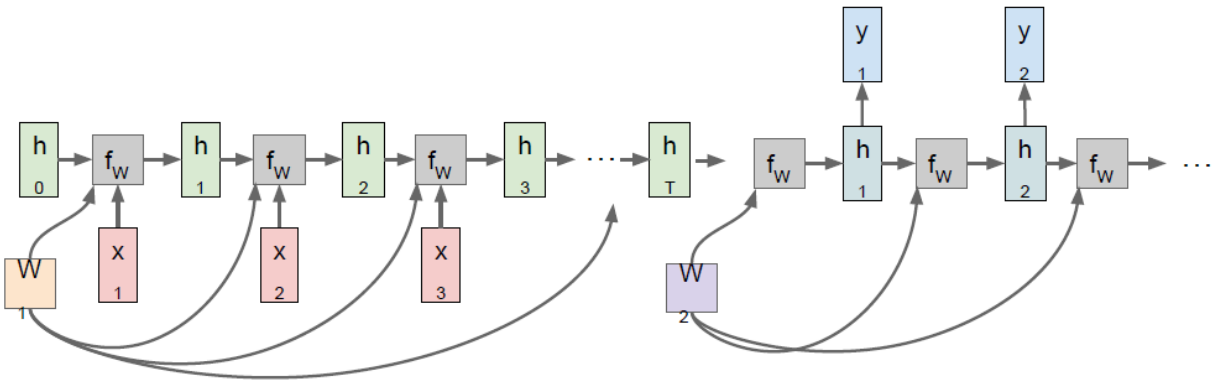
به نام خدا

تمرین سری ۶ یادگیری عمیق – دکتر فاطمی زاده

علی فتحی ۹۴۱۰۹۲۰۵

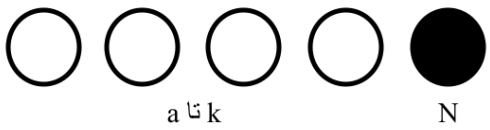
سوال ۱: Pattern

می‌دانیم LSTM ، هیچ‌گاه نمی‌تواند بدون آن که ورودی را کامل دیده باشد، ورودی را تشخیص دهد؛ زیرا در ایده‌ال ترین حالت هم، ما (انسان) اول سری را تا آمدن N (مانند aaaN) می‌بینیم سپس با توجه به دیدن سری‌ها هنگام آموزش، می‌گوییم خروجی aaaNbbb است؛ و این کار با مشاهده کامل ورودی در ابتدا و سپس نوشتن خروجی در پی آن است پس طرز کار انسان بیشتر شبیه ساختار Many-to-One, One-to-Many زیر است:



اما در صورتی که کاراکتر به کاراکتر ورودی بدهیم (مانند LSTM صورت سوال)، عاقلانه‌ترین کار آن است که ابتدا تا جایی که N نیامده‌است، خروجی a تولید کنیم و سپس با آمدن N ، شروع به خروجی b دادن به تعداد a های ورودی بکنیم و در انتها نیز کاراکتر end را بدهیم؛ پس در کم خطا ترین حالت نیز امکان تشخیص ۱۰۰ درصد درست نیست و در بهترین حالت، از تشخیص محل درست N معذوریم.

[علت آن که اگر تا قبل از آمدن N خروجی را a بدهیم بهتر است را بررسی می‌کنیم. سری خروجی ایده‌ال ما به شکل زیر است که در واقع با یک تاخیر پس از ورودی ظاهر می‌شود:



به صورت ساده، فرض می‌کنیم احتمال آنکه LSTM تا قبل از مشاهده N ، خروجی N تولید کند برابر p باشد (در واقع این احتمال با گذر زمان افزاینده است اما برای نشان دادن رفتار، ثابت گرفتن نیز به نتیجه دلخواه منجر می‌شود، و در واقع خطای پیش‌بینی در حالت افزاینده بودن احتمال p باید حتما از این برآورد ما کمتر باشد زیرا درجه آزادی بیشتری به آن داده شده است). خطای متوسط را برای این قسمت محاسبه می‌کنیم (به شکل تعداد انحراف‌ها با حالت درست):

$$\text{loss} = (1-\text{p}).\binom{k}{i}p^i(1-p)^{k-i}.(i+1)\ \{if\ last\ one\ is\ a\} +$$
$$p.\binom{k}{i}p^i(1-p)^{k-i}.i\ \{if\ last\ one\ is\ N\}$$

$$\Rightarrow \text{loss} = (1-\text{p}) + \binom{k}{i}p^i(1-p)^{k-i}.i\ = 1-p+kp = 1+(k-1)p$$

که با توجه به آن که k بزرگتر مساوی ۱ است، کمترین loss متناظر با احتمال N آمدن برابر با صفر است.]

