Learning Long Term Dependencies via Fourier Recurrent Units*

Jiong Zhang [1] Yibo Lin [2] Zhao Song [3] Inderjit S. Dhillon [4]

Abstract

It is a known fact that training recurrent neural networks for tasks that have long term dependencies is challenging. One of the main reasons is the vanishing or exploding gradient problem, which prevents gradient information from propagating to early layers. In this paper we propose a simple recurrent architecture, the Fourier Recurrent Unit (FRU), that stabilizes the gradients that arise in its training while giving us stronger expressive power. Specifically, FRU summarizes the hidden states $h^{(t)}$ along the temporal dimension with Fourier basis functions. This allows gradients to easily reach any layer due to FRU's residual learning structure and the global support of trigonometric functions. We show that FRU has gradient lower and upper bounds independent of temporal dimension. We also show the strong expressivity of sparse Fourier basis, from which FRU obtains its strong expressive power. Our experimental study also demonstrates that with fewer parameters the proposed architecture outperforms other recurrent architectures on many tasks.

1. Introduction

Deep neural networks (DNNs) have shown remarkably better performance than classical models on a wide range of problems, including speech recognition, computer vision and natural language processing. Despite DNNs having tremendous expressive power to fit very complex functions, training them by back-propagation can be difficult. Two main issues are vanishing and exploding gradients. These issues become particularly troublesome for recurrent neural networks (RNNs) since the weight matrix is identical at each layer and any small changes get amplified exponentially through the recurrent layers (Bengio et al., 1994). Although exploding gradients can be somehow mitigated by tricks like gradient clipping or normalization (Pascanu et al., 2013), vanishing gradients are harder to deal with. If gradients vanish, there is little information propagated back through back-propagation. This means that deep RNNs have great difficulty learning long-term dependencies.

Many models have been proposed to address the vanishing/exploding gradient issue for DNNs. For example Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) tries to solve it by adding additional memory gates, while residual networks (He et al., 2016) add a short cut to skip intermediate layers. Recently the approach of directly obtaining the statistical summary of past layers has drawn attention, such as statistical recurrent units (SRU) (Oliva et al., 2017). However, as we show later, they still suffer from vanishing gradients and have limited access to past layers.

In this paper, we present a novel recurrent architecture, Fourier Recurrent Units (FRU) that use Fourier basis to summarize the hidden statistics over past time steps. We show that this solves the vanishing gradient problem and gives us access to any past time step region. In more detail, we make the following contributions:

- We propose a method to summarize hidden states through past time steps in a recurrent neural network with Fourier basis (FRU). Thus any statistical summary of past hidden states can be approximated by a linear combination of summarized Fourier statistics.
- Theoretically, we show the expressive power of sparse Fourier basis and prove that FRU can solve the vanishing gradient problem by looking at gradient norm bounds. Specifically, we show that in the linear setting, SRU only improves the gradient lower/upper bound of RNN by a constant factor of the exponent (i.e, both have the form (e^{aT}, e^{bT})), while FRU (lower and upper) bounds the gradient by constants independent of the temporal dimension.
- We tested FRU together with RNN, LSTM and SRU

[1] UT-Austin, zhangjiong724@utexas.edu
[2] UT-Austin, yibolin@utexas.edu
[3] Harvard & UT-Austin, zhaos@seas.harvard.edu
[4] UT-Austin & Amazon, inderjit@cs.utexas.edu

*Full version is available at https://arxiv.org/pdf/1803.06585. Correspondence to: Jiong Zhang <zhangjiong724@utexas.edu>.

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

on both synthetic and real world datasets like pixel-(permuted) MNIST, IMDB movie rating dataset. FRU shows its superiority on all of these tasks while enjoying smaller number of parameters than LSTM/SRU.

We now present the outline of this paper. In Section 2 we discuss related work, while in Section 3 we introduce the FRU architecture and explain the intuition regarding the statistical summary and residual learning. In Sections 4 and 5 we prove the expressive power of sparse Fourier basis and show that in the linear case FRUs have constant lower and upper bounds on gradient magnitude. Experimental results on benchmarking synthetic datasets as well as real datasets like pixel MNIST and language data are presented in Section 6. Finally, we present our conclusions and suggest several interesting directions in Section 7.

2. Related Work

Numerous studies have been conducted hoping to address the vanishing and exploding gradient problems, such as the use of self-loops and gating units in the LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014). These models use trained gate units on inputs or memory states to keep the memory for a longer period of time thus enabling them to capture longer term dependencies than RNNs. However, it has also been argued that by using a simple initialization trick, RNNs can have better performance than LSTM on some benchmarking tasks (Le et al., 2015). Apart from these advanced frameworks, straight forward methods like gradient clipping (Mikolov, 2012) and spectral regularization (Pascanu et al., 2013) are also proposed.

As brought to wide notice in Residual networks (He et al., 2016), give MLP and CNN shortcuts to skip intermediate layers allowing gradients to flow back and reach the first layer without being diminished. It is also claimed this helps to preserve features that are already good. Although ResNet is originally developed for MLP and CNN architectures, many extensions to RNN have shown improvement, such as maximum entropy RNN (ME-RNN) (Mikolov et al., 2011), highway LSTM (Zhang et al., 2016) and Residual LSTM (Kim et al., 2017).

