# Chapter 10 - Part II

## Sequence Modeling: Recurrent and Recursive Nets

March
Brag
cesarbrma91@gmail.com
@MarchBragagnini

Universidad Católica
San Pablo

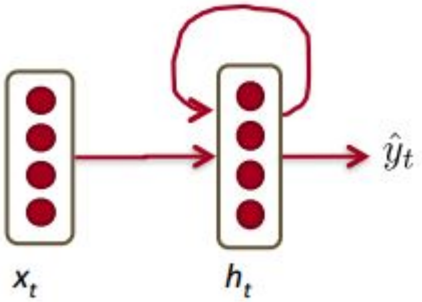
# Outline

1. The Challenge of Long-Term Dependencies.
2. Echo State Networks.
3. Leaky Units and Other Strategies for Multiple Time Scales
4. Gated RNNs.
5. Optimization for Long-Term Dependencies.
6. Explicit Memory.
7. Code.
8. References.

# The Challenge of Long-Term Dependencies

**Remembering Vanishing/Exploding gradient**

$$\begin{aligned} h_t &= Wf(h_{t-1}) + W^{(hx)}x_{[t]} \\ \hat{y}_t &= W^{(S)}f(h_t) \end{aligned}$$



$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_k}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\boxed{\frac{\partial h_t}{\partial h_k}}\frac{\partial h_k}{\partial W}$$

$$\frac{\partial h_t}{\partial h_k} = \boxed{\prod_{j=k+1}^{t}\frac{\partial h_j}{\partial h_{j-1}}}$$

[ ] CS224N/Ling284, Lecture 8, Stanford University, Manning , 2016.

# The Challenge of Long-Term Dependencies

## Remembering Vanishing/Exploding gradient

Each partial is a Jacobian

$$\frac{d\mathbf{f}}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Analyzing the norms of the Jacobians, yields:

$$\left\| \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq \|W^T\| \|\mathrm{diag}[f'(h_{j-1})]\| \leq \beta_W \beta_h$$

Where we defined $\beta$‘s as upper bounds of the norms

The gradient is a product of Jacobian matrices, each associated with a step in the forward computation.

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right\| \leq (\beta_W \beta_h)^{t-k}$$

This can become very small or very large quickly

[ ] CS224N/Ling284, Lecture 8, Stanford University, Manning , 2016.
[ ] Learning Long-Term Dependencies with Gradient Descent is Difficult, Bengio, 1994.

# The Challenge of Long-Term Dependencies

## Summary

The basic problem is that gradients propagated over many stages tend to either vanish (most of the time) or explode (rarely, but with much damage to the optimization)

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^{\top}\boldsymbol{h}^{(t-1)}$$

$$\boldsymbol{h}^{(t)} = \left(\boldsymbol{W}^{t}\right)^{\top}\boldsymbol{h}^{(0)}, \quad \boldsymbol{W} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\top}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{Q}^{\top}\boldsymbol{\Lambda}^{t}\boldsymbol{Q}\boldsymbol{h}^{(0)}$$

The eigenvalues are raised to the power of ***t***, causing eigenvalues with magnitude less than one to decay to zero and eigenvalues with magnitude greater than one to explode.

Any component of ***h(0)*** that is not aligned with the largest eigenvector will eventually be discarded

