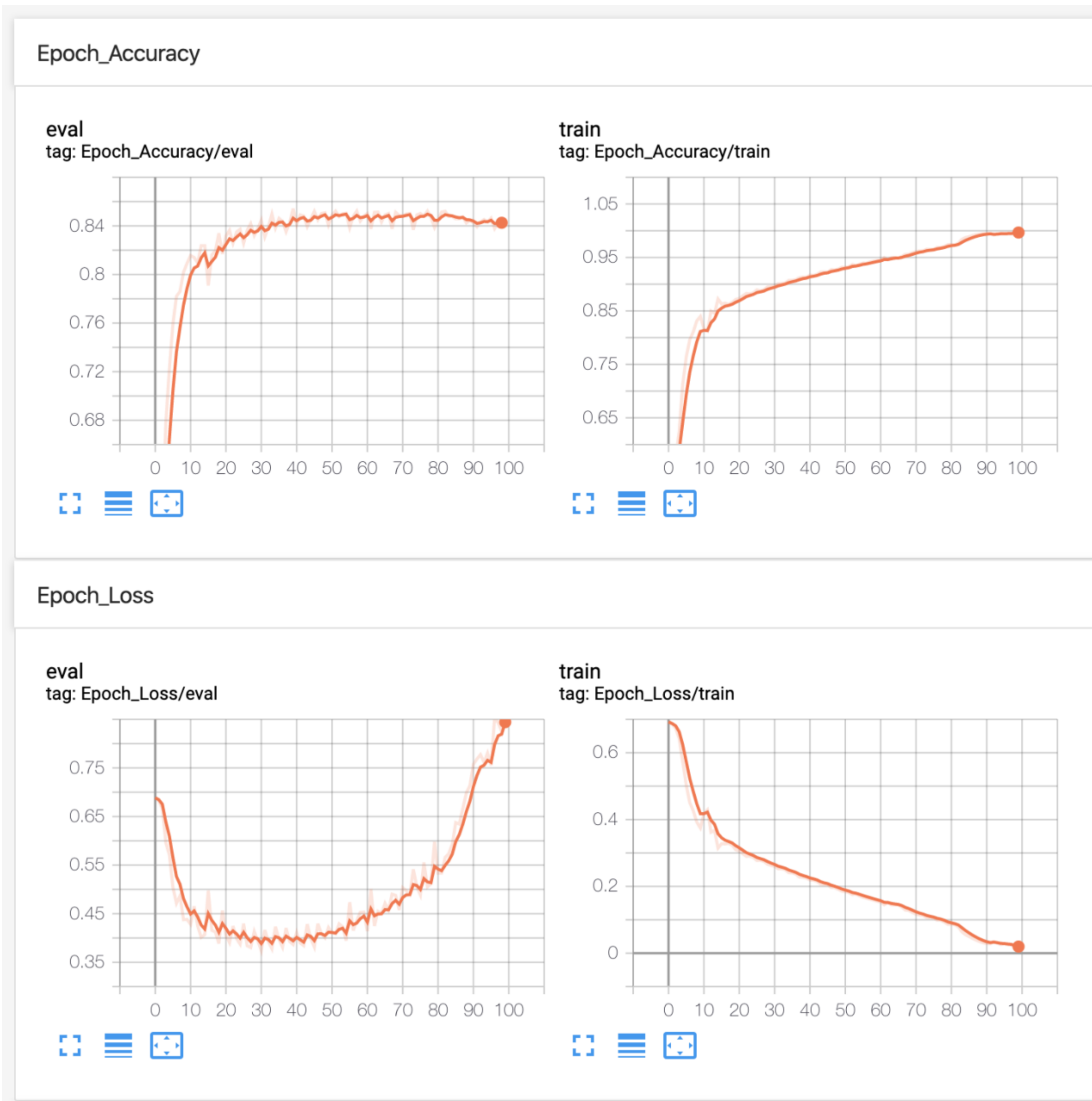
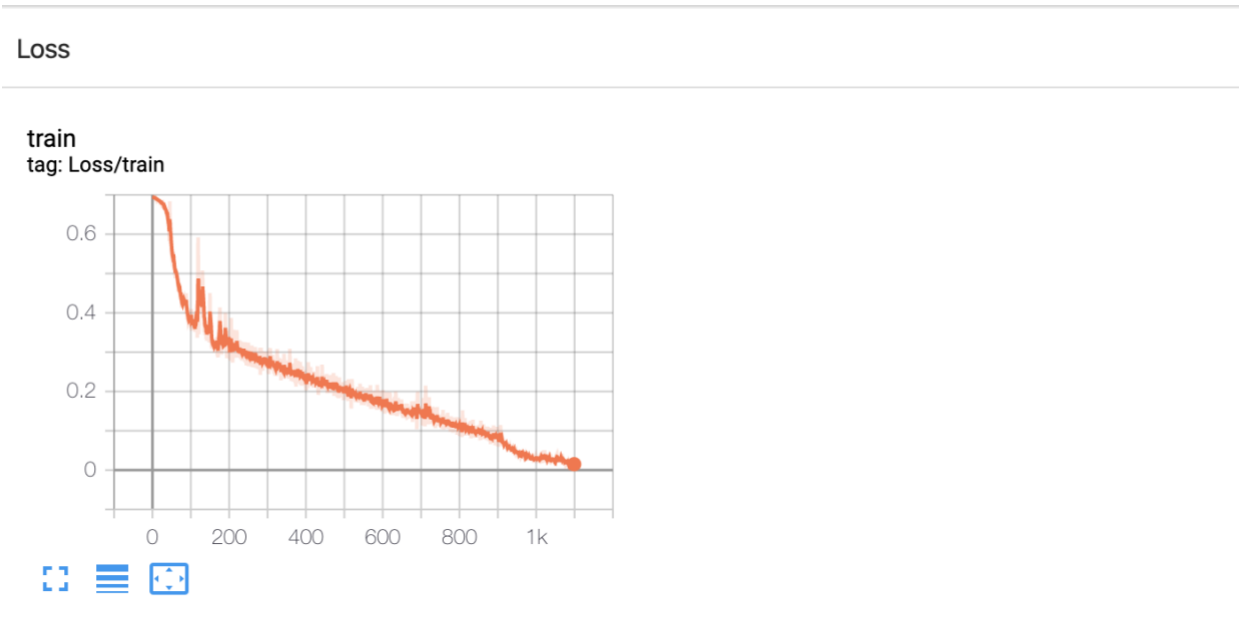
Mathew Varughese
CS 1699
HW 5

