**TheCEDL Homework 1**

李青峰、陳金博、羅羿牧

Selected Paper: Gated Feedback Recurrent Neural Networks

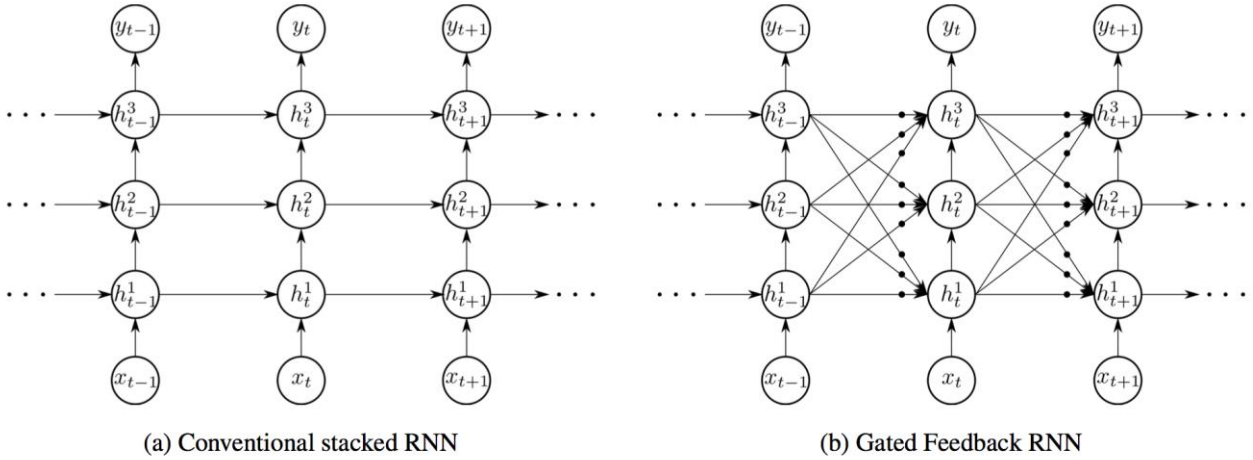
Paper Authors: Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio

---

# 1. 摘要介紹

這篇文章主打介紹的 RNN 架構是在 RNN 上的 state-to-state transition 改為 fully connected，使得整個深度學習網路架構具有從 coarse-grained timescale layer 往 fine-grained timescale layer feedback 的功能，並且在這些 connection 增加了 global reset gate，目的在於讓 RNN 具備可學習的、更彈性的時間性關聯表示。

本篇文章所謂的 Gate 指的是記憶單元在時間軸上的連接有著可以調控資訊流通的開關。在這篇文章的 Memory cell 選用 LSTM、GRU 兩種能夠處理長時記憶和短時記憶的的記憶單元，另外再用以下右圖的架構連接：



(a) Conventional stacked RNN　　(b) Gated Feedback RNN

*Figure 1.* Illustrations of (a) conventional stacking approach and (b) gated-feedback approach to form a deep RNN architecture. Bullets in (b) correspond to global reset gates. Skip connections are omitted to simplify the visualization of networks.

圖（一）

LSTM/GRU 被用在每個單元裡用來決定當前的 input $x_t$和前一個 state $h_{t-1}$是如何影響到當下的 output 及 state $h_t$。如圖一所示，每個 layer

的 state 透過 fully-connected 的 state-to-state transition 有機會能影響下一個 state 的任何一個 layer。這樣做的目的在於處理不同的 timescale（有些時候資訊在兩字之間，有時候資訊在相隔很多個字後出現）的語意，因為有可以 feedback 到前一層單元的結構，所以不同長度的 sequence dependencies 可以互相傳達，達到有效分析語意的目的。

其中每條 state-to-state 的 connection 透過 global reset gate 來調控，計算如下：

$$g^{i\to j} = \sigma\left(\mathrm{w}_g^{i\to j}\mathrm{h}_t^{j-1} + \mathrm{u}_g^{i\to j}\mathrm{h}_{t-1}^{*}\right)$$

其中 $\mathrm{w}_g^{i\to j}$ 是第 $i$ 層到第 $j$ 層間的 weight vector（在圖一中以垂直向上的箭頭代表），$\mathrm{u}_g^{i\to j}$ 是 previous state 到 current state 的 weight vector。

