Bilingual effect
-using SRN network to simulate

Writer:

成奕 F94061165 醫工系 110

于治維 C54076055 醫工系 111

劉育存 D84061625 心理系 110

I. 研究主題 & 神經網路架構選取

其實我們整體的脈絡比較像是：選取神經網路模型→決定研究主題。因為我們一開始也是直接先從研究主題進行發想，但是每次到了選哪一個神經網路的架構時，就會發現目前好像沒有一個適合我們研究主題的模型已經被證實確實有可以拿來研究這方面的主題，於是就又要打掉重想。

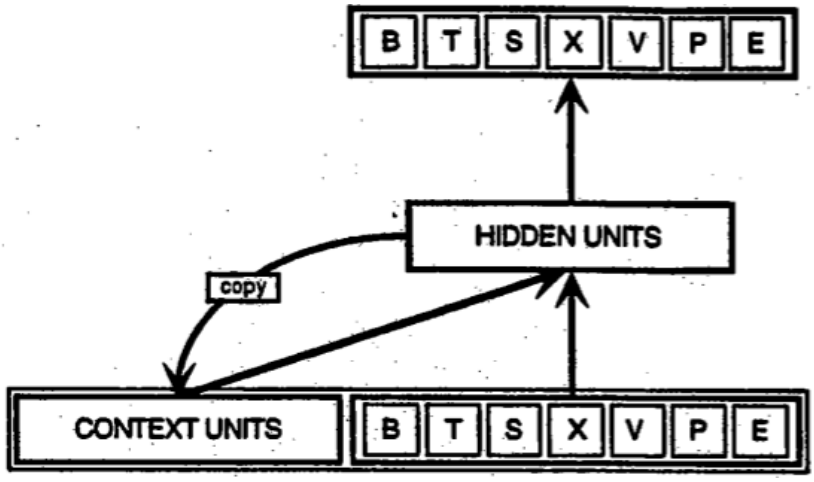
來來回回好幾次，我們終於發現問題在於新的模型架構要驗證對於在短時間內要能產生結果對於現在的我們來說有點困難。所以我們改變思維，決定先從模型選取後，再看看這個模型可以拿來做那些研究。

最後，我們決定要使用 simple recurrent network 來當作我們的模型，而目前利用 simple recurrent network 研究的方向比較有廣為人知的除了計算 1 的數量外，還有一個就是語言的學習。可以判斷一個單字出現再某單字後面的機率，藉此來學習字詞，也就是語言。

這裡說的學習語言主要就是前面說的-判斷某單字出現在某單字後面的機率。根據這個模型可以學習語言的特性，我們預計要做的主題是探討不同情況下的雙語學習者在學習第三語言所需的時間長短。第一種情況就是先學第一語言再學第二語言的雙語學習者，這也是大多雙語者的學習情況；第二種情況是同時學習第一和第二語言的雙語學習者，通常發生在具有雙語能力家庭中的小嬰兒身上。

之所以想要探討雙語學習者這個主題的原因在於，我們有找到文獻顯示雙語學習者在工作記憶的表現上會較為優異(Lukasik et al., 2018)，且對於第三種語言的學習表現也較為良好(Bartolotti & Marian, 2017)，所以我們才想以雙語者學習作為主題來進行討論。雖然一開始想說要觀察兩種不同的雙語者和單語者這三者間學習第三種語言的時間長短，但是後來發現，其實對於單語者來說，所謂“第三語言”就是他們的第二語言，所以可以直接觀察“先一後二”的模式即可。

1. 神經網路模型：

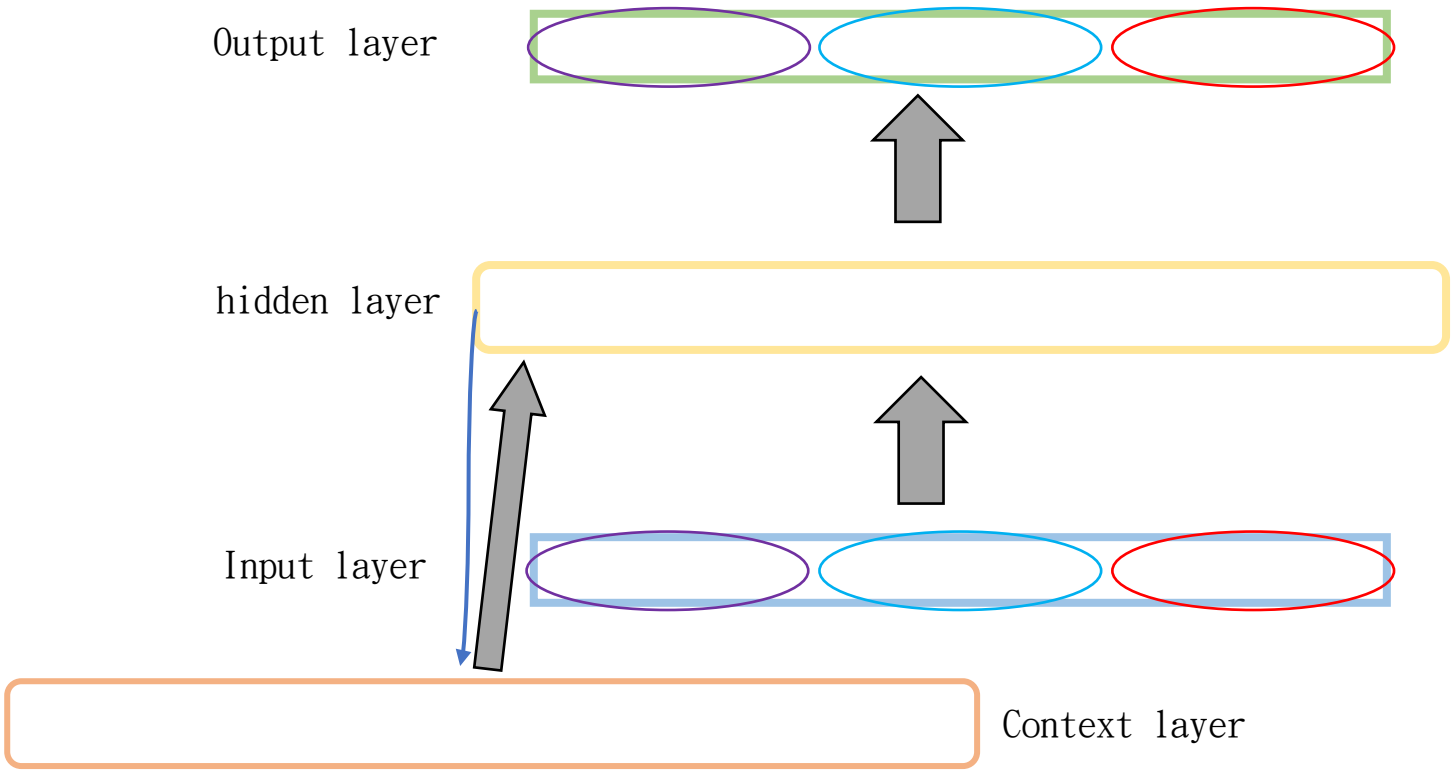


simple recurrent network 上圖為簡易版的 simple recurrent network 的示意圖。基本上 simple recurrent network 的概念就是，

context layer 會儲存上一次 hidden layer 的 activation 並在這一次傳回去給 hidden layer 裡面的節點。這樣一來，hidden layer 就可以知道上一次是甚麼樣的單字，並根據這次的單字來進行判斷後面的單字是哪一個。