Another recently proposed method, the statistical recurrent unit (SRU) (Oliva et al., 2017), keeps moving averages of summary statistics through past time steps. Rather than use gated units to decide what should be memorized, at each layer SRU memory cells incorporate new information at rate α and forget old information by rate $(1-\alpha)$. Thus by linearly combining multiple memory cells with different α's, SRU can have a multi-scale view of the past. However, the weight of moving averages is exponentially decaying through time and will surely go to zero given enough time steps. This prevents SRU from accessing the hidden states a few time steps ago, and allows gradients to vanish. Also, the expressive power of the basis of exponential functions is small which limits the expressivity of the whole network.

Fourier transform is a strong mathematical tool that has been successful in many applications. However the previous studies of Fourier expressive power have been concentrate in dense Fourier transform. Price and Song (Price & Song, 2015) proposed a way to define k-sparse Fourier transform problem in the continuous setting and also provided an algorithm which requires the frequency gap. Based on that (Chen et al., 2016) proposed a frequency gap free algorithm and well defined the expressive power of k-sparse Fourier transform. One of the key observations in the frequency gap free algorithm is that a low-degree polynomial has similar behavior as Fourier-sparse signal. To understand the expressive power of Fourier basis, we use the framework designed by (Price & Song, 2015) and use the techniques from (Price & Song, 2015; Chen et al., 2016).

There have been attempts to combine the Fourier transform with RNNs: the Fourier RNN (Koplon & Sontag, 1997) uses e^{ix} as activation function in RNN model; ForeNet (Zhang & Chan, 2000) notices the similarity between Fourier analysis of time series and RNN predictions and arrives at an RNN with diagonal transition matrix. For CNN, the FCNN (Pratt et al., 2017) replaces sliding window approach with the Fourier transform in the convolutional layer. Although some of these methods show improvement over current ones, they have not fully exploit the expressive power of Fourier transform or avoided the gradient vanishing/exploding issue. Motivated by the shortcomings of the above methods, we have developed a method that has a thorough view of the past hidden states, has strong expressive power and does not suffer from the gradient vanishing/exploding problem.

Notation. We use $[n]$ to denote $\{1, 2, \cdots, n\}$.

We provide several definitions related to matrix A. Let $\det(A)$ denote the determinant of a square matrix A, and $A^\top$ denote the transpose of A. Let $\|A\|$ denote the spectral norm of matrix A, and let A^t denote the square matrix A multiplied by itself $t-1$ times. Let $\sigma_i(A)$ denote the i-th largest singular value of A.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c and C.

The full version provides the detailed proofs and additional experimental results for comparison.

3. Fourier Recurrent Unit

In this section, we first introduce our notation in the RNN framework and then describe our method, the Fourier Recurrent Unit (FRU), in detail. Given a hidden state vector from the previous time step $h^{(t-1)} \in \mathbb{R}^{n_h}$, input $x^{(t-1)} \in \mathbb{R}^{n_i}$, RNN computes the next hidden state $h^{(t)}$ and output $y^{(t)} \in \mathbb{R}^{n_y}$ as:

$$\begin{aligned} h^{(t)} &= \phi(W \cdot h^{(t-1)} + U \cdot x^{(t-1)} + b) && \in \mathbb{R}^{n_h} \qquad (1) \\ y^{(t)} &= Y \cdot h^{(t)} && \in \mathbb{R}^{n_y} \end{aligned}$$

where ϕ is the activation, $W \in \mathbb{R}^{n_h \times n_h}$, $U \in \mathbb{R}^{n_h \times n_i}$ and $Y \in \mathbb{R}^{n_y \times n_h}$, $t = 1, 2, \ldots, T$ is the time step and $h^{(t)}$ is the hidden state at step t. In RNN, the output $y^{(t)}$ at each step is locally dependent to $h^{(t)}$ and only remotely linked with previous hidden states (through multiple weight matrices and activations). This give rise to the idea of directly summarizing hidden states through time.

Statistical Recurrent Unit. For each $t \in \{1, 2, \cdots, T\}$, (Oliva et al., 2017) propose SRU with the following update rules

$$\begin{aligned} g^{(t)} &= \phi(W_1 \cdot u^{(t-1)} + b_1) && \in \mathbb{R}^{n_g} \\ h^{(t)} &= \phi(W_2 \cdot g^{(t)} + U \cdot x^{(t-1)} + b_2) && \in \mathbb{R}^{n_h} \\ u_i^{(t)} &= D \cdot u_i^{(t-1)} + (I - D) \cdot (\mathbf{1} \otimes I) \cdot h^{(t)} && \qquad (2) \\ y^{(t)} &= Y \cdot u^{(t)} && \in \mathbb{R}^{n_y} \end{aligned}$$

where $\mathbf{1} \otimes I = [I_{n_h}, \ldots, I_{n_h}]^\top$. Given the decay factors $\alpha_k \in (0, 1), k = 1, 2 \cdots K$, the decaying matrix $D \in \mathbb{R}^{Kn_h \times Kn_h}$ is:

$$D = \operatorname{diag}\left(\alpha_1 I_{n_h}, \alpha_2 I_{n_h}, \cdots, \alpha_K I_{n_h}\right).$$

For each $i \in [Kn_h]$ and $t > 0$, $u_i^{(t)}$ can be expressed as the summary statistics across previous time steps with the corresponding α_k:

$$u_i^{(t)} = \alpha_k^t \cdot u_i^{(0)} + (1 - \alpha_k) \sum_{\tau=1}^{t} \alpha_k^{t-\tau} \cdot h^{(\tau)}. \qquad (3)$$

However, it is easy to note from (3) that the weight on $h^{(\tau)}$ vanishes exponentially with $t - \tau$, thus the SRU cannot access hidden states from a few time steps ago. As we show later in section 5, the statistical factor only improves the gradient lower bound by a constant factor on the exponent and still suffers from vanishing gradient. Also, the span of exponential functions has limited expressive power and thus linear combination of entries of $u^{(t)}$ also have limited expressive power.