Stacked LSTM 第 j 個 state 記憶內容的計算如下：

$$\tilde{\mathrm{c}}_t^{j} = \tanh\left(W^{j-1\to j}h_t^{j-1} + \sum_{i=1}^{L} g^{i\to j}U^{i\to j}h_{t-1}^{i}\right)$$

之後透過 $h_t^j = o_t^j \tanh\left(\tilde{\mathrm{c}}_t^j\right)$ 即可更新當前的 state，其中 $o_t^j$ 為 LSTM output gate 的值。

Stacked GRU 的更新也遵循相似的原則，不再贅述。

因此上面的公式可以理解成從狀態 $\mathrm{h}_{t-1}^{*}$到 $\mathrm{h}_t^{*}$ 的資訊透過 $g^{i\to j}$ 這個「控制閥」過濾，而這個控制閥與當前於時間 $t$ 的 input $\mathrm{x}_t$ 和所有之前的 state $\mathrm{h}_{t-1}^{*}$相關。因此這個架構在直覺上能夠透過學習權衡當前 input $\mathrm{x}_t$對比之前的 state $\mathrm{h}_{t-1}^{*}$的重要性，並選擇性關閉 state-to-state 的資訊流通。

以上就是這篇提到的 RNN 架構，它提出的這個架構用在 Python script 的語意理解上。Input 是 python script 共有 41 個不同的符號，output 是執行 Python script 的答案有 13 種符號。最後，作者證實用 Gated

Feedback Recurrent Neural Networks （GF-RNN）比一般在時間軸上平行連接的方法還要好，如圖二：

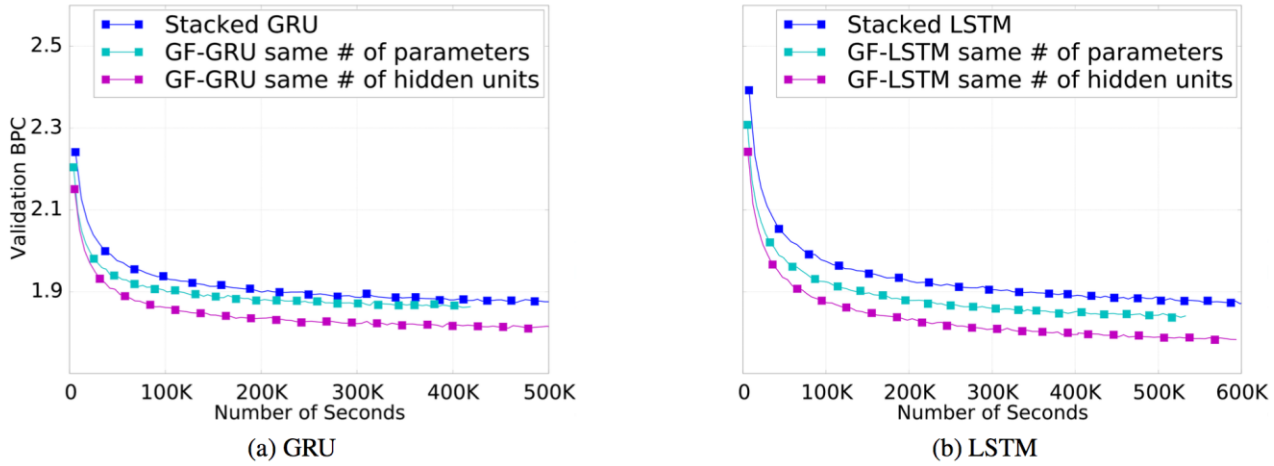



*Figure 2.* Validation learning curves of three different RNN architectures; Stacked RNN, GF-RNN with the same number of model parameters and GF-RNN with the same number of hidden units. The curves represent training up to 100 epochs. Best viewed in colors.

圖（二）

衡量方法用的是 BPC，一種值越低準確率越高的評量方法。

## 2. 特色

這種架構對顯著的特色在於能有效率地捕捉 long-term 和 short-term 的關聯性，應用和 LSTM 的開發目的是一樣的，只不過這是一種時間軸上的堆疊方式改良。

以往的 RNN 在時間軸上的連接方式只允許資訊從靠近 input layers 那端(對應短期關聯性)流向靠近 output layers 那端(對應長期關聯性)，GF-RNN 則透過 fully-connected 的 state-to-state transition 同時允許另一個方向（long-term to short-term）的資訊傳遞。

