Combining Mental Models with Neural Networks

Paweł Mąka, Jelle W.M. Jansen, Theodor Antoniou, Thomas Peter Maximilian Bahne, Kevin Müller, Can Türktaş, Nico Roos, and Kurt Driessens

Data Science and Knowledge Engineering, Maastricht University,
Maastricht, The Netherlands
{p.maka,jwm.jansen,t.antoniou}@student.maastrichtuniversity.nl,
{t.bahne,kevin.muller,c.turktas}@student.maastrichtuniversity.nl,
{roos,kurt.driessens}@maastrichtuniversity.nl
http://www.maastrichtuniversity.nl/dke/

Abstract. Human reasoning under uncertainty is conjectured to use Mental Models as a representation format. Each Mental Model characterizes a possible state of the world based on and constrained by the available information. Conclusion about the world must hold in each of these Mental Models. An important task in human reasoning is the construction these Mental Models using the available information. This paper investigates whether it is possible to design a neural network architecture that enables the construction of Mental Model, similarly to the conjectured way that humans reason. The paper investigates different architectures in an incremental way. The final architecture not only produces the correct mental models but also learns correct mental models for intermediate representation without being explicitly trained to do so. This contributes to the explainability of the approach.

Keywords: Machine Learning · Mental Models · Neural Networks · Reasoning

1 Introduction

Machine Learning and reasoning have been extensively researched in the past, with different attempts to combine those two research fields by encoding rule sets, with which a neural network learns the ability to reason through specified rules [4,12]. Human reasoning however, is not thought to work with a fixed set of reasoning rules that is encoded in the human natural neural network (the brain). Reasoning in the human mind comes with the concept of **Mental Models (MMs)**, a simplified representation of how humans understand the world [7]. A single MM is an abstract representation, an internal picture, of one distinct possible instantiation of the world. For example, if you are concerned if you can wear your new all-white sneakers the next day under the uncertainty of the weather, you might have two MMs for tomorrow: either it is good weather and you wear your sneakers, or it is bad weather and you will not wear your sneakers. These models allow complex situations to be simplified by getting rid of uncertainty through duplication and help make decisions based on similar situations. Reasoning can be defined as "Algebraic manipulation of previously acquired knowledge in order to answer a new question".[1] Humans can combine two or more mental models representing pieces of information in order to reach a conclusion. For example, knowing that both "$p \vee q$" and "$\neg q$" is true, we can conclude that p must be true. We should therefore be able to generate MMs and algebraically manipulate them in a machine learning setting to produce a conclusion (possibly a single or multiple MMs). Since it is not obvious what a mathematical definition of a MM could be, the first task of this paper is to translate this concept into a machine learning setting.

This paper investigates whether it is possible to design a neural network architecture that enable the construction of Mental Model, similarly to the conjectured way that humans reason. The approach differs from, for instance, neuro-symbolic computing by not explicitly encoding knowledge in link of the neural network. All information / knowledge is provided as input to the neural network. In this investigation, we start with information formulated in Boolean algebra sentences. Of course it is not difficult to create a neural network that answers queries for such inputs. However, that is not the goal of this investigation.

The remainder of this paper is organized as follows. The next section describes related work. Section 3 defines the setting in which we do our research. Section 4 describes the neural network architectures that we have developed and Section 5 describes the experiment that we have performed with these architecture. Section 6 concludes the paper.

2 Related Work

Machine Learning and reasoning originated as separate research fields of Artificial Intelligence in the past, but have recently seen different approaches of combining those in conjuncted research [4,12]. The research field of Neural-Symbolic Computing aims to embed the two most fundamental human cognitive abilities into a system: "the ability to learn from the environment, and the ability to reason from what has been learned." [4] They make use of neural networks, by encoding reasoning rules in between the layers of a neural network.

However, it has been argued that humans reason with the use of MMs, aiming to find conclusions that are true [9]. Those conclusions can be an outcome of a conjunction or a repetition of the premise concerned. Humans search for relations that are not explicitly asserted in the premises, reaching conclusions that seem the most probable [8]. Since the implementation of MMs in this project does not hard-code reasoning rules in between the layers of a neural network, it adds to current research by investigating a more general and flexible approach to reasoning in neural networks.

Since MMs are an integral component of this paper, we repeat some of the theory on MMs. As Johnson-Laird explains, "[...], each mental model represents what is common to a distinct set of possibilities." [8, p. 2]. They do not reflect every detail of that distinct state of the world, but reduce the available information to the aspects necessary for the context. In the sneaker example that was given in Section 1, the only necessary information in the context of deciding whether or not to wear new, white sneakers (without getting them dirty on the first day) are the weather conditions, which can be good or bad in this case. All other information that might be available is not reflected in the two MMs that arise from this context. It does not matter if the person had one or two cups of coffee in the morning, it is irrelevant what was in TV the night before. MMs reduce the mental load by excluding unnecessary information.

