Learning Arithmetic from Handwritten Images with the Aid of Symbols

Daniel L. Silver[1], Ahmed Galila[1]

[1]*Acadia University, Wolfville, NS, Canada B4P2R6*

Abstract

Human learners encounter difficulties when learning an algorithm from small numbers of noisy examples. The sequence of concepts that are manipulated by the algorithm must be recognized before the algorithm can be learned effectively. This is accomplished by learning to map the noisy examples of a concept to a the correct class label or symbol. In this way, human learners bias the development of more accurate models for an algorithm. Similarly, we propose RNNs can better learn an algorithm, if they concurrently learn to classify the concepts manipulated by that algorithm. We train recurrent LSTM neural networks to perform arithmetic operations (+,-,×,/) on sequences of images of noisy handwritten digits. The networks are trained in the presence and absence of symbols for the noisy digits. The results show that the models trained to classify the noisy handwritten digits are more effective at performing arithmetic than the models trained without symbols. Furthermore, we show that thermometer encoded symbols for digits develops internal representations that capture the quantity and ordinal relationship between those digits such that the recurrent networks are able to discover more accurate algorithms for all arithmetic operations for both seen and previously unseen combinations of digits.

Keywords

learning an algorithm, learning arithmetic, inductive bias from symbols, encoding symbolic knowledge, escaping local minima

1. Introduction

The ability of individual humans to learn a complex algorithm that accurately manipulates a sequence of concepts is hampered by the variety of real-world examples of these concepts. Consider the complexity of adding two hand-written digits. This difficulty is overcome as individuals learn to classify examples of each digit concept using symbols from a common language [1]. These symbolic class labels effectively transfer knowledge within a society from one person to another. Deep recurrent neural networks (RNNs) experience similar challenges when learning a new algorithm from a small dataset containing examples of input-output sequences. This can be attributed to the existence of many local minima in the hypothesis space of the network's loss function [2]. We propose that learning to map noisy examples of concepts to symbolic labels during recurrent network training provides an inductive bias that guides the learning algorithm to a more optimal representation in the hypothesis space. The conjecture is that shared symbols are important to the external exchange of knowledge between intelligent agents, even though the learned conceptual representation of these symbols are unique to each

NeSy 2021 Workshop, October 25-27, 2021
danny.silver@acadiau.ca (D. L. Silver); mail@ahmedabada.com (A. Galila)

agent [3].

The value of sharing concepts through symbols will be empirically demonstrated in this paper by training deep RNNs to perform basic arithmetic operations (+,-,×,/) given noisy images of handwritten digits and arithmetic operators. We limit our experiments to developing RNNs based on Long Short Term Memory units (LSTMs) using images from the MNIST dataset [4]. The models will accept a sequence of images as input and will output the result of the operation as a sequence of encoded vectors. The models will also use the output channel to map (classify) the noisy images to consistent symbols. Training will be performed with and without the presence of symbols, whereas testing will be done using only the noisy images of the digits. The test results will be analyzed to show the impact of the symbolic channel on the accuracy of the models and the models will be examined using activations clustering to show how the symbols affect the development of the networks internal representation and features.

The major contributions of this paper are: (1) We show that a RNN that is trained to map noisy examples of concepts to symbols while also learning an algorithm that manipulates those concepts performs significantly better than one trained without the use of symbols; and (2) We show that, with properly encoded symbols, a RNN can develop an algorithm for arithmetic operations that generalizes to unseen combinations of digits.

2. Background

2.1. Sequence Modeling with Recurrent LSTM Neural Networks

Many machine learning problems consider making one or more predictions based on a sequence of input data. A RNN is capable of developing such models, most notably sequence-to-sequence (seq2seq) models that develop a form of algorithm that manipulates a sequence of input values such as "3 + 2 =" to produce an output "5" [5]. In 1997, a type of RNN architecture was introduced that uses Long Short-Term Memory (LSTM) units versus standard hidden nodes with sigmoid activation functions [6]. LSTM RNNs provide a proven solution to overcoming what is know as the vanishing gradient problem that can cripple standard RNNs from learning long-term dependencies over many time steps [7].