這種架構在不同的 layer 的 neurons 會捕捉不同 timescale 的概念，愈靠近 output layers 的 neurons 會傾向捕捉 long-term dependencies，並且強迫同一個 layer 形成一個 module，捕捉同一個 timescale 的關聯性。這點與 CW-RNN 相同，但是達成的方式截然不同。CW-RNN 硬性設定 neurons update 的週期與 2 的層數次方相關；然而在實務上的 input sequence，某些時間上的關聯不見得會永遠在規律的時間間隔發生，因此 GF-LSTM 並沒有特別設定每個 layer 運作的週期，而是透過 global reset gates 的控制來 capture 不同 timescale 的 dependencies。最重要的是，global reset gates 的開關是可以學習的（learnable），免除了捕捉 timescale 特徵的先天限制，我們認為這點是本文的架構能夠超越 GF-LSTM 架構表現的最主要原因。

在圖二上面的 x 軸代表訓練所需時間，實驗證實在此種架構下的 RNN 不論是在「相同參數量」或是「相同 hidden units 數量」之下，訓練時間都較傳統時間軸上的連接的方式短。

作者在實驗中強制設定 global reset gate 的值為 1，得到明顯較差的效果，證實了 gated feedback 這種架構在學習這種 sequence input 的應用上是有提升訓練效率的。作者還執行了第二個實驗：

*Table 3.* Generated texts with our trained models. Given the seed at the left-most column (bold-faced font), the models predict next 200 ∼ 300 characters. Tabs, spaces and new-line characters are also generated by the models.

| Seed | Stacked LSTM | GF-LSTM |
|---|---|---|
| **[[pl:Icon]]**<br>**[[pt:Icon]]**<br>**[[ru:Icon]]**<br>**[[sv:Programspraket Icon]]</text>**<br>**</revision>**<br>**</page>**<br>**<page>**<br>**<title>Iconology</title>**<br>**<id>14802</id>**<br>**<revi** | **<revi**sion><br><id>15908383</id><br><timestamp><br>2002-07-20T18:33:34Z<br></timestamp><br><contributor><br><username>The Courseichi</userrand<br>vehicles in [[enguit]].<br>==The inhibitors and alphabetsy and moral/<br>hande in===In four [[communications]] and | **<revi**sion><br><id>41968413</id><br><timestamp><br>2006-09-03T11:38:06Z<br></timestamp><br><contributor><br><username>Navisb</username><br><id>46264</id><br></contributor><br><comment>The increase from the time |
| **<title>Inherence relation</title>**<br>**<id>14807</id>**<br>**<revision>**<br>**<id>34980694</id>**<br>**<timestamp>**<br>**2006-01-13T04:19:25Z**<br>**</timestamp>**<br>**<contributor>**<br>**<username>Ro** | **<username>Ro**bert]]<br>[[su:20 aves]]<br>[[vi:10 Februari]]<br>[[bi:16 agostoferosín]]<br>[[pt:Darenetische]]<br>[[eo:Hebrew selsowen]]<br>[[hr:2 febber]]<br>[[io:21 februari]]<br>[[it:18 de februari]] | **<username>Ro**ma</username><br><id>48</id><br></contributor><br><comment>Vly''' and when one hand<br>is angels and [[ghost]] borted and<br>''mask r:centrions]], [[Afghanistan]],<br>[[Glencoddic tetrahedron]], [[Adjudan]],<br>[[Dghacn]], for example, in which materials<br>dangerous (carriers) can only use with one |

實驗內容是輸入初期的 html code，讓程式產生出之後的程式（條件是必須符合語法）。Input 可以從左邊的 seed 繼續延伸到之後的程式語法，並且證實使用 GF-LSTM 比較能夠辨認出 HTML tag 。由上面的實驗可以看出，用 GF-LSTM 的方法可以學習到 tag 的概念，展示了強大的 sequence 跨越長短 timescale 時間性特徵的捕捉。

這類的應用可以在 Machine translation 裡面看到，我們認為在未來或許可以應用在程式語言輸出的 Generative NN，甚至是股票市場分析等需要重視「長周期與短周期特徵關聯」的應用。

## 3. 貢獻

李青峰：

- 「摘要介紹」部分的 GF-RNN 架構數學上的解釋
- 「特色」部分分析 GF-RNN 與 CW-RNN 的不同

陳金博：

- 「特色」部分於 Python 指令碼上的比較
- html 語言的生成實驗

羅羿牧：

- 「摘要介紹」部分的架構概覽
- 「特色」部分與 LSTM/RNN 的比較