Fourier Recurrent Unit. Recall that Fourier expansion indicates that a continuous function $F(t)$ defined on $[0, T]$ can be expressed as:

$$F(t) = A_0 + \frac{1}{T} \sum_{k=1}^{N} A_k \cos\left(\frac{2\pi kt}{T} + \theta_k\right)$$

where $\forall k \in [N]$:

$$\begin{aligned} &A_k = \sqrt{a_k^2 + b_k^2}, \theta_k = \arctan(b_k, a_k) \\ &a_k = 2\langle F(t), \cos\left(\frac{2\pi kt}{T}\right)\rangle, b_k = 2\langle F(t), \sin\left(\frac{2\pi kt}{T}\right)\rangle, \end{aligned}$$

where $\langle a, b \rangle = \int_0^T a(t)b(t)dt$. To utilize the strong expressive power of Fourier basis, we propose the Fourier recurrent unit model. Let $f_1, f_2, \cdots, f_K$ denote a set of K frequencies. For each $t \in \{1, 2, \cdots, T\}$, we have the following update rules

$$\begin{aligned} g^{(t)} &= \phi(W_1 \cdot u^{(t-1)} + b_1) && \in \mathbb{R}^{n_g} \\ h^{(t)} &= \phi(W_2 \cdot g^{(t)} + U \cdot x^{(t-1)} + b_2) && \in \mathbb{R}^{n_h} \\ u^{(t)} &= u^{(t-1)} + \frac{1}{T} C^{(t)} \cdot h^{(t)} && \in \mathbb{R}^{n_u} \qquad (4) \\ y^{(t)} &= Y \cdot u^{(t)} && \in \mathbb{R}^{n_y} \end{aligned}$$

where $C^{(t)} \in \mathbb{R}^{n_u \times n_h}$ is the Cosine matrix containing m square matrices:

$$C^{(t)} = \begin{bmatrix} C_1^{(t)} & C_2^{(t)} & \cdots & C_M^{(t)} \end{bmatrix}^\top,$$

and each $C_j^{(t)}$ is a diagonal matrix with cosine at $m = \frac{K}{M}$ distinct frequencies evaluated at time step t:

$$C_j^{(t)} = \operatorname{diag}\left(\cos\left(\frac{2\pi f_{k_1} t}{T} + \theta_{k_1}\right) I_d, \cdots, \cos\left(\frac{2\pi f_{k_2} t}{T} + \theta_{k_2}\right) I_d\right)$$

where $k_1 = m(j-1)+1$, $k_2 = mj$ and d is the dimension for each frequency. For every $t, j, k > 0$, $i = d(k-1) + j$ the entry $u_i^{(t)}$ has the expression:

$$u_i^{(t)} = u_i^{(0)} + \frac{1}{T} \sum_{\tau=1}^{t} \cos\left(\frac{2\pi f_k \tau}{T} + \theta_k\right) \cdot h_j^{(\tau)} \qquad (5)$$

As seen from (5), due to the global support of trigonometric functions, we can directly link $u^{(t)}$ with hidden states at any time step. Furthermore, because of the expressive power of the Fourier basis, given enough frequencies, $y^{(t)} = Y \cdot u^{(t)}$ can express any summary statistic of previous hidden states. As we will prove in later sections, these features prevent FRU from vanishing/exploding gradients and give it much stronger expressive power than RNN and SRU.

Connection with residual learning. Fourier recurrent update of $u^{(t)}$ can also be written as:

$$\begin{aligned} &u^{(t+1)} = u^{(t)} + \mathcal{F}(u^{(t)}) \\ &\mathcal{F}(u^{(t)}) = \frac{1}{T} C^{(t+1)} \phi(W_2 \phi(W_1 u^{(t)} + b_1) + U x^{(t)} + b_2) \end{aligned}$$

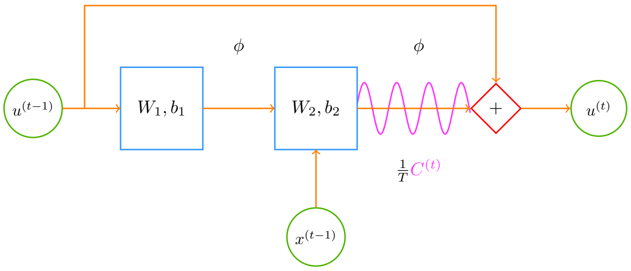

Figure 1: The Fourier Recurrent Unit

Thus the information flows from layer $(t-1)$ to layer t along two paths. The second term, $u^{(t-1)}$ needs to pass two layers of non-linearity, several weight matrices and scaled down by T, while the first term, $u^{(t-1)}$ directly goes to $u^{(t)}$ with only identity mapping. Thus FRU directly incorporates the idea of residual learning while limiting the magnitude of the residual term. This not only helps the information to flow more smoothly along the temporal dimension, but also acts as a regularization that makes the gradient of adjacent layers to be close to identity:

$$\frac{\partial u^{(t+1)}}{\partial u^{(t)}} = I + \frac{\partial \mathcal{F}}{\partial u^{(t)}}.$$

Intuitively this solves the gradient exploding/vanishing issue. Later in Section 5, we give a formal proof and comparison with SRU/RNN.