<u>Part I: Sentiment Analysis on IMDB Reviews</u>

1. Running GRU Cell



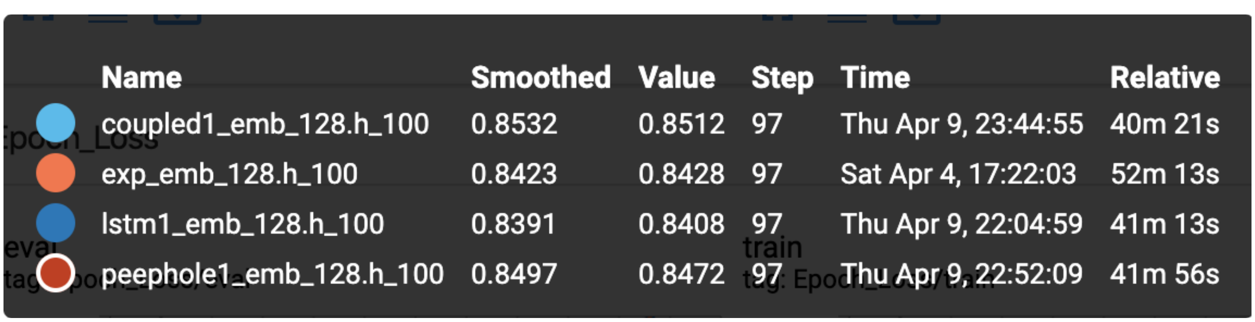


3. Total Parameters

GRU	LSTM	PEEPHOLED	COUPLED
Total Parameters: 68700	Total Parameters: 91600	Total Parameters: 121600	Total Parameters: 68700

