Can Inter-layer Recurrent Neural Networks Learn Algorithms?

Anthony Proschka [1]

Abstract

While recurrent neural networks have been proven to express universal Turing machines, *training* them to perform arbitrary computable functions exactly is difficult. In this paper, we introduce a novel architecture called Inter-layer Recurrent Neural Network (ILRNN, pronounced "iLearn"), a form of deep recurrent neural network that exhibits extended and top-down connectivity across layers. Despite being inspired by both biological neural networks and digital circuit design, we cannot yet show that an ILRNN using conventional optimization techniques learns to add arbitrarily large binary numbers (not even with informed initial hyperparameter configuration). It remains to be evaluated whether a learning algorithm exists that can reliably find global minima in such a network that represent the exact solution and generalize to arbitrarily large binary numbers.

1. Introduction & Motivation

In recent years, deep learning has shown remarkable progress in applications such as image recognition (Krizhevsky et al., 2012), speech recognition (Graves et al., 2013), games (Mnih et al., 2015; Silver et al., 2016) and photorealistic image generation (Goodfellow et al., 2014), either being the first technology to ever solve a problem or achieve human parity on it.

In this paper, we want to draw the attention to three core aspects of the study of artificial neural networks: expressivity, learnability and interpretability.

An important distinction that needs to be made is the fundamental difference between the expressivity of a deep learning model and its learnability. In the realm of real functions, several theoretical works have shown that feed forward artificial neural networks act as universal function approximators, i.e. under certain conditions functions of the form

$$FNN(x) = (\bigcirc_{k=1}^{n} f_k)(x),$$

$$\text{where } f_k(x) = \sigma(W_k x),$$

$\bigcirc$ is the repeated function composition operator, n is the number of layers, σ is the activation function, $W_k \in R^{l_k \times l_{k-1}}$ is the weight matrix of layer k and l holds the number of neurons for each layer, are able to approximate up to some tolerated error any arbitrary function (Cybenko, 1989; Hornik et al., 1989; Hornik, 1991). In the realm of theory of computation, Siegelmann and Sontag have shown that recurrent neural networks are able to simulate any Turing machine, which is equivalent to saying they can compute any partial recursive function (1995). Universal function approximation and Turing completeness are structurally different properties and should not be equated. However, they both illustrate that given the right set of hyperparameters and a viable weight configuration, artificial neural networks should in theory be able express a solution to a broad class of tasks. Whether the function that the human brain computes falls into this class is still unclear.

While expressivity seems to be a well established property of ANNs, their learnability is much less so. An early work showed that under certain circumstances, even the smallest networks are NP-hard to train (Blum & Rivest, 1989). In most scenarios, the system of polynomial or nonlinear equations resulting from modelling a given set of labelled data using a neural network cannot easily be solved analytically. Other solution and optimization techniques seem to have little guarantees that they find any solution or a global optimum.

Finally, ANNs are still mostly considered "black box" functions that lack interpretability. Once trained to perform a specific task, it is not straightforward to reconstruct what features (with semantic human understandable value) caused a certain prediction or decision, especially when the input features themselves mark low-level entities such as pixels in images or sound frequencies in audio streams. This paper also aims to provide an intuitive understanding of what type of computation is happening in the newly described recurrent neural network architecture.

[1]Candis GmbH, Berlin, Germany. Correspondence to: Anthony Proschka <~~anthony@candis.io~~>.

~~*Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).~~

Our contributions include the following:

- We introduce a novel architecture for deep recurrent neural networks called inter-layer recurrent neural networks (ILRNN). These networks exhibit extended and top-down connectivity across time steps without introducing cyclical paths in the computational graph.
- Building on top of well-known insights that neural networks can compute logic gates, we show by an example that recurrent neural networks can in fact represent any Boolean circuit perfectly. Thusly, ANNs with the appropriate activation functions can be interpreted as trainable versions of combinational logic.
- Formulating a (almost) memoryless binary addition task, we provide an interesting test bed for future work exploring means to teach computational tasks to neural networks.
- Using various of today's solving and optimization techniques, we obtain the empirical result that finding the weight configuration that solves the task perfectly is hard.

2. Related Work

In the following section we quickly recap important milestones in the advancements of recurrent neural networks (RNNs) and in the attempts to teach neural networks to learn exact and general solutions to algorithmic and computational problems.

2.1. Recurrent neural networks

Rumelhart et al. were possibly the first researchers to study recurrent neural networks (RNNs) as an extension of standard feedforward neural networks with the intention to model sequence and especially time series data (1986). They show how their then-novel backpropagation algorithm could be applied to RNNs by unrolling the net for a finite number of time steps and backpropagating error gradients through these.