2. 我們的 simple recurrent network 概念

基本上我們神經網路模型的架構跟上述的沒有太大的差別，唯一的區別在於我們增加了 input node 和 hidden node 的數量，因為我們要測驗三種語言，所以我們等倍數的增加了每一層 layer 的 node 數量。此外，我們也有特異將 input 區隔成三區，確認該模型能夠確實分別出這三種語言的差異，不是單純依據出現頻率這樣的概念而已。舉例來說：紫色區域的 input 就應該只會對應到紫色區域的 output，不應該跑到藍色或紅色的區域。



II.理想的資料

理想中，三種語言的資料應有以下幾點特徵：

- 需要是表音文字
- 彼此之間的語義應該接近
- 彼此之間的語言性質應該相去甚遠(不同語系分支)
- 文本資料需要足夠包含多種語境
- ~~資料不能太大~~(電腦資源不足)

故最後選擇新約聖經的「羅馬書」作為語言分析的基礎，
以及英文(NIV)、希臘文(NA26)與~~泰文(matlab 無法顯示)~~ 做為語言範本產生偽語言
並以範例附之練習字串作為測試語言的第三語言

語言分析

1. 列出所有字母(語素)及終止符

   以下這句話為例(Genesis 1:1 NIV)：

   ```
   In the beginning God created the heavens and the earth.
   ```

   所有的字母(含終止符)包括：

   ```
   [' ', 'G', 'I', 'a', 'b', 'c', 'd', 'e', 'g', 'h', 'i', 'n', 'o', 'r',
   's', 't', 'v']
   ```

2. 列出連續二字母(語素)及終止符之組合

```
[
' ', ' G', ' I', ' a', ' b', ' c', ' d', ' e', ' g', ' h', ' i', ' n', ' o', ' r', ' s', ' t', ' v',
'G ', 'GG', 'GI', 'Ga', 'Gb', 'Gc', 'Gd', 'Ge', 'Gg', 'Gh', 'Gi', 'Gn', 'Go', 'Gr', 'Gs', 'Gt', 'Gv',
'I ', 'IG', 'II', 'Ia', 'Ib', 'Ic', 'Id', 'Ie', 'Ig', 'Ih', 'Ii', 'In', 'Io', 'Ir', 'Is', 'It', 'Iv',
'a ', 'aG', 'aI', 'aa', 'ab', 'ac', 'ad', 'ae', 'ag', 'ah', 'ai', 'an', 'ao', 'ar', 'as', 'at', 'av',
'b ', 'bG', 'bI', 'ba', 'bb', 'bc', 'bd', 'be', 'bg', 'bh', 'bi', 'bn', 'bo', 'br', 'bs', 'bt', 'bv',
'c ', 'cG', 'cI', 'ca', 'cb', 'cc', 'cd', 'ce', 'cg', 'ch', 'ci', 'cn', 'co', 'cr', 'cs', 'ct', 'cv',
'd ', 'dG', 'dI', 'da', 'db', 'dc', 'dd', 'de', 'dg', 'dh', 'di', 'dn', 'do', 'dr', 'ds', 'dt', 'dv',
'e ', 'eG', 'eI', 'ea', 'eb', 'ec', 'ed', 'ee', 'eg', 'eh', 'ei', 'en', 'eo', 'er', 'es', 'et', 'ev',
'g ', 'gG', 'gI', 'ga', 'gb', 'gc', 'gd', 'ge', 'gg', 'gh', 'gi', 'gn', 'go', 'gr', 'gs', 'gt', 'gv',
'h ', 'hG', 'hI', 'ha', 'hb', 'hc', 'hd', 'he', 'hg', 'hh', 'hi', 'hn', 'ho', 'hr', 'hs', 'ht', 'hv',
'i ', 'iG', 'iI', 'ia', 'ib', 'ic', 'id', 'ie', 'ig', 'ih', 'ii', 'in', 'io', 'ir', 'is', 'it', 'iv',
'n ', 'nG', 'nI', 'na', 'nb', 'nc', 'nd', 'ne', 'ng', 'nh', 'ni', 'nn', 'no', 'nr', 'ns', 'nt', 'nv',
'o ', 'oG', 'oI', 'oa', 'ob', 'oc', 'od', 'oe', 'og', 'oh', 'oi', 'on', 'oo', 'or', 'os', 'ot', 'ov',
'r ', 'rG', 'rI', 'ra', 'rb', 'rc', 'rd', 're', 'rg', 'rh', 'ri', 'rn', 'ro', 'rr', 'rs', 'rt', 'rv',
's ', 'sG', 'sI', 'sa', 'sb', 'sc', 'sd', 'se', 'sg', 'sh', 'si', 'sn', 'so', 'sr', 'ss', 'st', 'sv',
't ', 'tG', 'tI', 'ta', 'tb', 'tc', 'td', 'te', 'tg', 'th', 'ti', 'tn', 'to', 'tr', 'ts', 'tt', 'tv',
'v ', 'vG', 'vI', 'va', 'vb', 'vc', 'vd', 've', 'vg', 'vh', 'vi', 'vn', 'vo', 'vr', 'vs', 'vt', 'vv'
]
```

3. 分析字母及終止符出現次數

```
' ': 10, 'G': 1, 'I': 1, 'a': 3, 'b': 1, 'c': 1, 'd': 3, 'e': 9, 'g':
2, 'h': 5, 'i': 2, 'n': 6, 'o': 1, 'r': 1, 's': 1, 't': 5, 'v': 1,
```

