The Statistical Recurrent Unit

Junier B. Oliva [1] **Barnabás Póczos** [1] **Jeff Schneider** [1]

Abstract

Sophisticated gated recurrent neural network architectures like LSTMs and GRUs have been shown to be highly effective in a myriad of applications. We develop an un-gated unit, the statistical recurrent unit (SRU), that is able to learn long term dependencies in data by only keeping moving averages of statistics. The SRU's architecture is simple, un-gated, and contains a comparable number of parameters to LSTMs; yet, SRUs perform favorably to more sophisticated LSTM and GRU alternatives, often outperforming one or both in various tasks. We show the efficacy of SRUs as compared to LSTMs and GRUs in an unbiased manner by optimizing respective architectures' hyperparameters for both synthetic and real-world tasks.

1. Introduction

The analysis of sequential data has long been a staple in machine learning. Domain areas like natural language (Zaremba et al., 2014; Vinyals et al., 2015), speech (Graves et al., 2013; Graves & Jaitly, 2014), music (Chung et al., 2014), and video (Donahue et al., 2015) processing have recently garnered much attention. While the study of sequences itself is broad and may be extended to general functional analysis (Ramsay & Silverman, 2002), most recent success has been from neural network based models, especially from recurrent architectures.

Recurrent networks are dynamical systems that represent time recursively. For example, the simple recurrent unit (Elman, 1990) contains a hidden state that itself depends on the previous hidden state. However, training such networks has been observed to be difficult in practice due to exploding and vanishing gradients when propagating error gradients through time (Hochreiter et al., 2001). While exploding gradients can be mitigated with techniques like gradient clipping and normalization (Pascanu et al., 2013), vanishing gradients may be harder to deal with. As a result, sophisticated gated architectures like Long-Short Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Unit (GRU) networks (Cho et al., 2014) have been developed. These gated architectures contain "memory cells" along with gates to control how much they decay through time thereby aiding the networks' ability to learn long term dependencies in sequences.

Notwithstanding, there are still challenges in capturing long term dependencies in gated architectures (Le et al., 2015). In this paper we present a simple un-gated architecture, the Statistical Recurrent Unit, that often outperforms these more complicated alternatives. Although the SRU keeps only simple moving averages of summary statistics, its novel architecture makes it more adept than previous gated units for capturing long term information in sequences and comparing them across different windows of time. For instance, the SRU, unlike traditional recurrent units, can obtain a multitude of viewpoints of the past by simple linear combinations of only a few averages. We shall illustrate the efficacy of the SRU below using both real-world and synthetic sequential data tasks.

The structure of the paper is as follows: first we detail the architecture of the SRU as well as provide several key intuitions and insights for its design; after, we describe our experiments comparing the SRU to popular gated alternatives, and we perform a "dissective" study of the SRU, gaining further understanding of the unit by exploring how various hyper-parameters affect performance; finally, we discuss conclusions from our study.

2. Model

The SRU maintains long term sequential dependencies in a rather intuitive fashion–through summary statistics. As the name implies, statisticians often employ summary statistics when trying to represent a dataset. Quite naturally then, we look to an algorithm that itself learns to represent data seen previously in much the same vein as a neural statistician (Edwards & Storkey, 2016).

Of course, unlike with unordered i.i.d. samples, simply averaging statistics of sequential points will lose valuable

[1] Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Junier B. Oliva <joliva@cs.cmu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

temporal information. The SRU maintains sequential information in two ways: first, we generate recurrent statistics that depend on a context of previously seen data; second, we generate moving averages at several scales, allowing the model to distinguish the type of data seen at different points in the past. We expound on these methods for creating temporally-aware statistics below.

We shall see that the statistical design of the SRU yields a powerful yet simple model that is able to analyze sequential data and, on the fly, create summary statistics for learning over sequences. Furthermore, through the use of ReLUs and exponential moving averages, the SRU is able to mitigate vanishing gradient issues that are common to many recurrent units.

2.1. Recurrent Statistics

We consider an input sequence of real valued points $x_1, x_2, \ldots, x_T \in \mathbb{R}^d$. As seen in the second row of Table 1, we can compute a vector of statistics $\phi(x_i) \in \mathbb{R}^D$ for each point. Here, each vector $\phi(x_i)$ is independent of other points x_j for $j \neq i$. One may then average these vectors as $\mu = \frac{1}{T}\sum_{i=1}^{T}\phi(x_i)$ to produce summary statistics of the sequence. This approach amounts to treating the sequence as a set of i.i.d. points drawn form some distribution and marginalizing out time. Clearly, here one will lose temporal information that will be useful for many sequence related ML tasks. It is interesting to note that global average pooling operations have gained a lot of recent traction in convolutional networks (Lin et al., 2013; Iandola et al., 2016). Analogously to the i.i.d. statistic approach, global averaging will lose spatial information, yet the high-level summary statistics provide an effective representation. Still, not marginalizing out time should provide a more robust approach for sequence tasks, thus we consider the following methods for producing statistics.