Around the same time, Jordan publishes a report on using RNNs for sequence modeling 1997. There, he also introduces simple nets that have recurrent connections feeding the outputs of a previous time step back to the hidden nodes of the next (note this is a first record of the idea generalized by ILRNNs that will be introduced later in this paper).

One of the major advances in RNN research is the long short-term memory or LSTM cell (Hochreiter & Schmidhuber, 1997). By incorporating gating mechanisms to an RNN cell, they circumvent the so-called vanishing or exploding gradient problem that arises when gradients are backpropagated through many layers of neurons in conventional networks (Hochreiter, 1991; Hochreiter et al., 2001). LSTMs and similar models such as GRUs are today widely used in practice (Cho et al., 2014; Salehinejad et al., 2018).

In more recent times, several studies have examined RNNs and methods to train them in greater detail. Graves gives thorough explanations on the different types of sequence labeling tasks that can be modelled using RNNs in the supervised setting both when input and output sequences are aligned or not (2012). In his doctoral thesis, Sutskever puts specific emphasis on the problem of properly training RNNs, and provides updates to the model, a second-order optimization algorithm as well as a new initialization scheme (Sutskever, 2013). This is particularly interesting for this paper as he recognizes the difficulty of finding a desired solution in an RNN for a given task.

Deep RNNs, RNNs that have multiple recurrent layers (or cells) stacked on top of each other, were first used to improve the state of the art in speech recognition (Graves et al., 2013). More specifically, the connectivity across neurons and layers in deep RNNs can be structured in a variety of ways proposed by Pascanu et al. (2013).

3. Model Definition

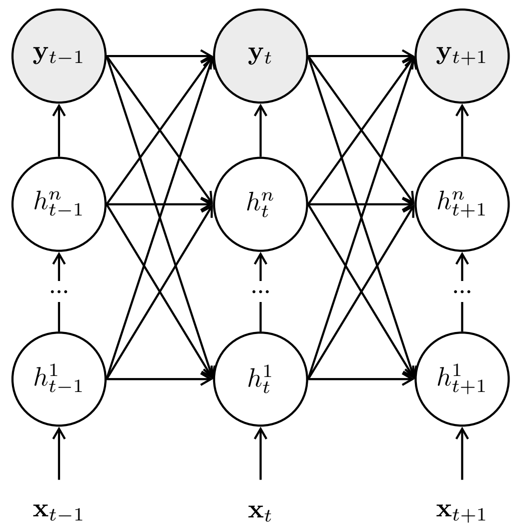


Figure 1. Visualization of ILRNN

We consider deep RNNs (Graves et al., 2013). In the novel architecture called inter-layer recurrent neural networks (ILRNNs), we now extend deep RNN's connectivity in such a way that both the hidden and output units receive incoming connections from all hidden and output units of the previous time step like so:

$$h_t^n = \sigma(W_{h^{n-1}h^n} h_t^{n-1} + \sum_i W^r_{h^i,h^n} h^i_{t-1} + b^n_h),$$

where h_t^i is the hidden activation at time step $t \in \{0, ..., T\}$ and hidden layer $i \in \{1, ..., N\}$, W is the set of feedforward weight matrices, W^r is the set of recurrent weight matrices, b_h^i is the bias vector for hidden layer i and σ is an arbitrary activation function.

Figure 1 shows the connectivity between the different hidden layers across time steps graphically. Note that the output layer is considered the final hidden layer in this notation. Also note that despite top-down connections between layers, since they occur across time steps no cyclical paths on the computational graph are added (allowing this architecture to be trained with conventional BPTT). While this increases the number of free parameters compared to the standard deep RNN considerably, this increase still fares well in comparison with converting a simple RNN cell to an LSTM cell. Surprisingly, to our knowledge this is the first paper to introduce this specific connectivity in recurrent neural networks.

This architecture is a general case of the net by Siegelman Sontag to prove their computational power (1995). It should thus exhibit the same guarantees in terms of expressivity.

4. Task Description

This section describes a specific formulation of the binary addition task that was used to evaluate both the expressibility and learnability of ILRNN.

As outlined in Section 2, considerable previous work has been done to enable ANNs to learn algorithmic tasks, programs and other programming language related problems. A popular sample task is *addition*. A variety of different formulations for the addition task exist, differing in the choice of numbering system (binary vs. decimal), encoding scheme (raw vs. one-hot encoding), model type (feedforward vs. recurrent) and external devices (none vs. scratchpad etc.) (cf. Reed & de Freitas 2015).

The literature most relevant to this paper has one trait in common: it uses large recurrent models (at times augmented with external memory devices) and formulates addition in a way that requires the model to memorize or actively read the entire sequence of addends before it outputs its result. This is where the binary addition task deployed differs with its predecessors: ILRNN is fed one pair of addends per time step, and can immediately output its corresponding sum. The only information it has to keep track of across time (and thus using its memory capabilities) is the potential carry that can result from the sum of two individual digits. A step-by-step illustration that deduces the representation fed to ILRNN from the original written base-10 addition that humans learn during elementary school is depicted in Table 1.