To further reduce load on our working memory, humans build MMs on the principle of "truth" [9]. The principle of truth dictates that MMs only represent propositions of the premise that are true and neglect those that are false, i.e. they follow the closed world assumption. For instance, when considering the exclusive disjunction "I can go on holiday or else I can finish the project I am working on", humans would build two MMs according to the principle of truth: "I go on holiday" and "I finish my project". Observe that each model does not include the falsification of the respective other premise. If one would to be precise and construct complete models, we would get "I go on holiday and I don't finish my project" and "I do not go on holiday and I finish my project". However, this shortcut used by our brains results in predictable misjudgement in deduction, which we do not want to imitate with neural networks. Therefore, we will disregard the principle of truth when constructing the MMs for our networks in Section 3.

Another assumption of MM theory states that MMs are iconic. "The structure of a [Mental Model] representation corresponds to the struc-ture of what it represents."[8, p. 2]. This is intuitive, as we think about different topics in different ways. The concept of biological evolution for example is entirely different from theory on radioactive decay. The considerations and dependencies that need to be taken into account greatly change from one context to another, implying that the structure of corresponding MMs also differ. This paper will take all of the three mentioned aspects of MM theory into account when defining the MMs in the next section.

3 Defining the Setting

In this section, the representation of MMs used in this paper will be described. As stipulated by iconicity, a MM needs to resemble the structure of what it represents. Therefore, it is necessary to first fix the context, i.e. whatever it is that the MM should represent. For this paper, a MM will be defined in the context of boolean algebra, because of its scalability, modularity [1], and relative simplicity. The sentences used in the datasets are composed of two simple sub-sentences, which are both assumed to be true. To increase complexity, sub-sentences can be combined with the "and" operator in order to obtain a single more complex sentence. This process can be repeated for further complexity.

[1] This results from the fact that any sentence can be transformed into conjunctive normal form [11, p. 253]

The truth table of a logical sentence represents a possible state of the world and can be interpreted as a MM, which allows us to think about the information contained in a logical sentence and helps us to perform further reasoning tasks. Even though the human mind might not need to fall back on truth tables and rather represents Boolean algebra in a more advanced way [9, p. 114], this does not compromise the validity of using truth tables for this context.

There is one modification to traditional truth tables used in this paper: when a variable does not appear in a sentence, or if the value of the variable does not influence the value of the sentence given the other variables, a value of "none" will be assigned to this variable, instead of "true" or "false". This modification is again inspired by MM theory, which states that MMs reduce the amount of stored information to a minimum in order to preserve cognitive capacity. The value of a variable that does not appear in a sentence has no effect on the value of the sentence. Hence, a single MM which assigns such variable the "none" value contains the same information of two other MMs which are identical to the first, except that the value of the variable is now changed to "true" and "false" respectively.

4 Methodology

This section describes the datasets we created to evaluate the architectures on a Boolean algebra reasoning task and gives a detailed description of the created neural networks. The neural networks can be divided into architectures predicting conclusion in the form of one MM or multiple MMs. This is also reflected in the structure of the datasets.

4.1 Creating the Learning Data

The datasets consist of inputs in the form of *two* logical sub-sentences[2] (Boolean expressions) and labels that represent a single or multiple MMs induced by both of the sub-sentences being true. For all datasets in this paper, it is always assumed that the logical sentence (or both sub-sentences) given as input is "true".

When creating the datasets a parameter representing the "depth" of a logical sentence is used. A sentence can be represented as a tree-structure with the variables in the leafs and non-terminal nodes containing logical operators. This means that a sentence of depth 1 consists of at most one logical operator and two variables. Examples for sentences of depth 1 are "x_1 or x_2" and "not x_3"; and of depth 2 are "x_1 or not x_2" and "(x_1 or x_2) and x_3".

The labels (MMs) use a vector representation. The length of the vector n corresponds to the number of logical variables used in the dataset. In the experiments $n = 5$, the logical variables are denoted with symbols x_1-x_5. Each element in the vector encodes the value of one variable in the MM. A value of 1 indicates

[2] If the evaluated neural network requires exactly one sentence as the input, the two sub-sentences can be concatenated with the "and" operator between them.

that the variable is true in the MM, the value -1 indicates false. Additionally, a variable in the vector can be assigned a value of 0, which corresponds to the “none” value described in Section 3.

All datasets were created by implementing the generate-and-test approach. We randomly generate a sentence and algorithmically determine what MMs are compatible with the sentence. If both the sentence and the resulting MMs (conclusion) fulfil the specifications, the sentence is added to the dataset. These steps were performed a specified number of times.