4. 分析連續二字母出現次數

```
:  0, G :  0, I :  0, a :  0, b :  0, c :  0, d :  0, e :  0, g :  0, h :  0, i :  0, n :  0, o :  0, r :  0, s :  0, t :  0, v :  0,
G  :  0, GG :  0, GI :  0, Ga :  0, Gb :  0, Gc :  0, Gd :  0, Ge :  0, Gg :  0, Gh :  0, Gi :  0, Gn :  0, Go :  1, Gr :  0, Gs :  0, Gt :  0, Gv :  0,
I  :  0, IG :  0, II :  0, Ia :  0, Ib :  0, Ic :  0, Id :  0, Ie :  0, Ig :  0, Ih :  0, Ii :  0, In :  1, Io :  0, Ir :  0, Is :  0, It :  0, Iv :  0,
a  :  0, aG :  0, aI :  0, aa :  0, ab :  0, ac :  0, ad :  0, ae :  0, ag :  0, ah :  0, ai :  0, an :  1, ao :  0, ar :  0, as :  0, at :  1, av :  1,
b  :  0, bG :  0, bI :  0, ba :  0, bb :  0, bc :  0, bd :  0, be :  1, bg :  0, bh :  0, bi :  0, bn :  0, bo :  0, br :  0, bs :  0, bt :  0, bv :  0,
c  :  0, cG :  0, cI :  0, ca :  0, cb :  0, cc :  0, cd :  0, ce :  0, cg :  0, ch :  0, ci :  0, cn :  0, co :  0, cr :  1, cs :  0, ct :  0, cv :  0,
d  :  3, dG :  0, dI :  0, da :  0, db :  0, dc :  0, dd :  0, de :  0, dg :  0, dh :  0, di :  0, dn :  0, do :  0, dr :  0, ds :  0, dt :  0, dv :  0,
e  :  3, eG :  0, eI :  0, ea :  2, eb :  0, ec :  0, ed :  1, ee :  0, eg :  1, eh :  0, ei :  0, en :  1, eo :  0, er :  0, es :  0, et :  0, ev :  0,
g  :  1, gG :  0, gI :  0, ga :  0, gb :  0, gc :  0, gd :  0, ge :  0, gg :  0, gh :  0, gi :  1, gn :  0, go :  0, gr :  0, gs :  0, gt :  0, gv :  0,
h  :  0, hG :  0, hI :  0, ha :  0, hb :  0, hc :  0, hd :  0, he :  4, hg :  0, hh :  0, hi :  0, hn :  0, ho :  0, hr :  0, hs :  0, ht :  0, hv :  0,
i  :  0, iG :  0, iI :  0, ia :  0, ib :  0, ic :  0, id :  0, ie :  0, ig :  0, ih :  0, ii :  0, in :  2, io :  0, ir :  0, is :  0, it :  0, iv :  0,
n  :  1, nG :  0, nI :  0, na :  0, nb :  0, nc :  0, nd :  1, ne :  0, ng :  1, nh :  0, ni :  1, nn :  1, no :  0, nr :  0, ns :  1, nt :  0, nv :  0,
o  :  0, oG :  0, oI :  0, oa :  0, ob :  0, oc :  0, od :  1, oe :  0, og :  0, oh :  0, oi :  0, on :  0, oo :  0, or :  0, os :  0, ot :  0, ov :  0,
r  :  0, rG :  0, rI :  0, ra :  0, rb :  0, rc :  0, rd :  0, re :  1, rg :  0, rh :  0, ri :  0, rn :  0, ro :  0, rr :  0, rs :  0, rt :  0, rv :  0,
s  :  1, sG :  0, sI :  0, sa :  0, sb :  0, sc :  0, sd :  0, se :  0, sg :  0, sh :  0, si :  0, sn :  0, so :  0, sr :  0, ss :  0, st :  0, sv :  0,
t  :  0, tG :  0, tI :  0, ta :  0, tb :  0, tc :  0, td :  0, te :  1, tg :  0, th :  3, ti :  0, tn :  0, to :  0, tr :  0, ts :  0, tt :  0, tv :  0,
v  :  0, vG :  0, vI :  0, va :  0, vb :  0, vc :  0, vd :  0, ve :  1, vg :  0, vh :  0, vi :  0, vn :  0, vo :  0, vr :  0, vs :  0, vt :  0, vv :  0,
```

p. s. 語素≠字母，但此處為方便分析，故使用字母作為最小單位

產生資料

1. 為簡約資料，只取次數最高的前 7 個字母作為基礎
   英語：

   e, o, t, h, a, i, s

   希臘文：

   ν, τ, α, o, ε, ι, σ

泰文：

```
า, ร, ่, น, ้, เ, อ
```

2. 任意抽取其中一字母作為開頭
3. 根據分析出的機率產生下一個字母

	e	o	t	h	a	i	s
e	0.147887	0.066398	0.103622	0.005030	0.250503	0.074447	0.352113
o	0.050781	0.150391	0.537109	0.001953	0.003906	0.054688	0.201172
t	0.052220	0.180099	0.008224	0.638980	0.027961	0.065789	0.026727
h	0.451351	0.152510	0.044788	0.000000	0.207722	0.143243	0.000386
a	0.019976	0.001175	0.427732	0.021152	0.003525	0.229142	0.297297
i	0.024816	0.101103	0.363051	0.000000	0.010110	0.000000	0.500919
s	0.320489	0.111842	0.149436	0.065789	0.138158	0.136278	0.078008

row 的元素總和為 1，代表前一個字母；column 代表將要產生的字母；對應的元素是其機率

範例

舉例來說：

1. 以 e 作為開頭：
   e
2. 找到 e-row，對應到 s-column 的機率為 0.352113，是機率最高的。
   假設抽到 s 做為下一個字母：
   es

3. 找到 s-row，對應到 t-column 的機率為 0.149436。假設抽到 t 做為下一個字母：
   est
4. 找到 t-row，對應到 a-column 的機率為 0.027961。假設抽到 a 做為下一個字母：
   esta
5. 找到 a-row，對應到 t-column 的機率為 0.427732。假設抽到 t 做為下一個字母：
   estat
6. 找到 t-row，對應到 e-column 的機率為 0.052220。假設抽到 e 做為下一個字母：
   estate

如此一來就產生出了一個單字，依此類推繼續產生出其他單字

資料集範例

英文：

```
e, o, t, h, a, i, s
aith
aitho
eesoos
ososttheo
heot
othes
esisothai
issothisise
the
eseo
theeseasti
tothiseit
eaisaeathea
ithe
sthease
ithaso
stastheai
shishe
```

asoshessese
ssesesi

希臘文：

ν, τ, α, ο, ε, ι, σ
ε τ ο
ι ν τ ο ν
α ι ν ε ν
ν ε ι ν α ι ν τ ε
ε ν ε ι
ο ν ε ν ο ν α ν
α τ ο ν
ο ν ε ι σ τ
ε ι ο ι τ
ν ο ν ε ο ν ε
α τ α ν ο ι
ι σ α τ
τ ο ι ν ο ι σ ι
ο ι α τ ο ν τ α ι ν
ο ν ε ν τ ο
σ α τ α ι σ α ν
ε ι ο ι ν α ι ο ν
τ ι ο ν ο ι ο ν α ι
τ ο σ ε τ ε ι ν τ ο ν
ο ι σ τ ι σ α τ ε ν τ

泰文：

า, ร, ่, น, ้, เ, อ
เรอารเนนเนเ
เรานำเร
านอเอนเรา
อเอเนเรารัน
รเอาร
านานำอา
นนันเนอ่ร
เนเอาราอเ
รอนน้ำเอนน
านนอเนนเนร
อนเอ