4. Fourier Basis

In this section we show that FRU has stronger expressive power than SRU by comparing the expressive power of limited number of Fourier basis (sparse Fourier basis) and exponential functions. On the one hand, we show that sparse Fourier basis is able to approximate polynomials well. On the other hand, we prove that even infinitely many exponential functions cannot fit a constant degree polynomial.

First, we state several basic facts which will be later used in the proof.

Lemma 4.1. *Given a square Vandermonde matrix V where $V_{i,j} = \alpha_i^{j-1}$, then $\det(V) = \prod_{1 \le i < j \le n}(\alpha_j - \alpha_i)$.*

Also recall the Taylor expansion of $\sin(x)$ and $\cos(x)$ is

$$\sin(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)!} x^{2i+1}, \cos(x) = \sum_{i=0}^{\infty} \frac{(-1)^i}{(2i)!} x^{2i}.$$

4.1. Using Fourier Basis to Interpolate Polynomials

(Chen et al., 2016) proved an interpolating result which uses Fourier basis ($e^{2\pi \mathbf{i} f t}$, $\mathbf{i} = \sqrt{-1}$) to fit a complex polynomial ($Q(t) : \mathbb{R} \to \mathbb{C}$). However in our application, the target polynomial is over the real domain, i.e. $Q(t) : \mathbb{R} \to \mathbb{R}$. Thus, we only use the real part of the Fourier basis. We extend the proof technique from previous work to our new setting, and obtain the following result,

Lemma 4.2. *For any $2d$-degree polynomial $Q(t) = \sum_{j=0}^{2d} c_j t^j \in \mathbb{R}$, any $T > 0$ and any $\epsilon > 0$, there always exists frequency $f > 0$ (which depends on d and ϵ) and $x^*(t) = \sum_{i=1}^{d+1} \alpha_i \cos(2\pi f i t) + \beta_i \cos(2\pi f i t + \theta_i)$ with coefficients $\{\alpha_i, \beta_i\}_{i=0}^{d}$ such that $\forall t \in [0, T]$, $|x^*(t) - Q(t)| \le \epsilon$.*

We provide the proof in the full version.

4.2. Exponential Functions Have Limited Expressive Power

Given k coefficients $c_1, \cdots, c_k \in \mathbb{R}$ and k decay parameters $\alpha_1, \cdots, \alpha_k \in (0,1)$, we define function $x(t) = \sum_{i=1}^{k} c_i \alpha_i^t$. We provide an explicit counterexample which is a degree-9 polynomial. Using that example, we are able to show the following result and defer the proof to full version.

Theorem 4.3. *There is a polynomial $P(t) : \mathbb{R} \to \mathbb{R}$ with $O(1)$ degree such that, for any $k \ge 1$, for any $x(t) = \sum_{i=1}^{k} c_i \alpha_i^t$, for any k coefficients $c_1, \cdots, c_k \in \mathbb{R}$ and k decay parameters $\alpha_1, \cdots, \alpha_k \in (0,1)$ such that*

$$\frac{1}{T}\int_0^T |P(t) - x(t)| \mathrm{d}t \gtrsim \frac{1}{T}\int_0^T |P(t)| \mathrm{d}t.$$

5. Vanishing and Exploding Gradients

In this section, we analyze the vanishing/exploding gradient issue in various recurrent architectures. Specifically we give lower and upper bounds of gradient magnitude under the linear setting and show that the gradient of FRU does not explode or vanish with temporal dimension $T \to \infty$. We first analyze RNN and SRU models as a baseline and show their gradients vanish/explode exponentially with T.

Gradient of linear RNN. For linear RNN, we have:

$$h^{(t+1)} = W \cdot h^{(t)} + U \cdot x^{(t)} + b$$

where $t = 0, 1, 2 \cdots T-1$. Thus

$$\begin{aligned} h^{(T)} &= W \cdot h^{(T-1)} + U \cdot x^{(T-1)} + b \\ &= W^{T-T_0} \cdot h^{(T_0)} + \sum_{t=T_0}^{T} W^{T-t-1}(U \cdot x^{(t)} + b) \end{aligned}$$

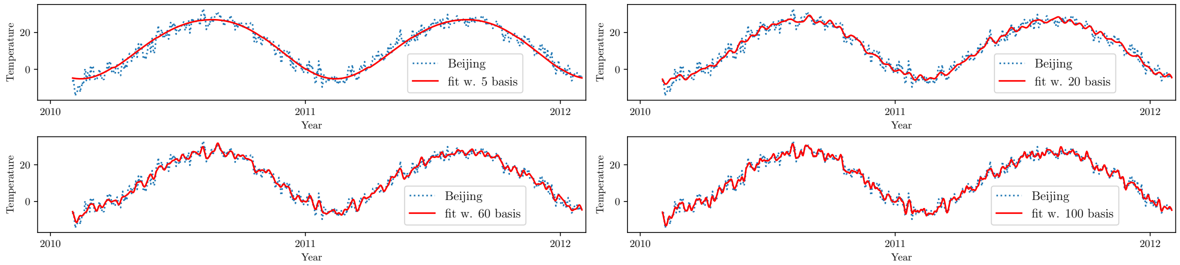

Figure 2: Temperature changes of Beijing from year 2010 to 2012, and the fit with Fourier basis: (a) 5 Fourier basis; (b) 20 Fourier basis; (c) 60 Fourier basis; (d) 100 Fourier basis.