Algorithm 1 Generate binary addition sample

```
**Input:** maximum digits per addend $n$,
Initialize $carry = false$, $sample = []$
**for** $i = 1$ **to** $n$ **do**
    Draw $x_1, x_2$ uniformly from $\{0, 1\}$
    $y_1 = (x_1 \oplus x_2) \oplus carry$
    $carry = x_1 \wedge x_2 + carry \wedge (x_1 \oplus x_2)$
    Append $(x_1, x_2, y_1)$ to $sample$
**end for**
**if** $carry == true$ **then**
    Append $(0, 0, 1)$ to $sample$
**end if**
Return $sample$
```

The procedure used to generate a sample of a given sequence length is depicted in Algorithm 1. Its assignment statements for y_1 and $carry$ prove interesting because these are exactly the set of expressions we wish ILRNN to internalize. We are effectively nudging the neural network to express Boolean algebra through its neural pathways.

One advantage of this task is that it naturally provides out-of-sample instances that can be used to test generalization: simply evaluate the trained model on adding numbers that are larger, i.e. have more digits than the ones seen during training. While our approach does not require the model to hold increasingly large binary numbers in memory, it simply evaluates whether the network is able to learn the exact addition logic needed to add arbitrarily large binary numbers. Ideally this addition logic is not degraded through spurious information passed through recurrent connections as the numbers become larger.

5. Experiments

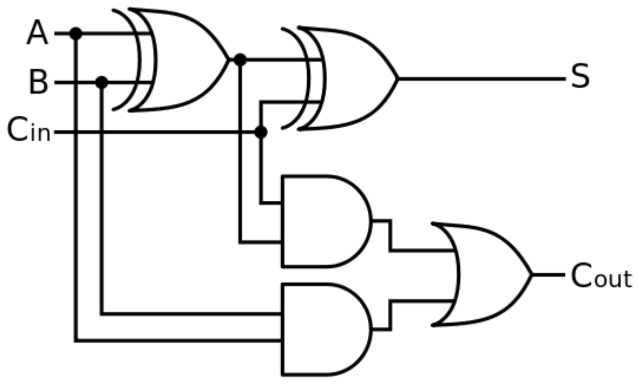
After introducing a new recurrent neural network model and specifying a specific form of binary addition task, the results of the experiments are reported. It is important to note that no experiment reported in this paper was conducted without incorporating prior knowledge into the model architecture. Accordingly, first the manual construction or rather deduction of the hyperparameter settings will be explained, before the attempt for an analytic solution and the training procedure using an gradient descent algorithm.

5.1. Manual construction

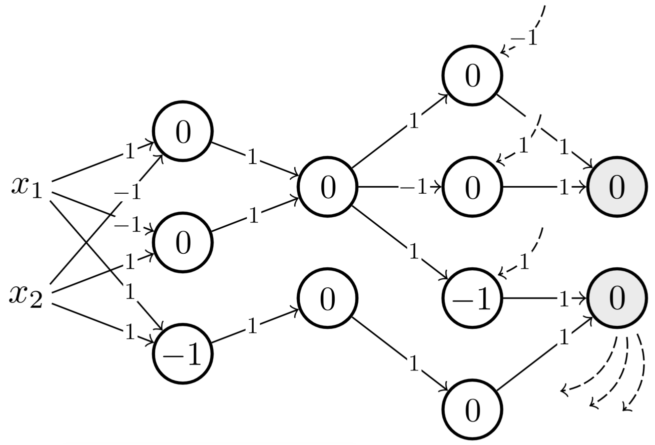
A structure that solves the binary addition task is the full adder circuit commonly used in the arithmetic logic units of CPUs. It implements Boolean algebra using nothing

Table 1. Different representations of the addition task

DECIMAL ADDITION	BINARY ADDITION	BINARY ADDITION (REVERSE ORDER)	NETWORK INPUTS & TARGETS (OVER TIME)									
				t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
387	110000011	110000011	x_1	1	1	0	0	0	0	0	1	1
+ 18	+ 000010010	+ 010010000	x_2	0	1	0	0	1	0	0	0	0
= 405	= 110010101	= 101010011	y_1	1	0	1	0	1	0	0	1	1



(a) Simple full adder circuit



(b) Simple full adder ILRNN

Figure 2. Full adder circuit and ILRNN equivalent

but basic logic gates. It is well known that simple neural networks with one hidden layer and nonlinear activation functions can represent the XOR logic gate as well as all other basic logic gates.