The first two datasets are called **Many-to-Single MM Small** and **Many-to-Single MM Big** respectively. Examples in these datasets induce one single MM, where any number of variables can be true or false (as long as at least one variable is not “none”). Additionally, The difference between the small and big version is the depth of the two sub-sentences. In Many-to-Single MM Small dataset, sub-sentences have a maximum depth of 1. This leads to a combined sentence of a maximum length of 11 (the length of the sequence of variables, operators and brackets that forms a sentence) with the sub-sentences of length 5 at most. This dataset contains 2369 sentences (consisting of two sub-sentences each). On the other hand, sub-sentences in the Many-to-Single MM Big dataset can have a maximum depth of 2 therefore the maximum length of a sentence is increased to 27. Each subsentence has $2 \cdot 13$ (including the outer brackets) plus 1 for the combining “and" operator. The size of this dataset is 276178 sentences and labels. The third dataset is called **Many-to-Many MM**. For this dataset, no restrictions were put on number of the induced MMs. Again, sentences are made up of two conjuncted sub-sentences, each of depth 2 at maximum. This dataset contains 3489 datapoints. Both Many-to-Single MM Small and Many-to-Many MM datasets consist of all possible sentences fulfilling the conditions (depth and number of conclusion MMs). The size of Many-to-Single MM Big dataset results from running the generate-and-test algorithm until less than 1% of generated sentences were not already included in the dataset.

4.2 Constructing the Architecture

The goal of our design is to encourage the network to not only output vectors which we interpret as MMs, but also to internally use these MMs in order to derive the desired output. In addition to testing if we can increase performance this way, we hope to achieve greater interpretability for the internal reasoning process of the neural network.

For the task of predicting MMs we designed a neural network that accepts two logical sub-sentences as input and predict a single or multiple mental models that are a conclusion of the input (with the assumption that both the sub-sentences are true). The first network we implement in this context is called **Single-mental model Net (Single-mmNet)** and is trained on the Many-to-single MM datasets. In addition, we define two other networks, that go by the names of **Multi-mental model Net with direct input (Multi-mmNet direct)**, and **Multi-mental model Net combination (Multi-mmNet combination)**. These networks are trained on the Many-to-multiple MM dataset.

To encourage our networks to internally use MMs, sub-sentences are fed into a shared sub-sentence encoder, before a final reasoning module combines the outputs for each sub-sequence (see Figure 1). Conceptually, we want our network to generate the MMs for every sub-sentence before merging the information in those sub-sentences in the final reasoning module. This concept is rooted in the modularity property described in Section 3. We introduce the inference layer that combines the MMs induced by sub-sentences. While not explicitly forced into a specific representation for sub-sentence MMs, the sub-sentence encoder adopts our definition of MMs during training.

To implement this architecture, it is sufficient to use a simple feed-forward architecture with an embedding layer and one fully-connected hidden layer. The activation functions were set to hyperbolic tangent for the output layer and ReLU for the hidden layer. The output is reshaped to a matrix $Y \in \mathbb{R}^{M \times D}$, where M is a constant number of MMs (specified as a hyper-parameter) and D is the number of logic variables (five in our case).

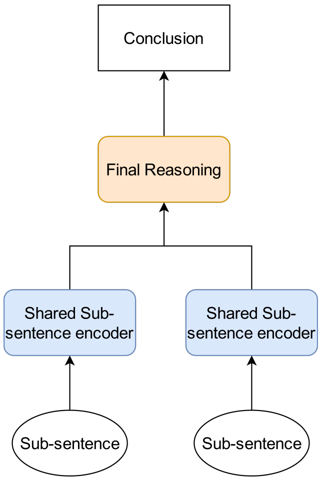


Fig. 1: General architecture of neural network consisting of shared sub-sentence encoders and a final reasoning module

Predicting a Single Mental Model The Single-mmNet architecture uses a fully connected network as a sub-sentence encoder and the reasoning module described in detail below. It is trained end-to-end using standard gradient-based optimization on the Many-to-single MM dataset.

The reasoning module dubbed "MM-Inference Layer" is meant to combine two outputs of the sub-sentence encoder (one output for every sub-sentence). The inference layer accepts two matrices, each representing the MMs induced by one sub-sentence. We denote the matrices with $Y^1 \in \mathbb{R}^{M \times D}$ and $Y^2 \in \mathbb{R}^{N \times D}$, where

M and N are the numbers of MMs induced by the first and second sub-sequence respectively, and D is the number of variables. It is assumed that all elements of the two matrices are in the range $[-1, 1]$. Let Y^1_m for $m \in \{1, 2, \ldots, M\}$ denote a MM model from the first sub-sentence (i.e. a row of Y^1) and Y^2_n for $n \in \{1, 2, \ldots, N\}$ a MM model from the second sub-sentence (i.e. a row of Y^2). For every pairing m, n, we calculate two quantities: $V_{m,n} \in \mathbb{R}^D$ and $C_{m,n} \in \mathbb{R}$, which we call *value* and *correctness*. To obtain the *value* $V_{m,n}$ between two MMs, we simply add the two MMs element wise and “clamp"[3] the resulting numbers between -1 and 1.