First, we provide temporal information whilst still utilizing averages through recurrent statistics that also depend on the values of previous points (see third row of Table 1). That is, we compute our statistics on the i^{th} point x_i not only as a function of x_i, but also as a function of the previous statistics of x_{i-1}, $\vec{\gamma}_{i-1}$ (which itself depends on $\vec{\gamma}_{i-2}$, etc.):

$$\vec{\gamma}_1 = \gamma(x_1, \vec{\gamma}_0),\ \vec{\gamma}_2 = \gamma(x_2, \vec{\gamma}_1),\ \ldots \tag{1}$$

where $\gamma(\cdot,\cdot)$ is a function for producing statistics given the current point and previous statistics, and $\vec{\gamma}_0$ is a constant initial vector for convention. We note that from a general standpoint if given a flexible model and enough dimensions, then recurrent summary statistics like (1) can perfectly encode ones sequence. Take for instance the following illustrative example where $x_i \in \mathbb{R}^+$ and statistics

$$\vec{\gamma}_i = (0, \ldots, 0, Tx_i, 0, \ldots) \tag{2}$$

$$\vec{\gamma}_{i+1} = (0, \ldots, 0, 0, Tx_{i+1}, 0, \ldots). \tag{3}$$

That is, one records the i^{th} input in the i^{th} index. When averaged the statistics will be $\frac{1}{T}\sum_{i=1}^{T}\vec{\gamma}_i = (x_1, x_2, \ldots)$, i.e. the complete sequence. Such recurrent statistics will undoubtedly suffer from the curse of dimensionality. Hence, we consider a more restrictive model of recurrent statistics which we expound on below (6).

Second, we provide even more temporal information by considering summary statistics at multiple scales. As a simple hypothetical example, consider taking multiple means across separate time windows (for instance taking means over indices 1-10, then over indices 11-20, etc.). Such an approach (4) will illustrate how summary statistics evolve through time.

$$\underbrace{\phi_1, \ldots, \phi_{10}}_{\mu_{1:10}},\ \underbrace{\phi_{11}, \ldots, \phi_{20}}_{\mu_{11:20}},\ \ldots. \tag{4}$$

In practice, we shed light on the dynamics of statistics through time by using several averages of the same summary statistics. The SRU will use exponential moving averages $\mu_i = \alpha\vec{\gamma}_i + (1-\alpha)\mu_{i-1}$ to compute means; hence, we consider multiple weights by taking the exponential means at various scales $\alpha_1, \ldots, \alpha_m$ as shown in the last row of Table 1. Later we show that this multi-scaled approach is capable of a combinatorial number of viewpoints of past statistics through simple linear combinations.

Table 1. Methods for keeping statistics of sequences.

inputs	$x_1,\ x_2,\ \ldots,\ x_T$
i.i.d. statistics	$\phi(x_1),\ \phi(x_2),\ \ldots,\ \phi(x_T)$
recurrent statistics	$\gamma\left(x_1, \vec{\gamma}_0\right),\ \gamma(x_2, \vec{\gamma}_1),\ \ldots, \gamma(x_T, \vec{\gamma}_{T-1})$
recurrent multi-scaled statistics	$\alpha_1^{T-1}\gamma(x_1,\vec{\gamma}_0),\ \alpha_1^{T-2}\gamma(x_2,\vec{\gamma}_1),\ \ldots$ $\ddots$ $\alpha_m^{T-1}\gamma(x_1,\vec{\gamma}_0),\ \alpha_m^{T-2}\gamma(x_2,\vec{\gamma}_1),\ \ldots$

2.2. Update Equations

We have discussed in broad terms how one may create temporally-aware summary statistics through multi-scaled recurrent statistics. Below, we cover specifically how the SRU creates and uses summary statistics for sequences.

Recall that our input is a sequence of ordered points: $x_1, x_2, \ldots, x_t \in \mathbb{R}^d$. Throughout, we apply an element-wise non-linearity $f(\cdot)$, which we take to be the ReLU (Jarrett et al., 2009; Nair & Hinton, 2010): $f(\cdot) = \max(\cdot, 0)$. The SRU operates via exponential moving averages, $\mu^{(\alpha)} \in \mathbb{R}^s$ (7), kept at various scales $\alpha \in A = \{\alpha_1, \ldots, \alpha_m\}$, where $\alpha_i \in [0, 1)$. These moving averages, $\mu^{(\alpha)}$, are of recurrent statistics φ (6) that are dependent not only on the current input but also on features of averages, r (5). The moving averages are then concatenated as $\mu = (\mu^{(\alpha_1)}, \ldots, \mu^{(\alpha_m)})$ and used to create an output o (8) that is fed upwards in the network.

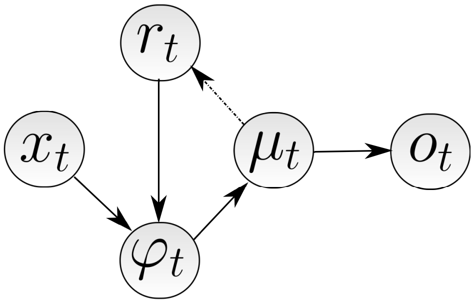

Figure 1. Graphical representation of the SRU. Solid lines indicate a dependence on the current value of a node. Dashed lines indicate a dependence on the previous value of a node. We see that both the current point x_t as well as a summary of the previous data r_t are used to make statistics φ_t, which in turn are used in moving averages μ_t, finally an output o_t is feed-forward through the rest of the network.