Let $L = L(h^{(T)})$ denote the loss function. By Chain rule, we have

$$\begin{aligned}\left\|\frac{\partial L}{\partial h^{(T_0)}}\right\| &= \left\|\left(\frac{\partial h^{(T)}}{\partial h^{(T_0)}}\right)^\top \frac{\partial L}{\partial h^{(T)}}\right\| \\ &= \left\|(W^{T-T_0})^\top \cdot \frac{\partial L}{\partial h^{(T)}}\right\| \\ &\geq \sigma_{\min}(W^{T-T_0}) \cdot \left\|\frac{\partial L}{\partial h^{(T)}}\right\|.\end{aligned}$$

Similarly for the upper bound:

$$\left\|\frac{\partial L}{\partial h^{(T_0)}}\right\| \leq \sigma_{\max}(W^{T-T_0}) \cdot \left\|\frac{\partial L}{\partial h^{(T)}}\right\|.$$

Gradient of linear SRU. For linear SRU, we have:

$$\begin{aligned}&h^{(t)} = W_1 W_2 \cdot u^{(t-1)} + W_2 b_1 + W_3 \cdot x^{(t-1)} + b_2, \\ &u^{(t)} = \alpha \cdot u^{(t-1)} + (1-\alpha) h^{(t)}.\end{aligned}$$

Denoting $W = W_1 W_2$ and $B = W_2 b_1 + b_2$, we have

Claim 5.1. *Let $\overline{W} = \alpha I + (1-\alpha) W$. Then using SRU update rule, we have $u^{(T)} = \overline{W}^{T-T_0} u^{(T_0)} + \sum_{t=T_0}^{T} \overline{W}^{T-t-1}(1-\alpha) W_3 (x^{(t)} + B)$.*

We provide the proof in the full version.

With $L = L(u^{(T)})$, by Chain rule, we have the lower bound:

$$\begin{aligned}\left\|\frac{\partial L}{\partial u^{(T_0)}}\right\| &= \left\|((\alpha I + (1-\alpha)W)^\top)^{T-T_0} \frac{\partial L}{\partial u^{(T)}}\right\| \\ &\geq (\alpha + (1-\alpha)\sigma_{\min}(W))^{T-T_0} \cdot \left\|\frac{\partial L}{\partial u^{(T)}}\right\|.\end{aligned}$$

And similarly for the upper bound:

$$\left\|\frac{\partial L}{\partial u^{(T_0)}}\right\| \leq (\alpha + (1-\alpha)\sigma_{\max}(W))^{(T-T_0)} \cdot \left\|\frac{\partial L}{\partial u^{(T)}}\right\|.$$

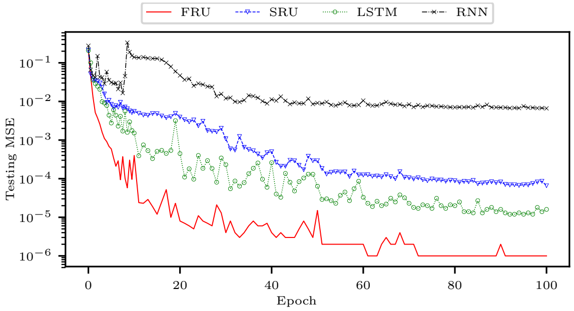

Figure 3: Test MSE of different models on mix-sin synthetic data. FRU uses $\text{FRU}_{120,5}$.

These bounds for RNN and SRU are achievable, a simple example would be $W = \sigma I$. It is easy to notice that with $\alpha \in (0,1)$, SRUs have better gradient bounds than RNNs. However, SRUs is only better by a constant factor on the exponent and gradients for both methods could still explode or vanish exponentially with temporal dimension T.

Gradient of linear FRU. By design, FRU avoids vanishing/exploding gradient by its residual learning structure. Specifically, the linear FRU has bounded gradient which is independent of the temporal dimension T. This means no matter how deep the network is, gradient of linear FRU would never vanish or explode. We have the following theorem:

Theorem 5.2. *With FRU update rule in (4), and ϕ being identity, we have: $e^{-2\sigma_{\max}(W_1 W_2)} \left\|\frac{\partial L}{\partial u^{(T)}}\right\| \leq \left\|\frac{\partial L}{\partial u^{(T_0)}}\right\| \leq e^{\sigma_{\max}(W_1 W_2)} \left\|\frac{\partial L}{\partial u^{(T)}}\right\|$ for any $T_0 \leq T$.*

We provide the proof in the full version.

6. Experimental Results

We implemented the Fourier recurrent unit in `Tensorflow` (Abadi et al., 2016) and used the standard implementation of `BasicRNNCell` and `BasicLSTMCell` for RNN and LSTM, respectively.

We also used the released source code of SRU (Oliva et al., 2017) and used the default configurations of $\{\alpha_i\}_{i=1}^5 = \{0.0, 0.25, 0.5, 0.9, 0.99\}$, g_t dimension of 60, and $h^{(t)}$ dimension of 200. We release our codes on github[1]. For fair comparison, we construct one layer of above cells with 200 units in the experiments. Adam (Kingma & Ba, 2014) is adopted as the optimization engine. We explore learning rates in {0.001, 0.005, 0.01, 0.05, 0.1} and learning rate decay in {0.8, 0.85, 0.9, 0.95, 0.99}. The best results are reported after grid search for best hyper parameters. For simplicity, we use FRU$_{k,d}$ to denote k sampled sparse frequencies and d dimensions for each frequency f_k in a FRU cell.