بخش ۱: حال LSTM را آموزش می‌دهیم:

در آموزش، از چهار کاراکتر N، a ، b و e استفاده شده که به ترتیب کدهای [1 0 0 0]، [0 1 0 0]، [0 0 1 0] و [0 0 0 1] دارند. سایز batch ، ۳۲ است و تعداد ۵۰۰ دور iteration اجرا می‌شود. نتیجه آموزش به صورت زیر است:

```
pred:  abe true:  Nbe
accuracy:  0.6666667
pred:  aabbe true:  aNbbe
accuracy:  0.8
pred:  aaabbbe true:  aaNbbbe
accuracy:  0.85714287
pred:  aaaabbbbe true:  aaaNbbbbe
accuracy:  0.8888889
pred:  aaaaabbbbbe true:  aaaaNbbbbbe
accuracy:  0.90909094
pred:  aaaaaabbbbbbe true:  aaaaaNbbbbbbe
accuracy:  0.9230769
pred:  aaaaaaabbbbbbbe true:  aaaaaaNbbbbbbbe
accuracy:  0.93333334
pred:  aaaaaaaabbbbbbbbe true:  aaaaaaaNbbbbbbbbe
accuracy:  0.9411765
pred:  aaaaaaaaabbbbbbbbbe true:  aaaaaaaaNbbbbbbbbbe
accuracy:  0.94736844
pred:  aaaaaaaaaNbbbbbbbbbbe true:  aaaaaaaaaNbbbbbbbbbbe
accuracy:  1.0
```

که همان پدیده توضیح داده شده تا حد خوبی مشاهده می‌شود و به جز در مواردی استثنا، مانند مورد آخر اجرای اتفاقی آورده شده، سری‌ها فاقد N می‌باشند. طبق انتظار صحت، با افزایش طول دنباله بیشتر می‌شود زیرا خطای شبکه، یک یا دو کاراکتر است و هرچه تعداد کل کاراکترها بیشتر باشد این یک یا دو خطا درصد کمتری از کل را تشکیل می‌دهند.

بخش ۲: در هنگام آموزش، ورودی شبکه Teacher Forcing است؛ یعنی ورودی شبکه توسط خود ما به آن داده می‌شود؛ اما هنگام تست لازم است ورودی دنباله در هر مرحله از خروجی مرحله قبل خود بیاید. تا رسیدن به N ، ورودی شبکه را Teacher Forcing می‌دهیم و ادامه دنباله را با خروجی خودش مرتبط می‌کنیم. برای اینکه کار، ابتدا دنباله‌ی ورودی مثل aaaaaN را به شبکه آموزش داده شده می‌دهیم. سپس کاراکتر آخر تولیدی آن را به انتهای دنباله می‌چسبانیم، و دوباره به آن می‌دهیم. برای مثال اگر کاراکتر آخر خروجی b باشد ورودی مرحله بعد aaaaaNb خواهد بود. سپس مجددا کاراکتر آخر خروجی را به انتهای این دنباله می‌چسبانیم و به عنوان ورودی دفعه بعد می‌دهیم تا جایی که کاراکتر آخر خروجی شبکه، e باشد ، و در اینجا خروجی نهایی را اعلام می‌کنیم.