We detail the update equations for the SRU below (and in Figure 1):

$$r_t = f\left(W^{(r)}\mu_{t-1} + b^{(r)}\right) \tag{5}$$

$$\varphi_t = f\left(W^{(\varphi)}r_t + W^{(x)}x_t + b^{(\varphi)}\right) \tag{6}$$

$$\forall \alpha \in A,\ \mu_t^{(\alpha)} = \alpha\mu_{t-1}^{(\alpha)} + (1-\alpha)\varphi_t \tag{7}$$

$$o_t = f\left(W^{(o)}\mu_t + b^{(o)}\right). \tag{8}$$

In practiced we noted that it suffices to use only a few α's such as $A = \{0, 0.25, 0.5, 0.9, 0.99\}$.

It is worth noting that previous work has considered capturing recurrent information at various timescales in the past. For instance, Koutnik et al. (2014) considers an RNN scheme that divides the hidden state into different modules for use at different frequencies. Furthermore, exponential averages in recurrent units have been considered previously, e.g. (Mikolov et al., 2015; Bengio et al., 2013; Jaeger et al., 2007). However, such works are more akin to un-gated GRUs since they consider only one scale per feature, limiting the views available per statistic to just one. The use of ReLUs in recurrent units has also been recently explored by Le et al. (2015), however there no statistics are kept and their use is limited to the simple RNN when initialized in a special manner.

2.3. Intuitions from Mean Map Embeddings

The design of the SRU is deliberately chosen to allow for long term dependencies to be learned. To better elucidate the design and its intuition, let us take a brief excursion to another use of (summary) statistics in machine learning for the representation of data: mean map embeddings (MMEs) of distributions (Smola et al., 2007). At its core, the concept of MMEs is that one may embed, and thereby represent, a distribution through statistics (such as moments). The MME for a distribution $\mathcal{D}$ given a positive semidefinite kernel k is:

$$\mu[\mathcal{D}] = \mathbb{E}_{X\sim\mathcal{D}}\left[\phi_k(X)\right], \tag{9}$$

where ϕ_k are the reproducing kernel Hilbert space (RKHS) features of k, which may be infinite dimensional. To represent a set $Y = \{y_1, \ldots, y_n\} \overset{iid}{\sim} \mathcal{D}$ one would use an empirical mean version of the MME:

$$\mu[Y] = \frac{1}{n}\sum_{i=1}^{n}\phi_k(y_i). \tag{10}$$

Numerous works have shown success in representing distributions and sets through MMEs (Muandet et al., 2016). One interpretation for the design of SRUs is that we are modifying MME's for use on sequences. Of course, one way of applying MMEs directly on sequences is to simply ignore the non-i.i.d. nature of sequences and treat points as comprising a set. This however loses important sequential information, as previously mentioned. Below we discuss the specific modifications we make from traditional MMEs and the benefits they yield.

2.3.1. Data-driven Statistics

First, we note the clear analogue between the mean embedding of a set Y, $\mu[Y]$ (10), and the moving average $\mu^{(\alpha)}$ (7). The moving averages $\mu^{(\alpha)}$ are clearly serving as summary statistics of previously seen data. However, the statistics we are averaging for $\mu^{(\alpha)}$, φ (6), are not comprised of a-priori RKHS features as is typical with MMEs, but rather are learned non-linear features. This has the benefit of using data-driven statistics, and may be interpreted as using a linear kernel in the learned features.

2.3.2. Recursive Statistics from the Past

Second, recall that typical MMEs use statistics that depend only on a single point x, $\phi_k(x)$. As aforementioned this is fine for i.i.d. data, however it loses sequential information when averaged. Instead, we wish to assign statistics that depend on the data we have seen so far, since it provides context for one's current point in the sequence. For instance, one may want to have a statistic that keeps track of the difference between the current point and the mean of previous data. We provide a context based on previous data by making the statistics considered at time t, φ_t (6), a function not only of x_t but also of $\{x_1, \ldots, x_{t-1}\}$ through r_t (5). r_t may be interpreted as a condensation of the sequence seen so far, and allows us to keep sequential information even through an averaging operation.

2.3.3. Multi-scaled Statistics

Third, the use of multi-scaled moving averages of statistics gives the SRU a simple and powerful rich view of past data that is unique to this recurrent unit. In short, by keeping moving averages at different scales $\{\alpha_1, \ldots, \alpha_m\}$, we are able to uncover differences in statistics at various times in the past. Note that we may unroll moving averages as:

$$\mu_t^{(\alpha)} = (1-\alpha)\left(\varphi_t + \alpha\varphi_{t-1} + \alpha^2\varphi_{t-2} + \ldots\right) \qquad (11)$$

Thus, a smaller α weighs current statistics more than older statistics; hence, a concatenated vector $\mu = (\mu^{(\alpha_1)}, \ldots, \mu^{(\alpha_m)})$ itself provides a multi-scale view of statistics through time (see Figure 2). For instance, keeping statistics for short and long terms pasts already yields information on the evolution of the sequence through time.