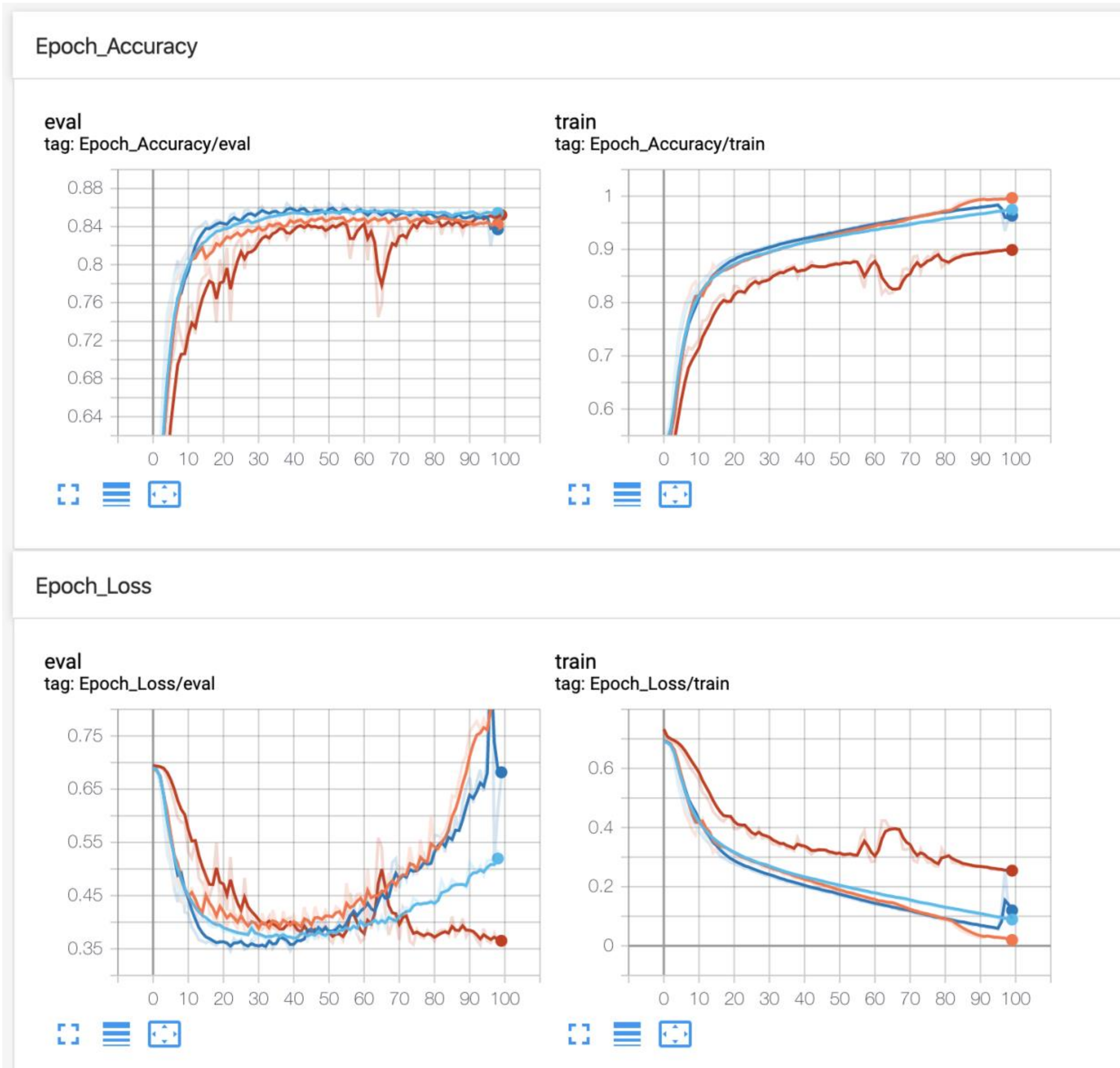
4. Results

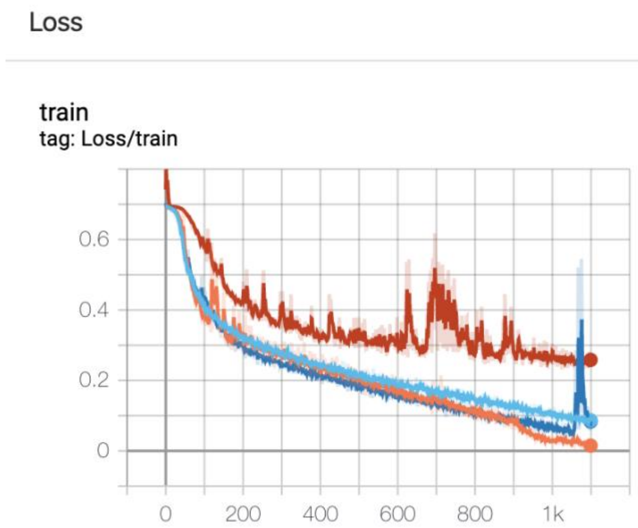
The next pages include the results. All LSTMs performed pretty similar. The coupled LSTM was a little better and slightly faster. The Peephole Loss graph shows it followed a slightly different trajectory than the other LSTMs. However, I think this exercise shows that different LSTM architectures do not make *that big* of a difference. As mentioned in the [Greff, et al (2015)] paper, these LSTM variants are about the same. This chart shows the validation accuracy and the time taken.



Name	Smoothed	Value	Step	Time	Relative
coupled1_emb_128.h_100	0.8532	0.8512	97	Thu Apr 9, 23:44:55	40m 21s
exp_emb_128.h_100	0.8423	0.8428	97	Sat Apr 4, 17:22:03	52m 13s
lstm1_emb_128.h_100	0.8391	0.8408	97	Thu Apr 9, 22:04:59	41m 13s
peephole1_emb_128.h_100	0.8497	0.8472	97	Thu Apr 9, 22:52:09	41m 56s

GRU
LSTM
PEEPHOLE
COUPLED

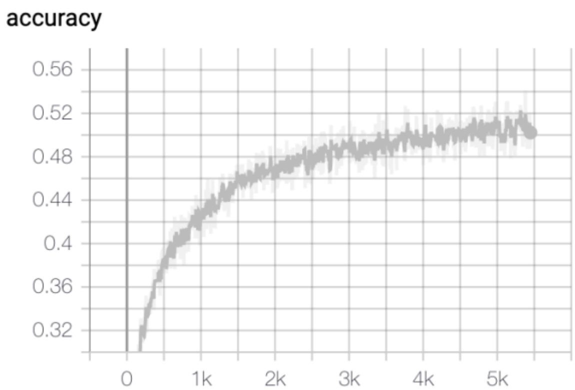


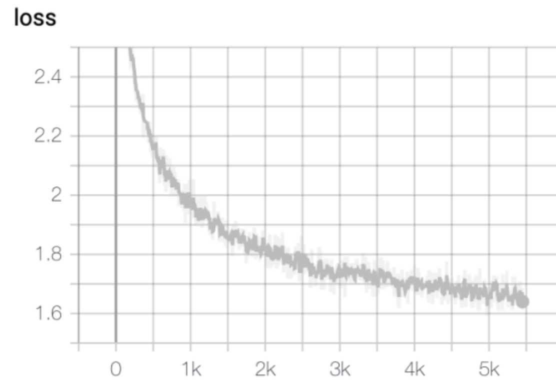


<u>Part II: Building a Shakespeare Writer</u>

2.
Model Architecture:
```
SentenceGeneration(
  (embedding): Embedding(65, 50, padding_idx=0)
  (rnn_model): GRUCell(input_size=50, hidden_size=50, bias=True)
  (classifier): Linear(in_features=50, out_features=65, bias=True)
)
```

Loss: 2.0736, Accuracy: 0.4115: : 545it [43:13, 4.76s/it]
Loss: 1.9892, Accuracy: 0.4310: : 545it [37:11, 4.09s/it]
Loss: 1.8494, Accuracy: 0.4666: : 545it [37:20, 4.11s/it]
Loss: 1.7587, Accuracy: 0.4979: : 545it [38:24, 4.23s/it]
Loss: 1.7559, Accuracy: 0.4809: : 545it [38:25, 4.23s/it]
Loss: 1.7565, Accuracy: 0.4843: : 545it [38:19, 4.22s/it]
Loss: 1.7184, Accuracy: 0.4903: : 545it [38:26, 4.23s/it]
Loss: 1.6420, Accuracy: 0.5258: : 545it [38:04, 4.19s/it]
Loss: 1.6337, Accuracy: 0.5182: : 545it [38:23, 4.23s/it]
Loss: 1.6120, Accuracy: 0.5038: : 545it [38:00, 4.19s/it]