Deep LSTM RNNs may increase the expressiveness of a model, however, it makes the search for a global minimum in the hypothesis space more challenging. The complex topology of the network adds regions in the loss function that act as local minima [8]. In the context of a small dataset, this makes it more likely for the LSTM RNN algorithm to converge to a sub-optimal minima [2]. Overcoming this difficulty in training deep networks requires increasing the number of labelled training examples used, pre-training the lower layers of the network using unsupervised learning with unlabelled examples [2], or building in inductive bias as is the case with convolutional neural networks [9].

2.2. Related Work

There has been prior work on learning arithmetic equations but almost all use noise-free symbolic encoding of the digits (ASCII code or one-hot vectors) and not images of handwritten digits [10, 11]. Even with symbolic encoding of the digits these approaches require neural components manually biased towards the arithmetic task at hand in order to generalize.

We found one prior effort to learn addition using images of integers of up to seven digits in a standard font [12]. Good accuracy is achieved but only after training on 150,000 examples. The authors take the reasonable approach of breaking the problem down into classifying each digit explicitly and then using a coded symbol for that digit to do addition. We take a similar direction, however we consider the situation where the symbols for the hand written digits may only be present for a portion of the training data and are not used explicitly to perform the arithmetic operation.

3. Theory and Approach

3.1. Theory

Humans have developed language to allow individuals to share knowledge of basic concepts in the form of symbols in a clear and concise manner [1]. These shared symbols introduce beneficial inductive bias when learning algorithmic models that manipulate these concepts; resulting in more accurate models [13]. Consider the following definitions:

Definition 1. *A* ***concept*** *is an abstract idea or notion that can be learned and represented internally in unique ways by an intelligent agent.*

Definition 2. *A* ***symbol*** *is a noise-free, concise and consistent external representation of a concept that captures its key regularities or semantics and is meant to be shared between learning agents.*

For example, the MNIST dataset contains a wide variety of handwritten examples of the concept “2”. A neural network would require a large number of noisy handwritten examples of this digit in order to learn to do arithmetic accurately. In contrast, a symbol for the digit “2”, such as the one-hot vector “001000000”, has a consistent representation for the concept. Far fewer examples of “2” encoded in this way is required by a neural network to learn to do arithmetic. This is why humans learn to classify digits before and concurrently while learning to do arithmetic.

Hypothesis 1. *Similar to how individual human learners overcome the difficulties they encounter while learning an algorithm that manipulates a sequence of noisy examples of concepts, a RNN can better learn an algorithm by concurrently learning to classify each noisy example using a symbolic label of the concept. Learning to classify the example guides the development of more beneficial internal representations for the concepts and therefore the algorithm.*

The conjecture is that symbols are predominantly external communication tools that allow agents to share otherwise complex noisy concepts that help them avoid local minima in model development. Clear symbols for concepts, provided on occasion, help a learner develop abstract representations that provide beneficial inductive bias during learning, therefore reducing the number of examples required to accurately learn a new concept. Furthermore, appropriately chosen symbols allow the learner to move beyond developing a simple mapping function from input to output, to developing an algorithm that represents a more general solution to the problem, in this case performing arithmetic operations.

3.2. Approach

To test the hypothesis presented above, we construct several LSTM RNN models that learn basic arithmetic operations (addition, subtraction, multiplication and division) on noisy images of handwritten digits. We use the MNIST dataset of images of handwritten digits [4] to train and test our models because the data has been prepared and does not require the neural network to contain convolutional components to produce good models [9]. This simplification allows us to better analyse the internal representations of the LSTM networks. Our goals are: (1) To show that it is possible to develop a RNN model that can learn to perform all four basic arithmetic operations more effectively with the presence of symbols than without; and (2) To show that networks that are trained using appropriate symbols are capable of discovering an algorithm from an impoverished set of examples that generalizes well to unseen combinations of digits.

Learning Arithmetic using LSTM networks. Figure 1 shows how the arithmetic expression "2 x 3 = 6" can be modeled with a two-layer LSTM RNN using the seq2seq approach [5] with the operands and operators presented in Reverse Polish Notation as "2 3 x 6". Reverse Polish Notation allows us to more easily analysis of the networks and to learn more complex arithmetic equations in future work without having to use brackets. On the first two time steps, the model accepts two 28x28 images of handwritten digits from the MNIST dataset. On the third time step, the model receives the operator (x), in the form of a 28x28 image, and outputs the least significant digit from performing the operation on the two operands (the quotient for the division operator). On the fourth time step the model receives the equals sign (=), again in the form of a 28x28 image, and outputs the most significant digit given the operation and the two operands (the remainderfor the division operator). The result is represented using one-hot encoded vectors or thermometer encoded vectors.

