Memorization in Recurrent Neural Networks

Tegan Maharaj [1 2] **David Krueger** [1 3] **Tim Coojimans** [1 3]

Abstract

We present work in progress on understanding generalization in deep networks by analyzing differences in learning behaviour of recurrent neural networks (RNNs) on noise data vs. real data - i.e., by investigating memorization. There has been a recent surge of interest in explaining the generalization performance of deep neural networks (DNNs), partially spurred by the observation that feedforward DNNs can fit random noise datasets with 0 training error. RNNs are typically extremely deep; analyzing learning behaviour in RNNs thus gives an interesting perspective for understanding memorization, generalization, and effective capacity in deep networks. We demonstrate that fitting noise with RNNs is more difficult than in feedforward networks; standard gradient-based optimization fails to reach 0 training error. We make several other observations comparing and contrasting our results with previous work, and suggest a suite of experiments for future work.

1. Introduction

The question of whether deep networks fit data by finding patterns or simply memorizing examples has been the subject of recent work. Zhang et al. (2016) point out that traditional measures of capacity do not explain the (very good) generalization performance of deep networks, given that these networks are typically massively over-parameterized, and are capable of fitting random noise. Arpit et al. examine the question from another angle, demonstrating that deep networks have qualitatively different behavior on random vs noise data, and suggesting that DNNs generalization is due to their propensity to learn simple patterns first. Arpit et al. call for a data-dependent notion of capacity to explain DNN generalization, and **?** provide one, establishing non-vacuous generalization bounds for stochastic DNNs.

[1]MILA (Montreal Institute for Learning Algorithms) [2]Montreal Polytechnique [3]University of Montreal. Correspondence to: Tegan Maharaj <tegan.maharaj@polymtl.ca>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

All these works, however, examine feed-forward networks; we extend this investigation to recurrent neural networks (RNNs), performing experiments on the tasks of character-level language modeling (Penn Treebank) and classification (sequential MNIST). While some of the results in these previous work on feedforward networks carry over to RNNs, we also find significant differences.

Our findings so far are:

1. Fitting random noise with RNNs is much harder than with feedforward nets. While we were able to fit small sets of noise data to some extent, we fail to substantially reduce training error on either task when training on the full (noised) datasets. This contrasts strongly with the results of Zhang et al. (2016) on feedforward nets.
2. Like Zhang et al. (2016) and Arpit et al. we find that label noise is more difficult to fit than input noise.
3. Like Arpit et al., we find that models trained on mixed datasets (containing both real and noise examples) first learn patterns which generalize to the validation set (i.e. to unseen *real* data).
4. We find that easy examples exist in the random input version of the sequential MNIST task. This contrasts with the results of Arpit et al.; in their experiments with MLPs, random MNIST inputs appear to have equal difficulty.

RNNs are known to be difficult to optimize in general, but the exceptional difficulty of fitting random data is surprising. Having noted the difficulty of fitting random data with RNNs, our next step will be a more thorough attempt to fit RNNs on a broader range of synthetic tasks that allow us to control relevant factors such as sequence length and dataset size. Going forward, we also plan to replicate more of the experiments of Arpit et al. on RNNs.

Unlike for feedforward nets, for RNNs, the ability to memorize examples may actually be desirable. At a high level, the distinguishing feature of RNNs is their ability to create a fixed-length representation of variable length data using a finite number of parameters. From this viewpoint, an idealized RNN would perform lossless compression of any input sequence, since the relevance of past and present inputs depends on future inputs and is hard to anticipate. Meanwhile,

the output mapping could take responsibility for discarding information stored in the current hidden state which is irrelevant to the current output. The difficulty of optimizing RNNs suggests that their learned compression is both 1) is lossy, and 2) strongly data-dependent.

1.1. Structure of this paper

We first define RNNs formally, then discuss concepts of depth, effective capacity, and memorization as they apply to RNNs, and motivate our empirical approach in line with previous work. We present and discuss results of our investigation of learning behaviour, generalization, and effective capacity as influenced by noise vs. real data, and conclude by proposing further experiments, conjectures to be investigated, and possible avenues for theoretical corroboration of our results.

