Diagnet: A Fast, Scalable Recurrent Neural Network Architecture

Thomas Lahore
tom.lahore@gmail.com

Morgan Weaver
morganjweaver@gmail.com

Abstract

Recurrent neural networks are a challenge to train, in large part due to the well known problem of vanishing and exploding gradients. Attempts to solve this have led to increasingly complex architectures and training methods. We propose Diagnet, a simple RNN design which consists of three core components: 1) a diagonal hidden-to-hidden matrix, 2) an absolute value function as the non-linearity, and 3) the application of constraints to both the recurrent parameter weights and gradient norms during training. Despite the segregated and simplistic nature of Diagnet's recurrent units, significant performance improvement is seen over Long Short-Term Memory on the 1000-example version of the bAbI tasks. Diagnet scores higher in testing on 19 out of the 20 tasks, by margins of up to 40% accuracy.

1 Introduction

Recurrent neural networks represent one of the most important deep learning architectures in current use. They are increasingly being used for many tasks, such as time series prediction [REF], unsegmented, connected handwriting recognition [REF], large-vocabulary speech recognition [REF], modeling chaotic phenomena [REF], language modeling and machine translation [REF], robot control [REF], Protein Homology Detection [REF].

We are interested in addressing two widely-cited [REF] pain points of RNN performance. The first being the vanishing and exploding gradient problem [REF], and the second being performance over increasing time steps [REF]. We address these issues by implementing three as kjfhakdf that have not yet been exibited in combination in neural net design. What we found Results were...impressive

In the following paper, we present a brief outline of foundational work in the problem space to contextualize Diagnet's emergence in RNN design. Then, we present a discussion of the architecture and mechanisms of Diagnet, along with a useful visual tool for building an intuitive understanding of Diagnet's units and their behavior. Next we present a series of common benchmarking tasks, the results of each task, and finally conclude with a discussion of Diagnet's performance and behavior based on experimental results. We hope to offer an interesting and powerful new tool for sequence modeling in our introduction of Diagnet, while laying the groundwork for further exploration, architectural innovation, and interaction with the ML community.

2 Previous Work

It is important to consider a new deep learning architecture with an awareness of existing models. We offer a brief and by no means comprehensive overview of existing RNN designs herein. Some of the earliest work consists of Hopfield Networks, developed by Hopfield in 1982. Subsequently, Schmidhuber [REF] developed the Neural History Compressor (1993) and was able to preserve information over time steps, while compressing information. In 1991 Fallman developed the Recurrent Cascade Correlation Algorithm Fahlman [1990], in which a single, previously trained hidden unit is

Preprint. Work in progress.

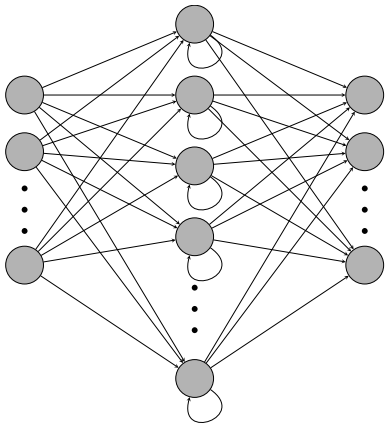

Figure 1: Figure showing nice diagram of the architecture.