نتیجه تست به صورت زیر است:

```
*** TESTING ***
in:  aNb pred:  abe true:  Nbe
test accuracy:  0.6666666666666666
in:  aaNbb pred:  aabbe true:  aNbbe
test accuracy:  0.8
in:  aaaNbbb pred:  aaabbbe true:  aaNbbbe
test accuracy:  0.8571428571428571
in:  aaaaNbbbb pred:  aaaabbbbe true:  aaaNbbbbe
test accuracy:  0.8888888888888888
in:  aaaaaNbbbbb pred:  aaaaabbbbbe true:  aaaaNbbbbbe
test accuracy:  0.9090909090909091
in:  aaaaaaNbbbbbb pred:  aaaaaabbbbbbe true:  aaaaaNbbbbbbe
test accuracy:  0.9230769230769231
in:  aaaaaaaNbbbbbbb pred:  aaaaaaabbbbbbbe true:  aaaaaaNbbbbbbbe
test accuracy:  0.9333333333333333
in:  aaaaaaaaNbbbbbbbb pred:  aaaaaaaabbbbbbbbe true:  aaaaaaaNbbbbbbbbe
test accuracy:  0.9411764705882353
in:  aaaaaaaaaNbbbbbbbbb pred:  aaaaaaaaNbbbbbbbbbe true:  aaaaaaaaNbbbbbbbbbe
test accuracy:  1.0
in:  aaaaaaaaaaNbbbbbbbbbb pred:  aaaaaaaaNNbbbbbbbbbbe true:  aaaaaaaaaNbbbbbbbbbbe
test accuracy:  0.9523809523809523
in:  aaaaaaaaaaaNbbbbbbbbbbb pred:  aaaaaaaaNNNbbbbbbbbbbbe true:  aaaaaaaaaaNbbbbbbbbbbbe
test accuracy:  0.9130434782608695
in:  aaaaaaaaaaaaNbbbbbbbbbbbb pred:  aaaaaaaaNNNNbbbbbbbbbbbbe true:  aaaaaaaaaaaNbbbbbbbbbbbbe
test accuracy:  0.88
in:  aaaaaaaaaaaaaNbbbbbbbbbbbbb pred:  aaaaaaaaNNNNNbbbbbbbbbbbbbe true:  aaaaaaaaaaaaNbbbbbbbbbbbbbe
test accuracy:  0.8518518518518519
in:  aaaaaaaaaaaaaaNbbbbbbbbbbbbbb pred:  aaaaaaaaNNNNNNbbbbbbbbbbbbbbe true:  aaaaaaaaaaaaaNbbbbbbbbbbbbbbe
test accuracy:  0.8275862068965517
```

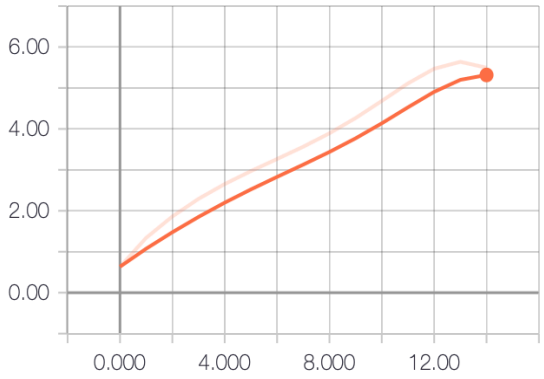
نکته جالب در این بخش آن است که سایز همه خروجی‌ها، دقیقا برابر خروجی مطلوب شده است و کاراکتر e در همه تست‌ها دقیقا در جای درست خود تولید شده است!

بخش ۳: فعالیت یا Cell State ها را برای k=15 رسم می‌کنیم:

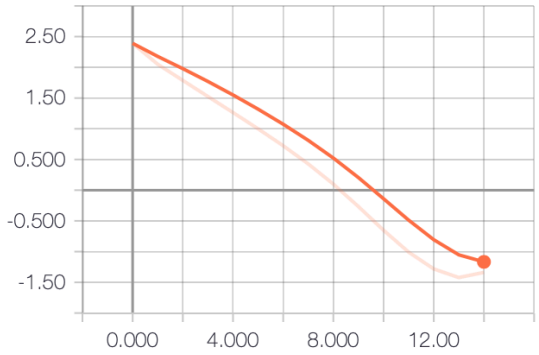
(خروجی تولید شده برای این حالت یه شکل زیر است که صحت ۷۷ درصد دارد:)

```
*** Saving and Testing for k=15 ***
in:  aaaaaaaaaaaaaaNbbbbbbbbbbbbbbb pred:  aaaaaaaaNNNNNNbbbbbbbbbbbbbbe true:  aaaaaaaaaaaaaaNbbbbbbbbbbbbbbe
same size: False test accuracy:  0.7741935483870968
```

