F-001

LSTM 派生型リカレントニューラルネットワークにおける 隠れ層重み行列について

A study on hidden layer weight matrix in LSTM variant recurrent neural networks

櫻井 孝憲† 浅井紀久夫†
Takanori Sakurai Kikuo Asai

1. はじめに

Neural Network(NN)を複数層積み重ねた深層学習は急速に発展しつつあり、さまざまな分野への応用が行われている。

NN の一分野に、時系列などの順序性のあるデータを取り扱う RNN(Recurrent Neural Network)がある。RNN は、過去の隠れ層を入力に加えたネットワーク構成になっている。RNN の一種である LSTM(Long Short-Term Memory)は、RNN の長期記憶が難しい、という欠点を補うために、勾配消失対策の内部記憶（CEC：Constant Error Carousel）を設け、さらに CEC の周囲に複数のゲートを備えるなどしている。LSTM は手書き文字認識やテキスト発声システムなど様々な応用がなされている。

しかし LSTM は複雑な構造をしており、計算量、メモリ共に多くを要する。複数の時系列を同時に取り扱う、非常に長期の時系列入力を取り扱う、などの応用を考えた場合、これらが障壁となる。そのため LSTM の構造を簡略化する派生型が複数提案されている。簡略化の基本的方向性は CEC およびゲートの削減である。

本論文では、それら派生型のなかで最も簡略化されている Recurrent Highway Networks[5]を主な題材にして、その隠れ層重み行列についてさらなる簡略化を図る。まず、RNN および LSTM の概略を説明し、その派生型で簡略化がどのように行われてきたかを整理する。そのうえで、隠れ層における出力とリカレント項とがバランス回路で表現できることを示し、それぞれの項の役割を考える。そして、隠れ層重み行列を含む項を除いた構成 SRNN(Single weight matrix RNN)を考案し、性能評価実験によりその有効性を検証する。

2. 関連研究

$\mathbf{x}(t)$を時系列の入力ベクトル、$\mathbf{y}(t)$をモデルの出力、$\mathbf{h}(t)$を隠れ層としたときに RNN は次の式で表される。

$$\mathbf{h}(t) = f(W\mathbf{x}(t) + U\mathbf{h}(t-1) + \mathbf{b}) \quad (1)$$

$$\mathbf{y}(t) = g(V\mathbf{h}(t) + \mathbf{d}) \quad (2)$$

f は活性化関数、g は出力を整形する関数である。W,U,V はそれぞれ、入力ベクトルに対する重み行列、隠れ層に対する重み行列、隠れ層を出力の形式に整形する行列である。モデルの出力$\mathbf{y}(t)$を計算する式の中に 1 時刻前の隠れ層の$\mathbf{h}(t-1)$項が含まれることが RNN の特徴である。

†放送大学 OUJ

本稿では、式(1)の活性化関数f内部の式$W\mathbf{x}(t) + U\mathbf{h}(t-1) + \mathbf{b}$を$\alpha$式、1 時刻前の隠れ層の項$\mathbf{h}(t-1)$をリカレント項と呼ぶ。RNN は過去の隠れ層の出力を取り入れることにより時系列データを予測できたが、勾配消失が発生し長期記憶ができなかった。

勾配消失問題を解決するために LSTM[1]が考案された。LSTM では勾配を保存する$\mathbf{c}(t)$（CEC）を設けるとともに、$\mathbf{c}(t)$を保全（保持または忘却）するために CEC の周囲に入力$\mathbf{i}(t)$、出力$\mathbf{o}(t)$、忘却 $\mathbf{fog}(t)$の３つのゲートを設けた。

$$\mathbf{i}(t) = \sigma(W_i\mathbf{x}(t) + U_i\mathbf{h}(t-1) + \mathbf{b}_i) \quad (3)$$

$$\mathbf{o}(t) = \sigma(W_o\mathbf{x}(t) + U_o\mathbf{h}(t-1) + \mathbf{b}_o) \quad (4)$$