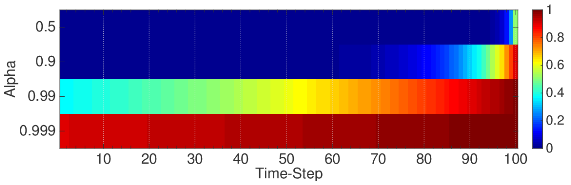

Figure 2. We may unroll the moving average updates as (11). To visualize the different emphasis in the past that varying α has on statistics we plot the values of weights in moving averages (i.e. α^i) for 100 points in the past across rows. We see that alpha values closer to 0 focus only on the recent past, where values close to 1 maintain an emphasis on the distant past as well.

2.4. Viewpoints of the Past

An interesting and useful property of keeping multiple scales for each statistic is that one can obtain a combinatorial number of viewpoints of the past through simple linear combinations of ones statistics. For instance, for properly chosen $w_j, w_k \in \mathbb{R}$, $w_j\mu^{(\alpha_j)} - w_k\mu^{(\alpha_k)}$ provides an aggregate of statistics from the past for $\alpha_j > \alpha_k$ (Figure 3). Of course, more complicated linear combinations may be performed to obtain richer viewpoints that are comprised of multiple windows. Furthermore, by using a linear projection of our statistics μ_t, as we do with o_t (8), we are able to compute output features of combined viewpoints of several statistics.

This kind of multi-viewpoint perspective of previously seen data is difficult to produce in traditional gated recurrent units since they must encode where in the sequence they currently are and then store an activation on separate nodes per each viewpoint for future use. SRUs, on the other hand, only need to take simple linear combinations to capture various viewpoints in the past. For example, as shown above, statistics from just the distant past are available via a simple subtraction of two moving averages (Figure 3, row 1). Such a windowed view would require a gated unit to learn to stop averaging after a certain point in the sequence, and the corresponding statistic would not yield any information outside of this window. *In contrast, each statistic kept by the SRU provides a combinatorial number of varying perspectives in the past through linear combinations and their multi-scaled nature.*

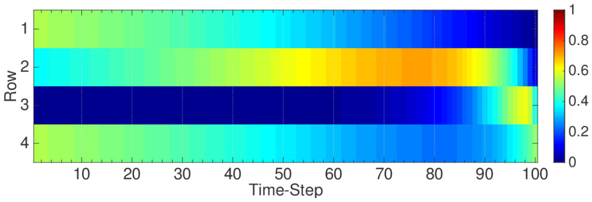

Figure 3. We visualize the power of taking linear combinations of $\mu^{(\alpha)}$'s for providing different viewpoints into past data. In row 1 we show the effective weights that would be used for weighing statistics φ_t if one considers $.001^{-1}\mu^{(.999)} - .01^{-1}\mu^{(.99)}$; we see that this is equivalent to considering only statistics from the distant past. Similarly, we show the effective weights when taking $.01^{-1}\mu^{(.99)} - .1^{-1}\mu^{(.9)}$ and $.1^{-1}\mu^{(.9)} - .5^{-1}\mu^{(.5)}$ on rows 2 and 3 respectively. We see that these linear combinations amount to considering viewpoints concentrated at various points in the past. Lastly its worth noting that more complicated linear combinations may lead to even richer views on previous statistics; for instance, we show $.001^{-1}\mu^{(.999)} - .01^{-1}\mu^{(.99)} + \frac{.5}{.09}\mu^{(.9)}$ on row 4, which concentrates on the statistics of the distant and very recent past, but de-emphasizes statistics of data from less recent past.

2.5. Vanishing Gradients

As previously mentioned, it has been shown that vanishing gradients make learning recurrent units difficult due to an inability to propagate error gradients through time. Notwithstanding its simple un-gated structure, the SRU features several safeguards to alleviate vanishing gradients. First, units and statistics are comprised of ReLUs. ReLUs have been observed to be easier to train for general deep networks (Nair & Hinton, 2010) and have had success in recurrent units (Le et al., 2015). Intuitively, ReLUs allow for the propagation on error on positive inputs without saturation and vanishing gradients as with traditional sigmoid units. The ability of the SRU to use ReLUs (without any special initialization) makes it especially adept at learning long term dependencies through time.

Furthermore, the explicit moving average of statistics allows for longer term learning. Consider the following derivative of the error signal E w.r.t. an element $\left[\mu^{(\alpha)}_{t-1}\right]_k$ of the unit's moving averages when $[\varphi_t]_k = 0$:

$$\frac{\partial E}{\partial \left[\mu^{(\alpha)}_{t-1}\right]_k} = \frac{\partial \left[\mu^{(\alpha)}_{t}\right]_k}{\partial \left[\mu^{(\alpha)}_{t-1}\right]_k} \frac{\partial E}{\partial \left[\mu^{(\alpha)}_{t}\right]_k} = \alpha \frac{\partial E}{\partial \left[\mu^{(\alpha)}_{t}\right]_k}.$$

That is, the factor α directly controls the decay of the error signal through time. Thus, by including an α explicitly near 1 (i.e. 0.999), the decay for that moving average can be made minuscule for the lengths of sequences in ones data. Also, it is interesting to note that, with a large α near 1, SRUs with ReLUs can implement part of the functionality of a gate ("remembering") by carrying through the previous moving average $[\mu^{(\alpha)}_{t-1}]_k$ when the corresponding statistic $[\varphi_t]_k$ has be zeroed out (7). The other functionality of a gate (forgetting) can be had by including an α near 0; if the ReLU statistic is not zeroed out, then the moving average for a small α will "forget" the previous value.