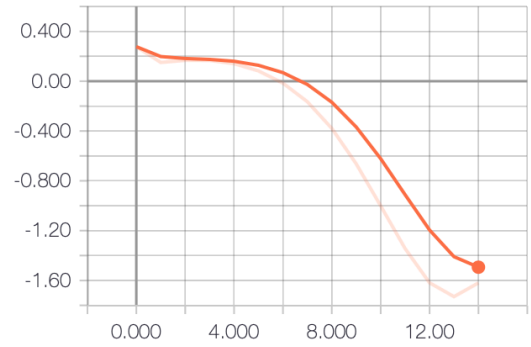
cell0_summary



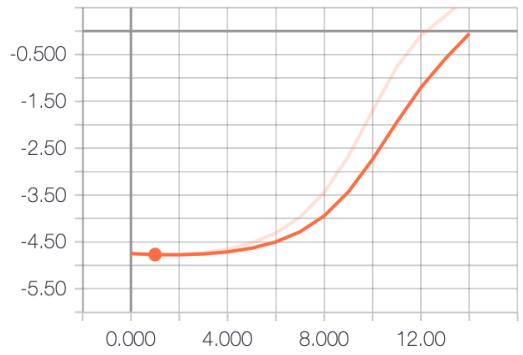
cell1_summary



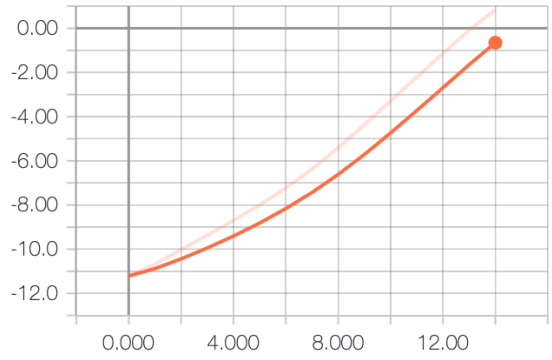
cell2_summary



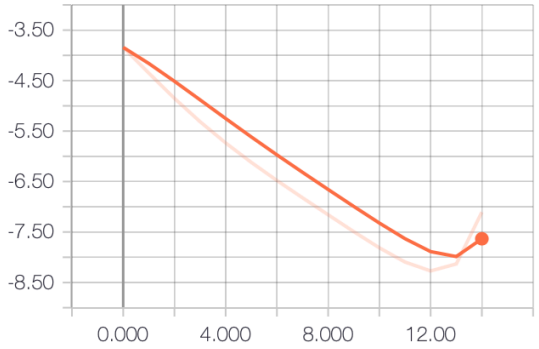
cell3_summary



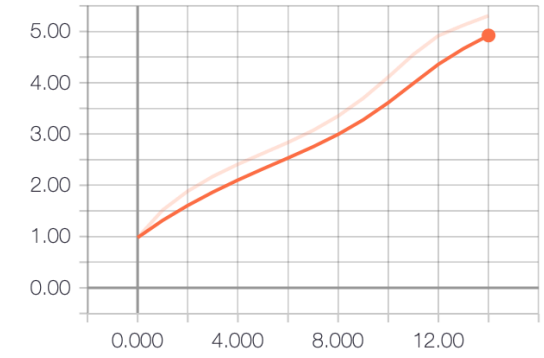
cell4_summary



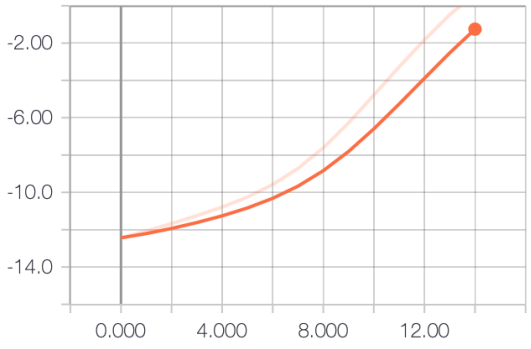
cell5_summary

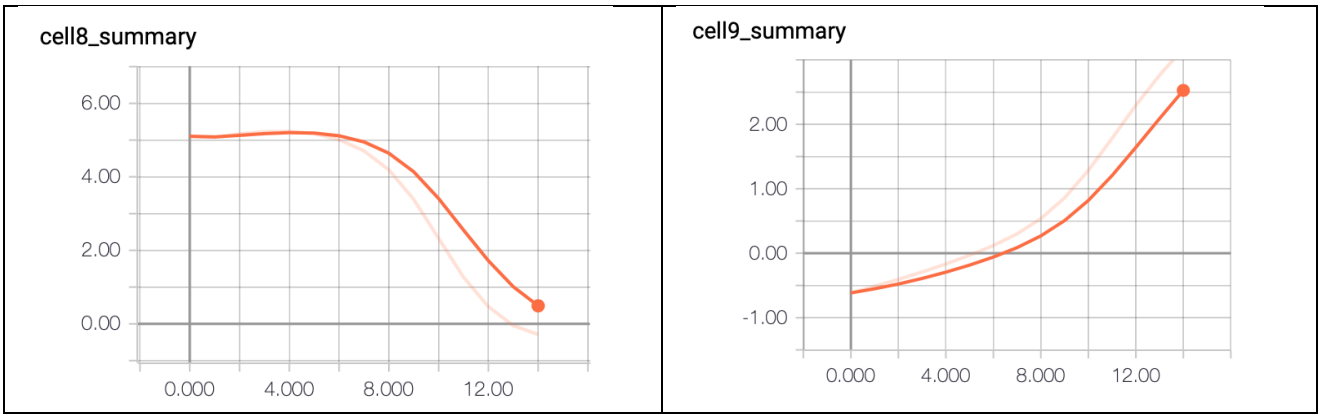


cell6_summary



cell7_summary





که محور افقی، فعالیت نورون‌های حافظه از لحظه وارد شدن N تا لحظه تولید e است (که این فاصله ۱۵ مقدار مختلف دارد).
همانطور که مشاهده می‌شود، با وارد شدن سری aaaaaaaaaaaaaaaN به شبکه، نورون‌های حافظه به وضعیت مشخصی می‌روند:

$$[C0, C1, ..., C9] = [6, 2.5, 0.3, -4.5, -11, -4, 1, -12, 5, -0.5]$$
$$= [H, M, 0, -M, -H, -M, 0, -H, M, 0]$$

(H: High, M: Middle)

که عدد ۱۵ در این حافظه ذخیره شده ، در این state ، می‌داند باید ۱۵ عدد b تولید کند.
در نهایت، به حالت زیر میل می‌کنند که آماده خروج کاراکتر e است:

$$[C0, C1, ..., C9] = [5, -1, -1.5, 0, -1, -7.5, 5, -1, 0.5, 2.5]$$
$$= [M, 0, -M, 0, 0, -H, M, 0, 0, M]$$

که یعنی حافظه‌های ۴ و ۷ که از مقدار بسیار بزرگی به صفر رسیده‌اند، بسیار تاثیر گذار بوده و عدد ۱۵ در آنها ذخیره شده است.