This was generated, after 10 epochs:

```
ROMEO and JULIET
:
A amamen for of your ragain's firs,
St. up bes othnil's breaks
Mague, dene say 'telf me are tannt,
no? carried an vione you not so was mind I do
he beward: he thing sweet it incad like look'
reser, to fare for'th the
can. Sawnal I tan shall I call known loves,
Poep and perpose,
Groy Eve live Apailes decang
yo med ouse mernects, neight,
There aydial Fremuped to hastly, soltas,
Look on that houndis.

JULIET:
The bood I.
Your suchence buge what a hast lold,
Thush this migh's most park Ancer
By, anlless, eart'gl the Gaid brace
Tee aboud by you so dan let neprit,
Chough hazs of your nothraves him? O prits to for sigs,
And pleack my Emnow, the subsuen he twes right.

POLIUS:
No ances and not prinfens
To supper! Wronder to arm and mit.
Thits od'y me mutar pulled my honess, leired.

CLOMETLANUS:
I, I day, breads on membuse of beserce:
Belard, them intell you meed I to mad of whilats
I worth for that scarte.

Why Vame is shall you preaves
They in but the heir pellest; up thy great;
And for we
```

That is pretty crazy.

I tried it again with different parameters, mainly a bigger hidden size, and got a slightly better accuracy.

Model Architecture:
SentenceGeneration(
  (embedding): Embedding(65, 60, padding_idx=0)
  (rnn_model): GRUCell(input_size=60, hidden_size=130, bias=True)
  (classifier): Linear(in_features=130, out_features=65, bias=True)
)

Loss: 1.8615, Accuracy: 0.4606: : 545it [02:32, 3.58it/s]
Loss: 1.6313, Accuracy: 0.5080: : 545it [02:32, 3.57it/s]
Loss: 1.6182, Accuracy: 0.5174: : 545it [02:32, 3.57it/s]
Loss: 1.4917, Accuracy: 0.5411: : 545it [02:32, 3.57it/s]
Loss: 1.5287, Accuracy: 0.5368: : 545it [02:32, 3.57it/s]
Loss: 1.5037, Accuracy: 0.5318: : 545it [02:32, 3.58it/s]
Loss: 1.4634, Accuracy: 0.5555: : 545it [02:32, 3.57it/s]
Loss: 1.4492, Accuracy: 0.5428: : 545it [02:32, 3.57it/s]
Loss: 1.4743, Accuracy: 0.5521: : 545it [02:32, 3.57it/s]
Loss: 1.3784, Accuracy: 0.5775: : 545it [02:32, 3.57it/s]
Loss: 1.4391, Accuracy: 0.5580: : 545it [02:32, 3.57it/s]

The results were a little better:

```
ROMEO and JULIET
ELANUS:
This pringess the two enterphecificing:
Feak.

Clonce:
Nay! dead, mother, my lord, opboligrownley!
They are dozen deposistance if thou.

ALOFFYRY:
And, give of these man else stirg his pain
A witte.

Second Senatol:
All against the Durtions of I would be saled?
Come, church: how troth I mothing midned cause;
Frail nor to stay the godses will you weak'd it
all old sprait show to my near, cames it ord,
Which you knist chered of desperate their leave:
What gentle go hath a holy propost you me Moves
But is to her forth did unto the crose.
```

```
AUDINGBELLA:
Ay, that I, then?

CLARENCE:
If it with I fort! that I am thy sort,
The tongue's far it be and love the vilbors,
As I so fulll on years are my was
Sweet kills and began do our bly hear not their
peining: that's firing and loved the gods;
For you citief's book, gendle Margaonry.

AUPIOLIUS:
Fale, no faced Butis seen men, and done
Of his breast come to my vain into was,
Bossions; say, be dear stones by your son?
Therefore she was enemy
```

Just for fun, I concatenated all of the React JavaScript source code into one file and trained the model on that. The results were equally as cool. Here is a snippet:

```javascript
import type {ProfilingDataFrontent} from './ReactInternalTypes';
import type {ProfilerCompo}, itemCount - treeBlumsizeURI, y: 'submcon'}
'ErrorBoundary just warn; id not invoked in ent has been variables it object', () =>
{
    const {useRef} = this.props.discreteSuspense.relation(context);
    const registerTarget = currentName(type);
    workInProgress.mode & ProwigressigntStartTime;
    dispatchClassComponent.previous = flarNestedContext(parentType, error);
    const node = node.return;
    validaterleventeractions(
      buttonEventNode,
      expirationTime,
    ) {
      // An unusedes beurce
      return pointerType.aria;
    },
    () => Scheduler.unstable_newdCancelObject(renderIntoRoot))
    pushRoot.findMouseEvent(
      listener, rowIndex,
      children,
      name,
    ).rsound() {
      inputChild(workInProgress);
      class = value._dispatchInstances;
      workInProgressRenderedType = renderElements(payload, numFilter);
      if () => {
        return null;
      }
    }

    function App(pressList, _asuep))) {
      return (
```