Providing a Symbolic Channel. We have examined the use of symbols on both input and output channels of RNNs and have found them equally useful [14]. In this paper, our approach is to provide a symbol via an output channel. The model is challenged to perform a classification on each incoming image from the MNIST dataset. By training the recurrent network to classify the digit images, the architecture establishes a conceptual representation of each digit and passes it as context to the next time step. The RNN must use its output nodes for two different tasks: (a) to associate each noisy handwritten digit with a symbol, and (b) to calculate the result of the arithmetic operation using internal representations for those symbols.

We can explain the role of the output symbol in terms of Bayesian inference. The probability of finding an optimum representation h is affected by the presence of evidence, in our case, the symbol classification C and the noisy input data D. Bayes' Theorem tells us:

$$p(h|C,D) = \frac{p(C,D|h) \cdot p(h)}{p(C,D)} \tag{1}$$

From this, the maximum likelihood hypothesis (assuming prior hypotheses are equally likely and the digits selected are in common) is given by:

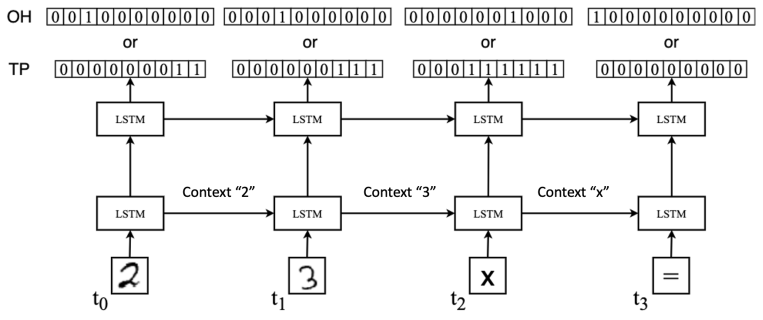


Figure 1: A sequential model learning to perform arithmetic (in this case, 2 x 3) on images of handwritten digits in the presence of symbols. The symbols are provided on the outputs of the first two intermediate steps in either one-hot (OH) or thermometer (TP) encoded format.

$$h_{ML} = \underset{h}{\operatorname{argmax}}\, p(C, D|h) = p(D|C, h) \cdot p(C|h) \tag{2}$$

That is, a RNN that correctly maps the noisy digits to their symbolic class labels increases the likelihood that it will discover a hypothesis that can accurately do arithmetic using those digits.

Encoding of Symbols. We desire a learning system that is able to learn: (1) the quantity that each digit represents; (2) the ordinal relationship between one digit and another, also known as successorship; and (3) the function of the operator.

Human learners are able to use the knowledge transferred through the use of the informative symbols to capture all three semantic aspects. The symbols provided to our neural network models must represent the same semantics. Although a one-hot vector representation, such as "000001000" for the digit "5" is a clear symbol, it lacks the representative ability to convey the first two semantics concerning digits: it does not convey quantity and it does not represent an ordering over the digits. We are looking for what semiotics refers to as an *index* type of symbol. We propose using unary or thermometer encoding [15]. A thermometer encoded integer is a vector of binary values that has as many elements sequentially set to one as the integer being represented. For example, the number 5 is encoded as "000011111". This symbol encoding conveys the first two semantics concerning digits. Consistent operator images allow the RNN to learn which arithmetic function it is being asked to perform.

Analysis of the LSTM Networks. To analyse the RNNs, we will use a visualization technique, known as activations clustering. It captures the output values of each hidden layer and renders these as pixel intensities in an image [16]. Figure 5 shows an activations cluster for a recurrent network with four time steps and two hidden layers, 20 units each.

Our expectation is that the activation clusters obtained from the hidden nodes of a model concurrently trained to classify the noisy inputs as symbols will be relatively consistent as compared to a model trained without having to classify the noisy inputs. We also expect to see the "carry one" signal that humans use to do addition represented in the network's hidden layers.

We wish to show that digit classification using symbols helps the RNN advance beyond learning simple mapping functions to learning a general algorithm for arithmetic operations. Toward this end, we develop RNNs that are trained on only a subset of the examples of operand combinations and then tested on the remaining combinations. We believe that by providing appropriately encoded symbols during training, our models will generate internal representations for unseen combinations of digits that allow the model to generalize to the correct arithmetic results.