$$\mathbf{fog}(t) = \sigma(W_{fog}\mathbf{x}(t) + U_{fog}\mathbf{h}(t-1) + \mathbf{b}_{fog}) \quad (5)$$

$$\mathbf{a}(t) = f(W_a\mathbf{x}(t) + U_a\mathbf{h}(t-1) + \mathbf{b}_a) \quad (6)$$

$$\mathbf{c}(t) = \mathbf{i}(t) \circ \mathbf{a}(t) + \mathbf{fog}(t) \circ \mathbf{c}(t-1) \quad (7)$$

$$\mathbf{h}(t) = \mathbf{o}(t) \circ g(\mathbf{c}(t)) \quad (8)$$

LSTM は長期記憶を行うことに成功し、さまざまな応用がなされている[2]。

LSTM には、中心となる RNN の活性化関数f（通常は tanh）が 1 つ、σ（シグモイド）が 3 つある。つまり、LSTM は 8 つの行列と 4 つのバイアスベクトルを持ち、多くの計算とメモリを使用する。そのため、これらを簡略化する試みがなされている。

2.1 GRU（Gated Recurrent Unit）

GRU[3]は LSTM から勾配保存のための$\mathbf{c}(t)$と、ひとつのゲートを除いても長期記憶ができることを示した。

$$\mathbf{r}(t) = \sigma(W_r\mathbf{x}(t) + U_r\mathbf{h}(t-1) + \mathbf{b}_r) \quad (9)$$

$$\mathbf{z}(t) = \sigma(W_z\mathbf{x}(t) + U_z\mathbf{h}(t-1) + \mathbf{b}_z) \quad (10)$$

$$\tilde{\mathbf{h}}(t) = f(W_h\mathbf{x}(t) + U_h(\mathbf{r}(t) \circ \mathbf{h}(t-1)) + \mathbf{b}_h) \quad (11)$$

$$\mathbf{h}(t) = \mathbf{z}(t) \circ \mathbf{h}(t-1) + (1 - \mathbf{z}(t)) \circ \tilde{\mathbf{h}}(t) \quad (12)$$

行列を含む、計算量とパラメータ数の多い式(9),(10),(11)に注目したときに、GRU においては式(11)の計算に式(9)の結果を用いている。このような場合を段と呼ぶことにする。一方で LSTM の式(7),(8)は式(4),(5),(6)の結果を入力としているが、式(7),(8)自体は行列計算ではないので、段とは数えない。式（12）も同様に段とは数えない。GRU は 2 段であり、LSTM は 1 段である。式(9)と式(11)は段があるので、GPU による同時並行計算が難しい。GPU を用いた並行計算処理の場合、段数が増えるのは計算時間の点で不利であり、またバックプロパゲーションの際に勾配消失の機会が増えることにつながる。

2.2 MGU（Minimal Gated Unit）

MGU[4]は、さらに 1 つのゲートを除き、1 ゲートでも長期記憶ができることを示した。

$$\mathbf{f}(t)=\sigma(W_f\mathbf{x}(t)+U_f\mathbf{h}(t-1)+\mathbf{b}_f) \quad (13)$$
$$\tilde{\mathbf{h}}(t)=f(W_h\mathbf{x}(t)+U_h\mathbf{f}(t)\circ\mathbf{h}(t-1))+\mathbf{b}_h) \quad (14)$$
$$\mathbf{h}(t)=(1-\mathbf{f}(t))\circ\mathbf{h}(t-1)+\mathbf{f}(t)\circ\tilde{\mathbf{h}}(t) \quad (15)$$