6.1. Synthetic Data

We design two synthetic datasets to test our model: mixture of sinusoidal functions (mix-sin) and mixture of polynomials (mix-poly). For mix-sin dataset, we first construct K components with each component being a combination of D sinusoidal functions at different frequencies and phases (sampled at beginning). Then, for each data point, we mix the K components with randomly sampled weights. Similarly, each data point in mix-poly dataset is a random mixture of K fixed D degree polynomials, with coefficients sampled at beginning and fixed. Alg. 1 and Alg. 2 (in the full version) explain these procedures in detail. Among the sequences, 80% are used for training and 20% are used for testing. We picked sequence length T to be 176, number of components K to be 5 and degree D to be 15 for mix-sin and $\{5, 10, 15\}$ for mix-poly. At each time step t, models are asked to predict the sequence value at time step $t+1$. It requires the model to learn the K underlying functions and uncover the mixture rates at beginning time steps. Thus we can measure the model's ability to express sinusoidal and polynomial functions as well as their long term memory.

Figure 3 and 4 plots the testing mean square error (MSE) of different models on mix-sin/mix-poly datasets. We use learning rate of 0.001 and learing rate decay of 0.9 for training. FRU achieves orders of magnitude smaller MSE than other models on mix-sin and mix-poly datasets, while using about half the number of parameters of SRU. This indicates FRU's ability to easily express these component functions.

To explicitly demonstrate the gradient stability and ability to learn long term dependencies of different models, we analyzed the partial gradient at different distance. Specifically, we plot the partial derivative norm of error on digit t w.r.t. the initial hidden state, i.e. $\frac{\partial(\widehat{y}^{(t)} - y^{(t)})^2}{\partial h^{(0)}}$ where $y^{(t)}$ is label and $\widehat{y}^{(t)}$ is model prediction. The norms of gradients for FRU are very stable from $t = 0$ to $t = 300$.

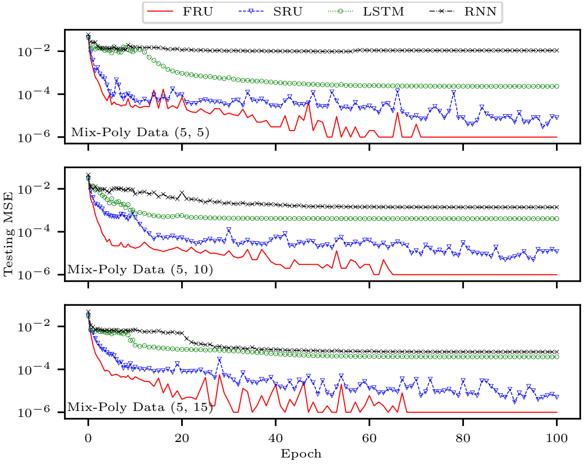

Figure 4: Test MSE of different models on mix-poly synthetic data with different maximum degrees of polynomial basis. FRU uses FRU$_{120,5}$.

Table 1: Testing Accuracy of MNIST Dataset

Networks	Testing Accuracy	#Variables	Variable Ratio
RNN	10.39%	42K	0.26
LSTM	98.17%	164K	1.00
SRU	96.20%	275K	1.68
FRU$_{40,10}$	96.88%	107K	0.65
FRU$_{60,10}$	97.61%	159K	0.97

With the convergence of training, the amplitudes of gradient curves gradually decrease. However, the gradients for SRU decrease in orders of magnitudes with the increase of time steps, indicating that SRU is not able to capture long term dependencies. The gradients for RNN/LSTM are even more unstable and the vanishing issues are rather severe.

6.2. Pixel-MNIST Dataset

We then explore the performance of Fourier recurrent units in classifying MNIST dataset. Each 28×28 image is flattened to a long sequence with a length of 784. The RNN models are asked to classify the data into 10 categories after being fed all pixels sequentially. Batch size is set to 256 and dropout (Srivastava et al., 2014) is not included in this experiment. A softmax function is applied to the 10 dimensional output at last layer of each model. For FRU, frequencies f are uniformly sampled in log space from 0 to 784.

Fig. 6 plots the testing accuracy of different models during training. RNN fails to converge and LSTM converges very slow. The fastest convergence comes from FRU, which achieves over 97.5% accuracy in 10 epochs while LSTM reaches 97% at around 40th epoch. Table 1 shows the accuracy at the end of 100 epochs for RNN, LSTM, SRU, and different configurations of FRU. LSTM ends up with