อนนเราน

้านอนร

้าอน้

นาเร้เ

่านอเ

เอานอ

นเอเราเนอ้

ารานราเร้

課本範例：

b, t, s, x, v, p, e

b t s x s e

b t x s e

b t s s s x s e

b t s s x x v v e

b t s s s x x v v e

b t x x v p x v v e

b t x x t t v v e

b t s x x t v v e

b t s s x x v p s e

b t s x x t v p s e

b t x x t v p s e

b t x x v p s e

b p v v e

b p t t v v e

b p t v p x v v e

b p v p x v v e

b p t v p x t v v e

b p v p x v p s e

b p t v p s e

b p v p x t v p s e

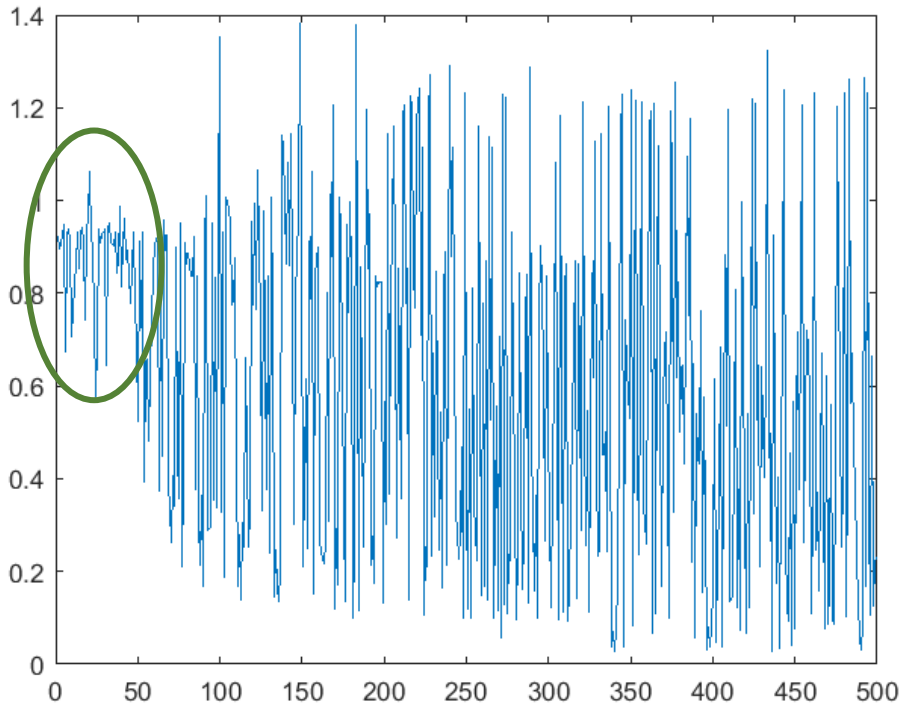
b p t t v p s e

III. 模型結果解釋：

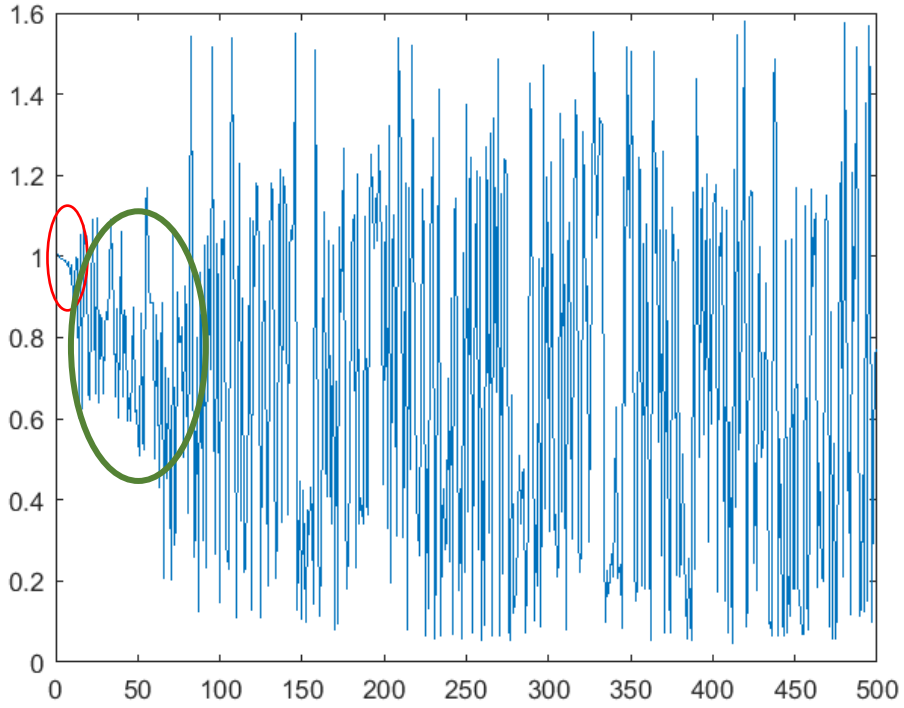
1. 模型外架構

我們稍微修改了 pdp tool 裡面的 simple recurrent network 架構，把 input、hidden layer、context layer 和 output 的 node 數量增加。並且稍微區隔成三種語言(分別用橘、藍、綠色框起)。