3. Experiments

We compared the performance of the SRU[1] to two popular gated recurrent units, the GRU and LSTM unit. All experiments were performed in `Tensorflow` (Abadi et al., 2016) and used the standard implementations of `GRUCell` and `BasicLSTMCell` for GRUs and LSTMs respectively. In order to perform a fair, unbiased comparison of the recurrent units and their hyper-parameters, which greatly affect performance (Bergstra & Bengio, 2012), we used the `Hyperopt` (Bergstra et al., 2015) hyper-parameter optimization package. We believe that such an approach gives each algorithm a fair shot to succeed without injecting biases from experimenters or imposing gross restrictions on architectures considered.

In all experiments we used SGD for optimization using gradient clipping (Pascanu et al., 2013) with a norm of 1 on all algorithms. Unless otherwise specified 100 trials were performed to search over the following hyper-parameters on a validation set: one, `initial_learning_rate` the initial learning rate used for SGD, in range of $[\exp(-10), 1]$; two, `lr_decay` the multiplier to multiply the learning rate by every 1k iterations, in range of $[0.8, 0.999]$; three, `dropout_keep_rate`, percent of output units that are kept during dropout, in range $(0, 1]$; four, `num_units` number of units for recurrent unit, in $\{1, \ldots, 256\}$. In addition, the following two parameters were searched over for the SRU: `num_stats`, the dimensionality of φ (6), in $\{1, \ldots, 256\}$; `summary_dims`, the dimensionality of r (5), in $\{1, \ldots, 64\}$.

3.1. Synthetic Recurrent Unit Generated Data

First we provide evidence that traditional gated units have difficulties capturing the same type of multi-scale recurrent statistic based dependencies that the SRU offers. We show the relative inefficiency of traditional gated units at learning long term dependencies of statistics by considering 1d synthetic data from a ground truth SRU.

We begin the sequences with $x_1 \overset{iid}{\sim} \mathcal{N}(0, 100^2)$, and x_t is the results of a projection of o_t. We generate a total of 176 points per sequence for 3200 training sequences, 400 validation sequences, and 400 testing sequences.

The ground truth statistical recurrent unit has three statistics ϕ_t (6): the positive part of inputs $(x)_+$, the negative part of inputs $(x)_-$, and an internal statistic, z. We use $\alpha \in \{\alpha_i\}_{i=1}^5 = \{0.0, 0.5, 0.9, 0.99, 0.999\}$. Denote $\mu_+^{(\alpha)}$, $\mu_-^{(\alpha)}$, $\mu_z^{(\alpha)}$ as the moving averages using α for each respective statistic. The internal statistic z does not get used (through r_t (5)) in updating the statistics for $(x)_+$ or $(x)_-$. z is itself updated as:

$$\begin{aligned} z_t = {} & (z_{t-1})_+ \\ & + \left(\mu_+^{(\alpha_4)} - \mu_+^{(\alpha_5)} - 0.01\right)_+ - \left(-\mu_-^{(\alpha_4)} + \mu_-^{(\alpha_5)} - 0.01\right)_+ \\ & - \left(-\mu_+^{(\alpha_4)} + \mu_+^{(\alpha_5)} - 0.05\right)_+ + \left(\mu_-^{(\alpha_4)} - \mu_-^{(\alpha_5)} - 0.05\right)_+, \end{aligned}$$

where each of the summands are r_t features. Furthermore we have $o_t \in \mathbb{R}^{15}$ (8):

$$o_t = \left((x_t)_+,\ -(x_t)_-,\ v_1^T \mu_t,\ \ldots,\ v_{13}^T \mu_t\right),$$

where v_j's where initialized and fixed as $(v_j)_k \overset{iid}{\sim} \mathcal{N}(0, (\frac{1}{100})^2)$. Finally the next point is generated as:

$$x_{t+1} = (x_t)_+ - (x_t)_- + w^T o_{t,3:},$$

where w was initialized and fixed as $(w)_k \overset{iid}{\sim} \mathcal{N}(0, 1)$, and $o_{t,3:}$ are the last 13 dimensions of o_t.

After the ground truth SRU was constructed we generated the training, validation, and testing sequences. As can be seen in Figure 4, the sequences follow a simple pattern: at the start negative values are quickly pushed to zero and positive values follow a parabolic line until hitting zero, at which point they slope downward depending on initial values. While simple, it is clear that trained recurrent units must be able to hold long-term information since all sequences converge at one point and future behaviour depends on initial values.

We look to minimize the mean of squared errors (MSE); that is, the loss we consider per sequence is $\frac{1}{175}\sum_{t=1}^{175} |x_{t+1} - p_t|^2$, where p_t is the output of the network after being fed x_t. We conducted 100 trials of hyper-parameter optimization as described above and obtained the following results in Table 2.

[1] See `https://github.com/junieroliva/recurrent` for code.