[1] https://github.com/limbo018/FRU

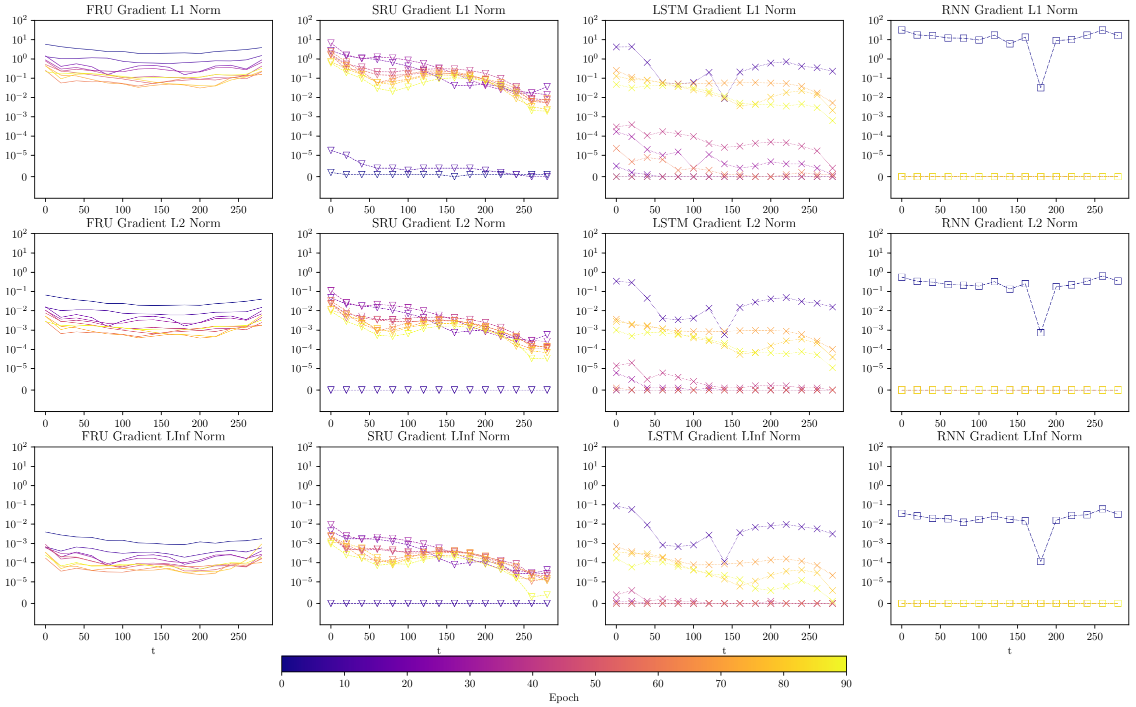

Figure 5: L1, L2, and L_∞ norms of gradients for different models on the training of mix-poly (5, 5) dataset. We evaluate the gradients of loss to the initial state with time steps, i.e., $\frac{\partial(\widehat{y}^{(t)}-y^{(t)})^2}{\partial h^{(0)}}$, where $(\widehat{y}^{(t)} - y^{(t)})^2$ is the loss at time step t. Each point in a curve is averaged over gradients at 20 consecutive time steps. We plot the curves at epoch $0, 10, 20, \ldots, 90$ with different colors from dark to light. FRU uses $\mathrm{FRU}_{120,5}$ and SRU uses $\{\alpha_i\}_{i=1}^5 = \{0.0, 0.25, 0.5, 0.9, 0.95\}$.

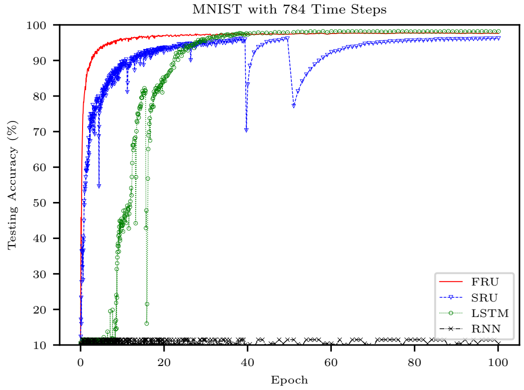

Figure 6: Testing accuracy of RNN, LSTM, SRU, and FRU for pixel-by-pixel MNIST dataset. FRU uses $\mathrm{FRU}_{60,10}$, i.e., 60 frequencies with the dimension of each frequency f_k to be 10.

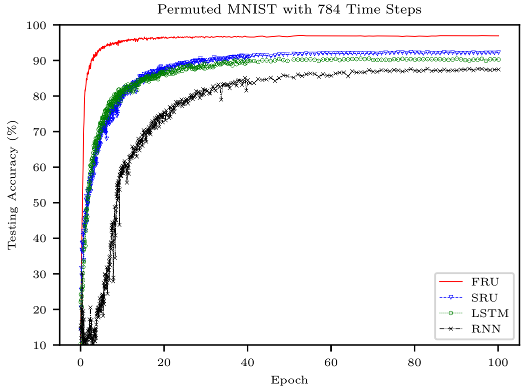

Figure 7: Testing accuracy of RNN, LSTM, SRU, and FRU for permuted pixel-by-pixel MNIST. FRU uses 60 frequencies with the dimension of 10 for each frequency.

98.17% in testing accuracy and SRU achieves 96.20%. Different configurations of FRU with 40 and 60 frequencies provide close accuracy to LSTM. The number and ratio of trainable parameters are also illustrated in the table. The amount of variables for FRU is much smaller than that of SRU, and comparable to that of LSTM, while it is able to achieve smoother training and high testing accuracy. We ascribe such benefits of FRU to better expressive power and more robust to gradient vanishing from the Fourier representations.

6.3. Permuted MNIST Dataset

We now use the same models as previous section and test on permuted MNIST dataset. Permute MNIST dataset is generated from pixel-MNIST dataset with a random but

Table 2: Testing Accuracy of Permuted MNIST Dataset

RNN	LSTM	SRU	FRU
87.46%	90.26%	92.21%	96.93%

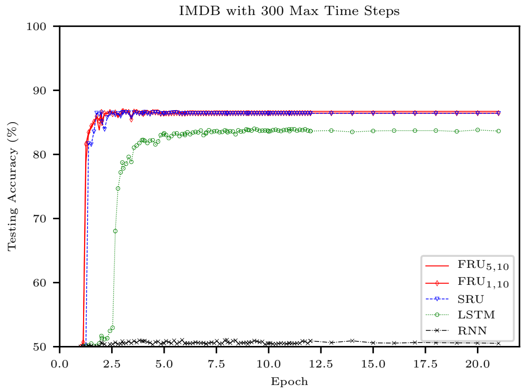

Figure 8: Testing accuracy of RNN, LSTM, SRU, and FRU for IMDB dataset. $\text{FRU}_{5,10}$ uses 5 frequencies with the dimension of 10 for each frequency f_k. $\text{FRU}_{1,10}$ is an extreme case of FRU with only frequency 0.

fixed permutation among its pixels. It is reported the permutation increases the difficulty of classification (Arjovsky et al., 2016). The training curve is plotted in Fig. 7 and the converged accuracy is shown in Table 2. We can see that in this task, FRU can achieve 4.72% higher accuracy than SRU, 6.67% higher accuracy than LSTM, and 9.47% higher accuracy than RNN. The training curve of FRU is smoother and converges much faster than other models. The benefits of FRU to SRU are more significant in permuted MNIST than that in the original pixel-by-pixel MNIST. This can be explained by higher model complexity of permuted-MNIST and stronger expressive power of FRU.