4. Empirical Studies

In this section we present a sequence of three experiments to test our hypothesis. The first examines the fundamental benefit of learning an algorithm while concurrently learning to classify noisy examples of concepts used by the algorithm. The second experiment uses a thermometer encoding for the symbols in an effort to develop an algorithm that can generalize to previously unseen combinations of digits when doing addition. The third experiment extends the second to all four arithmetic operations. The Python deep learning stack including Tensorflow and Keras was used to architect, train and test the RNN models.

4.1. Exp 1: Learning the Arithmetic Operations

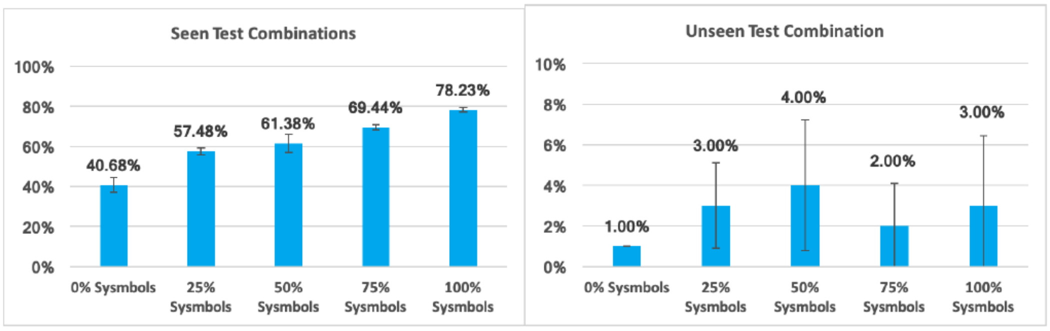
Objective. To demonstrate that (1) the presence of symbols of the noisy input data improves the learning accuracy of the RNN models developed for all four arithmetic operators (+,-,×,/), and (2) the models are able to generalize to previously unseen combinations of digits. To verify this, we train our models on a subset of all combinations of digits and test on the remaining combinations.

Method. For each of the four arithmetic operations and for each of the 100 combinations of two digits, MNIST image examples were randomly selected. For addition and multiplication, all combinations are included in the dataset. However, for subtraction, no combinations that would result in negative outputs are included and, similarly, divide by zero combinations are avoided.

For each operator, the combinations of digits are randomly split into a "seen" set of 80 combinations and an "unseen" set of 20 combinations. For each of the seen combinations, eight pairs of MNIST digits are selected; four are used for training, two for validation and the remaining two for testing. The "unseen" combinations are used to test if the models generalize to an algorithm for arithmetic over all possible digit combinations.

We develop five different models using different quantities of symbols to assist learning arithmetic using the MNIST images. The dataset of seen combinations is replicated five times, each replica includes a different percentage of symbols present: 0, 25, 50, 75 and 100%. The symbol (if provided) is presented as an output label during the first two time steps and the model is trained to classify the handwritten digit based on the label. When a symbol is not provided, the label vector is composed of ten 0.5 "dummy" values. When tested the models are provided with only the noisy MNIST images as input.

The five different models have the architecture described in Section 3.2 that accepts a sequence of four 28x28 images. Two hidden layers are used with 512 LSTM units each. The models output



(a) Seen Test Combinations *(b)* Unseen Test Combinations

Figure 2: A comparison of the mean accuracy and 95% CI for models trained with different percentage of symbols when tested on both the **seen** and **unseen** test combinations.

a one-hot vector. Each model is trained using 5-fold cross-validation and the results averaged. The networks are trained using the Adam optimizer with the mean square error as the loss function and a learning rate of 0.001. Training is performed over 200 epochs in batches of 100 and the model performing best on the validation set is saved and used for testing.

Results and Discussion. Figure 2 shows the mean accuracy and 95% confidence intervals for the various models tested on the seen and unseen test sets. These models are trained using only four MNIST image examples of each digit combination, thus the low accuracies.

The results show that the RNNs using symbols encoded as one-hot output vectors are capable of learning all four arithmetic operations using images of handwritten digits. They show that the accuracy of the models on previously seen combinations of digits increases linearly with the percentage of symbols provided as class labels for the noisy digits. However, the results also show that the RNNs were unable to discover an algorithm for the arithmetic operations that generalizes for previously unseen combinations of operands (mean accuracies are less than 5%).

