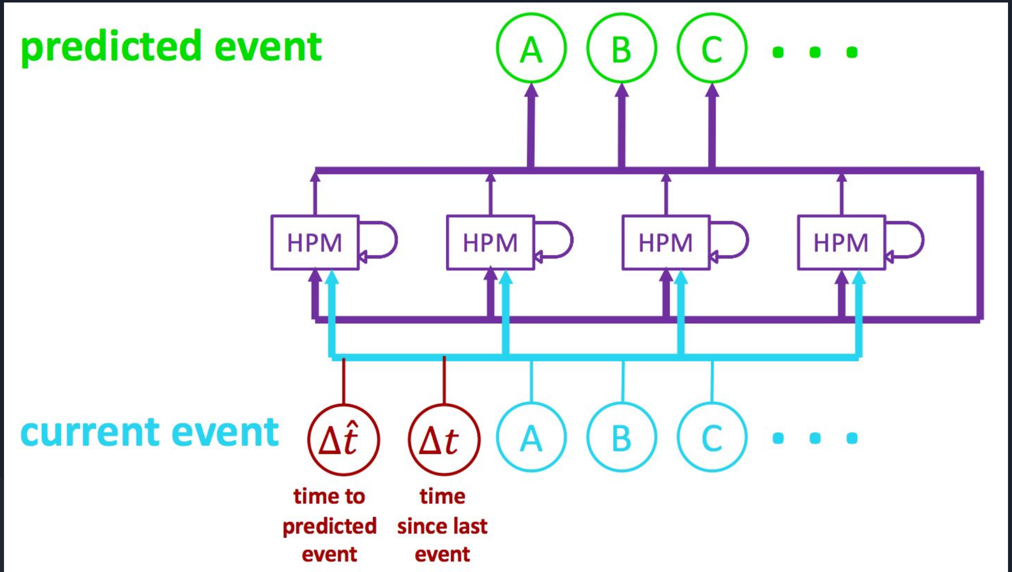
Continuous RNNs



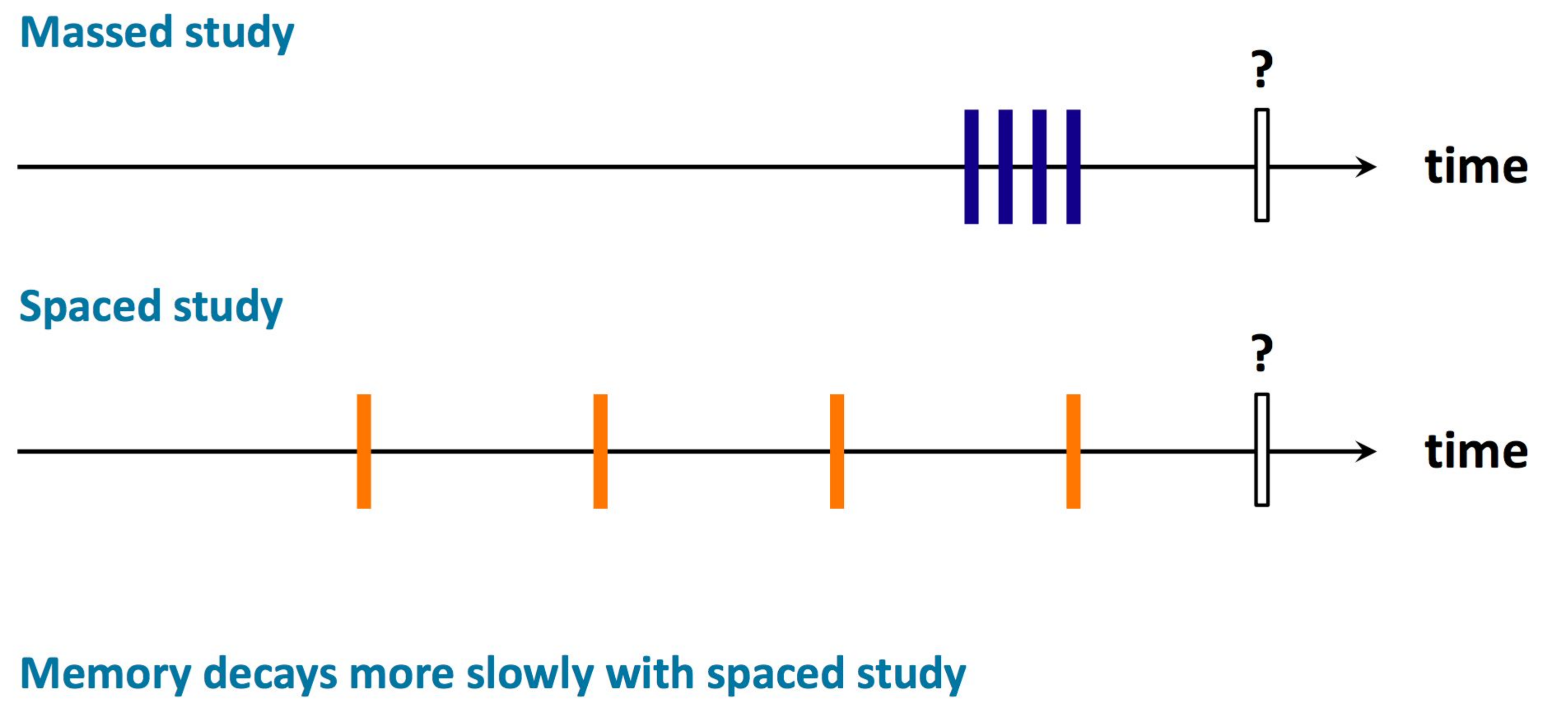
http://www.cs.colorado.edu/~mozer/Research/Selected%20Publications/reprints/MozerKazakovLindsey2017.pdf