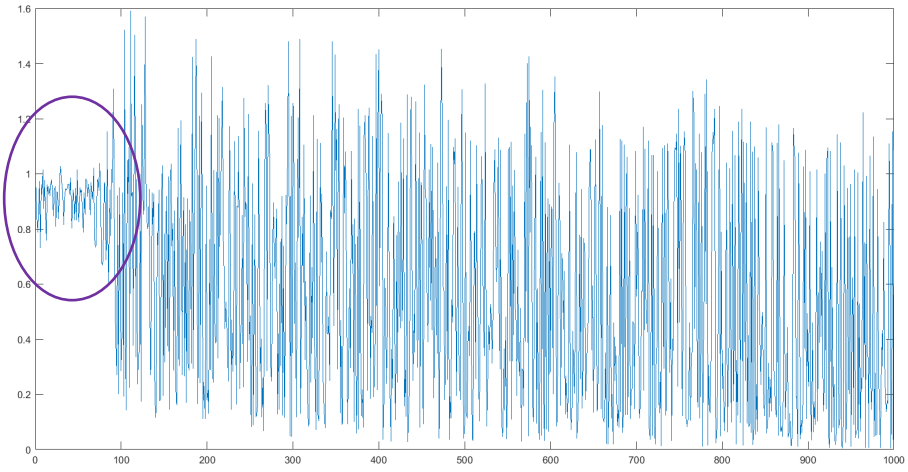
2. 單獨訓練第一語言：



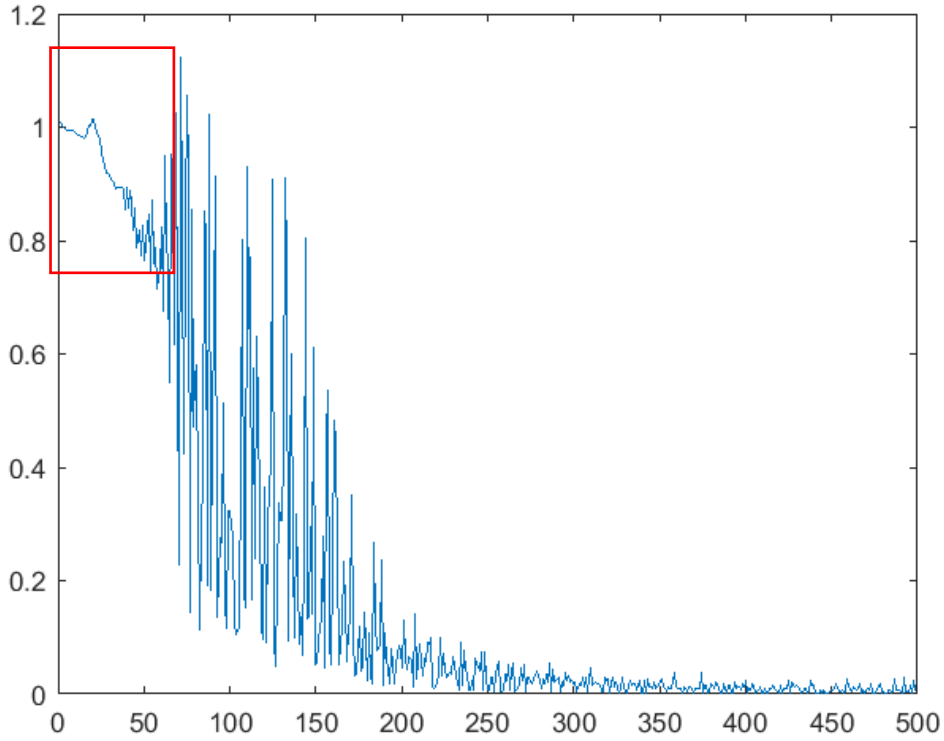
3. 第一語言學成後，訓練第二語言：



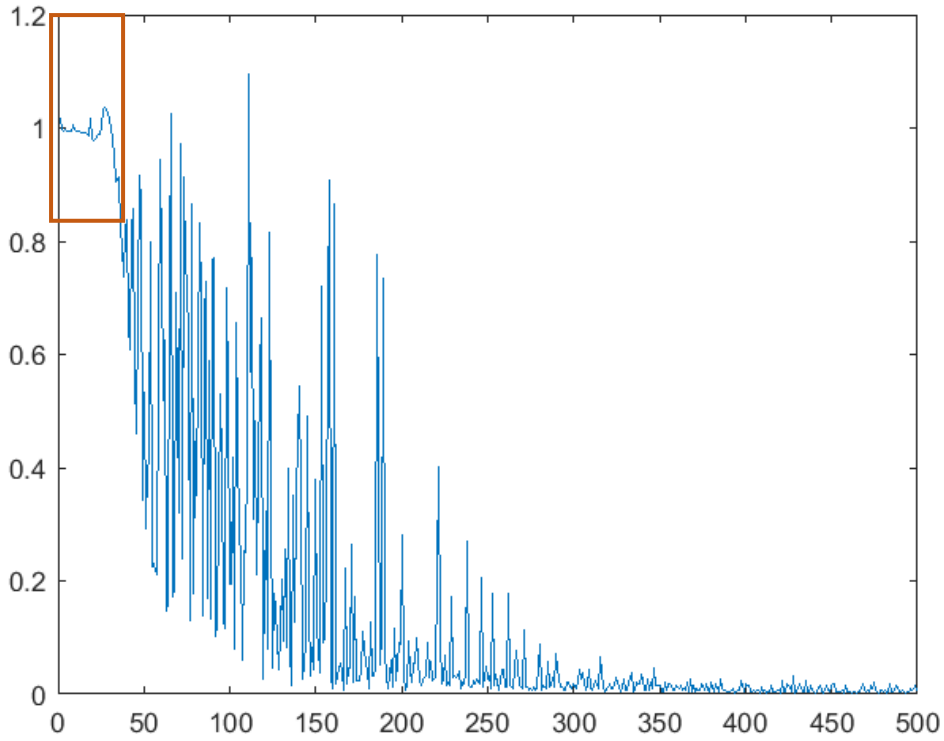
4. 同時訓練第一、二語言：



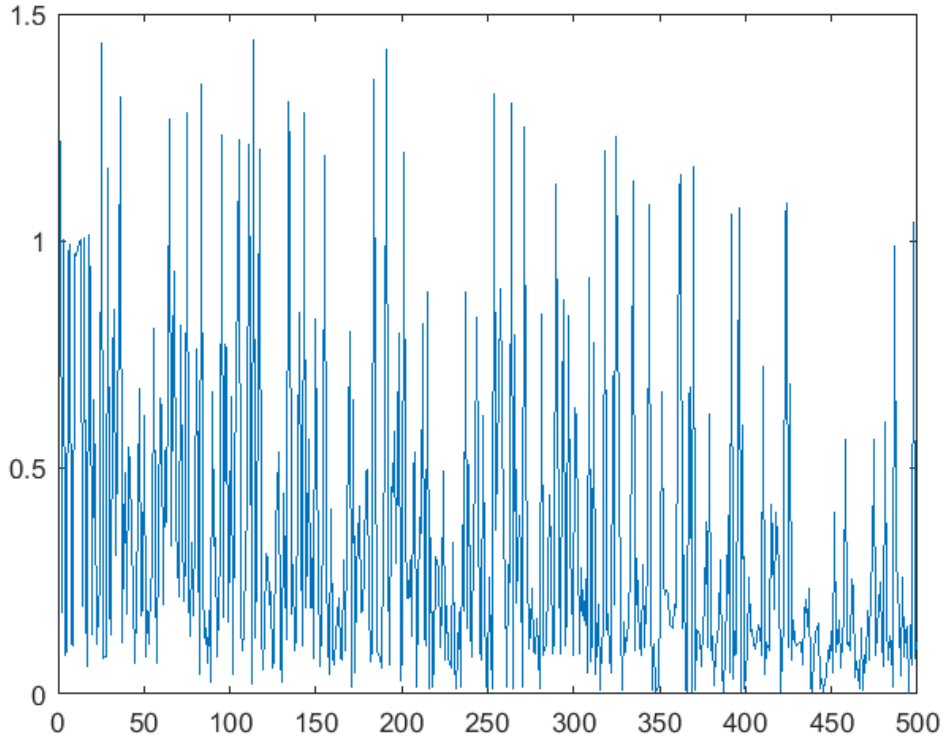