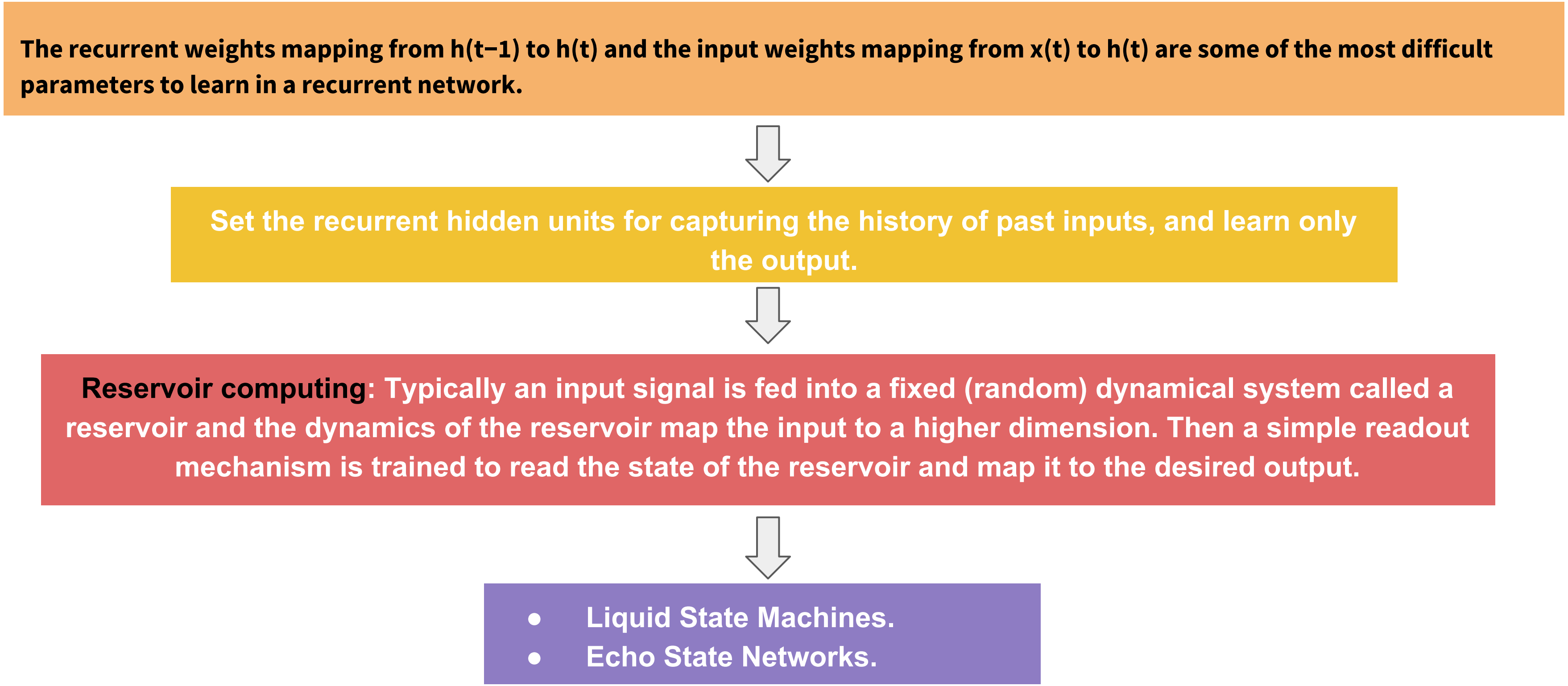
**Problem: Long Term Dependencies, information whose outputs depend on very early entries can not be remembered.**

**Input: Teo es la mascota de Lari, es un perrito pequeño y bien cariñoso, siempre sale a pasear todos los domingos, le gusta ladrar a la gente y a las perritas. A su mascota no le gustan las chalinas**

**Can RNN predict the missing word? : La mascota de Lari se llama ____**

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

# Echo State Network



**The recurrent weights mapping from h(t−1) to h(t) and the input weights mapping from x(t) to h(t) are some of the most difficult parameters to learn in a recurrent network.**

**Set the recurrent hidden units for capturing the history of past inputs, and learn only the output.**

**Reservoir computing: Typically an input signal is fed into a fixed (random) dynamical system called a reservoir and the dynamics of the reservoir map the input to a higher dimension. Then a simple readout mechanism is trained to read the state of the reservoir and map it to the desired output.**