Willie Boag
July 28, 2017

A Tour of This Talk

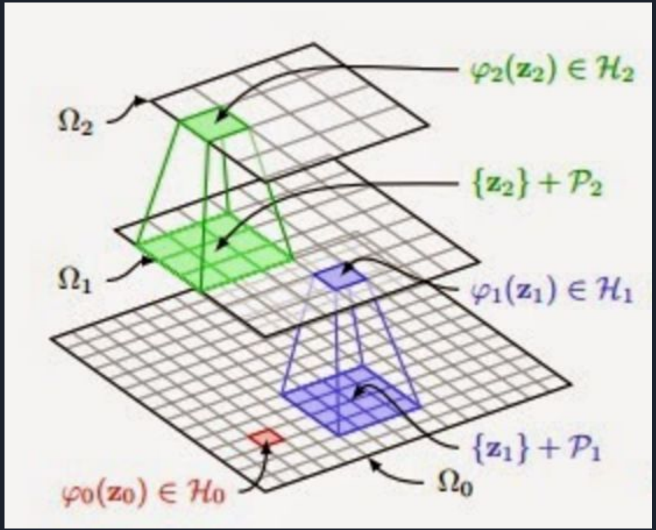
1. Background
   a. Domain-Appropriate Bias.
2. Model
   a. A new recurrent unit.
3. Experiments
   a. 6 synthetic datasets. 5 real datasets.
4. Conclusions
   a. Doesn’t beat standard GRU + time_features :(

Event Sequences



Depends on timing, not just ordering

Domain Appropriate Bias: ConvNets



CNN

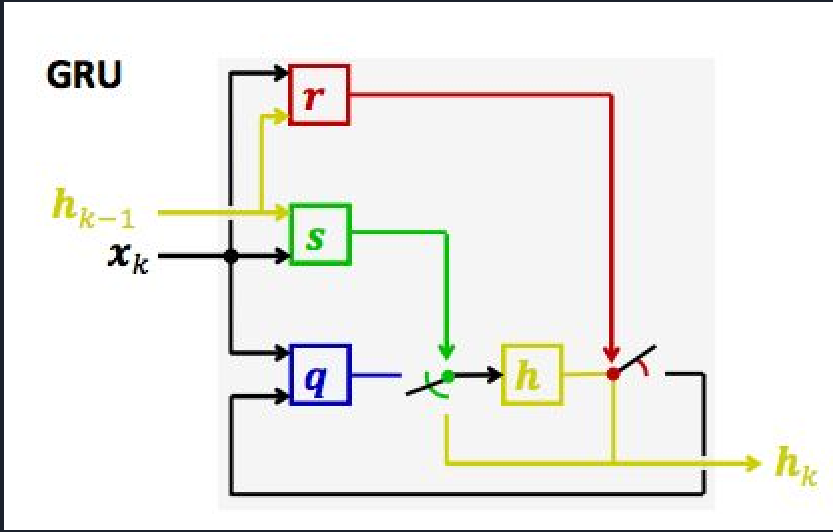
- Spacial Locality (x_i influences x_{i+1})
- Spacial Position Homogeneity (translation invariance)

