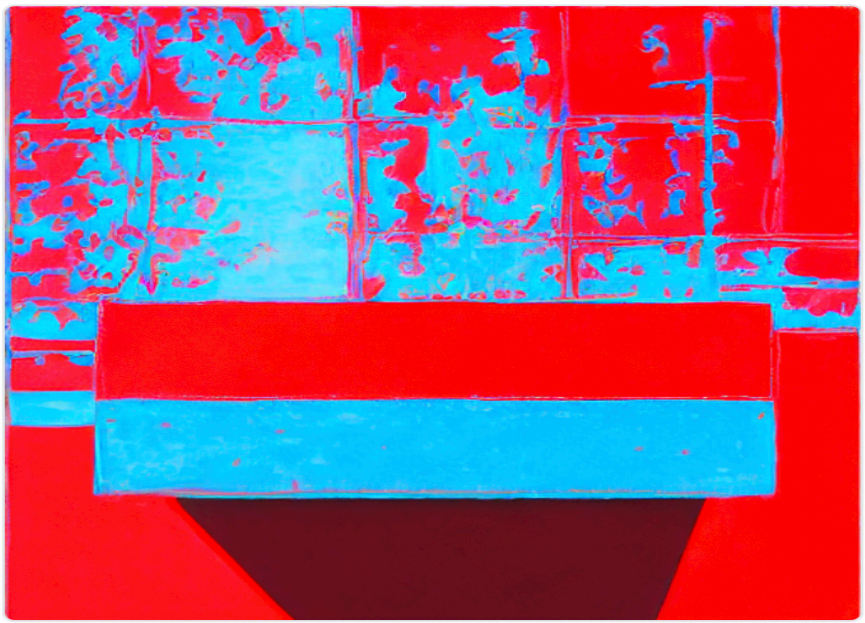
https://huggingface.co/spaces/stabilityai/stable-diffusion