- **Liquid State Machines.**
- **Echo State Networks.**

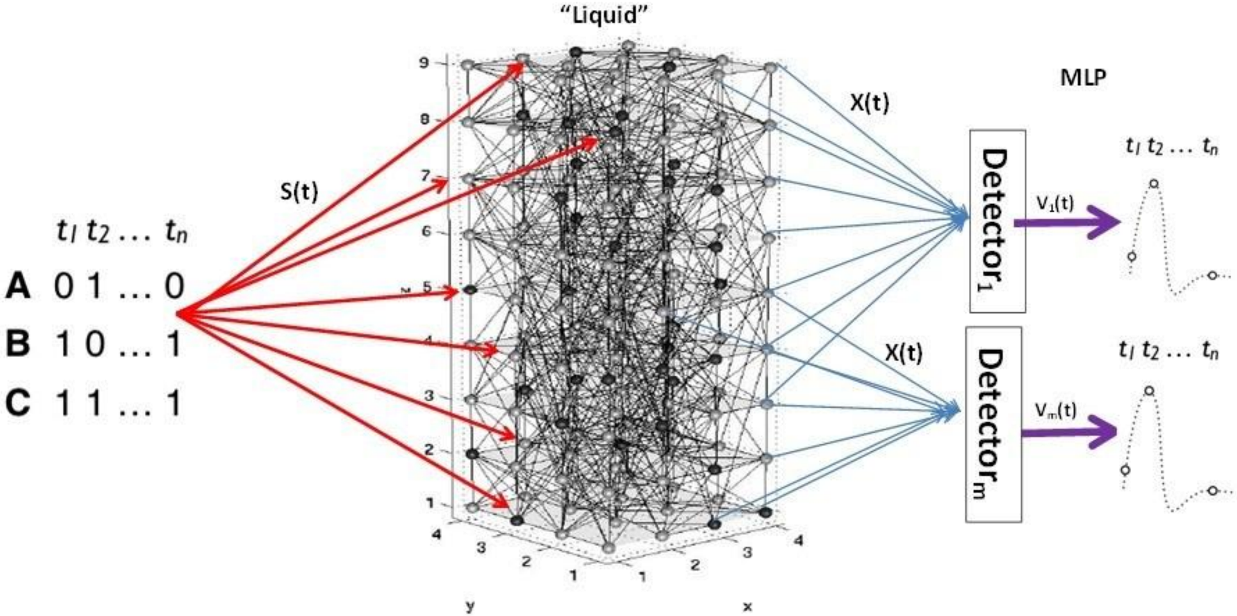
[ ] Understanding LSTM Networks, Colab Blog, 2015.
[ ] Exploring LSTMs, Edwin Chen, 2016.
[ ] Reservoir computing, wikipedia.

# Echo State Network

## Liquid State Machine

A liquid state machine (LSM) consists of a large collection neurons. Each node receives time varying input from external sources (the inputs) as well as from other nodes. Nodes are randomly connected to each other. The recurrent nature of the connections turns the time varying input into a spatio-temporal patterns. These spatio-temporal patterns are learned.