5. 雙語(先一後二)，訓練第三語言：



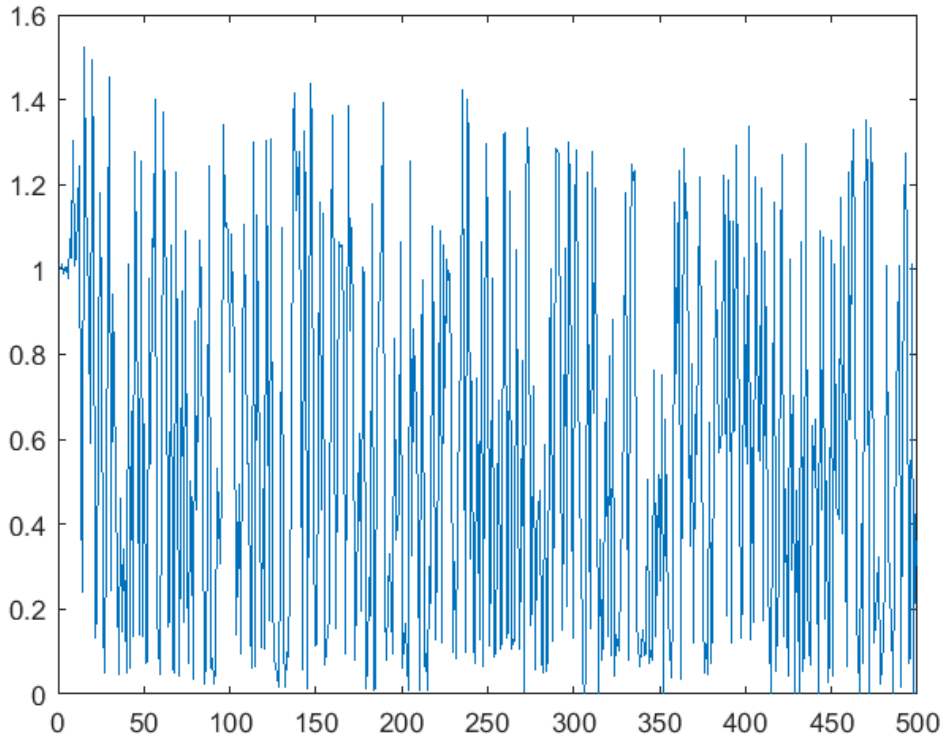
6. 雙語(同時)，訓練第三語言：



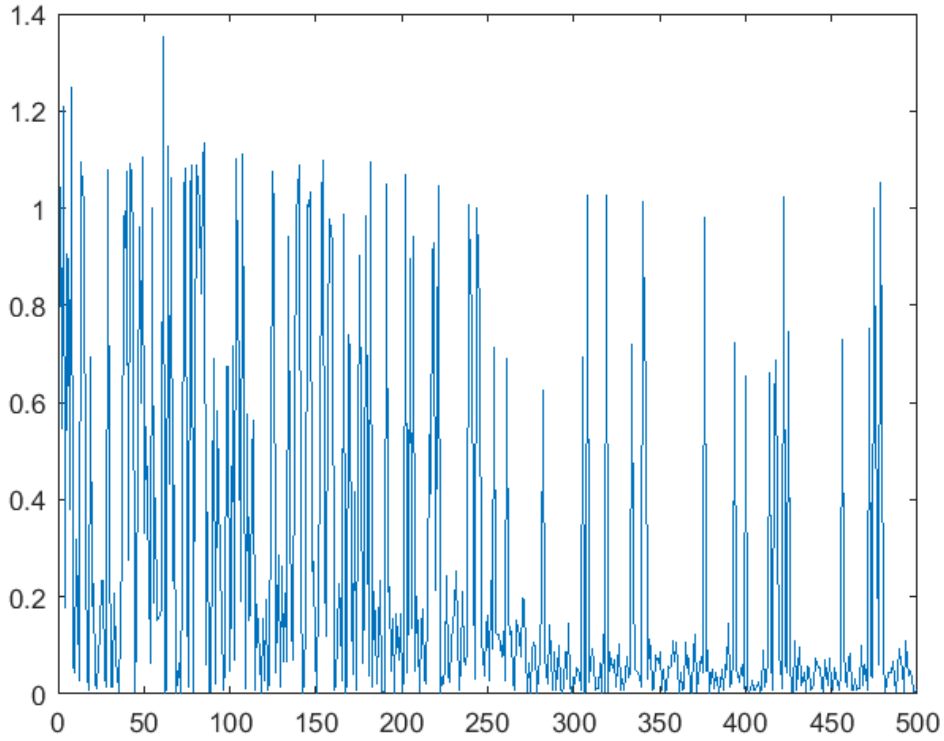
7. 雙語(先一後二)學習完第三種語言後，測試第一語言能力



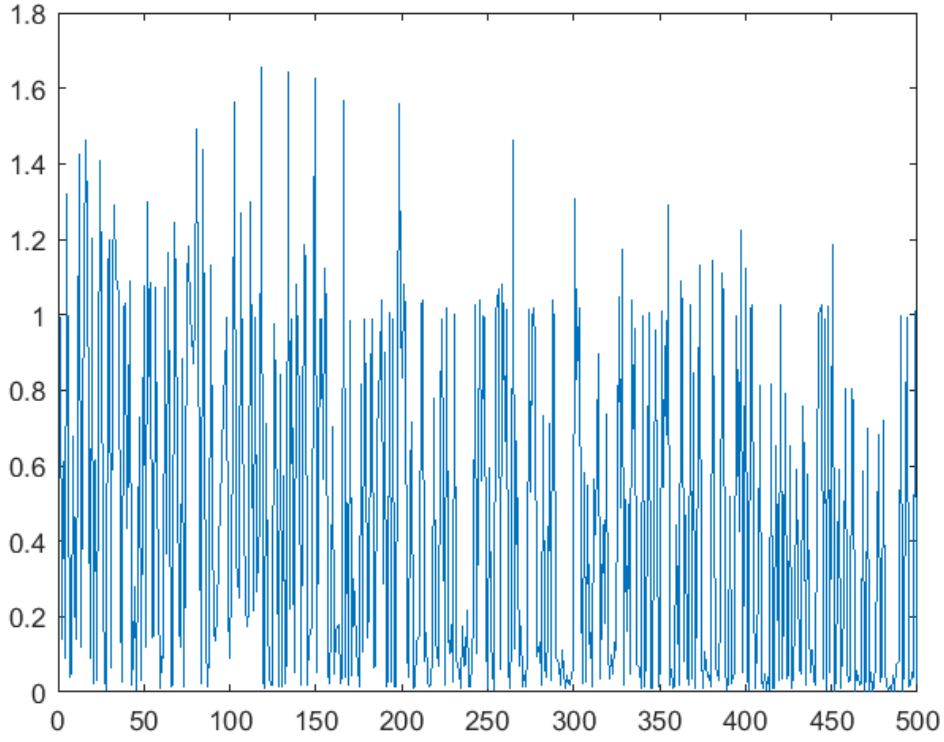
8. 雙語(先一後二)學習完第三種語言後，測試第二語言能力



