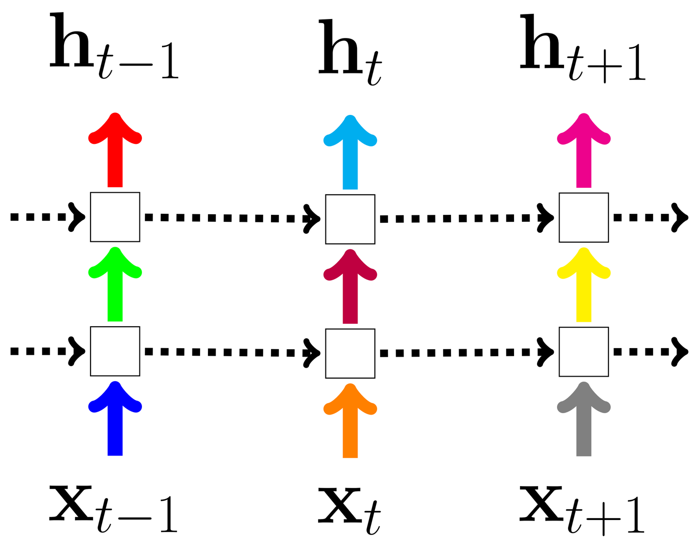
Dropout in Recurrent Neural Networks

A Theoretically Grounded Dropout Variant in RNNs using Variational Inference

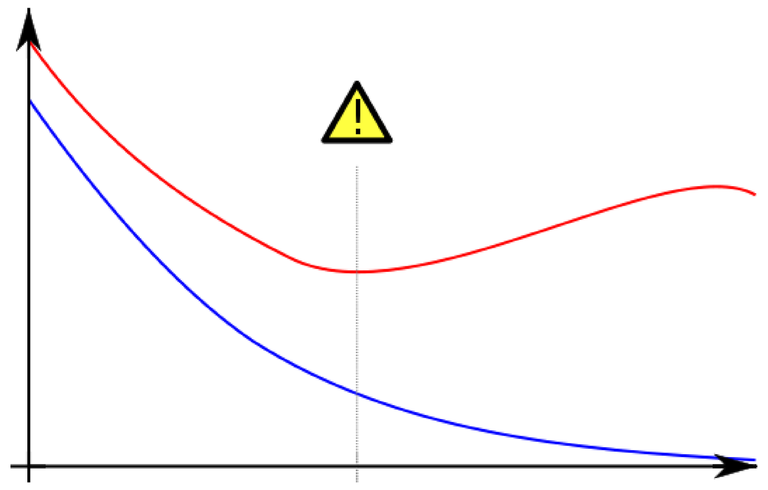
Yarin Gal yg279@cam.ac.uk

UNIVERSITY OF CAMBRIDGE

RNNs overfit quickly



RNNs are **awesome**;



but also **overfit** very quickly.

This means...

- We can't use **large** models
- We have to use **early stopping**
- We can't use **small data**
- We have to **waste data** for validation sets

Existing dropout in RNNs

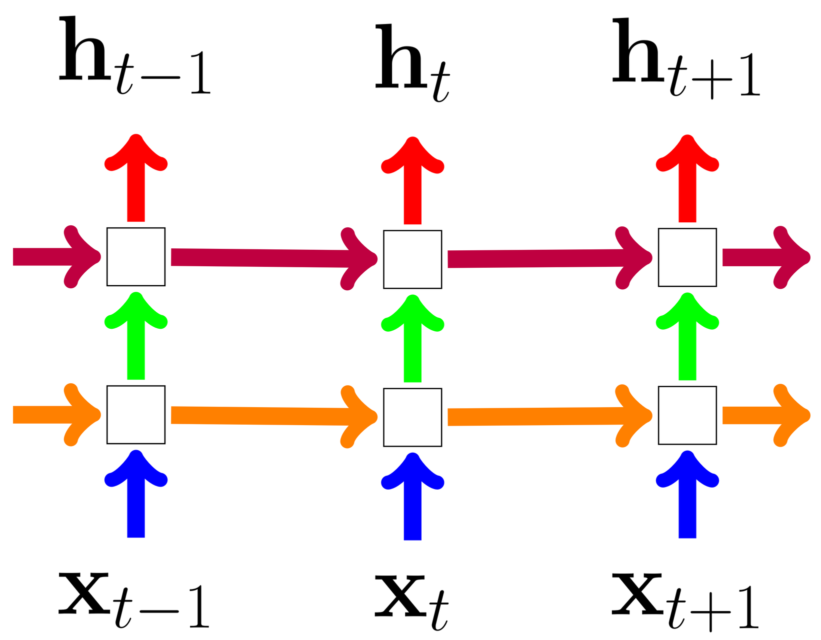
Let's **use dropout** then. But lots of research has claimed that that's a **bad idea**:

- **Pachitariu & Sahani**, 2013
  - noise added in the recurrent connections of an RNN leads to model **instabilities**
- **Bayer et al.**, 2013
  - with dropout, the RNN's **dynamics change** dramatically
- **Pham et al.**, 2014
  - dropout in recurrent layers **disrupts** the RNN's ability to model sequences
- **Zaremba et al.**, 2014
  - applying dropout to the **non-recurrent connections alone** results in improved performance
- **Bluche et al.**, 2015
  - **exploratory analysis** of the performance of dropout before, inside, and after the RNN's
- **Moon et al.**, 2015
  - Drop elements in the **LSTM's cell** using the same mask at every time step.

Many settled on using dropout for **inputs and outputs alone**.

VI based dropout in RNNs

Uses the **same dropout mask** at **each time step**, including recurrent layers, and **drops word types** at random throughout the sentence:



Why does it make sense?

- **Input**: sequence of vectors $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ with T time steps
- Let $\boldsymbol{\omega} = \{$**all** model weight matrices$\}$ and put prior $p(\boldsymbol{\omega})$ (e.g. standard Gaussian)
- Define $\mathbf{h}_t = \mathbf{f}^{\boldsymbol{\omega}}_{\mathbf{h}}(\mathbf{x}_t, \mathbf{h}_{t-1})$
  - single **recurrent unit** transition. E.g. $\tanh(W\mathbf{x}_t + U\mathbf{h}_{t-1} + b)$ (similarly for **LSTM, GRU**)
- Set $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}) = \mathbf{f}^{\boldsymbol{\omega}}_{\mathbf{y}}(\mathbf{h}_T)$
  - model **output** (e.g. affine transformation of last state, or function of all states)
- Lastly, define $p(\mathbf{y}|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}))$
  - model **likelihood** on random function output $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x})$. E.g. $\mathcal{N}(\mathbf{y}; \mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}), \sigma^2)$

- **Variational interpretation of dropout** [Gal and Ghahramani, 2015]:
  Dropout objective minimises

$$\mathrm{KL}\big(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{X},\mathbf{Y})\big) \propto -\int q(\boldsymbol{\omega}) \log p(\mathbf{Y}|\mathbf{X},\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega}))$$
$$= -\sum_{i=1}^{N} \int q(\boldsymbol{\omega}) \log p(\mathbf{y}_i|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}_i)) \mathrm{d}\boldsymbol{\omega} + \mathrm{KL}(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})).$$

with $q(\boldsymbol{\omega})$ factorising over weight columns $\mathbf{w}_{ik}$, e.g. $q(\mathbf{w}_{ik}) = p\delta_0 + (1-p)\delta_{\mathbf{m}_{ik}}$.

- But

$$\int q(\boldsymbol{\omega}) \log p(\mathbf{y}|\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x})) \mathrm{d}\boldsymbol{\omega} = \int q(\boldsymbol{\omega}) \log p\left(\mathbf{y}\middle|\mathbf{f}^{\boldsymbol{\omega}}_{\mathbf{y}}\big(\mathbf{f}^{\boldsymbol{\omega}}_{\mathbf{h}}(\mathbf{x}_T, ...\mathbf{f}^{\boldsymbol{\omega}}_{\mathbf{h}}(\mathbf{x}_1, \mathbf{h}_0)...)\big)\right) \mathrm{d}\boldsymbol{\omega},$$