2. Background and Related Work

2.1. Recurrent neural network definitions and notation

A recurrent neural network (RNN) is a neural network which processes sequential input $(x_1, x_2, .., x_t, ..., x_n)$, using a nonlinear function f to construct corresponding representations (called hidden states, or activations) $(h_1, h_2, ..., h_t, ..., h_n)$, each of which depends on the input and on the previous timestep:

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

For a **simple RNN**, or 'vanilla' RNN, this is most commonly implemented with two sets of recurrent parameters; W for the input, and U for the hidden-to-hidden transition, and where the activation function f is usually a logistic sigmoid or hyperbolic tangent (tanh), sometimes rectified linear (ReLU).:

$$h_t = f(Wx_t, Uh_{t-1}) \tag{2}$$

RNNs are usually trained with stochastic gradient descent (SGD) via backpropagation through time (BPTT).The repeated use of the same parameters for each timestep of input can be viewed as repeated applications of a transition operator with learned parameters. While this characteristically allows RNNs to maintain a 'memory' of information from past timesteps, because of repeated multiplications they are vulnerable to problems of both vanishing and exploding gradients, as demonstrated by Bengio et al. (1994). This observation motivates the use of gated architectures, as used in Long short-term memory networks (LSTM) (Hochreiter & Schmidhuber, 1997a) and Gated recurrent units (GRU).

2.2. Recurrence and Depth

RNNs are the deepest neural networks. But unlike with feedforwards nets, the depth in recurrent networks mostly comes from the repeated application of the same transition operator. This may cause an RNNs hidden activations (and their gradients) to increase or decrease exponentially (Bengio et al., 1994) (although inputs, noise, and nonlinearities may all counter-act this effect to some extent).

2.3. RNN capacity

Recurrent neural networks are theoretically capable of representing universal Turing machines (Siegelmann & Sontag, 1995) (although in practice finite numerical precision limits their capacity), so effective capacity is particularly important when trying to look at RNNs. We define effective capacity as in Arpit et al.; effective capacity is an attribute of a *learning algorithm* (including a model and training procedure), and denotes the set of hypotheses which that algorithm could reach given *some* training set.

3. Experiments and Discussion

Experiments are designed to examine differences between noise vs. real data, in order to assess memorization behaviour. We perform experiments on sequential MNIST (Le et al., 2015) (**sMNIST**) and character-level Penn Treebank (Marcus et al., 1993) **cPTB**). sMNIST is a classification task: given the sequence of pixels in a vector representing an image, classify the digit shown in that image. cPTB is a language-modeling task: given the sequence of characters thus far, predict the next. cPTB is usually measured in log-likelihood/bits-per-character (BPC)[1]), but in order to assess memorization, we look at per-character and per-sequence *accuracy* on cPTB.

Unless otherwise noted: all experiments for sMNIST involve a single-layer LSTM (Hochreiter & Schmidhuber, 1997b) with orthogonal initialization and Layer Normalization (Ba et al., 2016) optimized with RMSProp (Tieleman & Hinton, 2012) with decay rate 0.5 and learning rate $1e^{-3}$. All experiments for cPTB use single-layer LSTMs with 1000 units, initialized orthogonally, optimized with ADAM (Kingma & Ba, 2014), with sequences of 100, gradient norm clipping set to 1, and learning rate of $2e^{-3}$.

3.1. Differences on noise vs. real data early in training

Results on sMNIST, shown in 1, are similar to findings of (Arpit et al.) with feed-forward nets on MNIST in that some examples are easier than others, but differ in that this is also true for X noise (random data).

[1]BPC is a deterministic function of NLL; $BPC = \log_2(NLL)$

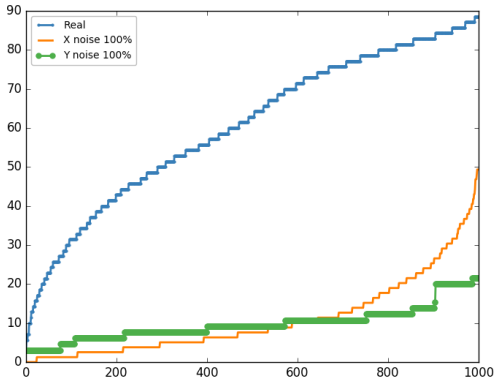

Figure 1. Normalized percentage of times a given example is correctly classified after 100 epochs of training, averaged over 70 experiments with different random seeds on 1000 examples from sMNIST. More examples are consistently learned earlier for real data, and random data (X Noise, orange) is harder to fit than random labels (Y Noise, green). More significantly, we note that all three curves show evidence of variability in the difficulty of fitting examples, in contrast to previous results with feedforward nets.