4.2. Exp 2 - Encoding the Semantics of Digits

Objective. Exp 1 shows that the current RNN approach fails to go beyond learning a mapping function and actually discover an algorithm to perform the arithmetic operations. This experiment explores the use of symbols encoded as thermometer vectors, as compared to one-hot vectors, as a possible solution. The objective is the same as that of Exp 1; to develop a network that can generalize to unseen combinations of digits. We constrain the problem to (1) learning an algorithm for only addition and (2) using consistent symbolic inputs and not the noisy MNIST handwritten digits. This reduces the size of the RNN and makes it's hidden node features easier to analyse using the activations clustering method described in Section 3.2.

Method. The models developed for this experiment accept a sequence of four vectors at its input, each vector is composed of thirteen elements. Figure 3 depicts the structure of the input vectors. The first ten elements represent a digit using one-hot encoding. The remaining three elements represent the context of the input. Element 11 (C) is set to one on the first two time

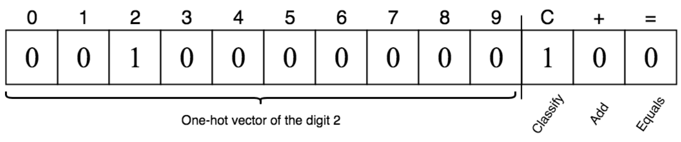


Figure 3: An example of an input vector for the digit 2 on the first time step.

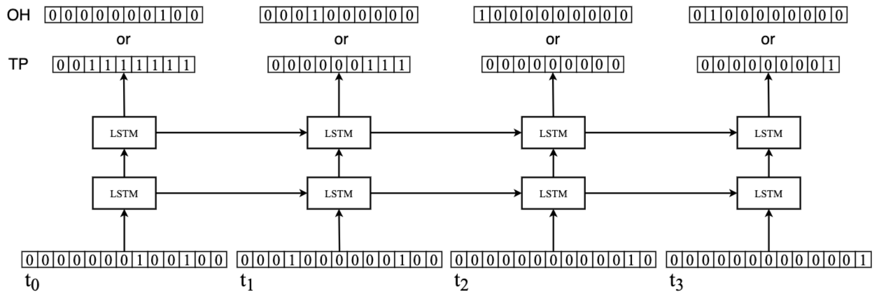


Figure 4: The constrained symbol-only architecture trained to add 7 and 3. The output is represented as a one-hot vector (OH) or as a thermometer vector (TP).

steps when we want the network to output a vector that classifies the input digit. Element 12 (+) is set to one at the third time step to indicate that the addition operation should be performed and that the least significant digit of the output should be produced. Finally, element 13 (=) is set to one on the fourth time step when the most significant digit of the result of the addition is expected at the output. The output layer of the models consists of 10 values representing the output as either a one-hot vector or a thermometer vector. Figure 4 depicts how the input and output layers interact on each of the four time steps.

Several RNN architectures were tried using the input and output layers described above. The best models had two hidden layers of 20 LSTM units each. A dataset was created using the set of 100 possible combinations of two digits. A subset of 80 combinations are selected as the "seen" examples of addition, making sure that each unique digit would be present at least once on either side of the addition operator. The remaining 20 combinations are used as the "unseen" test set. A 5-fold cross-validation approach is used and the mean accuracy of each trained model is recorded when applied to the seen and unseen test sets.

Results and Discussion. All models performed at 100% accuracy on the seen test sets. However the models that use one-hot vector symbols had only 5% accuracy on the unseen test sets, whereas the models that use thermometer vector symbols performed at 90% accuracy. We conclude that thermometer encoded symbols provide the information needed for the RNN to better discover the representation of an algorithm for addition of two operands.