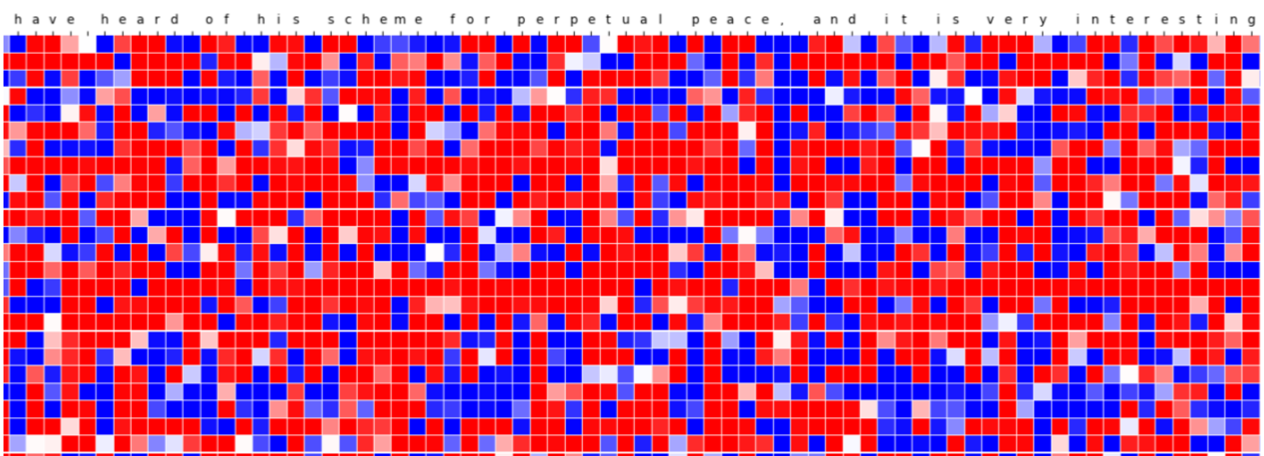
```
        <Bexpressing development ref={emptyFanstort?: boundDerivedSetIsFiberError()}
/>
        () =>
          Co(this.props.children).toHaveBeenCalled();

          expect(Scheduler).toFlushAndYield([
            'Entry with an useContext value foundDepveached app['center']',
          );
        },
      ),
    },
  [],
};
```

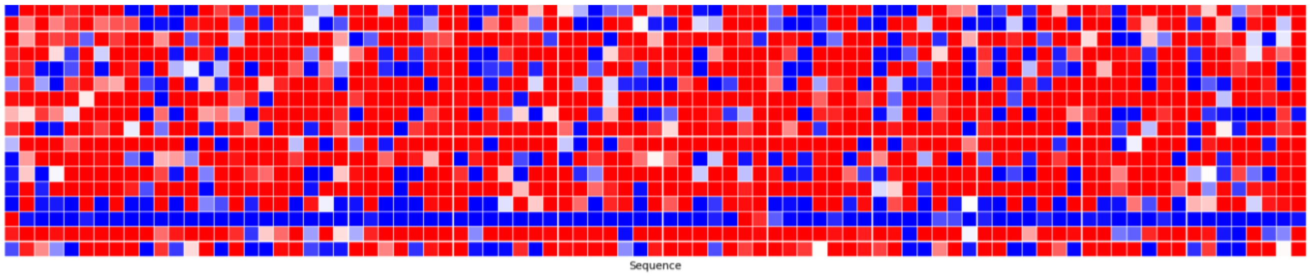
Interesting how it learned in JavaScript to close braces, close comments, arrow notation () => {}, import statements, and so on. It still feels like it will not be able to write functioning code like this though. It knows to declare variables, but not how to use them.

<u>Part III: Visualizing LSTM Gates</u>

2.



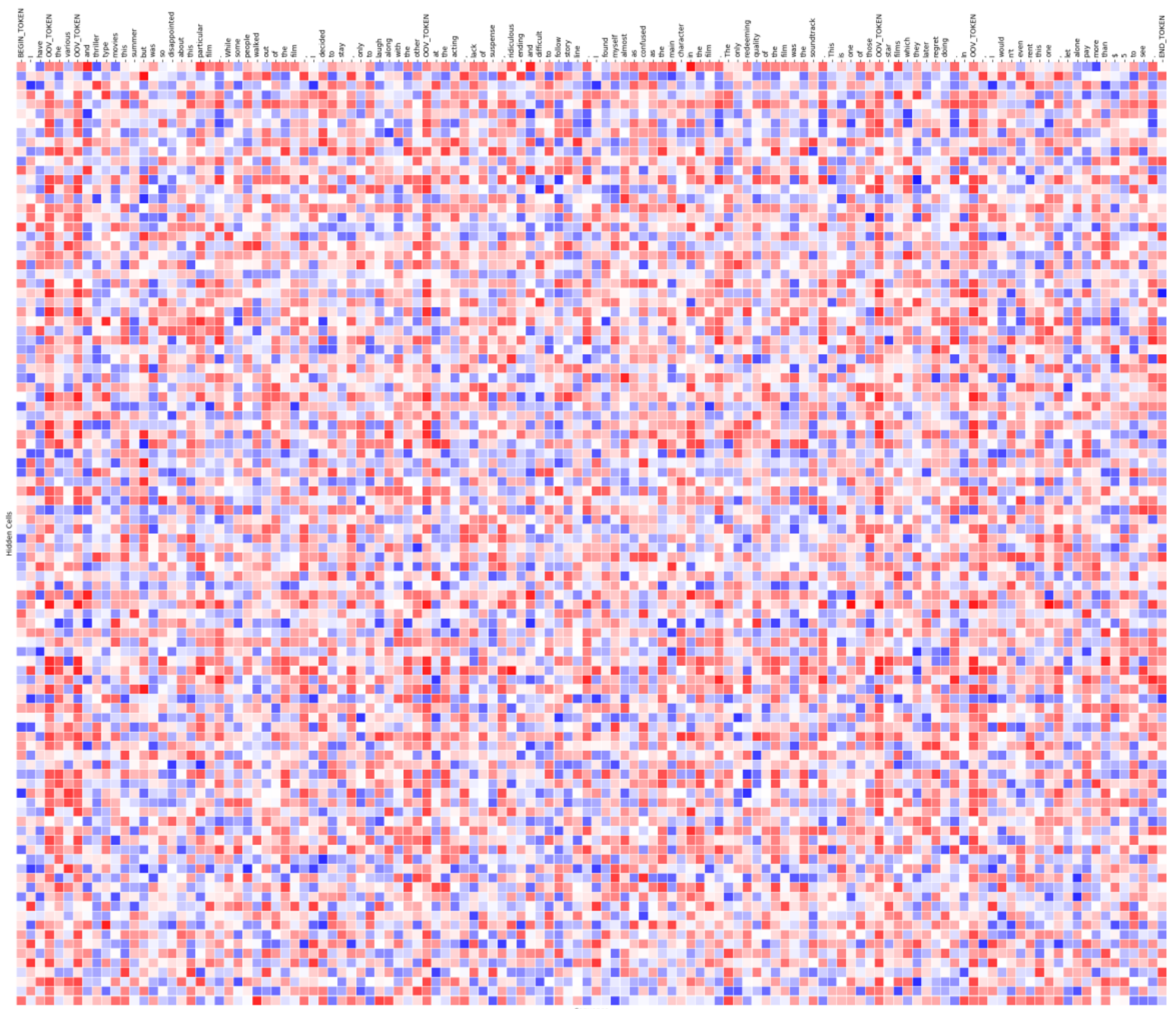
This is a snippet of the visualization from the reset signal. It seems like it resets on periods! That is cool. Also, on spaces it resets a little too it seems.