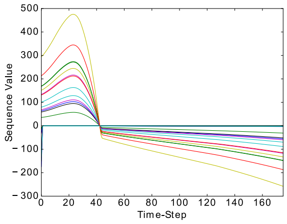

Figure 4. 25 sequences generated from the ground truth SRU model.

Table 2. MSEs for synthetically generated dataset.

	SRU	GRU	LSTM
Error	**0.62**	21.72	161.62

Not surprisingly, the SRU performs far better than traditional gated recurrent units. This suggests that the types of long-term statistical relationships captured by the SRU are indeed different than those of traditional recurrent units. As previously mentioned, the SRU is able to obtain a multitude of different views from its statistics, a task that traditional units achieve less efficiently since they must devote one whole memory cell per viewpoint and statistic. As we show below, the SRU is able to outperform traditional gated units in long term problems even for real data that is not generated from its model class.

3.2. MNIST Image Classification

Next we explore the ability of recurrent units to use long-term dependencies in ones data with a synthetic task using a real dataset. It has been observed that LSTMs perform poorly in classifying a long pixel-by-pixel sequence of MNIST digits (Le et al., 2015). In this synthetic task, each 28×28 gray-scale MNIST digit image is flattened and observed as a sequence $\{x_1, \ldots, x_{784}\}$, where $x_i \in [0, 1]$ (see Figure 5). The task is, based on the output observed after feeding x_{784} through the network, to classify the digit of the corresponding image in $\{0, \ldots, 9\}$. Hence, we project the output after x_{784} of each recurrent unit to 10 dimensions and use a softmax activation.

We report the hyper-parameter optimized results below in Table 3; due to resource constraints each trial consisted only of 10K training iterations. We see that the SRU is able to out-perform both GRUs and LSTMs. Given the long length and dependencies of pixel sequences in this experiment, it is not surprising that SRUs' abilities to capture long-term dependencies are aiding it to achieve a much lower error.

Table 3. Test error rate for MNIST pixel sequence classification.

	SRU	GRU	LSTM
Error Rate	**0.11**	0.28	0.48

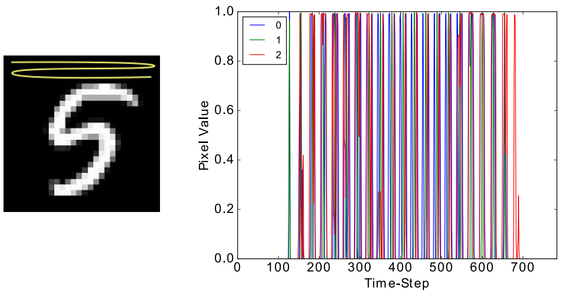

Figure 5. Right: example MNIST 28×28 image, which is taken as a pixel-by-pixel sequence of length 784 unrolled as shown in yellow. Left: example pixel sequences for 0, 1, and 2 digit images.

3.2.1. Dissective Study

Next, we study the behavior of the statistical recurrent unit with a dissective study where we vary several parameters of the architecture. We consider variants to a base model with: `num_stats=200`; base architecture `r_dims=60`; `num_units=200`. We keep the parameters `initial_learning_rate`, `lr_decay` fixed at the the optimal values found (0.1, 0.99 respectively) unless we find no learning, in which case we also try learning rates of 0.01 and 0.001.

The need for multi-scaled recurrent statistics. Recall that we designed the statistics used by the SRU expressly to capture long term time dependencies in sequences. We did so both with recurrent statistics, i.e. statistics that themselves depend on previous points' statistics, and with multi-scaled averages. We show below that both of these time-dependent design choices are vital to capturing long term dependencies in data. Furthermore, we show that the use of ReLU statistics lends itself to better learning.

We explored the impact that time-dependent statistics had on learning by first considering naive i.i.d. summary statistics for sequences. This was achieved by using `r_dims=0` and $\alpha \in A = \{0.99999\}$. Here no past-dependent context is used for statistics, i.e. we used i.i.d.-type statistics as is typical for unordered sets. Furthermore, the use of a single scale α near 1 means that all of the points' statistics will be weighted nearly identically (11) regardless of index. We optimized the SRU when using no recurrent statistics and a single scale (`iid`), when using recurrent statistics with a single scale (`recur`), and when using no recurrent statistics with multiple scales (`multi`). We report errors below in Table 4.

Table 4. Test error rate for MNIST pixel sequence classification.

	`iid`	`recur`	`multi`
Error Rate	0.88	0.88	0.63

Predictably, we cannot learn by simply keeping i.i.d. type statistics of pixel values at a single scale. Furthermore, we find that only using recurrent statistics (`recur`) in the SRU is not enough. It is interesting to note, however, that keeping i.i.d. statistics at multiple scales is able to predict digits with limited success. This lends evidence for the need of *both* recurrent statistics and multiple scales.

Next, we explored the effects of the scales at which we keep our statistics by varying from $\alpha \in A = \{0.0, 0.5, 0.9, 0.99, 0.999\}$ considering $\alpha \in A = \{0.0, 0.5, 0.9\}$, $\alpha \in A = \{0.0, 0.5, 0.9, 0.99\}$. We see in Table 5 that additional, longer scales aid our learning for this dataset. This is not very surprising given the long term nature of the pixel sequences.