To verify this conclusion, we generated activations clusters for a series of four digit combinations from the unseen test set using the best one-hot vector and thermometer vector models. Figure 5 shows hidden node activations produced by the two methods for the same combinations of digits of the unseen test set. The figure shows four rows of two hidden layers, one for each digit combination. Each row represents a single time step moving from top to bottom. For each time step, the figure depicts, from left to right, the digit input to the network, the digit output

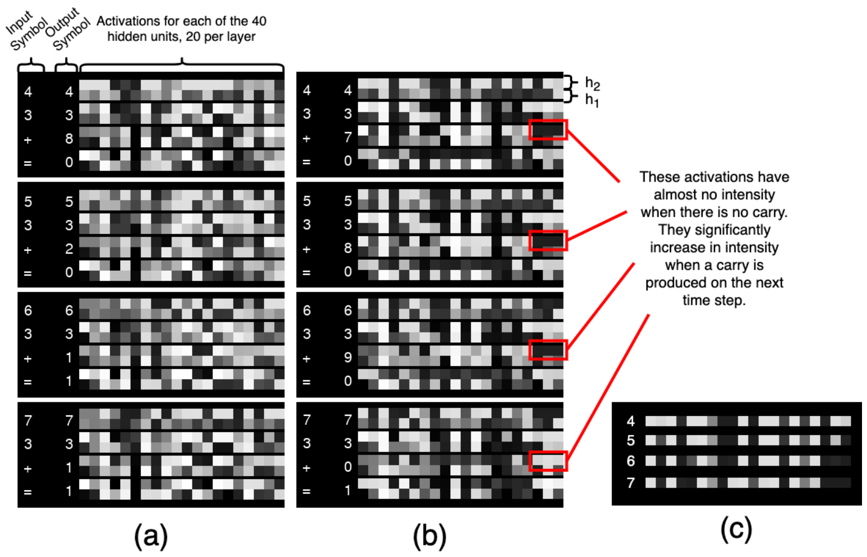

Input Symbol
Output Symbol
Activations for each of the 40 hidden units, 20 per layer
h2
h1
These activations have almost no intensity when there is no carry. They significantly increase in intensity when a carry is produced on the next time step.
(a)
(b)
(c)

Figure 5: A series of activations produced when applying examples from the unseen test set to a model trained using (a) one-hot encoded and (b) thermometer encoded symbols. The features created by hidden layer 2, h_2, of the thermometer encoded model for the four first operands are compared in (c).

by the network, and the LSTM node activations in the two hidden layers, h_1 and h_2. Note that both models correctly classify the digits, but the model that uses one-hot encoded symbols errs on all four additions.

Figure 5(b) shows that when the "+" input is received following the "7" and then "3" operands, several of the h_2 node activations (observe the last three on the right) change sharply to signal a carry-forward. This same signal is not observed in the activations of the one-hot vector output model of Figure 5(a). Subsequently, the model that uses one-hot vector outputs does poorly, erring on all four additions.

4.3. Exp 3: Noisy Digits with thermometer Encoded Symbols

Objective. The purpose of this experiment is to show that symbols represented as thermometer encodings can improve the generalization accuracy of a RNN model trained to learn how to perform arithmetic from an impoverished dataset. It extends Exp 2 in two ways: (1) it returns to using the noisy MNIST digits, and (2) it trains a RNN to learn all four arithmetic operations.

Method. This experiment uses the same LSTM RNN network architecture used in Exp 1 and shown in Figure 1. The model accepts a sequence of four 28x28 images, as described in Section 3.2. The same base dataset is used except thermometer encoded symbols are used as output labels for all time steps. When output symbols are not provided, a vector composed of 0.5 "dummy" values is used instead. As in Exp 1, the dataset of seen combinations is replicated five times, each replica includes a different percentage of symbols present, specifically 0, 25, 50, 75 and 100%. Each model is trained five times using 5-fold cross-validation and the statistics calculated.

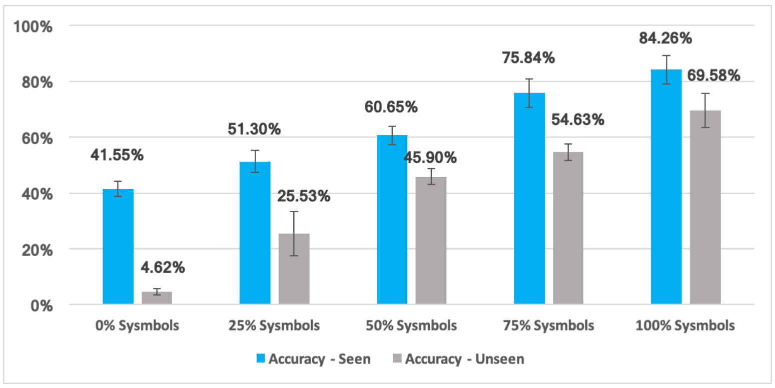


Figure 6: A comparison of the mean accuracy and 95% CI for models trained with different percentage of symbols when tested on both the **seen** and **unseen** test combinations.