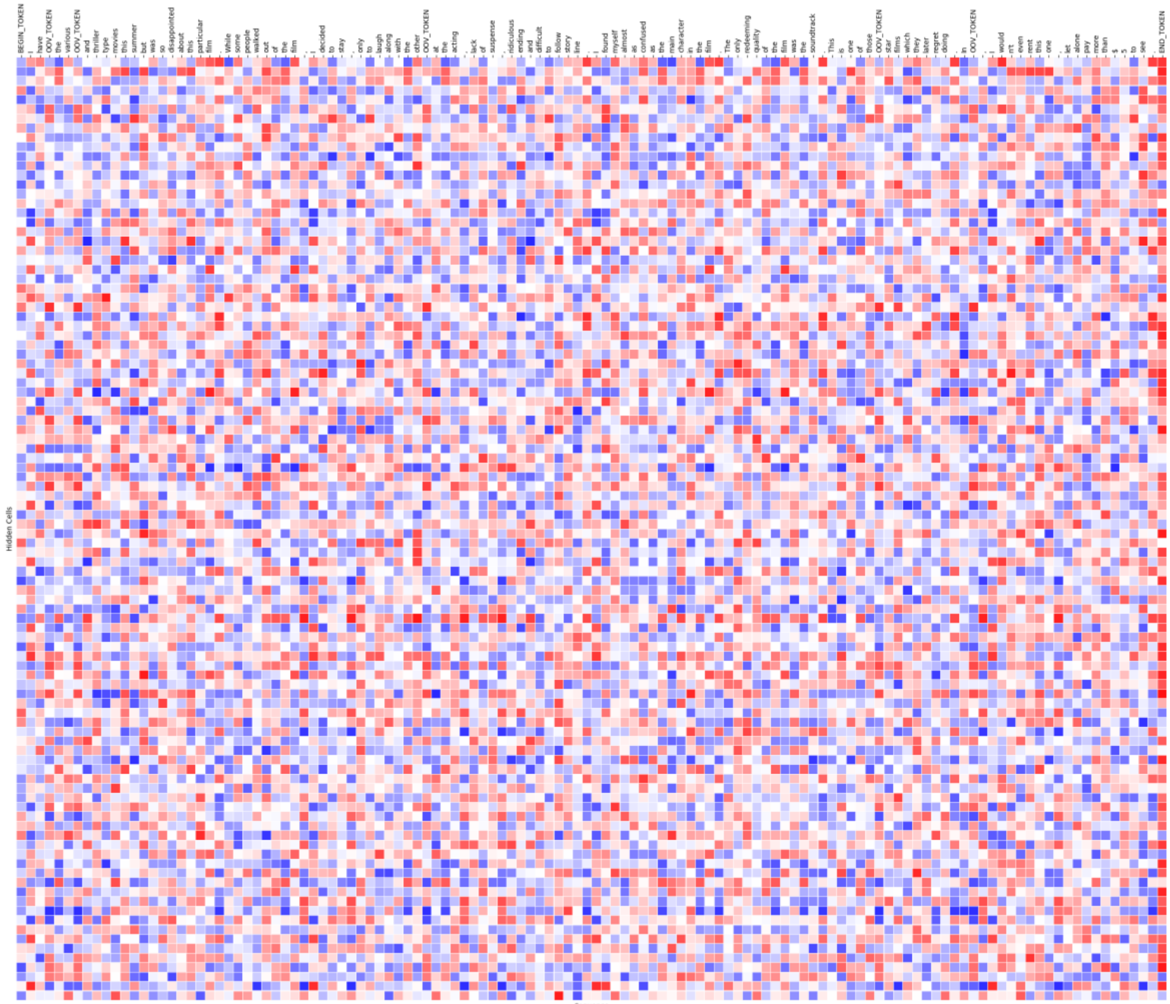
This is a segment of an update signal. It is interesting because the blue lines correspond to spots where the text is inside of a quotation.