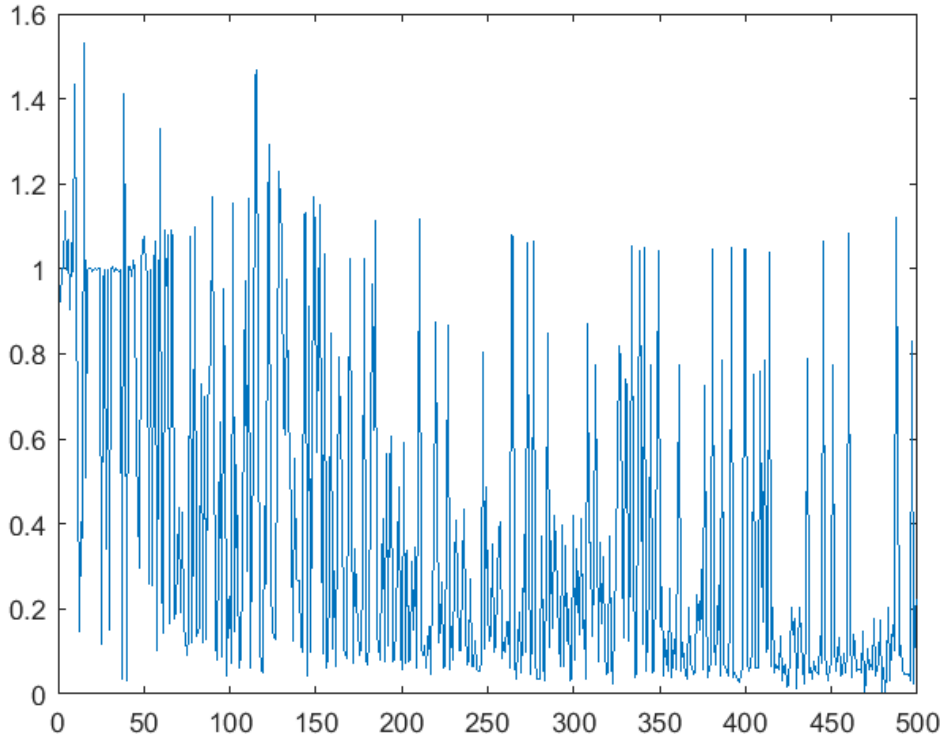
9. 雙語(同時)學習完第三種語言後，測試第一語言能力



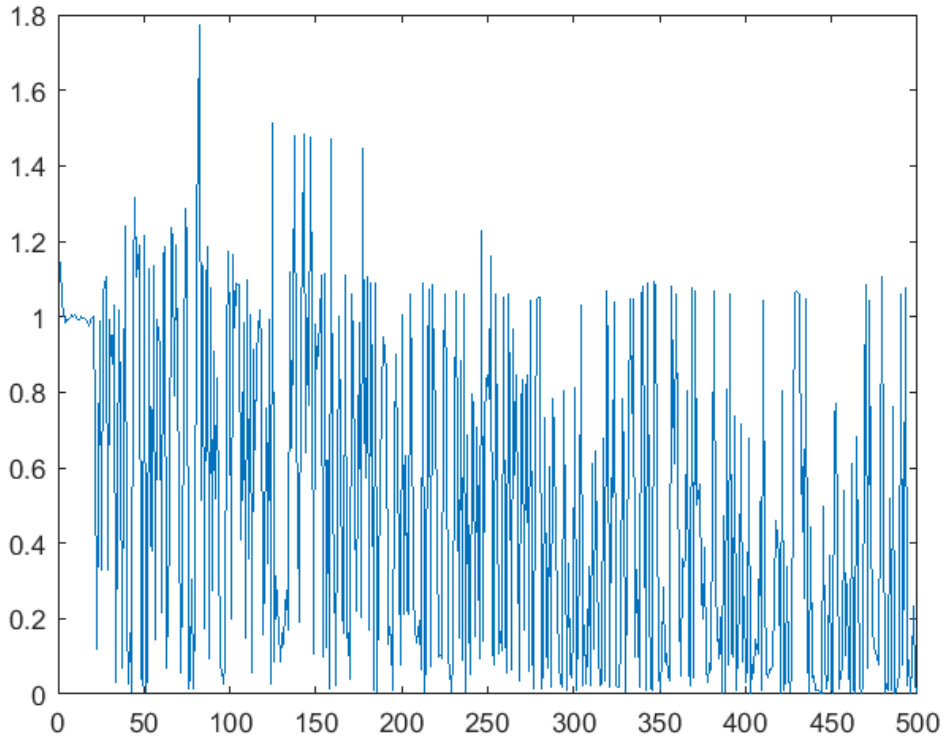
10. 雙語(同時)學習完第三種語言後，測試第二語言能力



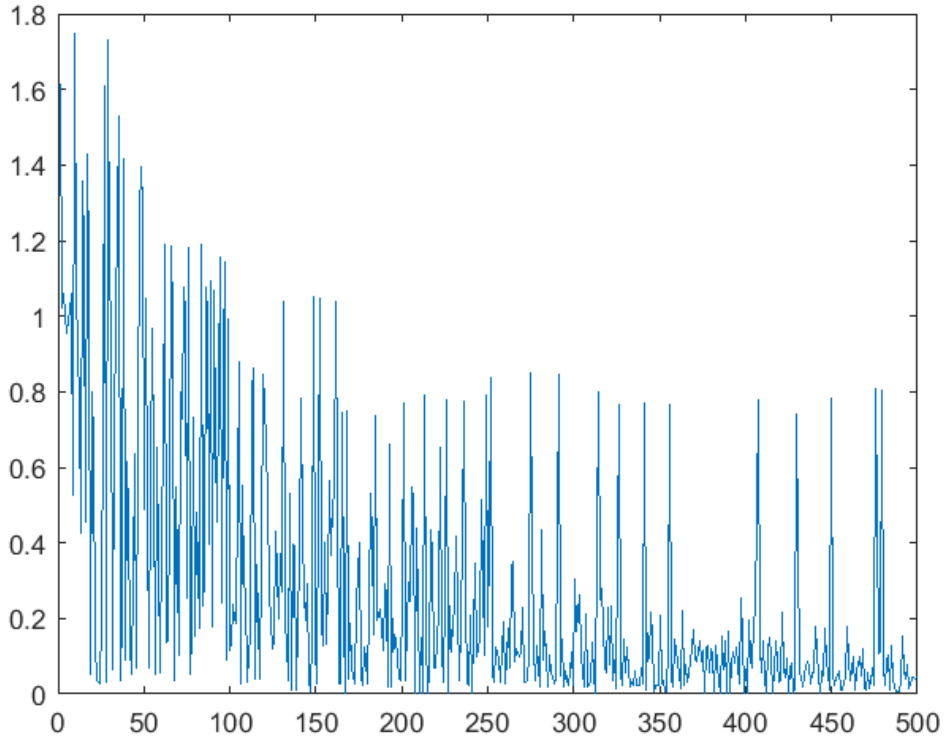
11. 雙語(先一後二，4000 次第三語言，測試第一語言)