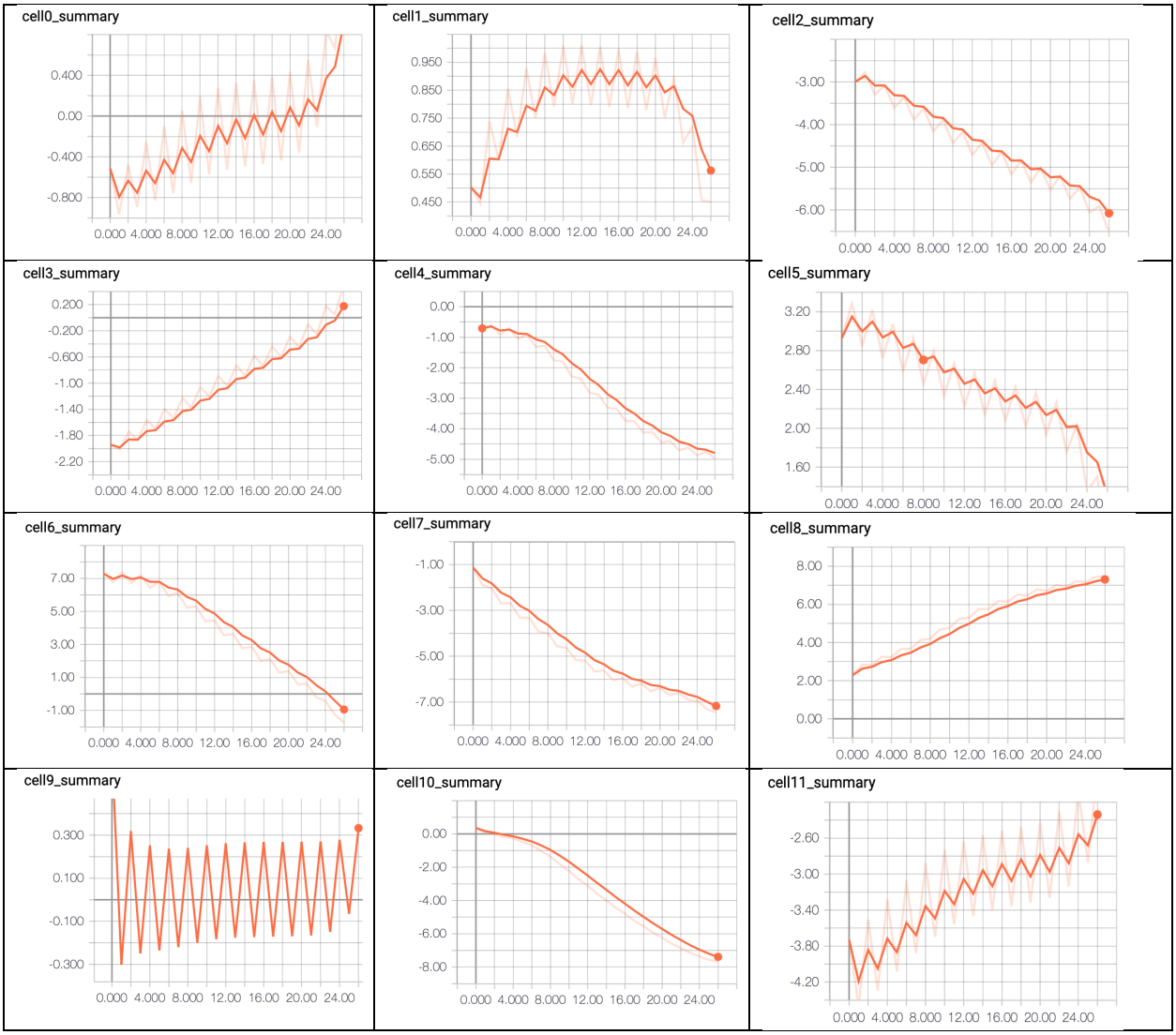
بخش ۴: فرآیند آموزش:

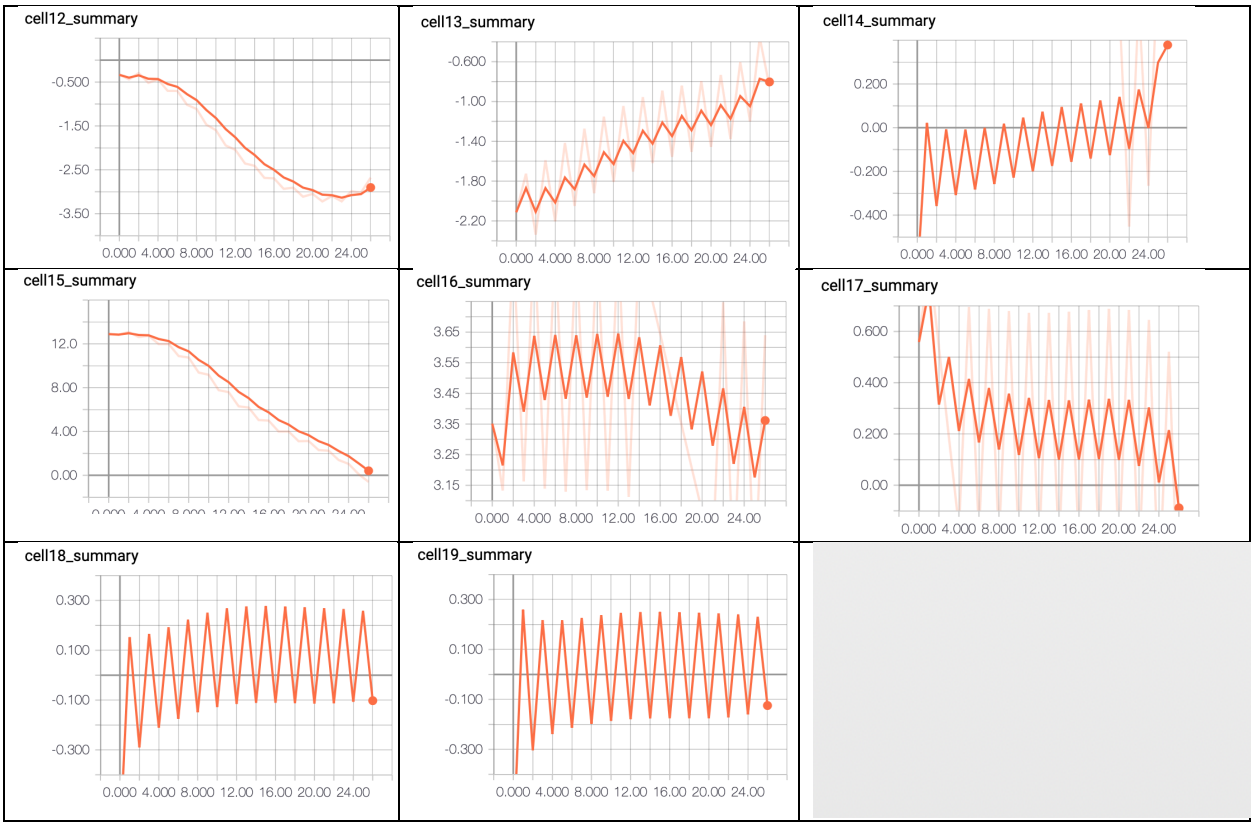
```
pred:  babae true:  bNbae
accuracy:  0.8
pred:  babababae true:  babNbabae
accuracy:  0.8888889
pred:  babababababae true:  bababNbababae
accuracy:  0.9230769
pred:  bababababababababae true:  babababNbababababae
accuracy:  0.9411765
pred:  bababababababababababab true:  bababababNbababababae
accuracy:  0.9047619
pred:  bababababababababababab true:  bababababNbababababae
accuracy:  0.92
pred:  bababababababababababababab true:  bababababababNbababababae
accuracy:  0.9310345
pred:  babababababababababababababae true:  bababababababNbabababababae
accuracy:  0.969697
pred:  babababababababababababababababae true:  bababababababNbabababababae
accuracy:  0.972973
pred:  babababababababababababababababababae true:  bababababababababNbabababababababae
accuracy:  0.9756098
```

بخش تست (چند دنباله برای مثال):

```
in:  abababababNbababababaa pred:  babababababababababababae true:  babababababNbababababae
same size: True test accuracy:  0.96
in:  abababababababNbababababababa pred:  babababababababababababababae true:  babababababababNbababababababae
same size: True test accuracy:  0.9655172413793104
in:  abababababababNbababababababa pred:  babababababababababababababae true:  babababababababNbababababababae
same size: True test accuracy:  0.9696969696969697
in:  abababababababababNbababababababa pred:  babababababababababababababae true:  babababababababNbababababababae
same size: True test accuracy:  0.972972972972973
in:  abababababababababNbababababababa pred:  babababababababababNbababababababae true:  babababababababababNbababababababae
same size: True test accuracy:  1.0
in:  abababababababababababNbababababababa pred:  babababababababababNbNbababababababababae true:  babababababababababNbababababababae
same size: True test accuracy:  0.9777777777777777
in:  abababababababababababababNbababababababa pred:  babababababababababNbNbNbababababababababae true:  babababababababababababNbababababababae
same size: False test accuracy:  0.8979591836734694
```

و بخش فعالیت‌ها:





از این قسمت، متوجه می‌شویم LSTM قابلیت یادگیری سری‌های پیچیده‌تر را نیز دارد. در مورد سلول‌های حافظه اتفاق جالبی رخ داده است؛ اینکه برای تولید سری متناوب a و b مقدار بعضی عناصر حافظه Fluctuate می‌کنند! این به ما می‌گوید که در یک LSTM ، برخی عناصر حافظه عمق سری (k) را ذخیره می‌کنند و مقدار آنها در ابتدا به عددی بزرگ رسیده و سپس به جلو رفتن در زمان به مقدار صفر می‌رسند. برخی دیگر از عناصر حافظه آموزش دیده شده‌اند که تناوب کنند و سری جابجا شونده مد نظر را تولید کنند. نتیجه کلی تخمین این سری نیز بسیار قابل قبول است (برای دنباله‌های نه چندان بزرگ).