Cell State candidate signals look fairly random. It seems like they are more blue as well.

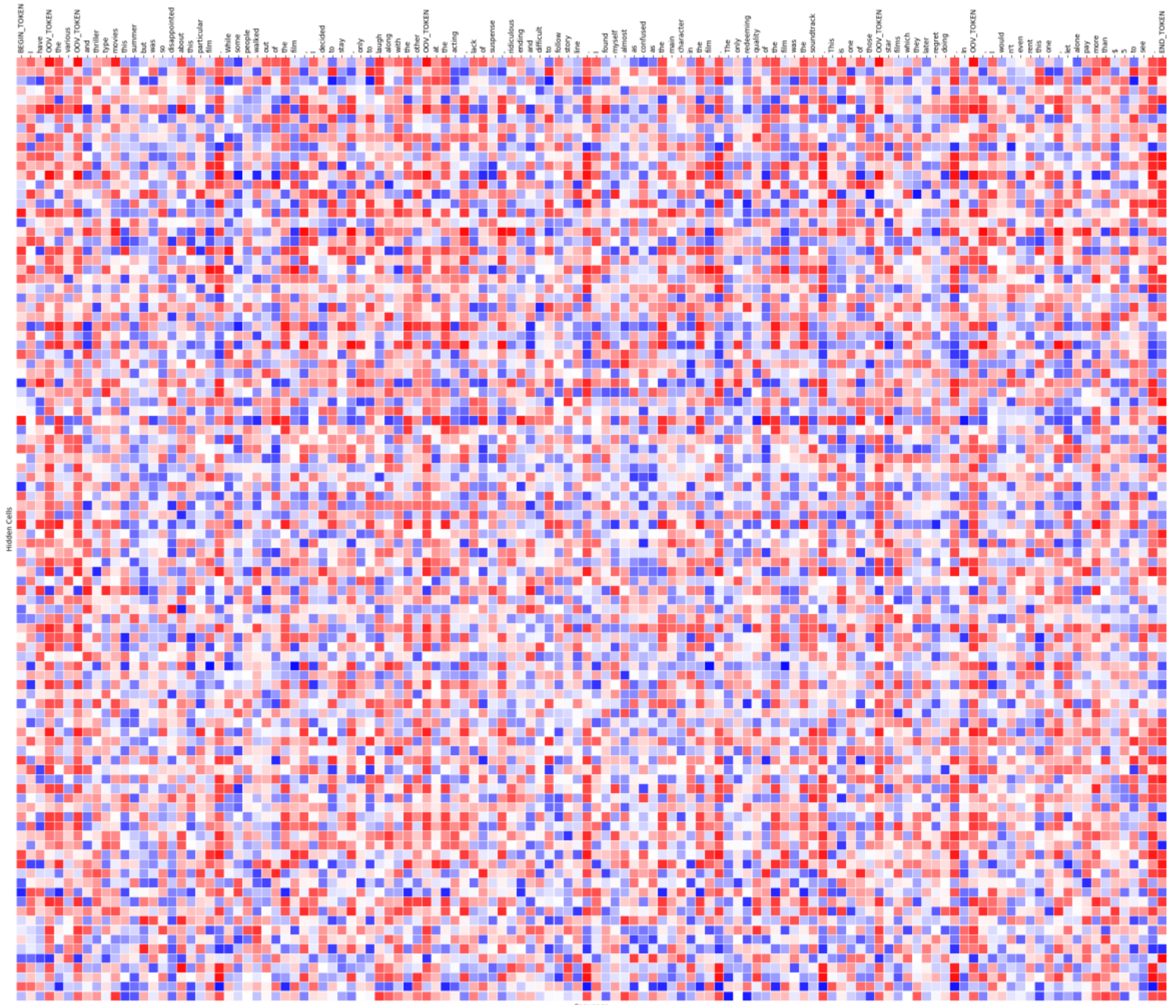
3.



This is the forget signal for a coupled LSTM. It seems pretty random. Out of vocab (OOV_TOKEN) seems to have cell activations be more red which is interesting. It is probably learning to ignore those.