MGU の場合も、段数は 2 段である。

2.3 RHN（ Recurrent Highway Networks）

RHN[5]は、1 ゲートで、かつ段数が 1 段であっても収束することを示した。

$$\mathbf{h}(t)=f(W_h\mathbf{x}(t)*+R_h\mathbf{s}(t-1))+\mathbf{b}_h) \quad (16)$$
$$\mathbf{t}(t)=\sigma(W_t\mathbf{x}(t)*+R_t\mathbf{s}(t-1)+\mathbf{b}_t) \quad (17)$$
$$\mathbf{s}(t)=\mathbf{h}(t)\circ\mathbf{t}(t)+\mathbf{s}(t-1)\circ(1-\mathbf{t}(t)) \quad (18)$$

＊のついた項は、RHN を多層化した場合には、第一層のみに現れる。また、RHN では MGU までと異なり隠れ層を表すベクトルは$\mathbf{s}(t)$であり、$\mathbf{h}(t)$ベクトルは第一層では RNN と同じ構成で、二層目以降は RNN から入力ベクトルの項、$W_h\mathbf{x}(t)$を除いたものである。また、原論文[5]によると、RHN は Highway Networks（HN）[6]をリカレント化したものである。本研究では、長期記憶を持ちながら簡略化を実現するため、RHN と HN の中間に位置する構成を考案している。

HN は以下の式で表され、リカレント項は持たない。

$$\mathbf{h}(t)=f(W_h\mathbf{x}(t)+\mathbf{b}_h) \quad (19)$$
$$\mathbf{t}(t)=\sigma(W_t\mathbf{x}(t)+\mathbf{b}_t) \quad (20)$$
$$\mathbf{y}=\mathbf{t}(t)\circ\mathbf{h}(t)+(\mathbf{1}-\mathbf{t}(t))\circ\mathbf{x}(t) \quad (21)$$

HN は 100 層以上の深い層での学習を目的としている。深層学習で成果を挙げた Residual Net[7]と同様に、式(21)に$\mathbf{x}(t)$項を持っており、さらにゲート$\mathbf{t}(t)$を設けて式(19)の NN の結果と$\mathbf{x}(t)$の間を調整している。

3. 隠れ層重み行列の検討

3.1 バランス回路

式(12)、(15)、(18)に共通して現れる式を本稿ではバランス回路と呼ぶ。バランス回路と呼ぶのは、オーディオ・セットの左右音量を調整するバランスノブに機能が似ているからである。次式は、**T** による **A** と **B** ベクトルのバランス回路を表す。

$$\mathbf{y}=\mathbf{T}\circ\mathbf{A}+(1-\mathbf{T})\circ\mathbf{B} \quad (22)$$

式(22)は式(23)のように変形することができる。

$$\mathbf{y}=\mathbf{B}+(\mathbf{A}-\mathbf{B})\circ\mathbf{T} \quad (23)$$

このように表記した場合には、**T** は **A**,**B** の間の適切な部分に **y** が来るように調節するという役割をもっているといえる（図１）。

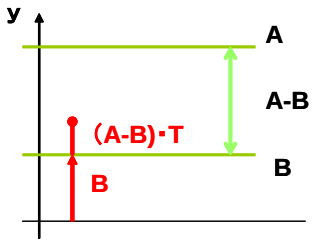

図1　バランス回路

RHN においては、**A**、**B**、**T** はそれぞれ$\mathbf{h}(t)$、$\mathbf{s}(t-1)$、$\mathbf{t}(t)$にあたる。学習が進むに従って重み行列内のパラメータが調整されていき、$\mathbf{h}(t)$と$\mathbf{s}(t-1)$の間は狭まっていく。

すなわちバランス回路を持つブロック GRU、MGU、RHN ではバランス回路が収束の役割を主に担っていると考えられる。

3.2 RHN 隠れ層重み行列

RHN の式(18)の$\mathbf{t}(t)$に注目すると、これは$\mathbf{t}(t)$による$\mathbf{s}(t-1)$と$\mathbf{h}(t)$のバランス回路だということができる。

$$\mathbf{s}(t)=\begin{cases}\mathbf{s}(t-1) & , \quad if \quad \mathbf{t}(t)=0\\ \mathbf{h}(t)=f(W_h\mathbf{x}(t)+R_h\mathbf{s}(t-1)+\mathbf{b}_h), & if \quad \mathbf{t}(t)=1\end{cases} \quad (24)$$