Table 5. Test error rate for MNIST pixel sequence classification.

A	$\{0.0, 0.5, 0.9\}$	$\{0.0, 0.5, 0.9, 0.99\}$
Error Rate	0.79	0.21

Lastly, we considered the use of non-ReLU statistics by changing the element-wise non-linearity $f(\cdot)$ (5)-(8) to be the hyperbolic tangent $f(\cdot) = \tanh(\cdot)$. We postulated that the use of ReLUs would help our learning since they have been observed to better handle the problem of vanishing gradients. We find evidence of this when swapping ReLUs for hyperbolic tangent units in SRUs: we get an error rate of 0.18 when using hyperbolic tangent units. Although the previous uses of ReLUs in RNN required careful initialization (Le et al., 2015), SRUs are able to use ReLUs for better learning without any special considerations.

Dimension of recurrent summary. Next we explore the effect of varying the number of dimensions used for the recurrent summary of statistics r_t (5). We consider `r_dims` in $\{5, 20, 240\}$. As previously discussed r_t provides a context based on past data so that the SRU may produce non-i.i.d. statistics as it moves along a sequences. As one would expect the dimensionality of r_t will limit the information flow from the past and values that are too small will hinder performance. It is also interesting to see that after enough dimensions, there are diminishing returns to adding more.

Table 6. Test error rate varying recurrent summary r_t.

dim of r `r_dims`	5	20	240
Error Rate	0.25	0.20	0.10

Number of statistics and outputs. Finally, we vary the number of statistics `num_stats`, and outputs `units`. Interestingly the SRU seems robust to the number of outputs propagated in the network. However, performance is considerably affected by the number of statistics considered.

Table 7. Test error rate varying number of units.

dim of phi	`num_stats`		`units`		dim of o
	10	50	10	50	
Error Rate	0.88	0.32	0.15	0.15	

remember dim of phi = dim of mu

3.3. Polyphonic Music Modeling

Henceforth we consider real data and sequence learning tasks. First, we used the polyphonic music datasets from Boulanger-Lewandowski et al. (2012). Each time-step is a binary vector representing the notes played at the respective time-step. Since we were required to predict binary vectors we used the element-wise sigmoid σ. I.e., the binary vector of notes x_{t+1} was modeled as $\sigma(p_t)$, where p_t is the output after feeding x_t (and previous values $x_1, \ldots, x_{t-1}$) through the recurrent network.

It is interesting to note in Table 8 that the SRU is able to outperform one of the traditional gated units in every dataset and it outperforms both in two datasets.

Table 8. Test negative log-likelihood for polyphonic music data.

Data set	SRU	GRU	LSTM
JSB	**8.260**	8.548	8.393
Muse	6.336	6.429	**6.293**
Nottingham	3.362	3.386	**3.359**
Piano	**7.737**	7.929	7.931

3.4. Electronica-Genre Music MFCC

In the following experiment we modeled the Mel frequency cepstrum coefficients (MFCCs) in a dataset of nearly 18 000 scraped 30s sound clips of electronica-genre songs. MFCCs are perceptually based spectral features positioned logarithmically on the mel scale, which approximates the human auditory system's response (Müller, 2007). We looked to model the 13 real-valued coefficients using the recurrent units, by modeling x_{t+1} as a projection of the output of a recurrent unit after being fed $x_1, \ldots, x_t$.

Table 9. Test-set MSEs of MFCC Music data.

	SRU	GRU	LSTM
Error	**1.176**	2.080	1.183

As can be seen in Table 9, SRUs again are outperforming gated architectures and are especially beating GRUs by a wider margin.

3.5. Climate Data

Next we consider weather data prediction using the North America Regional Reanalysis (NARR) Project (NAR). The dataset provides a long-term set of consistent climate data on a regional scale for the North American domain. The period of the reanalyses is from October 1978 to the present and analyses were made 8 times daily (3 hour intervals).

We take our input sequences to be year-long sequences of weather variables in a location for the year 2006. I.e. an input sequence will be a 2920 length sequence of weather variables at a given lat/lon coordinate. We considered the following 7 variables: `pres10m`, 10 m pressure (pa); `tcdc`, total cloud cover (%); `rh2m`, relative humidity 2m (%); `tmpsfc`, surface temperature (k); `snod`, snow depth surface (m); `ugrd10m`, u component of wind 10m above ground; `vgrd10m`, v component of wind 10m above ground. The variables were standardized, see Figure 6 for example sequences.

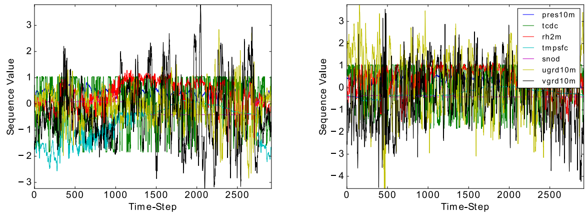

Figure 6. Two example sequences for weather variables at distinct locations for the year 2006.

Below we see results using 51 200 training location sequences and 6 400 validation and testing instances. Again, we look to model the next point in a sequence as a projection of the output of the recurrent unit after feeding the previous points. One may see in Table 10 that SRUs and LSTMs perform nearly identically; perhaps the cyclical nature of climate data was beneficial to the gated units.