added to the network at a time, with incoming weights frozen as the incoming unit is connected in a feed-forward manner with the previous output units. Each new hidden neuron must be maximally correlated with current remaining error per step. Now make recurrent: each neuron has single link to self. Rinse n repeat algo; truncated backprop at the time bc feasible. At time, proved not useful for certain grammars like seq parity (using saturating, not abs val fxn). First paper T knows of where neuron self-connected. (IndRNN first layer is the same) In 1997, Hochreiter and Schmidhuber introduced Long Short-Term Memory (LSTM), which sought to address the vanishing and exploding gradient problem with constant error flow through specialized units in addition to multiplicative gated memory units, resulting in more efficient storage of information at little computational cost. GRU (2014)-just ref it, no significant influence IRNN (2015)-identity RNN–HINTON, recurrent neurons are relus, init at ID matrix. 2 big ideas. Prob: not used bc unstable, need right params. Good init results > LSTM simpler architecture. Init at ID matrix was the big idea (unitary matrix) Diagnet too–but off-dig values can't ever be anything but 0; diag are all 1 fast Hadamard product instead of tiresome n$\hat{2}$. SCRN (2015)–diagonal entries connected constrained to 1 or a min val, so self-recurrent. Recognizing value of separate way of handling given recurrent unit connections to self. Better task results but not less complex computationally. Higher quality learning process. uRNN (2015)-Bengio et al. addresses vanishing gradient prob, not exploding. Transition matrix is unitary. Enorced. Complex numbers, etc. Decent results. DizzyRNN (2016)-abs val. weight matrix unitary transformations, broken into pairs of 2, use 2d rotations. reasonable results. 1-2 tasks IndRNN (2018)–uses self-connection in first layer, loses much efficiency. Many-to-many connection. Very expensive. Uses relu for non-linearity rather than our absval. Gradient clipping–2 uses here–1)take norm of gradient over all params and limit to N; throw away. If larger, rescale. 2) Gradients on indiv params, scale down if exceeds limit. Lossy, loses vector direction information. Directionality is better preserved in this multi-targeted approach. First pass then prune outliers.

Besides architectural innovations, progress has been made in controlling the gradient directly.

Clipped Gradients (Paskanu, Mikolov, Bengio)

Make a chart of architectures vs features? Make a feature matrix showing what overlaps the various solutions have? That might be of more interest, and faster than simply talking about all of them individually. We can still cite them.

3 Architecture

Densely connected from input to hidden layer. Only a single, "shallow" hidden layer. Hidden layer has no biases. Hidden layer is recurrently connected by a diagonal matrix; thus every hidden neuron is solely recurrently self-connected. Self connections of hidden neurons are clamped to lie in the range [-1.0, 1.0] Self connections of hidden neurons are initialized at 1.0. Each hidden neuron makes use of the absolute value function as a nonlinearity. There are no gates Densely connected from hidden layer to output

$$\mathbf{h}_t := |\mathbf{u} \circ \mathbf{h}_{t-1} + \mathbf{V}\mathbf{x}_t| \tag{1}$$

3.1 Absolute Value Function

The absolute value function was chosen because it is has a number of desirable properties:

- Nonlinear: This is a requirement for useful computations to be possible
- Unbounded: Can I make an argument for why this is an important thing? Something somthing Turing-complete?
- Idempotent: This aspect of stability may allow for long term, high precision storage of information without specialized architectural design
- (Almost) Everywhere differentiable: It is (almost) everywhere differentiable, with unit-length derivatives of either -1 or 1
- Norm-preserving: It is norm-preserving, both for each individual component of the vector, and also for the vector as a whole. The absolute value function cannot, by itself, cause gradients to explode, nor vanish

Generic restatements about why norm preservation is important when training recurrent neural networks.

LSTM and GRU preserve the gradient over time by mostly avoiding the direct application of nonlinearities as much as possible. ResNet is intuitively similar in this sense. (TODO: Highway networks?) However, it is likely the case that many potentially useful algorithms or dynamical behaviors require the application of a long sequence of nonlinear transformations, rather than simply preserving a small number of computations/transformations over extensive spans of time.

One toy problem that requires a long, unbroken chain of nonlinear transformations is sequential parity, which will be visited below (and which LSTM and GRU are completely unable to solve for T > 10.)

Others [REF] [REF] have experimented with using the absolute value function in RNNs, but overall this is a largely unexplored activation function.

We speculate that it has not received much attention as of yet, because very special care must be taken to control the spectral radius of the recurrent matrix.