$\mathbf{s}(t)$は$\mathbf{t}(t)=0$のときには 1 つ前の時刻の隠れ層$\mathbf{s}(t-1)$と同一になり、 $\mathbf{t}(t)=1$のときは RNN となる。$\mathbf{t}(t)$は、$\mathbf{s}(t)$が$\mathbf{s}(t-1)$と$\mathbf{h}(t)$の間の適切な値を取るように調節している。

RHN をバランス回路としてみたときに、以下の 3 点について疑問がある。

- $\mathbf{s}(t-1)$が 3 か所に現れるのは多すぎないか
- $\mathbf{s}(t)$は$\mathbf{h}(t)$と$\mathbf{s}(t-1)$の間のバランス回路であるのに、$\mathbf{h}(t)$を計算する式の中に$\mathbf{s}(t-1)$項があるのは不自然ではないか
- f及びσの中はリカレント項を含むα式であるが、リカレント項がなくても十分ではないか

これらの疑問に対して、以下のように考察した。

式(16)は、LSTM 内部にある RNN 活性化関数(6)式由来であると考えることができる。LSTM が RNN の外側に CEC やゲートを設けて、RNN の弱点を補強しようとしたものと考えられるからである。さらに、式(6)は式(1)の RNN 由来である。

RNN が考案された時点では、過去の隠れ層をこのようにシンプルに繰り込んでいた。ところが、RHN ではリカレント項$\mathbf{s}(t-1)$が式(18)にもあり、この項はバランス回路の一端を担っている。従って、式(16)のR_hが掛かっている$\mathbf{s}(t-1)$は式(18)の$\mathbf{s}(t-1)$と、隠れ層出力を戻す役割として重複しており、式(16)のリカレント項$\mathbf{s}(t-1)$は省略することができるのではないかと考える。

さらに、式(18)は$\mathbf{h}(t)=f(\alpha$ 式)と$\mathbf{s}(t-1)$項のバランス回路である。この配分を決めるにあたり、$\mathbf{h}(t)$側のα式に$\mathbf{x}(t)$と$\mathbf{s}(t-1)$の両方が必要であるかも疑問である。活性化関数が tanh でもシグモイドでも単調増加であることを考慮したとき、活性化関数内部のα式に$\mathbf{s}(t-1)$項が入るのは$\mathbf{h}(t)$の嵩上げであり、バランス回路の収束を遅らせる要因になると考える。

つぎに、式(17)のシグモイド内部の α 式は入力ゲートの式(3)由来である。入力ゲートは、CEC をノイズから保護することを目的に設けられた。その後、GRU，MGU，RHN では、入力ゲートは忘却ゲートの式(5)と一本化されている。保護ないしは忘却する判断をするためには入力（$\mathbf{x}(t)$）が過去と大幅に異なっていると認識しなければならないので、過去の状態（$\mathbf{s}(t-1)$）との比較が必要となる。そこで、ゲートには両者を併せ持つ式が、シグモイドの内部として用いられている。

しかし、RHN（GRU 以降）では、ゲートが保全（保護ないしは忘却）すべき CEC は存在しない。そのため、式(17)は CEC の保全以外の目的に使用されているはずである。このときに、式(17)に$\mathbf{x}(t)$と$\mathbf{s}(t-1)$の双方が必要であるか疑問である。

そこで、これら隠れ層重み行列を含む項を除いたブロックを考案して、その有効性を確認することにした。この考案したブロックを SRNN(Single weight matrix RNN)と呼ぶことにする。