This approach disregards the situation when two MMs are incompatible with each other. Two MMs are incompatible when the same variable is true in one model and false in the other. (We will sometimes refer to such a variable as an incompatible variable). To indicate when two MMs are incompatible, we introduce *correctness*. For “perfect” values for variables of either exactly -1, 0, or 1, the *correctness* of a pair of MMs will be 1 if the two models are compatible (i.e. no variable is true in one of the models and false in the other) and 0 otherwise. During training and testing however, the sub-sentence encoder could assign any value between -1 and 1 to the variables. As a consequence, the correctness becomes a number between 0 and 1. Therefore in practice, two MMs become increasingly incompatible, as the absolute difference between the variables increases. The two quantities (value and correctness) are calculated using Eq. 1 and 2 respectively.

$$\begin{aligned} &V_{m,n} = \min(1,\ \max(-1, Y^1_m + Y^2_n)) \\ &\forall m = 1, ..., M, \quad \forall n = 1, ..., N \end{aligned} \tag{1}$$

$$\begin{aligned} &C_{m,n} = \prod_{d=1}^{D} [1 - \max(0,\ |Y^1_m - Y^2_n| - 1)]_d \\ &\forall m = 1, ..., M, n = 1, ..., N \end{aligned} \tag{2}$$

The Single-mmNet network only outputs one MM, which is calculated using Eq. 3. In essence, Z is a sum of all values weighted with their respective correctness and normalised by the sum of all correctness.

$$Z = \frac{\sum_{m=1,n=1}^{M,N} V_{m,n} \cdot C_{m,n}}{\sum_{m=1,n=1}^{M,N} C_{m,n}} \tag{3}$$

The use of a fully-connected network as a sub-sentence encoder means that the number of MMs is set beforehand and identical for each sub-sentence. To allow a variable number of sub-sentence MMs, we added a second type of output to the fully-connected network - scores $S^1 \in \mathbb{R}^M$ and $S^2 \in \mathbb{R}^N$ for the first and second sub-sentence respectively. Each MM has a corresponding score, where the value of 1 indicates that the MM is correct and contains important information and value of 0 means that the MM is erroneous or redundant. In contrast to value

[3] Clamping indicates setting all values < -1 to -1 and all values > 1 to 1

and correctness, scores are taken directly from the outputs of sub-sentence encoders. The MM-Inference Layer is subsequently adapted to accept these scores as additional inputs and take them into account when calculating the output - Eq. 4 shows the formulation used. The introduction of scores does not change the dimension of the output $Z^s \in \mathbb{R}^D$. It was empirically found that normalizing the sum by the summed correctness (hence without scores) yields more stable results in terms of test accuracy.

$$Z^s = \frac{\sum_{m=1,n=1}^{M,N} V_{m,n} \cdot C_{m,n} \cdot S_m^1 \cdot S_n^2}{\sum_{m=1,n=1}^{M,N} C_{m,n}} \tag{4}$$

Predicting Multiple Mental Models Allowing more MMs as a conclusion is inherently a many-to-many problem. For this problem we propose two encoder-decoder architectures built around a modified version of the MM-Inference Layer for the encoder part, and an LSTM layer for the decoder part. Both models use the same shared fully-connected sub-sentence encoder with scores. The MM-Inference Layer is modified to produce values (see Eq. 1) and scores based on correctness and input scores as defined in Eq. 5.