Something about absolute value function being the only continuous norm preserving non-linearity (is that true?).

3.2 Constraints

Postulate that absolute value function hasn't been explored much, particularly for recurrent neural nets, because without the proper constraints, its values tend to rapidly blow up.

3.2.1 Limited [-1, 1] range

Important to have these constraints, otherwise blows up. This is an extremely simple way of limiting the spectral radius of the matrix. In fact (double check) because of the constrained structure, these "recurrent factors" can straightforwardly tell us the singular values, eigenvalues, spectral radius of the linear transformation. (I think the fact that we use no biases helps because it remains linear rather than affine.)

Mention that on some tasks, it often ends up using very different "signatures" of recurrent factor values.

Mention importance of initializing to 1.0 here? Different section?

3.2.2 Two complementary types of gradient clipping

Some form of gradient clipping is standard practice for RNNs [REF], but to our knowledge, this is the first application of both global and localized clipping (and it is important to do both!)

Appears to be critical to limit the size of the overall norm of the gradient, particularly for some of the really long-delay tasks.

Critical to do a combination of both global and local norm clipping. Either alone is insufficient.

4 Experiments

Experimental settings, like learning rates, cooling schedules, hidden layer sizes, etc..

This experiment uses the following settings:

RMSprop*, with a learning rate of 0.001 Recurrent parameters are all initialized to 1.0 Feed forward parameters use Xavier's initialization [REF] Global gradient norm clipped at 30.0 Per-parameter gradients clipped at [-1.0, 1.0]

Initialize all recurrent factors to 1.0 Derivative clipping Weight initialization scheme *Try with mini-batches Hidden layer sizes Poor-man's gate sizes

Maybe make a table of these things for all the experiments? That would be pleasant to read

4.1 Facebook bAbI Question-Answering

bAbI is a set of 20 synthetic natural language understanding tasks created by Weston et al. [2015]. Each task has been designed to test for particular logical and reasoning capabilities, such as induction, deduction, chaining of facts, temporal ordering of events, as well as others. It is suggested that solving these "toy" problems is a necessary (although not sufficient) precondition to the development of truly intelligent systems.

In the paper, it was shown that a plain LSTM network was unable to solve any of the 20 tasks, where success had been defined as attaining an accuracy of 95% or more.

The initial, significant improvement over a plain RNN baseline was in the form of MemNNs, previously introduced in [REF]. And since then a number of new techniques have been successfully applied to bAbI, generally falling into two overall domains: augmented memories, and attentional mechanisms.

Augmented memories are a family of approaches that involve fitting a neural network (generally a recurrent one) with an "external" - and differently structured - form of memory. These external memory stores are generally more inspired by computer science than biology, and include concepts such as content addressable memory, random access memory, stacks, queues, Turing tapes, cellular automata, etc.. Almost all external memories make use of fully differentiable read and write mechanisms, although there are a few notable exceptions that are discrete in nature; requiring reinforcement learning rather than pure backpropagation in order to train.

The other significant family of approaches involve attentional mechanisms. Attentional mechanisms have met with great recent success, particularly in the domain of machine translation. Rather than having the neural network improve its ability to store information initially, attentional mechanisms all the model to go back over the sequential history of its inputs, often multiple times, conditioned on the type of output that is expected. In the case of machine translation, that conditioning will be driven by the next word in the sentence it is translating combined with what it has already produced. In the case of natural language understanding, it is the question being asked that conditions the attention to the relevant subset of facts in the preceding material. In essence, attention allows models to selectively reread do further processing on what has already been seen.

Diagnet does none of this, yet. But future experiments will undoubtedly involve memory enhancing alterations, with Diagnet as the core "controller" of the architecture.

In the accompanying paper, Weston et al introduce Memory Networks in order to overcome many of the shortcomings of a plain LSTM network.

Additionally, it has been found that adding attentional mechanisms to input history has resulted in significantly improved performance.