3.2. Differences in learning behaviour and generalization

First in Figure 2 we plot. Observe that we can't fit noise on the full dataset or 1/8th, so we examine different noise levels only for 1/64th (3162 examples), 1/512th (790 examples), and 1/4096th (127 examples), shown in Figure 3.

3.3. Differences in effective capacity

Unlike typical feed-forward nets trained on common benchmark tasks like MNIST and CIFAR, typical RNNs trained on common benchmark tasks like sequential MNIST and Penn Treebank do not usually get close to 100% training accuracy. We observe in Figure 3 that it requires more effective capacity to fit noise.

Language models cannot hope to achieve 100% accuracy on tasks, since the past sequence of characters does not determine the next character. In a finite dataset, such ambiguities may not arise. For instance, the Penn Treebank corpus is composed of a single continuous example, and in principle, an RNN could be trained to memorize this string. In practice, however, RNNs are trained on this task using truncated backprop through time (TBPTT), and the hidden state is often reset at regular intervals, for instance, after every 100 steps, or at the end of every sentence.

In future work, we plan to design tasks which do not suffer from such ambiguities. Designing synthetic tasks in such a way is straightforward, but removing ambiguities from cPTB requires more thought and is subject to design choices.

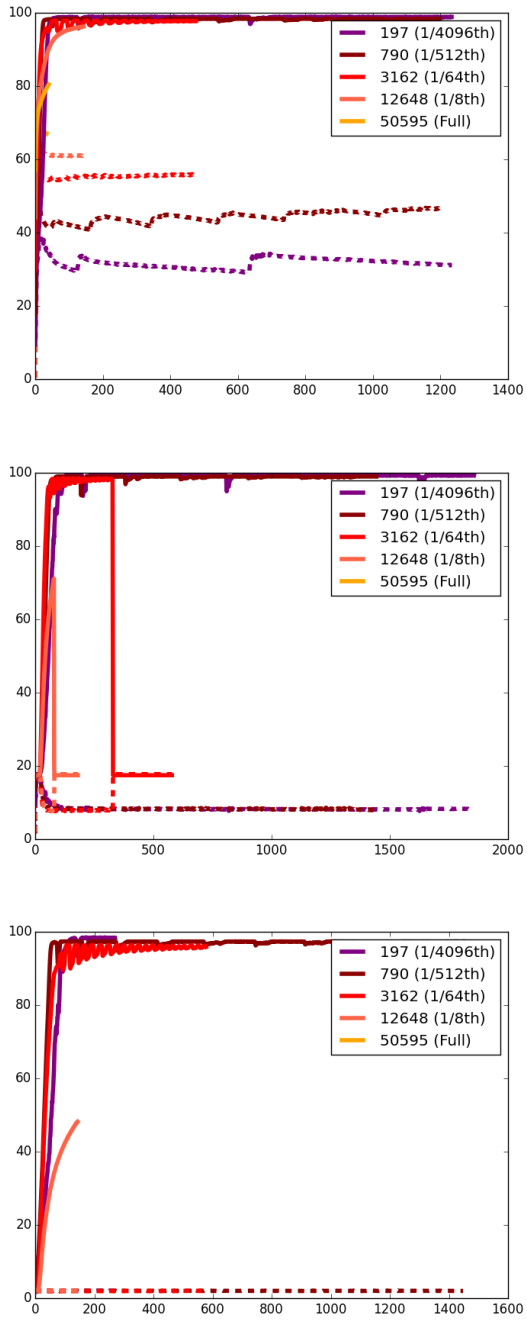

Figure 2. Training (solid) and validation (dotted) curves for differently sized subsets of the data in increasing powers of 8, for real data (top), 100% X noise (middle), and 100% Y noise (bottom) on cPTB. Except on very small datasets (1/64th and less), optimization fails. on noise data.

One simple idea which would ameliorate but not solve the problem is to measure performance on the end of training sequences only (where the model has more context).