We use this fact to construct a baseline neural network of manually defined architecture and weight configuration that solves the task exactly. This proves that a ILRNN exists that expresses the full adder functionality. Note that this correspondence would not have been possible using a standard deep RNN, it was made possible through top-down recurrent connections across layers.

Figure 2 depicts both the origin full adder microcircuit as well as its neural equivalent. The numbers on the visible connections (drawn as arrows) mark the respective weights for that connection. Missing connections can be regarded as having a weight of 0. The numbers within the hidden and output nodes resemble the biases chosen for these nodes. Dotted arrows indicate inter-layer recurrent connections (that pass on the potential carry). Only 3 recurrent connections are needed to reconstruct the full adder as a neural circuit. Note that for this weight configuration for ILRNN to work, the binary step needs to be chosen as activation.

5.2. Analytical solution

When one considers a linear activation function (i.e. $f(x) = x$), the learning problem is reduced to a system of polynomials. Calculcating the Gröbner basis for this system expectedly returns 0 (Buchberger, 2006), signifying that there exists no solution. This is insofar expected, as the XOR gate alone requires a nonlinear activation function in order to be represented by a neural net.

5.3. Learning / iterative optimization

Finally, we regard the task as a supervised sequence learning problem and train the network using backpropagation through time (BPTT) (Rumelhart et al., 1986). We tried a custom activation function used to resemble the binary outputs of the full adder microcircuit gates, for which we also implemented custom gradients following the work on binarized convolutional neural networks (Rastegari et al., 2016; Courbariaux & Bengio, 2016).

Unfortunately, our attempts to train the resulting architecture using conventional methods did not lead to the desired results. Standard optimizers such as Adam with different learning rates, different activation functions employed in the network (binary step, ReLU, Sigmoid) converged to a suboptimal solution. Even initializing the weights to the target solution perturbed by small amounts of white noise would not converge as desired.

Bland publish a thorough analysis on the learning behavior and the error surface of the XOR function for a simple neural network (1998). This is relevant here in the sense that the desired full adder consists of several XOR gates that should be learned. Bland finds that empirically only a small share of initial weight configurations converge to a solution computing XOR (1998).

6. Conclusion & Future Work

All in all, this paper introduces a new recurrent network architecture called ILRNN (iLearn) and a binary addition task that is used to evaluate it. While we find that the new architecture can express the exact solution, conventional training algorithms fail at finding it. The binary addition task seems like a good test bed for further research about exact learning in neural networks.

So far, architecture was hand-crafted (and weights needed to be initialized close to hand-crafted solution?), which is far from practical. Possible extensions to address the witnessed learnability problems are the following:

- The proposed network could be a sub-network of a larger network.
- Other (biologically inspired) learning algorithms could be tried out (f.e. differential plasticity, hebbian learning, local learning), new forms of regularization.
- Find a distributed representation of the logic gates also used in this paper
- It still needs to be evaluated how well the new architecture fares with larger problems and in combination with other recent deep learning-related refinements such as LSTM cells, regularization, bi-directionality etc.
- Regarding larger problems, those not requiring exact learning could be considered.
- Now that per-layer constraints in connectivity have been loosened, it would also be interesting to ask whether the law of synchrony of computation of neurons in artificial neural networks can be loosened.

References

Bland, R., of Stirling. Dept. of Computing Science, U., and Mathematics. Learning xor : exploring the space of a classic problem. 1998. Includes bibliographical references.

Blum, A. and Rivest, R. L. Training a 3-node neural network is np-complete. In *Advances in neural information processing systems*, pp. 494–501, 1989.

Buchberger, B. Bruno buchbergers phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of symbolic computation*, 41(3-4):475–511, 2006.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Courbariaux, M. and Bengio, Y. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016. URL http://arxiv.org/abs/1602.02830.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989. ISSN 0932-4194. doi: 10.1007/BF02551274. URL http://dx.doi.org/10.1007/BF02551274.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Graves, A. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pp. 5–13. Springer, 2012.

Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91 (1), 1991.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jordan, M. I. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pp. 471–495. Elsevier, 1997.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016. URL http://arxiv.org/abs/1603.05279.

Reed, S. and de Freitas, N. Neural programmer-interpreters. Technical Report arXiv:1511.06279, 2015. URL http://arxiv.org/abs/1511.06279.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

Salehinejad, H., Baarbe, J., Sankar, S., Barfett, J., Colak, E., and Valaee, S. Recent advances in recurrent neural networks. *CoRR*, abs/1801.01078, 2018. URL http://arxiv.org/abs/1801.01078.

Siegelmann, H. T. and Sontag, E. D. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016. URL http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

Sutskever, I. *Training Recurrent Neural Networks*. PhD thesis, Toronto, Ont., Canada, Canada, 2013. AAINS22066.