Table 10. Test MSEs for weather data.

	SRU	GRU	LSTM
Error	**0.465**	0.487	0.466

3.6. SportVu NBA Tracking data

Finally, we look to predict the positions of National Basketball Association (NBA) players based on previous court positions during a play. Optical tracking data for this project were provided by STATS LLC from their SportVU product and obtained from (NBA). The data are composed of x and y coordinates for each of the ten players and the ball. We again minimize the squared norm of errors for predictions.

Table 11. Test-set MSEs of NBA data.

	SRU	GRU	LSTM
Error	**34.505**	329.921	296.908

We observed a large margin of improvement for SRUs over gated architectures in Table 11 that is reminiscent of the synthetic data experiment in §3.1. This suggests that this dataset contains long term dependencies that the SRU is able to exploit.

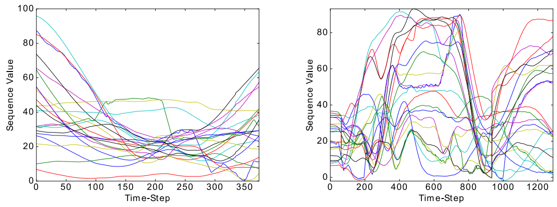

Figure 7. Example player/ball x, y positions for two plays.

4. Discussion

We believe that the use of summary statistics has been under-explored in modern recurrent units. Although recent studies in convolutional networks have considered global average pooling, which is essentially using high-level summary statistics to represent images, there has been little exploration of summary statistics for modern recurrent networks. To this end we introduce the Statistical Recurrent Unit, a novel architecture that seeks to capture long term dependencies in data using only simple moving averages and rectified-linear units.

The SRU was motivated by the success of mean-map embeddings for representing unordered datasets, and may be interpreted as an alteration of MMEs for sequential data. The main modifications are as follows: first, the SRU uses data-driven statistics unlike typical MMEs, which will use RKHS features from an a-priori selected class of kernels; second, SRUs will use recurrent statistics that are dependent not only on a current point, but on previous points' statistics through a condensation of kept moving averages; third, the SRU will keep moving averages at various scales. We provide evidence that the combination of these modifications yield much better results than any one of them in isolation.

The resulting recurrent unit is especially adept for capturing long term dependencies in data and readily has access to a combinatorial number of viewpoints of past windows through simple linear combinations. Moreover, it is interesting to note that even though the SRU is gate-less, it may implement part of both "remembering" and "forgetting" functionalities through ReLUs and moving averages.

We showed empirically that the SRU is better equipped that traditional gated units for long term dependencies via synthetic and real-world data experiments.

Acknowledgements

This research is supported in part by DOE grant DESC0011114 and NSF grant IIS1563887.

References

Ncep north american regional reanalysis. https://data.noaa.gov/dataset/ncep-north-american-regional-reanalysis-narr-for-1979-to-present. Accessed: 2016-10-17.

Nba movement data. https://github.com/sealneaward/nba-movement-data. Accessed: 2016-10-17.

Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

Bengio, Yoshua, Boulanger-Lewandowski, Nicolas, and Pascanu, Razvan. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 8624–8628. IEEE, 2013.

Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Bergstra, James, Komer, Brent, Eliasmith, Chris, Yamins, Dan, and Cox, David D. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

Boulanger-Lewandowski, Nicolas, Bengio, Yoshua, and Vincent, Pascal. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Donahue, Jeffrey, Anne Hendricks, Lisa, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Edwards, Harrison and Storkey, Amos. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Elman, Jeffrey L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pp. 1764–1772, 2014.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. 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, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hochreiter, Sepp, Bengio, Yoshua, Frasconi, Paolo, and Schmidhuber, Jürgen. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Iandola, Forrest N, Han, Song, Moskewicz, Matthew W, Ashraf, Khalid, Dally, William J, and Keutzer, Kurt. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Jaeger, Herbert, Lukoševičius, Mantas, Popovici, Dan, and Siewert, Udo. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

Jarrett, Kevin, Kavukcuoglu, Koray, LeCun, Yann, et al. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153. IEEE, 2009.

Koutnik, Jan, Greff, Klaus, Gomez, Faustino, and Schmidhuber, Juergen. A clockwork rnn. pp. 1863–1871, 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.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Mikolov, Tomas, Joulin, Armand, Chopra, Sumit, Mathieu, Michael, and Ranzato, Marc'Aurelio. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2015.

Muandet, Krikamol, Fukumizu, Kenji, Sriperumbudur, Bharath, and Schölkopf, Bernhard. Kernel mean embedding of distributions: A review and beyonds. *arXiv preprint arXiv:1605.09522*, 2016.

Müller, Meinard. *Information retrieval for music and motion*, volume 2. Springer, 2007.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

Ramsay, J.O. and Silverman, B.W. *Applied functional data analysis: methods and case studies*, volume 77. Springer New York:, 2002.

Smola, Alex, Gretton, Arthur, Song, Le, and Schölkopf, Bernhard. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pp. 13–31. Springer, 2007.

Vinyals, Oriol, Kaiser, Łukasz, Koo, Terry, Petrov, Slav, Sutskever, Ilya, and Hinton, Geoffrey. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.