6.4. IMDB Dataset

We further evaluate FRU and other models with IMDB movie review dataset (25K training and 25K testing sequences). We integrate FRU and SRU into `TFLearn` (Damien et al., 2016), a high-level API for `Tensorflow`, and test together with LSTM and RNN. The average sequence length of the dataset is around 284 and the maximum sequence length goes up to over 2800. We truncate all sequences to a length of 300. All models use a single layer with 128 units, batch size of 32, dropout keep rate of 80%. FRU uses 5 frequencies with the dimension for each frequency f_k as 10. Learning rates and decays are tuned separately for each model for best performance.

Table 3: Testing Accuracy of IMDB Dataset

Networks	Testing Accuracy	#Variables	Variable Ratio
RNN	50.53%	33K	0.25
LSTM	83.64%	132K	1.00
SRU	86.40%	226K	1.72
$\text{FRU}_{5,10}$	86.71%	12K	0.09
$\text{FRU}_{1,10}$	86.44%	4K	0.03

Fig. 8 plots the testing accuracy of different models during training and Table 3 gives the eventual testing accuracy. $\text{FRU}_{5,10}$ can achieve 0.31% higher accuracy than SRU, and 3.07% better accuracy than LSTM. RNN fails to converge even after a large amount of training steps. We draw attention to the fact that with 5 frequencies, FRU achieves the highest accuracy with 10X fewer variables than LSTM and 19X fewer variables than SRU, indicating its exceptional expressive power. We further explore a special case of FRU, $\text{FRU}_{1,10}$, with only frequency 0, which is reduced to a RNN-like cell. It uses 8X fewer variables than RNN, but converges much faster and is able to achieve the second highest accuracy.

Besides the experimental results above, we provide more experiments on different configurations of FRU for MNIST dataset, detailed procedures to generate synthetic data in the full version.

7. Conclusion

In this paper, we have proposed a simple recurrent architecture called the Fourier recurrent unit (FRU), which has the structure of residual learning and exploits the expressive power of Fourier basis. We gave a proof of the expressivity of sparse Fourier basis and showed that FRU does not suffer from vanishing/exploding gradient in the linear case. Ideally, due to the global support of Fourier basis, FRU is able to capture dependencies of any length. We empirically showed FRU's ability to fit mixed sinusoidal and polynomial curves, and FRU outperforms LSTM and SRU on pixel MNIST dataset with fewer parameters. On language models datasets, FRU also shows its superiority over other RNN architectures. Although we now limit our models to recurrent structure, it would be very exciting to extend the Fourier idea to help gradient issues/expressive power for non-recurrent deep neural network, e.g. MLP/CNN. It would also be interesting to see how other basis functions, such as polynomial basis, will behave on similar architectures. For example, Chebyshev's polynomial is one of the interesting case to try.

Acknowledgments This research was supported by NSF grants CCF1320746, IIS-1546452 and CCF-1564000. The authors would like to thank Danqi Chen, Yu Feng, Rasmus Kyng, Eric Price, Tianxiao Shen, and Huacheng Yu for useful discussions.

References

Abadi, M., Agarwal, A., Barham, P., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016. URL http://arxiv.org/abs/1603.04467.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pp. 1120–1128, 2016.

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

Chen, X., Kane, D. M., Price, E., and Song, Z. Fourier-sparse interpolation without a frequency gap. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pp. 741–750. IEEE, https://arxiv.org/pdf/1609.01361.pdf, 2016.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Damien, A. et al. Tflearn. https://github.com/tflearn/tflearn, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Kim, J., El-Khamy, M., and Lee, J. Residual lstm: Design of a deep recurrent architecture for distant speech recognition. *arXiv preprint arXiv:1701.03360*, 2017.

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

Koplon, R. and Sontag, E. D. Using Fourier-neural recurrent networks to fit sequential input/output data. *Neurocomputing*, 15(3-4):225–248, 1997.

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

Mikolov, T. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 2012.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černockỳ, J. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pp. 196–201. IEEE, 2011.

Oliva, J. B., Póczos, B., and Schneider, J. The statistical recurrent unit. In *International Conference on Machine Learning*, pp. 2671–2680, 2017.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Pratt, H., Williams, B., Coenen, F., and Zheng, Y. FCNN: Fourier convolutional neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 786–798. Springer, 2017.

Price, E. and Song, Z. A robust sparse Fourier transform in the continuous setting. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pp. 583–600. IEEE, https://arxiv.org/pdf/1609.00896.pdf, 2015.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Zhang, Y., Chen, G., Yu, D., Yaco, K., Khudanpur, S., and Glass, J. Highway long short-term memory RNNs for distant speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5755–5759. IEEE, 2016.

Zhang, Y.-Q. and Chan, L.-W. Forenet: Fourier recurrent networks for time series prediction. In *Citeseer*, 2000.