$$S_{m,n} = C_{m,n} \cdot S_m^1 \cdot S_n^2 \tag{5}$$

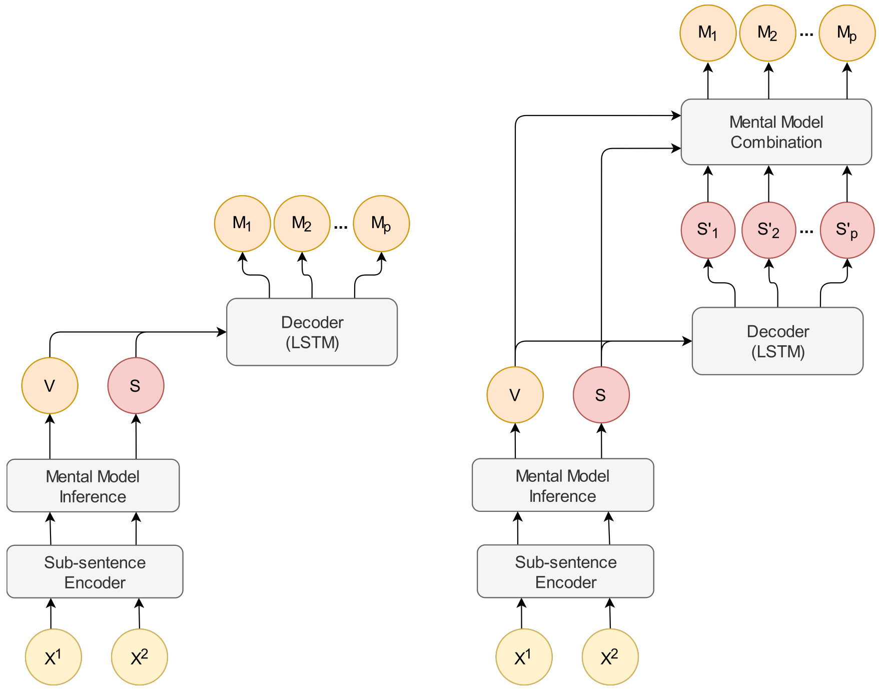
The output values V and scores S are flattened and concatenated, and are used as the initial hidden state and cell state of the LSTM in the decoder part. The output of the LSTM feeds into the fully-connected layer.

In the *first* architecture **Multi-mmNet (direct output)** the fully-connected layer has a number of neurons equal to the number of variables n and uses a hyperbolic tangent activation function. The outputs of this layer are interpreted as predicted MMs (see Figure 2a), and are auto-regressively fed back as the input for the prediction of the next MM. We stop the model when the end-of-sequence token is reached.

In the *second* architecture **Multi-mmNet (combination)** the fully-connected layer has a number of neurons equal to the number of mental models returned by the encoder $(M \cdot N)$, and a sigmoid activation function. Its output S' is interpreted as scores for combining the MMs obtained from the MM-Inference Layer of the decoder part. This combination happens through a MM-Combination Layer that computes the sum of MMs weighted by the scores predicted by the decoder (see Figure 2b). The following shows the calculation for the p-th output of the network:

$$Z_p^c = \frac{\sum_{i=1}^{M \cdot N} V_i \cdot S_i \cdot S'_{p,i}}{\max(1, \sum_{j=1}^{M \cdot N} S_j \cdot S'_{p,j})}. \tag{6}$$

To avoid division by 0, we do not allow the deliminator to be less than 1. During training we use teacher forcing [14,5], where the training data is used as the input to the decoder instead of the output generated by the network in the previous step. During inference the resulting output is used auto-regressively



(a) Multi-mmNet (direct output) (b) Multi-mmNet (combination)

Fig. 2: Schematic connections of Multi-mmNet (direct output) architecture on the left and Multi-mmNet (combination) on the right, where X^1 and X^2 are the input sub-sentences, M_1 to M_p are the output MMs, and S'_1 to S'_p are the scores returned by the decoder of the Multi-mmNet (combination).

to predict the next output, until the end-of-sequence token is reached. This technique is used to mitigate the network instability, and make it converge faster. In the experiments, we chose a MM containing only 0s as the end-of-sequence token for both architectures.

5 Experiments

Each dataset is split into training, validation and test subsets according to 80%, 10% and 10% ratios respectively. Training used the Adam optimizer [10] and a mean squared error as loss function for all models. Training was terminated early based on the validation loss.

5.1 Many-to-single Mental Model Architectures

As discussed above the Single-mmNet architecture predicts one MM given two sub-sentences. The number of mental models of the fully-connected sub-sentence encoder was set to 3 for the Many-to-Single MM Small dataset experiments. For the Many-to-Single MM Big dataset it was experimentally found that using 6 mental models resulted in the best performance. The network was evaluated both without and with the use of scores. With the use of scores, the Single-mmNet with scores theoretically has the ability to distinguish which MM is important and which is not. In practice however, scores converge to 1.0 and are not used by the network as intended. For comparison we trained a standard LSTM [6] on the datasets concatenating the two sub-sentences using an *and* operator. We choose the LSTM as baseline after empirically comparing alternatives on the simple task of predicting a single true variable in a logical sentence.[4]

Results reported on the small version Many-to-Single MM dataset summarise 6 repetitions of the experiment, while those on the big version stem from 3 repetitions. Because Single-mmNet with scores performed better than the one without, only this architecture was evaluated on the big dataset.

Observations and Discussion Table 1 summarises the performances. The addition of scores seems to increase the robustness of the single-mmNet architecture, despite the fact that they converged to 1.0 during training. The model with scores achieved similar accuracy as the LSTM.