Liquid ~ pond water

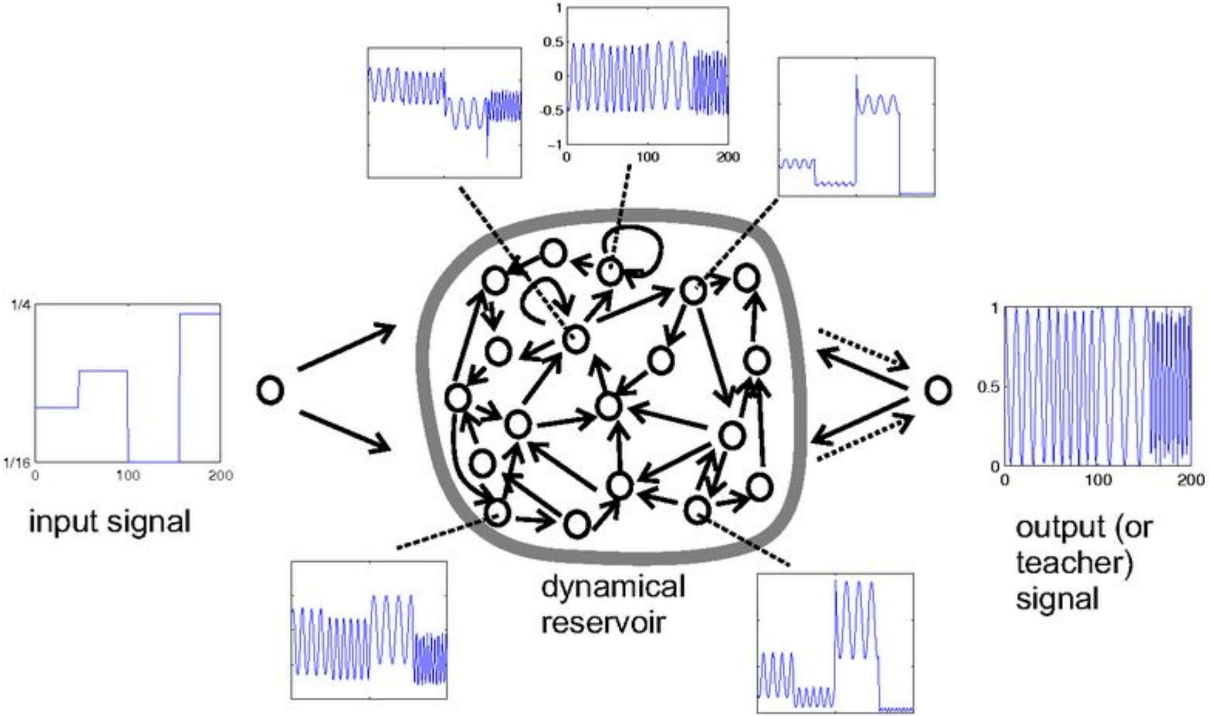


Model of Liquid State Machine.

[ ] The Liquid State Machine (LSM), Hananel Blog.
[ ] Real-time computing without stable states: a new framework for neural computation based on perturbations, Maass, 2002.

# Echo State Network

## Echo State Network



Output feedback

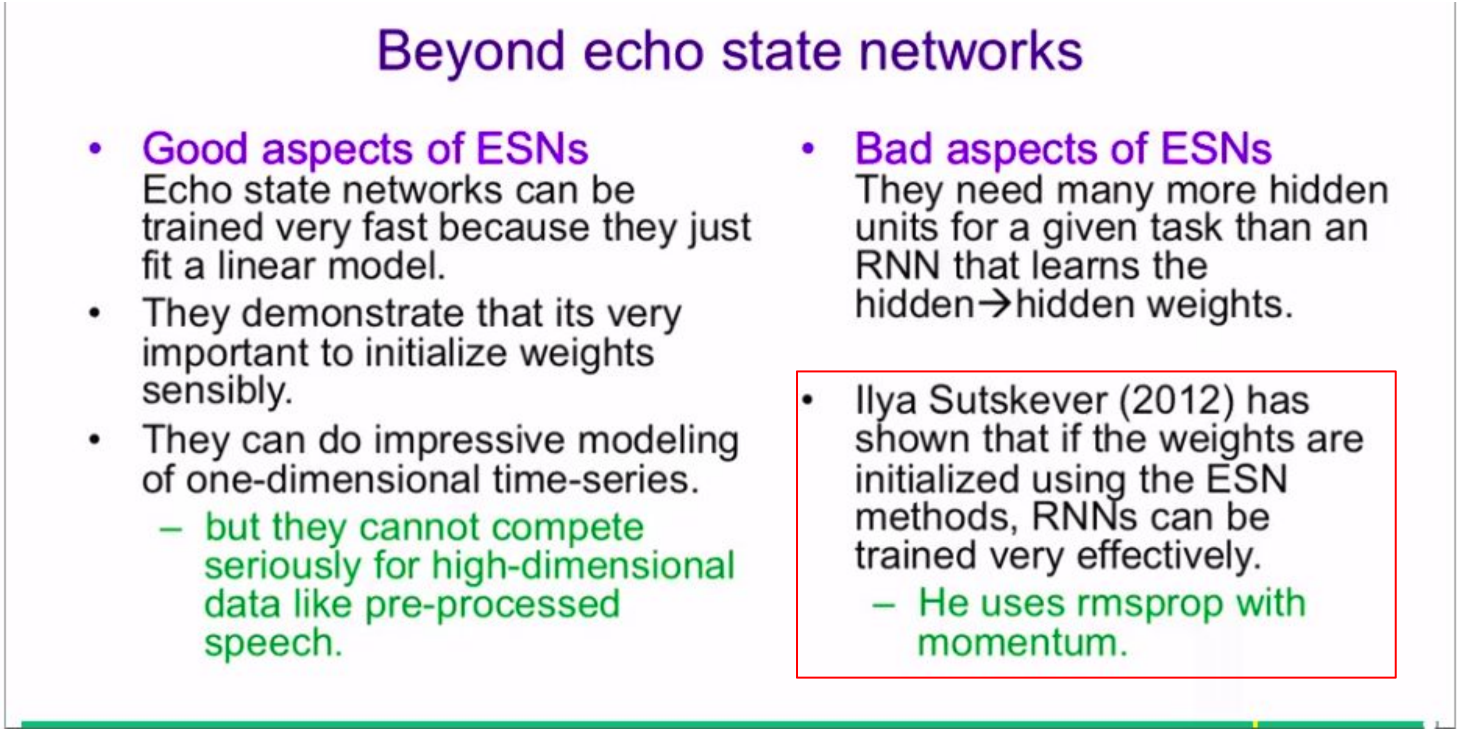
Only it train the output weights.