SRNN の Highway Networks（HN）、RHN に対する関係を図２に示す。

$$\text{HN}: \quad \mathbf{h}(t)=f(W_h\mathbf{x}(t)+\mathbf{b}_h),\ \mathbf{t}(t)=\sigma(W_t\mathbf{x}(t)+\mathbf{b}_t),\ \mathbf{y}=\mathbf{t}(t)\circ\mathbf{h}(t)+(\mathbf{1}-\mathbf{t}(t))\circ\mathbf{x}(t)$$

$$\text{SRNN}: \quad \mathbf{h}(t)=f(W_h\mathbf{x}(t)+\mathbf{b}_h),\ \mathbf{t}(t)=\sigma(W_t\mathbf{x}(t)+\mathbf{b}_t),\ \mathbf{s}(t)=\mathbf{t}(t)\circ\mathbf{h}(t)+(1-\mathbf{t}(t))\circ\mathbf{s}(t-1)$$

x(t)をs(t−1)に変更することにより、リカレント化

削除

$$\text{RHN}: \quad \mathbf{h}(t)=f(W_h\mathbf{x}(t)+R_h\mathbf{s}(t-1))+\mathbf{b}_t),\ \mathbf{t}(t)=\sigma(W_t\mathbf{x}(t)+R_t\mathbf{s}(t-1)+\mathbf{b}_t),\ \mathbf{s}(t)=\mathbf{t}(t)\circ\mathbf{h}(t)+(1-\mathbf{t}(t))\circ\mathbf{s}(t-1)$$

図2　SRNNとHN，RHNの関係

4. 実験

LSTM 派生型 RNN において、隠れ層重み行列を含む項を除いたブロックの有効性を確かめるために、RHN との比較実験を行う。性能評価に用いたタスク（あるいはデータセット）は MNIST、PTB で、前者は手書き文字認識、後者は英文における次の文字予測である。

4.1 MNIST による性能評価

MNIST 試験では、RHN と SRNN に加えて、両者の中間として、2 つの隠れ層重み行列の片方のみを除いたもの 2 種、合計 4 つの場合について実験を行った。

RHN を表す式の中の、R_hを含む項を ACTH（活性化関数項の隠れ層重み行列）、R_t を含む項を TRANH（トランスフォームゲートの隠れ層重み行列）と、それぞれ名づける。また、RHN 式から R_hを除いたものを NOACTH、R_t を除いたものを NOTRANH と呼ぶこととする。双方を除いたものは SRNN である。それぞれ、または双方を除いたものと RHN の学習の様子を比較する。

MNIST データセットの入力は 28×28 ピクセルの数字画像であり、出力（ラベル）は対応する数字である。実験にあたって正規化はおこなっていない。またドロップアウトも使用しなかった。

エポックは 200、RNN ブロックは 1 層、隠れ層は 100、活性化関数は tanh、学習率変化法は Adam、ラーニングレートは 0.0015 とした。

実験にあたっては、１エポック毎に、学習済みモデルで validation データを用いて損失関数値、確度値を計算し、学習の進展とともに精度が上がっていく様子を観察した。