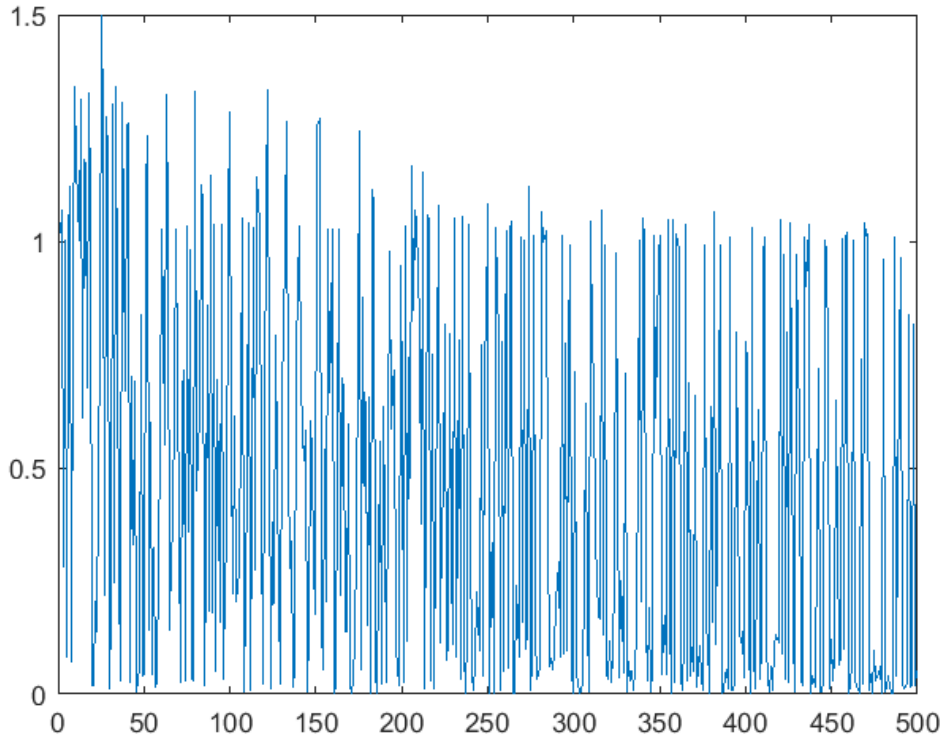
12. 雙語(先一後二，4000 次第三語言，測試第二語言)



13. 雙語(同時，4000 次第三語言，測試第一語言)



14. 雙語(同時，4000 次第三語言，測試第二語言)



總結：

上面畫的圖是以 pss 這個數據進行繪圖的，pss 是 patern sum of squares，會把每一次的 Output 和 target 之間的平方差相加。我們觀察到通常學習第一個語言的時候，會需要大約 50 個 epoch 才會進入到我們認為是學會的狀態，也就是高震幅的擺動。接著學習第二語言的時候，在前面幾十個 epoch 會呈現一個非常高錯誤的狀態(紅色圈)接著才進入中震幅的震盪(綠色圈)，最後才進入高振幅震盪，代表學會第二語言。

至於同時學習兩種語言的模型，他如我們所預料的花費比較長的時間才把兩種語言一次性地學會，大約花了 100 個 epoch。不過如果看“先學一再學二”的總學習時長大約也是 50+50=100 個 epoch。所以其實整體花費的時長是一樣的，只是單次性的時間較長而已。

接著觀察“先一後二”組別學習第三語言的時候，發現大約也是要到 50 多個 epoch 的時候 error 值才會下降(紅色方框)，而且發現前面的錯誤會先維持在一個高水平的位置後，逐步下降到了 200 個 epoch 的時候產生非常低的錯誤率。

相對的，“同時學習雙語”組別學習第三種語言時，也會先產生高水平的 error 值，但是在接近 50 個 epoch 時(橘色方框)，就會顯著下降到中震幅的位置。然而大概要到 300 多個 epoch 才會產生非常低的錯誤率。最後，我們再分別進行原本第一和第二語言的維持率測試(圖 7～圖 10)。這兩種看似都

還記得原本的語言，因為看起來都在 25 個 epoch 以內就呈現高振福震盪的模式。然後為了測試 attrition 效果，我們讓學完雙語的模型分別再次讓他學習 4000epoch 的第三語言，為了看看是否能徹底忘記前面學習過的兩種語言。在(圖 11～圖 14)我們可以觀察到確實產生 attrition 效果，因為兩者都需要接近 50epoch 才會進入高振幅振盪。

但是值得注意的在於“同時學習雙語者”遺忘的效果似乎比“先一後二雙語者”還要來的差一點。換言之，“同時學習雙語者”記憶前面兩種語言的能力似乎比“先一後二雙語者”還要來的強一點。因為我們發現“同時”的 error 值比“先一後二雙語者”還要來的快一點進入高震幅震盪。

IV. 討論:

從上述的結果看來，順序性的學習雙語和同時性的學習雙語在學習第三種語言所花費的時間上並沒有太大的區別，似乎說明順序性和同時性雙語學習者在工作記憶的表現並沒有哪方特別優勢的情況。

從另一個角度來看，一開始為了確保我們的模型能夠順利完成三種語言的學習我們將它等倍數的增加了每一層 layer 的 node 數量，進行了一系列的模擬，或許在我們對人類腦容量的極限有更深的了解也可以在做相關的研究，討論如何有效率的利用有限的工作記憶進行學習。

透過簡單的 simple recurrent network 模型來模擬人類的語言學習，還有很多的語言學習的因素都還是尚未人類了解的，在 simple recurrent network 裡也都是以隱式的方式表示，光是這樣已經能夠讓我們發現許多有趣的內部表徵，也幫助我們更加了解人類可以如何進行語言學習。

V. 參考資料:

Bartolotti, J., & Marian, V. (2017). Bilinguals' Existing Languages Benefit Vocabulary Learning in a Third Language. *Lang Learn, 67*(1), 110-140. doi:10.1111/lang.12200

Lukasik, K. M., Lehtonen, M., Soveri, A., Waris, O., Jylkkä, J., & Laine, M. (2018). Bilingualism and working memory performance: Evidence from a large-scale online study. *PLoS One, 13*(11), e0205916. doi:10.1371/journal.pone.0205916