- So using MC integration with $\widehat{\boldsymbol{\omega}}_i \sim q(\boldsymbol{\omega})$,

$$\mathcal{L}_{VI} \approx -\sum_{i=1}^{N} \log p\left(\mathbf{y}_i\middle|\mathbf{f}^{\widehat{\boldsymbol{\omega}}_i}_{\mathbf{y}}\big(\mathbf{f}^{\widehat{\boldsymbol{\omega}}_i}_{\mathbf{h}}(\mathbf{x}_{iT}, ...\mathbf{f}^{\widehat{\boldsymbol{\omega}}_i}_{\mathbf{h}}(\mathbf{x}_{i1}, \mathbf{h}_0)...)\big)\right) + \mathrm{KL}\big(q_\theta(\boldsymbol{\omega}) \mid\mid p(\boldsymbol{\omega})\big).$$

$$=$$

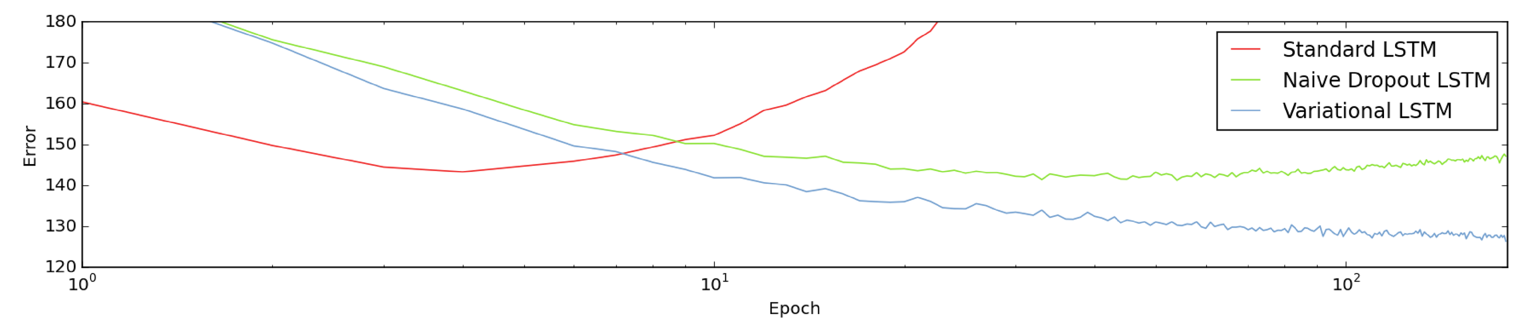
using random mask to set weight columns to zero (dropping units), repeating **the same mask at each time step** for **all weight matrices** (including embedding layer)

Results

- **Penn Treebank** language modelling

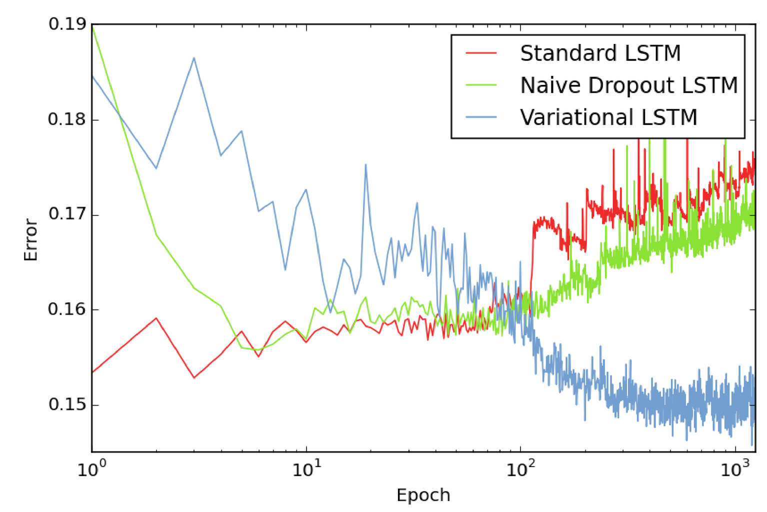
	Medium LSTM			Large LSTM		
	Validation	Test	WPS	Validation	Test	WPS
Non-regularized (early stopping)	121.1	121.7	5.5K	128.3	127.4	2.5K
Moon et al. [2015]	100.7	97.0	4.8K	122.9	118.7	3K
Moon et al. [2015] +emb dropout	88.9	86.5	4.8K	88.8	86.0	3K
Zaremba et al. [2014]	86.2	82.7	5.5K	82.2	78.4	2.5K
Variational (tied weights)	81.8 ± 0.2	79.7 ± 0.1	4.7K	77.3 ± 0.2	75.0 ± 0.1	2.4K
Variational (tied weights, MC)	–	79.0 ± 0.1	–	–	74.1 ± 0.0	–
Variational (untied weights)	81.9 ± 0.2	79.7 ± 0.1	2.7K	77.9 ± 0.3	75.2 ± 0.2	1.6K
Variational (untied weights, MC)	–	$\mathbf{78.6 \pm 0.1}$	–	–	$\mathbf{73.4 \pm 0.0}$	–

Single model perplexity (on test and validation sets). Two LSTM sizes are compared using Zaremba, Sutskever, and Vinyals [2014]'s setup.