Domain Appropriate Bias: Temporal Events

- Temporal Locality (event x_t influences event x_{t+1})
- Temporal Position Homogeneity (imagine a “burst” of events being invariant across days)
- Temporal Scale Homogeneity (locality & position homogeneity should apply across many scales)
- Temporal Scale Interactions (when events from varying scales affect one another)

GRU: Gated Recurrent Units (Review)

1. *Determine reset gate settings*	$\boldsymbol{r}_k \leftarrow \text{logistic}(\boldsymbol{W}^R \boldsymbol{x}_k + \boldsymbol{U}^R \boldsymbol{h}_{k-1} + \boldsymbol{b}^R)$
2. *Detect relevant event signals*	$\boldsymbol{q}_k \leftarrow \tanh(\boldsymbol{W}^Q \boldsymbol{x}_k + \boldsymbol{U}^Q (\boldsymbol{r}_k \circ \boldsymbol{h}_{k-1}) + \boldsymbol{b}^Q)$
3. *Determine update gate settings*	$\boldsymbol{s}_k \leftarrow \text{logistic}(\boldsymbol{W}^S \boldsymbol{x}_k + \boldsymbol{U}^S \boldsymbol{h}_{k-1} + \boldsymbol{b}^S)$
4. *Update hidden state*	$\boldsymbol{h}_k \leftarrow (1 - \boldsymbol{s}_k) \circ \boldsymbol{h}_{k-1} + \boldsymbol{s}_k \circ \boldsymbol{q}_k$



Refresher:

- r_k tells us how much to read from h_{k-1} memory
- h_{k-1} memory never decays

New perspective:

- it's like we threw away $(1-r_k)$ of the h_{k-1} memory
- Instead, imagine:

$r_k \odot h_{k-1} \odot 1$ (i.e. NEVER_DECAY)

+

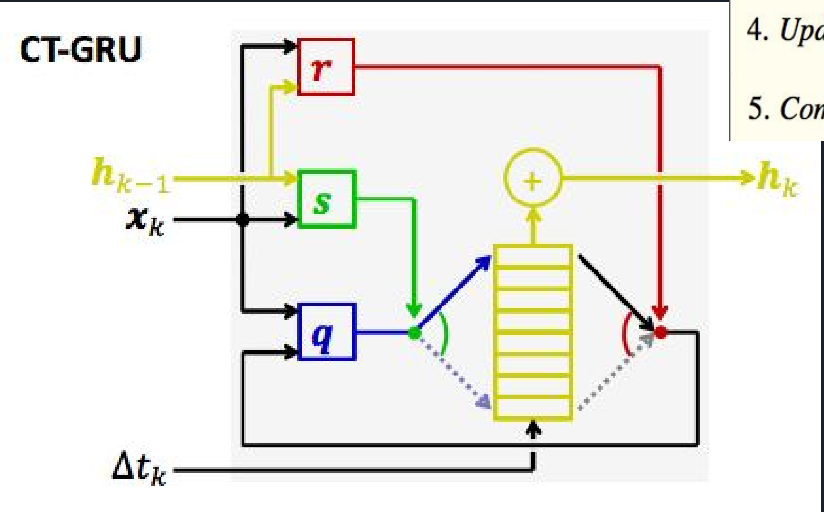
$(1-r)_k \odot h_{k-1} \odot 0$ (i.e. INSTANTLY_DECAY)

CT-GRU: Continuous-Time GRU

Note: $\tilde{\tau}_i$ is from a pre-defined timescale

$$\tilde{T} \equiv \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots, \tilde{\tau}_M\}$$