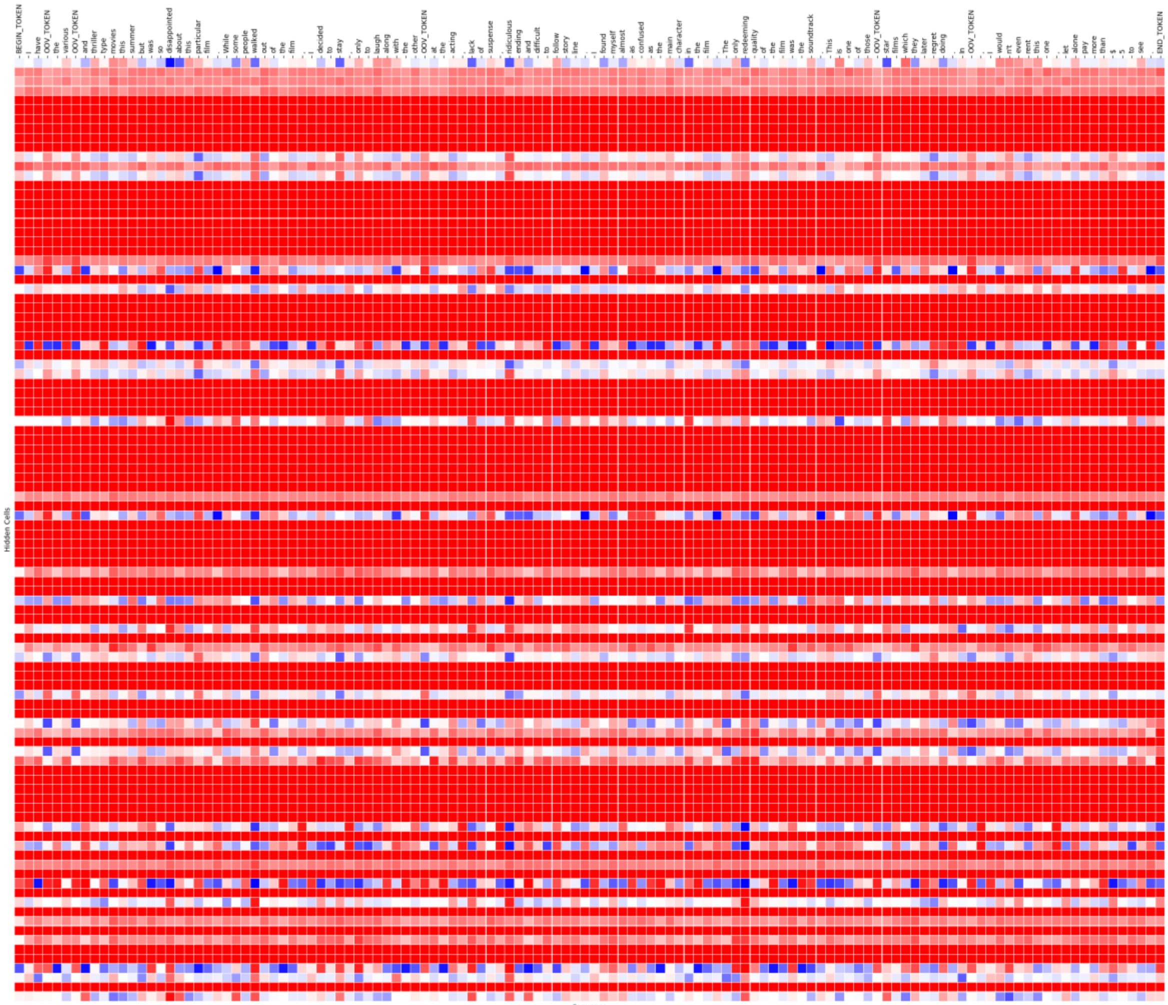


In this image that is the output signal for a coupled LSTM, it is clear the model learned what the END_TOKEN does (the last line).

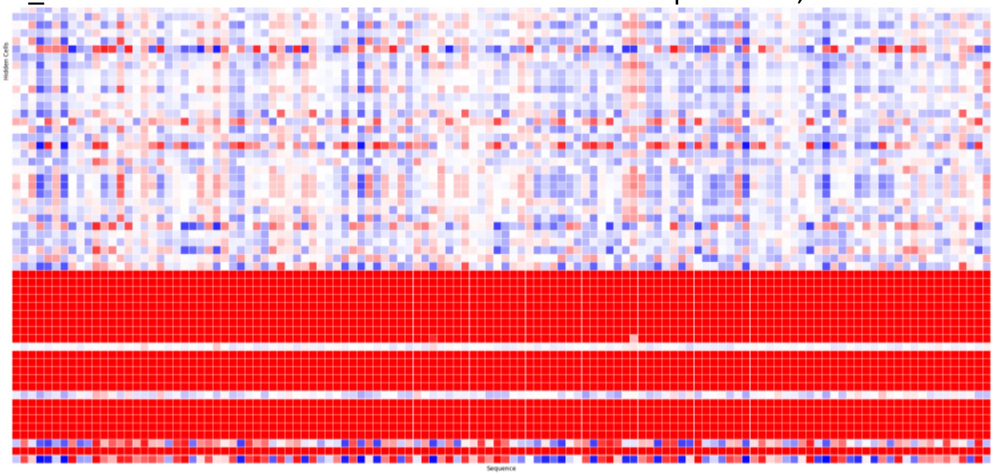


This is a forget state of a vanilla LSTM. It seems to be forgetting periods.

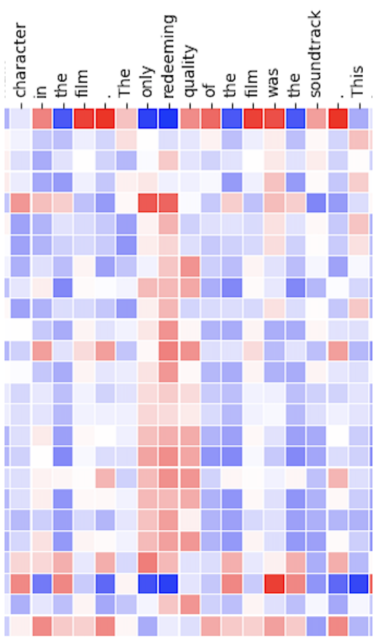
Below is the Peephole LSTM. I updated the vmin and vmax variables to change the Seaborn plot but got visualizations that look like this. It is interesting how this cell states candidates visualization has huge rows of red.



The forget signals are below. It is interesting how there are columns whenever there is an OOV_TOKEN. There are also rows of reds in different spots. And,



This is a zoomed in version below that shows the word “redeeming quality” seems to be learned as a trigger word by the model.



I have a lot more images, but it seemed silly to include them all in this report. A lot of hidden states seemed sort of random. But, in general, I was able to make out some patterns that made sense. For example, periods seem to reset, out of vocab tokens seemed to be most ignored.