損失関数値、確度値の推移を図３、４に示す。横軸はエポックであり、縦軸は損失関数値及び確度値である。

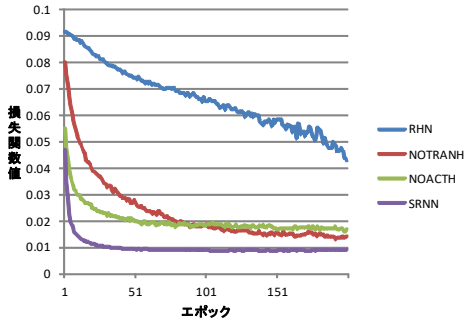

図３　隠れ層重み行列を除いたＭＮＩＳＴ試験（損失関数）

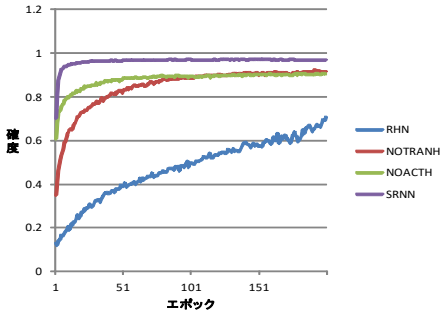

図４　隠れ層重み行列を除いたＭＮＩＳＴ試験（確度）

MNIST 試験では、SRNN は RHN より早いエポックで収束している。また片方の隠れ層重み行列のみを除いた NOTRANH、NOACTH も RHN より早く収束している。これらの結果を見る限り、ACTH、TRANH の 2 つの隠れ層重み行列を含む項は、双方共に学習の進展を阻害しているように見える。

4.2 PTB による性能評価

PTB 試験では、SRNN、RHN、LSTM の３種類の RNN ブロックについての比較を行った。

SRNN、RHN のハイパーパラメータは、それぞれ試行の結果もっとも良い収束を示したものを用いた。LSTM は Tensorflow チュートリアルの medium[8]を用いた。

SRNN、RHN では、エポックは 60、RNN ブロックは２層、隠れ層は SRNN が 450、RHN が 250 である。活性化関数は tanh、学習率変化法は当初定数を 0.2 とし、6 エポック以降 0.8 を乗じた（Tensorflow チュートリアルと同じ方法）。ラーニングレートは当初 0.2、バッチサイズは 20、ボキャブラリサイズは 10000 である。

PTB 試験での SRNN、RHN、LSTM のパープレキシティの推移を図５に示す。横軸はエポック、縦軸はパープレキシティである。

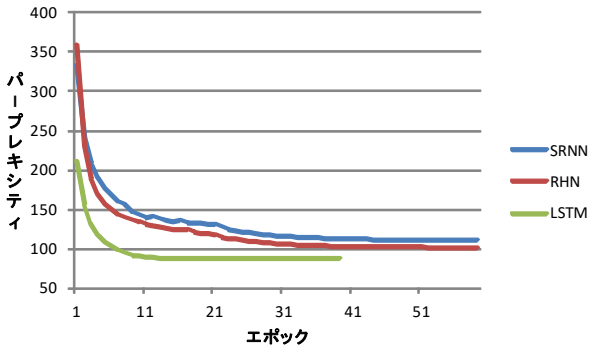

図５　隠れ層重み行列を除いたＰＴＢ試験

LSTM がもっとも早く、かつ高い収束を示している。SRNN と RHN では RHN の方が少し早く、収束結果は若干 RHN の方がよい。

4.3 GRU

GRU もバランス回路式(12)を持つ。そこで GRU でも RHN の場合と同様に隠れ層重み行列を含む項が不必要か、または収束を遅らせている可能性があると考えた。

GRU の式(9)の$U_{\mathrm{r}}\mathbf{h}(t-1)$項と式(10)の$U_{\mathrm{z}}\mathbf{h}(t-1)$項の双方を取り除いたものについても同様に PTB 試験を行った。

これら隠れ層重み行列を除いた GRU を SGRU(Single weight matrix GRU)と呼ぶこととする。

GRU,SGRU 共に、エポックは 60、RNN ブロックは２層、隠れ層は 450 で、それ以外のパラメータは SRNN の試験と同一とした。両者のパープレキシティの推移を図６に示す。

横軸はエポック、縦軸はパープレキシティである。

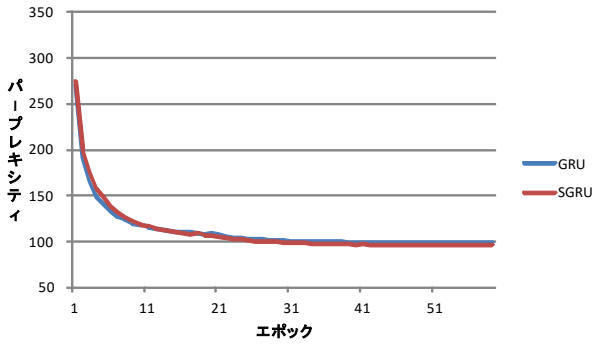

図６　ＧＲＵ、ＳＧＲＵによるＰＴＢ試験

PTB 試験では、２つの隠れ層重み行列項を除いても、SGRU は GRU と同等の収束を示している。

5. 考察

RHN において、２つの隠れ層重み行列項を除いた場合、MNIST 試験では収束が早まり、PTB 試験ではほぼ同等の収束がなされた。このことから、これら隠れ層重み行列は、収束の阻害をしているか、ないしは収束に寄与していないことが考えられる。