Table 1: Experiments with the small and big version of the Many-to-single MM dataset

Model	**Average Accuracy**	
	Small Dataset	**Big Dataset**
LSTM	100%	99.80%
Single-mmNet without scores	96.84%	-
Single-mmNet with scores	100%	99.96%

The LSTM and Single-mmNet with scores both reached perfect accuracy on the small version of the dataset. They also performed very well on the big dataset with an average accuracy of 99.80% and 99.96% respectively. The Single-mmNet without scores reached this perfect accuracy only in some runs (on the small dataset).

While the scores don't seem to fulfill the purpose of indicating which MM is relevant, their use improved accuracy and stability of the architecture, possibly

[4] The LSTM outperformed a Vanilla RNN [3], a GRU [2], and a simplified Transformer [13] that consisted of the encoder part of the Transformer with a fully-connected output layer.

being of use during the training phase, so it was decided to keep them in the subsequent, more complex architectures as well.

5.2 Many-to-many Mental Model Architectures

Many sentences imply multiple MMs, so the Many-to-Many MM dataset was used to reflect this fact. Based on the performance of the Vanilla LSTM, the Single-mmNet architecture was expanded by feeding the encodings into an LSTM decoder as discussed in Section 4.2. The MMs and scores are used as initialisation of the LSTM-decoder, which outputs MMs directly for "Multi-mmNet (direct output)" or outputs scores used in the Mental Model Combination Layer for "Multi-mmNet (combination)". The number of mental models of the fully-connected sub-sentence encoder was once again set to 3.

As benchmarks, we used encoder-decoder networks based on an LSTM. The first two architectures use sub-sentence representations where one uses specific Start of Sequence (SOS) and End of Sequence (EOS) tags, while the other does not. The third model uses a symbolic concatenation of the two sub-sentences and without specific SOS/EOS tags.

Observations and Discussion The performance of the Multi-mmNet architectures (both "direct output" and "combination") and benchmark networks can beeen seen in Table 2. These results are achieved after fine-tuning the models' parameters. A perfect accuracy was achieved by the model outputting MMs and an average 99.67% accuracy by the model outputting scores for combining MMs, actually reaching perfect accuracy 4 out of 6 times.

The two LSTM networks using sub-sentences exhibit the similar accuracy, while the encoder-decoder model using symbols performed even slightly better, but the differences are very small. In fact, all models performed comparable to the benchmarks when judging accuracy. Beside predicting the correct MM, the model predicts the MMs in the same order they were listed in the dataset. This is an expected result for an LSTM network.

Table 2: Experiments with Many to Many dataset

Model	**Average Accuracy**
LSTM – Encoder decoder sub sentences no start index	99.70%
LSTM – Encoder decoder sub sentences with start index	99.70%
LSTM – Encoder decoder symbol no index	99.95%
Multi-mmNet (direct output)	100%
Multi-mmNet (combination)	99.67%

Contrary to single-mmNet with scores however (see Section 5.1) Multi-mmNet (combination) *does* use the scores to mark the importance of the MMs. This was not observed for the Multi-mmNet (direct output) - the sub-sentence encoder of this model did not output MMs at all. Using MMs as output of the decoder, resulted in reverting back to an uninterpretable latent representation of sub-sentences, not unlike the LSTM benchmark models. In case of Multi-mmNet (combination) however, without being explicitly trained to do so, the sub-sentence encoder produced MMs as we hypothesized they could be used.

5.3 Sub-sentence encodings

To illustrate the MMs (and scores if applicable) produced by the fully-connected sub-sentence encoder, Table 3 shows the rounded outputs for a selection of sub-sentences. The outputs of Single-mmNet without scores are easily interpreted as MMs corresponding to the sub-sentence. When only one MM is sufficient to represent the information in the sub-sentence, it is copied to all three outputs. Despite using scores to allow the network to use less MMs for each sub-sentence in the Single-mmNet with scores, the network learned to output all scores as 1.

The encoder of Multi-mmNet (direct output) did not learn to output MMs at all. Although the network achieves perfect accuracy, the output of the sub-sentence encoder is not easily interpreted: the vectors do not correspond to MMs of sub-sentences, with scores close to 0 for all outputs. The introduction of the Mental Model Combination layer in Multi-mmNet (combination) enabled sub-sentence encoder to output MMs, and subsequently improved the interpretability of the encodings. Additionally, the encoder learned to use less outputs by setting corresponding scores to 0. That said, the encoder still sets two scores to 1 for most of the sub-sentences, therefore the duplication of MMs is still present in the output.

The networks were trained in end-to-end fashion and were not directly optimized to internally employ MMs. The usage of MMs as an intermediate representation is imposed through MM-Inference Layer in all three architectures exhibiting this behaviour. In case of these architectures - and in contrast to the Multi-mmNet (direct output) - this layer is the last (output) layer of the networks, which were trained to predict MMs. The layer preserves the diminsionality of the input as it is being processed, and the processing itself was designed utilize of the semantics of the introduced representation of MMs. This leads to a substantial improvement for the interpretability of the latent space of the proposed architectures.