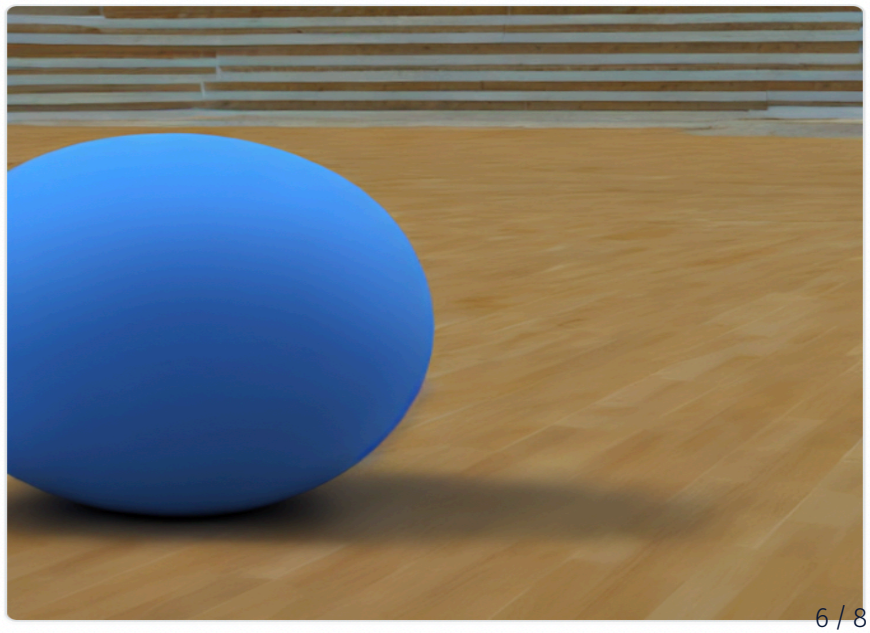
blue ball behind a red cube

Generate image

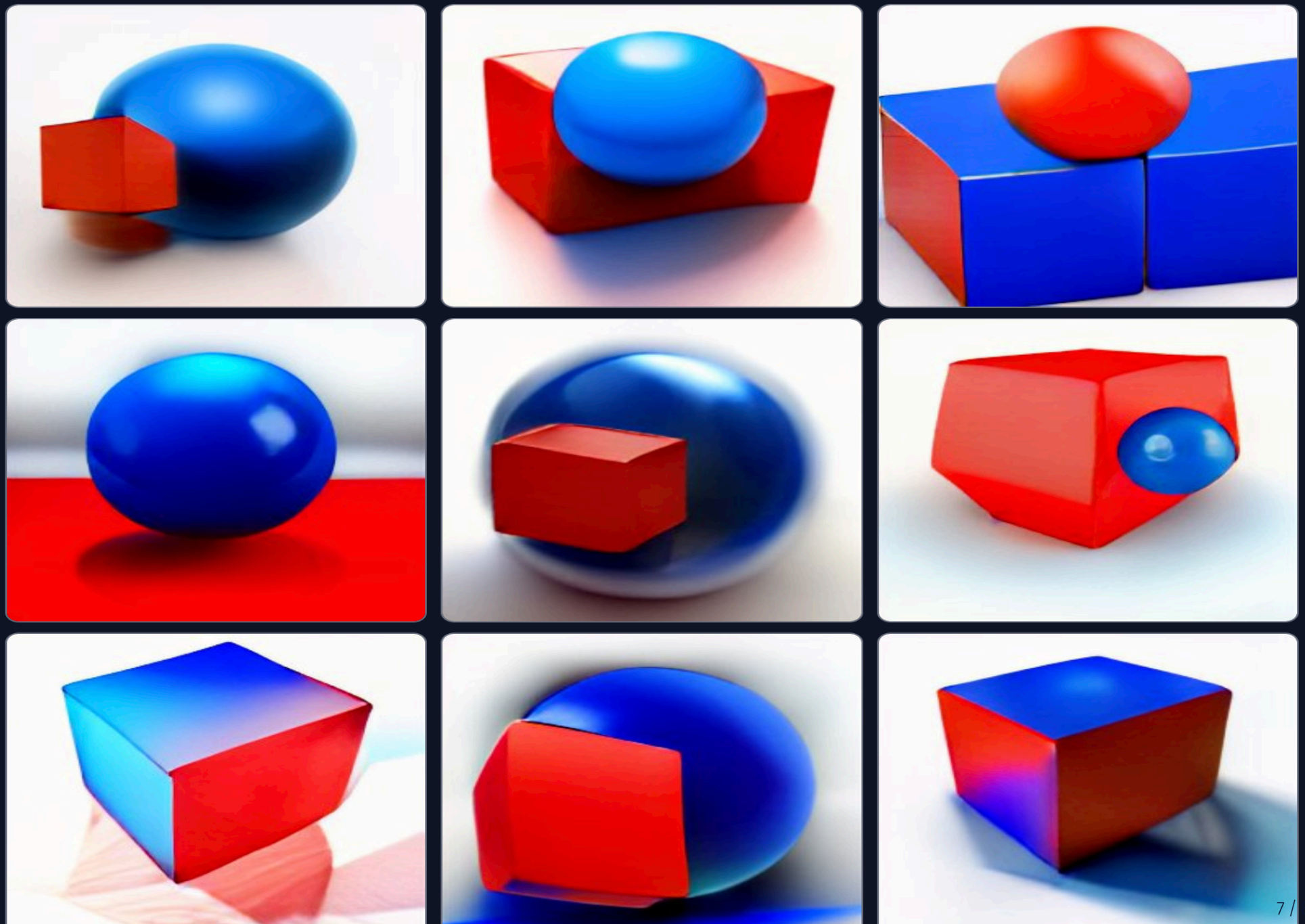








red cube behind a blue ball



https://huggingface.co/spaces/stabilityai/stable-diffusion

red cube behind a blue ball

Generate image