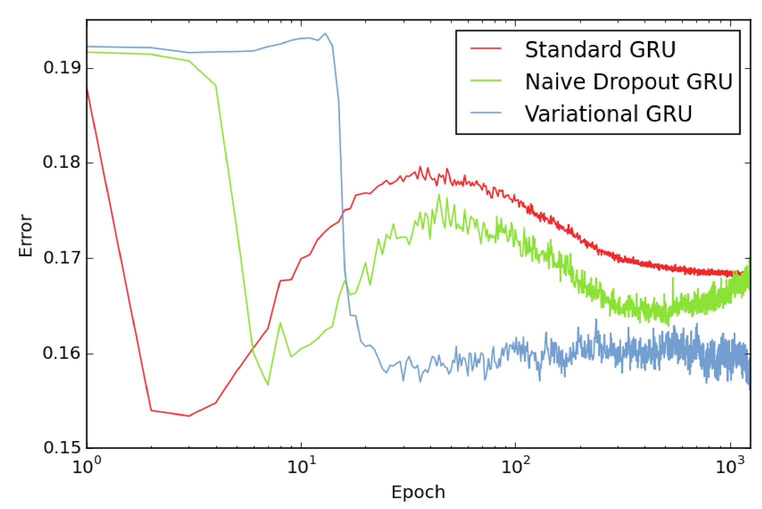


Validation perplexity (medium model) with dropout regularisation alone

- **Sentiment analysis** (raw Cornell film reviews corpus, Pang and Lee [2005])

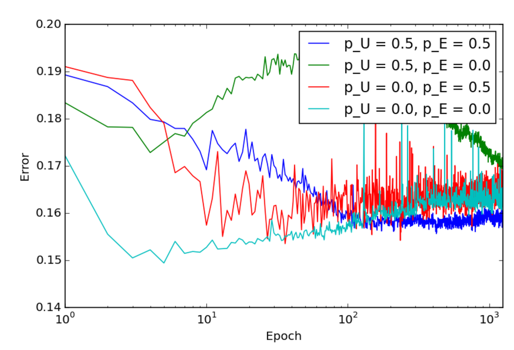


LSTM test error

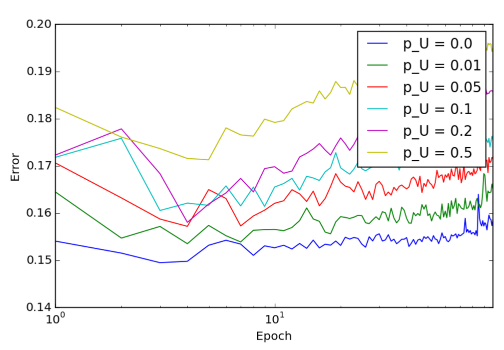


GRU test error

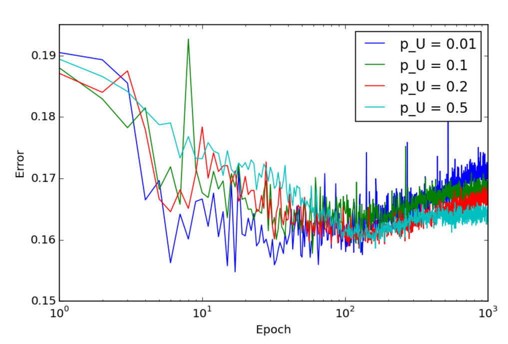
Different dropout probabilities used with the recurrent layer (p_U) and embedding layer (p_E):



$p_E = 0, 0.5 \times p_U = 0, 0.5$



$p_U = 0, ..., 0.5$ & $p_E = 0$



$p_U = 0, ..., 0.5$ & $p_E = 0.5$