Results and Discussion. Figure 6 shows the mean accuracies and the 95% confidence intervals for all five models tested on both the seen and unseen test sets. These results show that increasing the number of symbols available per combination during training improves the accuracy of the recurrent network when tested on combinations that the model has seen during training. This confirms the findings in the prior experiments and supports our hypothesis that artificial neural networks similar to human learners can benefit from the presence of symbols when training with limited datasets.

5. Conclusion

Much of RNN modeling has focused on developing accurate algorithmic models using sufficient numbers of training examples. In contrast, humans learn algorithms from impoverished datasets by using shared knowledge from other humans in the form of symbols. In this paper, we propose that RNNs can overcome similar learning challenges caused by limited training sets if they concurrently learn to classify noisy input examples of the concepts upon which they operate.

A series of experiments are conducted to test this theory when training RNNs to perform basic arithmetic on images of handwritten digits. The results show that: (1) It is possible to develop a RNN model that can learn to perform all four basic arithmetic operations from an impoverished set of examples more effectively in the presence of symbols than without; and (2) Networks trained using appropriate symbols are capable of discovering an algorithm that generalizes better to all combinations of digits, instead of simply learning an input to output mapping function. By using thermometer encoded versus one-hot encoded symbols along with a four step arithmetic sequence we were able to develop LSTM RNNs that generalize well to combinations of digits that were not seen during training.

In future work we will consider longer sequences of arithmetic operands and operators using Reverse Polish Notation as well as sequences written in standard notation using brackets to convey the order of calculations. Such sequences will test the RNNs ability to develop subroutines. We will also consider applying this theory of learning with symbols to other application domains such as natural language processing and the explanation of image classification.

References

[1] Y. Bengio, Evolving culture vs local minima, Computing Researh Repository (CoRR) abs/1203.2990 (2012). URL: http://arxiv.org/abs/1203.2990. arXiv:1203.2990.

[2] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (2009) 1–40. URL: http://dl.acm.org/citation.cfm?id=1577069.1577070.

[3] A. Newell, Physical symbol systems, Cognitive Science 4 (1980) 135–183. URL: https://doi.org/10.1207/s15516709cog0402_2. doi:10.1207/s15516709cog0402_2.

[4] Y. LeCun, C. Cortes, C. J. Burges, The mnist database of handwritten digits, 1998. URL: http://yann.lecun.com/exdb/mnist/, [Online; accessed 04-June-2018].

[5] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112. URL: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[6] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

[7] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans on NN 5 (1994) 157–166. URL: http://dx.doi.org/10.1109/72.279181. doi:10.1109/72.279181.

[8] Y. Dauphin, R. Pascanu, Ç. Gülçehre, K. Cho, S. Ganguli, Y. Bengio, Identifying and attacking the saddle point problem in high-dimensional non-convex optimization, Computing Researh Repository (CoRR) abs/1406.2572 (2014). URL: http://arxiv.org/abs/1406.2572. arXiv:1406.2572.

[9] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[10] A. Trask, F. Hill, S. E. Reed, J. W. Rae, C. Dyer, P. Blunsom, Neural arithmetic logic units, CoRR abs/1808.00508 (2018). URL: http://arxiv.org/abs/1808.00508. arXiv:1808.00508.

[11] W. Zaremba, I. Sutskever, Learning to execute, CoRR abs/1410.4615 (2014). URL: http://arxiv.org/abs/1410.4615. arXiv:1410.4615.

[12] Y. Hoshen, S. Peleg, Visual learning of arithmetic operations, CoRR abs/1506.02264 (2015). URL: http://arxiv.org/abs/1506.02264. arXiv:1506.02264.

[13] S. Thrun, Lifelong Learning Algorithms, in Learning to learn, Springer US, Boston, MA, 1998, pp. 181–209.

[14] A. Galilia, Learning with symbols, 2019. Master Thesis, Acadia University.

[15] J. Buckman, A. Roy, C. Raffel, I. J. Goodfellow, Thermometer encoding: One hot way to resist adversarial examples, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018. URL: https://openreview.net/forum?id=S18Su--CW.

[16] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, B. Srivastava, Detecting backdoor attacks on deep neural networks by activation clustering, CoRR abs/1811.03728 (2018). URL: http://arxiv.org/abs/1811.03728. arXiv:1811.03728.