ADD MORE STUFF ABOUT THAT HERE

Table 1: Example from bAbI problem set

Fact	Question	Answer
Mary moved to the bathroom. John went to the hallway.	Where is Mary?	bathroom
Daniel went back to the hallway. Sandra moved to the garden.	Where is Daniel?	hallway
John moved to the office. Sandra journeyed to the bathroom.	Where is Daniel?	hallway
Mary moved to the hallway. Daniel travelled to the office.	Where is Daniel?	office
John went back to the garden. John moved to the bedroom.	Where is Sandra?	bathroom

Table 2: bAbI test set accuracies

Task	LSTM	Diagnet	+/-
Single Supporting Fact	50	**72.1**	+22.1
Two Supporting Facts	20	**31.8**	+11.8
Three Supporting Facts	20	**32.4**	+12.4
Two Arg. Relations	61	**98.9**	+37.9
Three Arg. Relations	70	**82.4**	+12.4
Yes/No Questions	48	**72.4**	+24.4
Counting	49	**82.1**	+33.1
Lists/Sets	45	**75.1**	+30.1
Simple Negation	64	**77**.0	+13.0
Indefinite Knowledge	44	**66.8**	+22.8
Basic Coreference	72	**81.3**	+9.3
Conjunction	74	**90.4**	+6.4
Compound Coreference	94	**94.2**	+0.2
Time Reasoning	27	**39.6**	+12.6
Basic Deduction	21	**49.1**	+28.1
Basic Induction	23	**46.5**	+23.5
Positional Reasoning	**51**	50.1	-0.9
Size Reasoning	52	**92.0**	+40.0
Path Finding	8	**12.4**	+4.4
Agent's Motivations	91	**96.3**	+5.3

Note that in this work we are not comparing Diagnet directly to any of the aforementioned "augmented" neural architectures. Instead, we demonstrate that Diagnet significantly outperforms LSTM and GRU on many of these 20 tasks, even despite using a much simpler and computationally efficient internal architecture.

As far as we are aware, these are the strongest results to date using a "plain" neural neural architecture on the 1k, weakly supervised version of the bAbI tasks.

This suggests that Diagnet may represent

5 Discussion

5.1 Future Work

Diagnet's architecture suggests many avenues for additional tasks and performance-enhancing modifications. Its segregated unit design is ideal for parallelization, and we have yet to tap its potential for GPU operations. Another intuitive addition would be to include an attentional mechanism or

memory augmentation, which we leave to future publications. As RNNs have historically been utilized with great success in NLP tasks, we intend to continue exploring Diagnet's capabilities in this problem space. Another area in which complex long-range sequences are of great interest is computational genomics. The roughly 1.5-gigabytes of information in the human genome may benefit from an RNN architecture supporting wider hidden layers.

6 Conclusion

Here we outline the features, mechanisms, and benchmarking performance of Diagnet. Results were robust over a variety of tasks, most notably in bAbI question-answering tasks, where Diagnet's accuracy scores showed up to 36% percent improvement over existing architectures such as LSTM and GRU. While memory augmentation and attentional mechanisms offer clear advantages as architectural additions, we choose to focus on the core RNN functionality in this paper. Future work may incorporate attentional mechanisms, memory augmentation, and additional task spaces such as semantic embeddings, genetic sequencing, and computer vision. Diagnet's novel architecture offers great flexibility in hidden layering, and number of steps in various memory tasks, as well as potential for distributed computing. Due to the fact that Diagnet does not have gating mechanisms in its current iteration, it is largely input-driven, and tends to perform best in tasks where an output sequence is expected after arbitrary delays. The work presented here represents a first glance at Diagnet's mechanisms and potential, which the authors hope to expand in future work.

References

Scott E. Fahlman. The recurrent cascade-correlation architecture. In *NIPS*, 1990.

J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *ArXiv e-prints*, February 2015.