1. *Determine retrieval scale and weighting*	$\ln \boldsymbol{\tau}_k^R \leftarrow \boldsymbol{W}^R \boldsymbol{x}_k + \boldsymbol{U}^R \boldsymbol{h}_{k-1} + \boldsymbol{b}^R$ $\boldsymbol{r}_{ki} \leftarrow \mathrm{softmax}_i \left(-(\ln \boldsymbol{\tau}_k^R - \ln \tilde{\tau}_i)^2\right)$
2. *Detect relevant event signals*	$\boldsymbol{q}_k \leftarrow \tanh(\boldsymbol{W}^Q \boldsymbol{x}_k + \boldsymbol{U}^Q (\sum_i \boldsymbol{r}_{ki} \circ \hat{\boldsymbol{h}}_{k-1,i}) + \boldsymbol{b}^Q)$
3. *Determine storage scale and weighting*	$\ln \boldsymbol{\tau}_k^S \leftarrow \boldsymbol{W}^S \boldsymbol{x}_k + \boldsymbol{U}^S \boldsymbol{h}_{k-1} + \boldsymbol{b}^S$ $\boldsymbol{s}_{ki} \leftarrow \mathrm{softmax}_i \left(-(\ln \boldsymbol{\tau}_k^S - \ln \tilde{\tau}_i)^2\right)$
4. *Update multiscale state*	$\hat{\boldsymbol{h}}_{ki} \leftarrow \left[(1 - \boldsymbol{s}_{ki}) \circ \hat{\boldsymbol{h}}_{k-1,i} + \boldsymbol{s}_{ki} \circ \boldsymbol{q}_k\right] e^{-\Delta t_k / \tilde{\tau}_i}$
5. *Combine time scales*	$\boldsymbol{h}_k \leftarrow \sum_i \hat{\boldsymbol{h}}_{ki}$



1. basically predicts “what time scale should I read from in memory?” via softmax
2. Interpolates the histories according to predictions from 1
3. basically predicts “how much should I write to each time scale?” via softmax
4. Compute the new info & then decay it according to your particular time scale
5. The new state is the sum of the various frequencies

Intuition: Decomposing Into Time Scales

$$s_{ki} \leftarrow e^{-[\ln(\tilde{\tau}_i/\tau_k^S)]^2} / \sum_j e^{-[\ln(\tilde{\tau}_j/\tau_k^S)]^2}$$

This mixture decomposes $\boldsymbol{\tau}_k^S$ into the of time scales using:

- Half-lifes: break it into time scales
- Softmax: get the weights for the sum

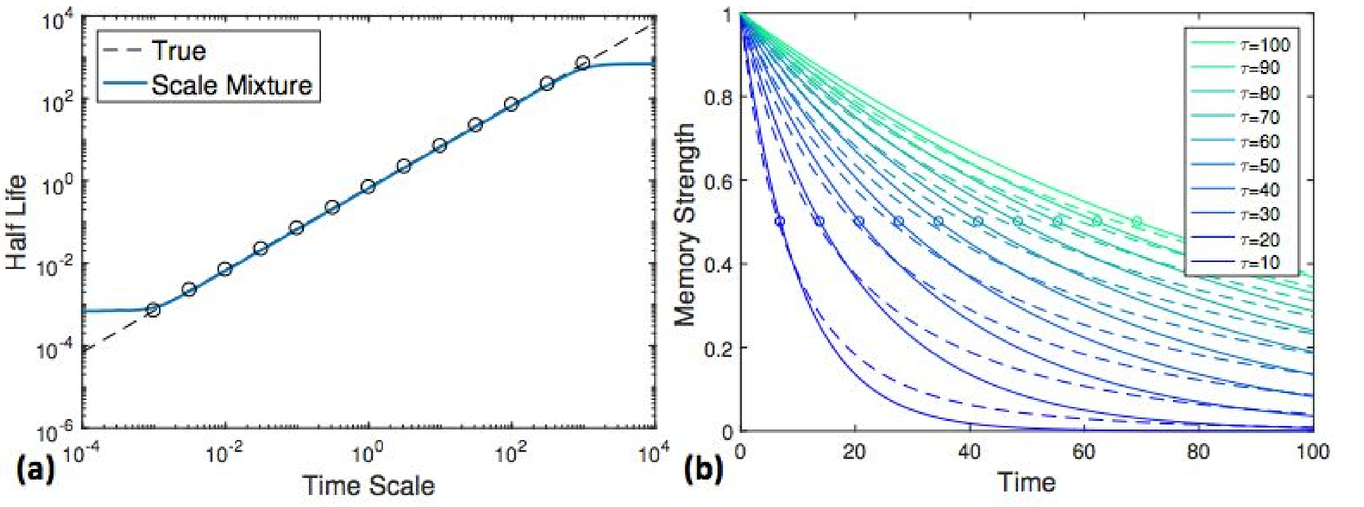


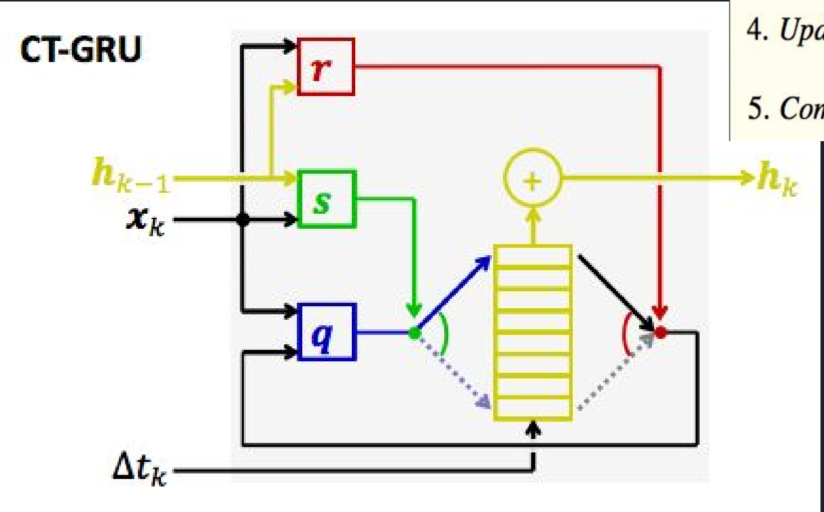
Figure 2: (a) Half life for a range of time scales: true value (dashed black line) and mixture approximation (blue line). (b) Decay curves for time scales $\tau \in [10, 100]$ (solid lines) and the mixture approximation (dashed lines).

CT-GRU: Continuous-Time GRU

Note: $\tilde{\tau}_i$ is from a pre-defined timescale

$$\tilde{T} \equiv \{\tilde{\tau}_1, \tilde{\tau}_2, \ldots \tilde{\tau}_M\}$$

1. *Determine retrieval scale and weighting*	$\ln \boldsymbol{\tau}_k^R \leftarrow \boldsymbol{W}^R \boldsymbol{x}_k + \boldsymbol{U}^R \boldsymbol{h}_{k-1} + \boldsymbol{b}^R$ $\boldsymbol{r}_{ki} \leftarrow \mathrm{softmax}_i \left(-(\ln \boldsymbol{\tau}_k^R - \ln \tilde{\tau}_i)^2\right)$
2. *Detect relevant event signals*	$\boldsymbol{q}_k \leftarrow \tanh(\boldsymbol{W}^Q \boldsymbol{x}_k + \boldsymbol{U}^Q (\sum_i \boldsymbol{r}_{ki} \circ \hat{\boldsymbol{h}}_{k-1,i}) + \boldsymbol{b}^Q)$
3. *Determine storage scale and weighting*	$\ln \boldsymbol{\tau}_k^S \leftarrow \boldsymbol{W}^S \boldsymbol{x}_k + \boldsymbol{U}^S \boldsymbol{h}_{k-1} + \boldsymbol{b}^S$ $\boldsymbol{s}_{ki} \leftarrow \mathrm{softmax}_i \left(-(\ln \boldsymbol{\tau}_k^S - \ln \tilde{\tau}_i)^2\right)$
4. *Update multiscale state*	$\hat{\boldsymbol{h}}_{ki} \leftarrow \left[(1 - \boldsymbol{s}_{ki}) \circ \hat{\boldsymbol{h}}_{k-1,i} + \boldsymbol{s}_{ki} \circ \boldsymbol{q}_k\right] e^{-\Delta t_k / \tilde{\tau}_i}$
5. *Combine time scales*	$\boldsymbol{h}_k \leftarrow \sum_i \hat{\boldsymbol{h}}_{ki}$



1. basically predicts “what time scale should I read from in memory?” via softmax
2. Interpolates the histories according to predictions from 1
3. basically predicts “how much should I write to each time scale?” via softmax
4. Compute the new info & then decay it according to your particular time scale
5. The new state is the sum of the various frequencies