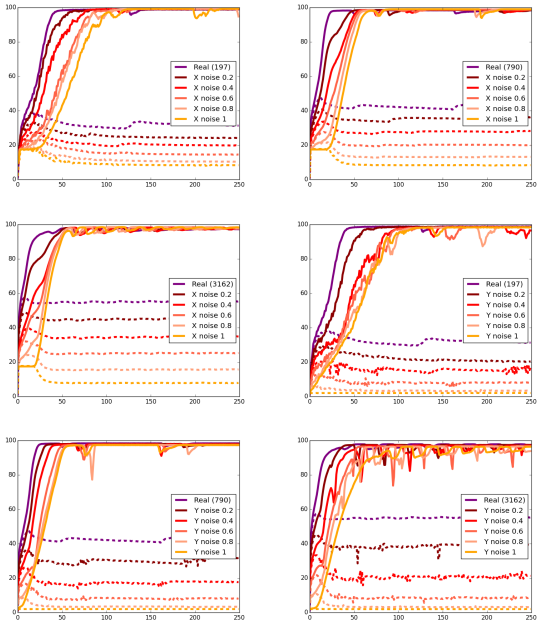

Figure 3. Training (solid) and validation (dotted) curves for different values of X noise (left column) and Y noise (right column) and differently sized subsets of the data (rows: [197, 790, and 3162] examples from top to bottom)

4. Discussion

Our results so far reveal some interesting differences between RNNs and feedforward nets. The most significant is the difficulty of fitting random noise with RNNs. The difficulty of fitting noise with RNNs suggests that these models might be inherently more reliant on learning patterns than feedforward models, and thus might generalize better. Intuitively, the repeated application of the same transition operator might cause repeated subsequences within a given input to be processed in a similar way (although in principle, the difference in context could cause them to be treated entirely differently).

On the other hand, the difficulty of fitting noise makes is attractive as challenge for optimization algorithms. We would expect optimizers which can fit noise datasets to yield substantial improvements on training performance for real data as well. This could be especially useful because RNN optimization is more difficult than feed-forward optimization, and remains a larger obstacle. Still it is unclear whether this would improve generalization, since poor optimization may actually contribute to generalization in RNNs. We hypothesize that better optimization would improve validation performance up to some point (dependent on other aspects of the problem and method), and degrade it thereafter. If overfitting becomes a challenge, regularization might still allow one to reap the benefits of improved optimization.

5. Future Work

In future work we plan to evaluate RNN optimization on a broader range of synthetic tasks. These experiments are designed to measure how sequence length, capacity, and dataset characteristics (categorical vs. real-vector values inputs/outputs; the number of examples, inputs, outputs) affect the ability of RNNs to fit a training set (e.g. with 100% accuracy).

We propose the following types of synthetic tasks:

1. **next-step prediction**: given input and output sequences of length N, predict y_n from $x_{1:n}, y_{1:n-1}$ (for all $1 \leq n \leq N$). Language modeling is an instance of this setting where $y_n \doteq x_{n-1}$.

2. **sequence-to-sequence (seq2seq)**: given an input $x \doteq x_{1:i}$ and output $y \doteq y_{1:j}$, predict y_n from $x_{1:i}$ (for all $1 \leq n \leq N$).

3. **vector-to-sequence / sequence-to-vector**: Special cases of seq2seq where the input / output (respectively) has only one time-step. Sequential MNIST is an instance of sequence-to-vector.

In every case, the inputs and targets can be independently chosen to be categorical or real-valued. Combining all of the options yields a total of 16 synthetic task settings. We note that previous work focused entirely on classification tasks on real-valued inputs, so changing the type of input and output for feedforward experiments would also be novel.

References

Arpit, Devansh, Jastrzbski, Stanisaw, Ballas, Nicolas, Krueger, David, Bengio, Emmanuel, Kanwal, Maxinder S., Maharaj, Tegan, Fischer, Asja, Courville, Aaron, Bengio, Yoshua, and Lacoste-Julien, Simon. A closer look at memorization in deep networks. *ICML 2017 (to appear)*.

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.

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

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

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Le, Quoc V, Jaitly, Navdeep, and Hinton, Geoffrey E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

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

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.