α式から隠れ層重み行列を含む項を除く試みは、RNN 派生型の不安定性を除くことを目的とした CFN[9]、活性化関数式を RNN ブロックと分離しようとした minimalRNN[10]などで行われているが、これらは収束の促進を目的としてはいないことが SRNN とは異なる。

また、必要メモリについて SRNN は RHN に対して有利である。MNIST 試験においては、入力ベクトル長が 28、隠れ層行列が 100 である。パラメータ数は RHN が 25,600 であるのに対して SRNN が 5,600 と４分の１のメモリサイズであり、メモリサイズ、CPU と GPU 間の転送時間に対して SRNN の方が有利である。

GRU による PTB 試験では、RHN 以外の、RNN 式とバランス回路を用いた他の派生型でも同じことが言える可能性を示している。

一方で、SRNN は LSTM、GRU ほど強く学習できない場合がある。PTB 試験において SRNN の収束時パープレキシティが１１０、RHN が１０１であるのに対して LSTM は８７、GRU,SGRU が９６である。このことは LSTM、GRU が持っていて SRNN、RHN にはない、学習のための機能があることを示唆している。

6. まとめ

本論文ではバランス回路を持つ LSTM 派生型 RNN について、その隠れ層重み行列項を除いたときの振る舞いを考察した。

最後に、SRNN が RHN より有利であるかは今後の検討課題である。RHN や GRU が解決できて、SRNN が解決できない種類の試験が出てくる可能性もある。その際には、リカレントニューラルネットワークを構成するブロック内の隠れ層重み行列の役割及び存在意義がより深く理解されるようになると考える。

謝辞

本研究の一部は JSPS 科研費 17K00493 の助成を受けました。

参考文献

[1] Sepp Hochreiter,Jürgen Schmidhuber, “Long Short-Term Memory”, Neural Computation, Vol.9, No.8 (1997).
[2] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber, LSTM: A Search Space Odyssey”, IEEE Transactions on Neural Networks and Learning Systems, Vol.28, No.10 (2017).
[3] Kyunghyun Cho et al, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”, IN Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734(2014).
[4] Guo-Bing Zhou Wu, Chen-Lin Zhang, Zhi-Hua Zhou Jianxin, “Minimal Gated Unit for Recurrent Neural Networks”, International Journal of Automation and Computing , Vol.13,Issue3 pages 226-234(2016).
[5] Zilly Georg Julian, Srivastava Kumar Rupesh, Koutník Jan, Schmidhuber ürgen, “Recurrent highway networks”, In Proc. 34th ICML Vol .70 Pages 4189-4198, (2016).
[6] Srivastava Kumar, Greff, Klaus, and Schmidhuber, Jürgen.Rupesh, “Highway networks”, In ICML Deep Learning Workshop, (2015).
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition”, The IEEE Conference on Computer Vision and Pattern Recognition pages 770-778(2016)
[8] https://www.tensorflow.org/tutorials/sequences/recurrent
[9] Laurent Thomas, von Brecht James, “A recurrent neural network without chaos”, In Proc. ICLR 2017,(2017)
[10] Chen Minmin, Pennington Jeffrey, Schoenhols Samuel S., “Dynamical Isometry and a Mean Field Theory of RNNs:Gating Enables Signal Propagation in Recurrent Neural Networks”, In Proc. 35th ICML vol. 80 pages 873–882,(2018)