6 Conclusion

This paper investigated enabling neural networks to make use of Mental Models for solving reasoning tasks. We conclude that it is possible to construct and train neural network architecture to generate Mental Models for the input information. This can been done by introducing vector encoding of Mental Models,

Table 3: Rounded output of the sub-sentence encoder in MM architectures

Architecture	Sub-sentence	Y_1	S_1	Y_2	S_2	Y_3	S_3
Single-mmNet without scores	(x_2 or x_1)	[1, 1, 0, 0, 0]	-	[-1, 1, 0, 0, 0]	-	[1, -1, 0, 0, 0]	-
	not x_1	[-1, 0, 0, 0, 0]	-	[-1, 0, 0, 0, 0]	-	[-1, 0, 0, 0, 0]	-
	x_5	[0, 0, 0, 0, 1]	-	[0, 0, 0, 0, 1]	-	[0, 0, 0, 0, 1]	-
	(x_1 and x_5)	[1, 0, 0, 0, 1]	-	[1, 0, 0, 0, 1]	-	[1, 0, 0, 0, 1]	-
	(x_3 and x_2)	[0, 1, 1, 0, 0]	-	[0, 1, 1, 0, 0]	-	[0, 1, 1, 0, 0]	-
	(x_2 or x_1)	[1, 1, 0, 0, 0]	-	[-1, 1, 0, 0, 0]	-	[1, -1, 0, 0, 0]	-
	(x_1 and x_3)	[1, 0, 1, 0, 0]	-	[1, 0, 1, 0, 0]	-	[1, 0, 1, 0, 0]	-
	not x_3	[0, 0, -1, 0, 0]	-	[0, 0, -1, 0, 0]	-	[0, 0, -1, 0, 0]	-
	x_1	[1, 0, 0, 0, 0]	-	[1, 0, 0, 0, 0]	-	[1, 0, 0, 0, 0]	-
Single-mmNet with scores	(x_2 or x_1)	[1, -1, 0, 0, 0]	1	[-1, 1, 0, 0, 0]	1	[1, 1, 0, 0, 0]	1
	not x_1	[-1, 0, 0, 0, 0]	1	[-1, 0, 0, 0, 0]	1	[-1, 0, 0, 0, 0]	1
	x_5	[0, 0, 0, 0, 1]	1	[0, 0, 0, 0, 1]	1	[0, 0, 0, 0, 1]	1
	(x_1 and x_5)	[1, 0, 0, 0, 1]	1	[1, 0, 0, 0, 1]	1	[1, 0, 0, 0, 1]	1
	(x_3 and x_2)	[0, 1, 1, 0, 0]	1	[0, 1, 1, 0, 0]	1	[0, 1, 1, 0, 0]	1
	(x_2 or x_1)	[1, -1, 0, 0, 0]	1	[-1, 1, 0, 0, 0]	1	[1, 1, 0, 0, 0]	1
	(x_1 and x_3)	[1, 0, 1, 0, 0]	1	[1, 0, 1, 0, 0]	1	[1, 0, 1, 0, 0]	1
	not x_3	[0, 0, -1, 0, 0]	1	[0, 0, -1, 0, 0]	1	[0, 0, -1, 0, 0]	1
	x_1	[1, 0, 0, 0, 0]	1	[1, 0, 0, 0, 0]	1	[1, 0, 0, 0, 0]	1
Multi-mmNet (direct output)	(x_2 or x_1)	[1, 0, 0, -1, -1]	0	[0, 1, 0, -1, 0]	0	[0, -1, 1, 0, 0]	0
	not x_1	[1, 1, -1, 1, -1]	0	[-1, 1, 1, 0, 0]	1	[-1, 1, 0, 1, 1]	0
	x_5	[1, 0, 1, 1, -1]	1	[1, 0, 0, 1, 0]	1	[0, 1, 1, 1, 0]	0
	(x_1 and x_5)	[0, -1, 1, 1, -1]	1	[1, 0, 0, 1, 1]	1	[1, -1, 1, 1, 0]	1
	(x_3 and x_2)	[1, 1, -1, -1, -1]	0	[1, 1, -1, 0, 0]	0	[1, 1, -1, 0, -1]	1
	(x_2 or x_1)	[1, 0, 0, -1, -1]	0	[0, 1, 0, -1, 0]	0	[0, -1, 1, 0, 0]	0
	(x_1 and x_3)	[-1, 1, -1, 0, -1]	1	[1, 0, 0, 1, 1]	1	[1, -1, -1, 0, -1]	1
	not x_3	[0, -1, 1, -1, -1]	1	[-1, 0, 1, 0, 0]	1	[0, 1, 1, 0, 1]	0
	x_1	[-1, 0, 0, -1, -1]	1	[1, 0, 0, 1, 1]	1	[1, -1, 1, 0, 0]	1
Multi-mmNet (combination)	(x_2 or x_1)	[1, 0, 0, 0, 0]	1	[-1, 1, 0, 0, 0]	1	[1, 1, 0, 0, 0]	0
	not x_1	[0, 1, 1, 0, 1]	0	[-1, 0, 0, 0, 0]	1	[-1, 0, 0, 0, 0]	1
	x_5	[-1, 1, 1, 1, 0]	0	[0, 0, 0, 0, 1]	1	[0, 0, 0, 0, 1]	1
	(x_1 and x_5)	[1, 1, 1, 1, 1]	0	[1, 0, 0, 0, 1]	1	[1, 0, 0, 0, 1]	1
	(x_3 and x_2)	[-1, 1, 1, 1, 1]	0	[0, 1, 1, 0, 0]	1	[0, 1, 1, 0, 0]	1
	(x_2 or x_1)	[1, 0, 0, 0, 0]	1	[-1, 1, 0, 0, 0]	1	[1, 1, 0, 0, 0]	0
	(x_1 and x_3)	[1, 1, 1, 1, 1]	0	[1, 0, 1, 0, 0]	1	[1, 0, 1, 0, 0]	1
	not x_3	[0, 1, 1, 0, 0]	0	[0, 0, -1, 0, 0]	1	[0, 0, -1, 0, 0]	1
	x_1	[1, 1, 1, 0, 1]	0	[1, 0, 0, 0, 0]	1	[1, 0, 0, 0, 0]	1