Experiments

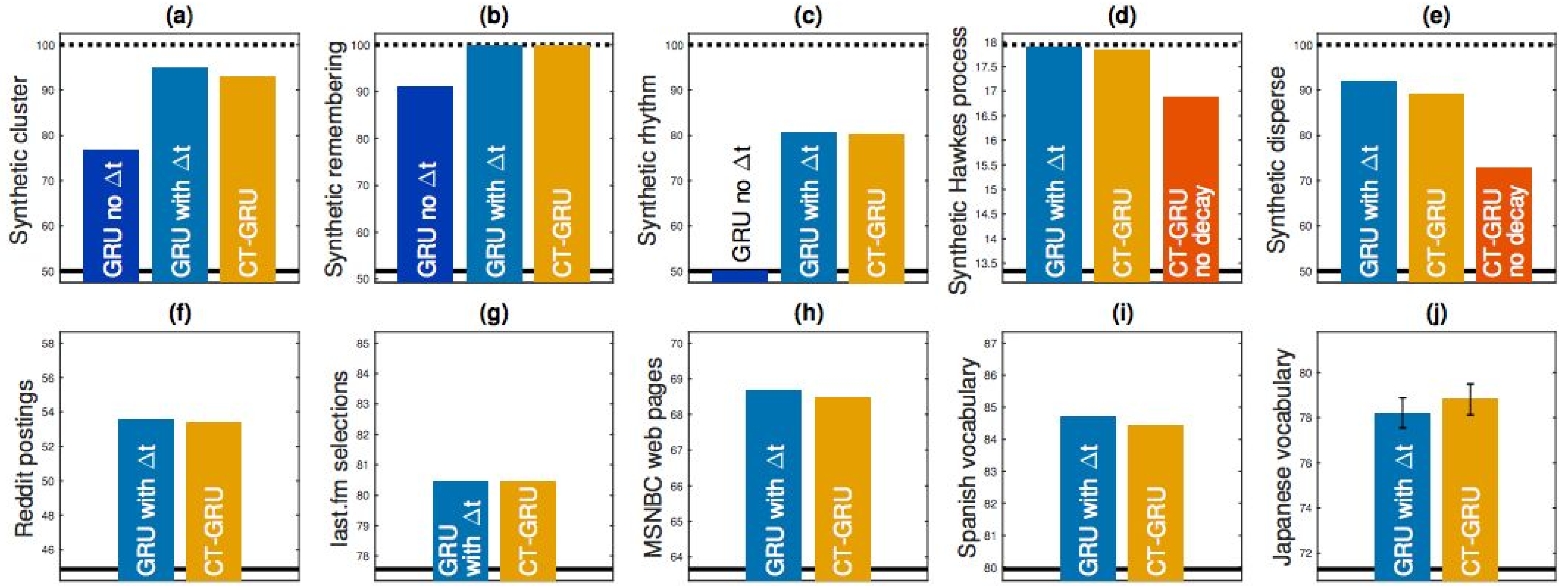


Figure 5: Comparison of GRU, CT-GRU, and variants. Data sets (a)-(i) consist of at least 10k training and test examples and thus a single train/test split is adequate for evaluation. Smaller data set (j) is tested via 8-fold cross validation. Solid black lines represent a reference baseline performance level, and dashed lines indicate optimal performance (where known).

Discussion

- First, we hoped that with smaller data sets, the value of the inductive bias in the CT-GRU would give it an advantage over the GRU, but it did not.
- Second, we tested other natural and synthetic data sets, but the pattern of results is as we report here.
- Third, we considered additional tasks that might reveal an advantage of the CT-GRU such as sequence extrapolation and event-timing prediction.
- And finally, we developed literally dozens of alternative neural net architectures that, like the CT-GRU, incorporate the forms of inductive bias described in the introduction that we expected to be helpful for event-sequence processing. Some of these models are easier to train than others, but, in the end, none beat the performance of generic LSTM or GRU architectures provided with additional delta_t inputs.

Conclusions

- Despite the thoughtful motivation, interesting novelty, and meticulous experiments, the model performs basically identically to the GRU with time-based features.
- CT-GRU and GRU-with-time both outperformed vanilla-GRU, which means the time information WAS being utilized.
- Interestingly, CT-GRU and GRU-with-time even produced nearly identical predictions (when one made an error, the other made the same error).

- My Favorite Line: “ We also note, somewhat cynically, that a large fraction of the novel architectures that are claimed to yield promising results one year seem to fall by the wayside a year later.”