and formulating neural network layers that perform differentiable operations to combine those encodings. By incorporating those layers with existing neural networks, we created several architectures and trained them using gradient-based methods for the Boolean algebra reasoning tasks. All proposed neural networks achieved accuracy comparable to existing architectures. Additionally, three out of four architectures exhibited the internal usage of Mental Models in the latent space (the exception was Multi-mmNet with direct output). Only when there exists a direct path through the MM layers to the output, which is also an encoded MM, we observe that the simple sub-sentence encoder learned to output human-interpretable encodings - even though we trained the architectures in the end-to-end fashion. We see this fact as the advantage of those architectures as it can be used to achieve greater explainability of neural networks. The code-base of the project can be found on Github. [5]

Currently mental models are being processed in a specific order in the neural networks. The networks are good at predicting what to expect. However, in real world problems, the order of the mental models is irrelevant. This could be solved by using a permutation invariant loss function but is left for future work. A restriction of our research is how this theoretical setting can be translated to a real world problem. In this work, a specific Boolean algebra problem was explored. The presented experiments were intended as a proof-of-concept and the experiments involving larger datasets (in terms of both the number of variables and the depth of the Boolean expressions) should be conducted. The difficulties could arise when the architectures are adapted to accept other forms of input (ultimately, natural language). Additionally, our architecture is limited to reason from exactly two sub-sentences. This is left for future research.

References

1. Bottou, L.: From machine learning to machine reasoning. Machine Learning **94**(2), 133–149 (Feb 2014). https://doi.org/10.1007/s10994-013-5335-x
2. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics (Oct 2014). https://doi.org/10.3115/v1/D14-1179
3. Elman, J.L.: Finding structure in time. Cognitive Science **14**(2), 179–211 (1990). https://doi.org/10.1016/0364-0213(90)90002-E
4. Garcez, A., Gori, M., Lamb, L., Serafini, L., Spranger, M., Tran, S.: Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. Journal of Applied Logics – IfCoLog Journal of Logic and their Applications (FLAP) **6**, 611–632 (2019)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), `http://www.deeplearningbook.org`
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**, 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
7. Johnson-Laird, P.: The history of mental models, pp. 179–212 (09 2004). https://doi.org/10.4324/9780203506936
8. Johnson-Laird, P.N.: Mental models and human reasoning. Proceedings of the National Academy of Sciences **107**(43), 18243–18250 (2010). https://doi.org/10.1073/pnas.1012933107
9. Johnson-Laird, P.N.: How we reason. Oxford University Press, USA (2006). https://doi.org/10.1093/acprof:oso/9780199551330.003.0028

[5] `github.com/Pawel-M/Machine-Learning-and-Reasoning`

10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR (2015)
11. Russell, S., Norvig, P.: Artificial intelligence: a modern approach (2002). https://doi.org/10.1145/201977.201989
12. Shi, S., Chen, H., Ma, W., Mao, J., Zhang, M., Zhang, Y.: Neural logic reasoning. p. 1365–1374. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3411949
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
14. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation **1**(2), 270–280 (1989). https://doi.org/10.1162/neco.1989.1.2.270