Model of Echo State Network.

[ ] Echo State Networks, Scholarpedia, 2007.

# Echo State Network

## Echo State Network



### Beyond echo state networks

- Good aspects of ESNs
  Echo state networks can be trained very fast because they just fit a linear model.
- They demonstrate that its very important to initialize weights sensibly.
- They can do impressive modeling of one-dimensional time-series.
  – but they cannot compete seriously for high-dimensional data like pre-processed speech.

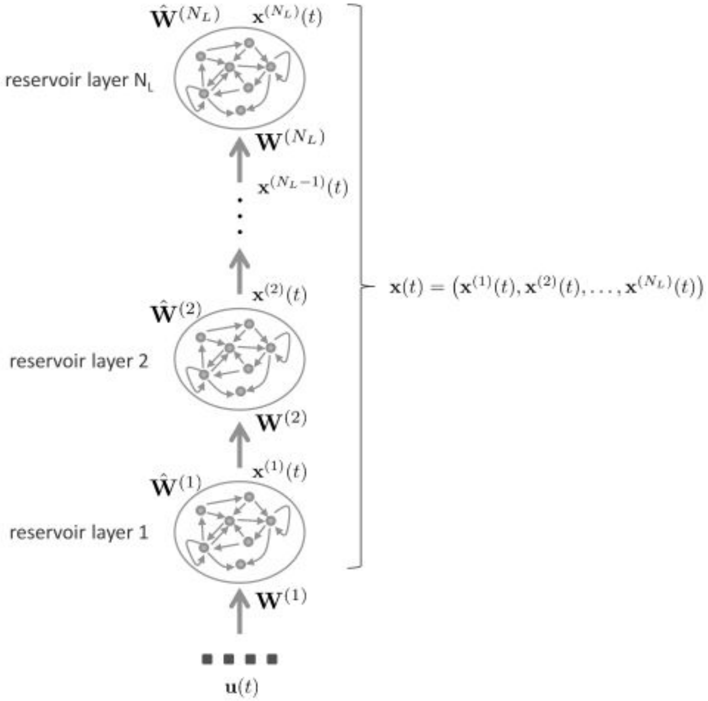
- Bad aspects of ESNs
  They need many more hidden units for a given task than an RNN that learns the hidden→hidden weights.

- Ilya Sutskever (2012) has shown that if the weights are initialized using the ESN methods, RNNs can be trained very effectively.
  – He uses rmsprop with momentum.

[ ] Echo State Networks, Neural Networks for Machine Learning, Video Lecture, Coursera.

# Echo State Network

## Deep Echo State Network

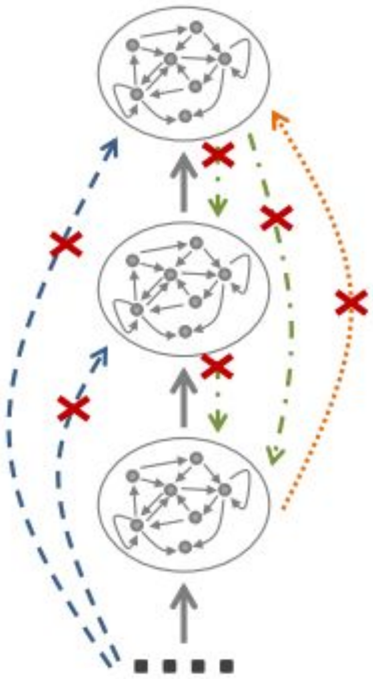


Model of Deep Echo State Network.

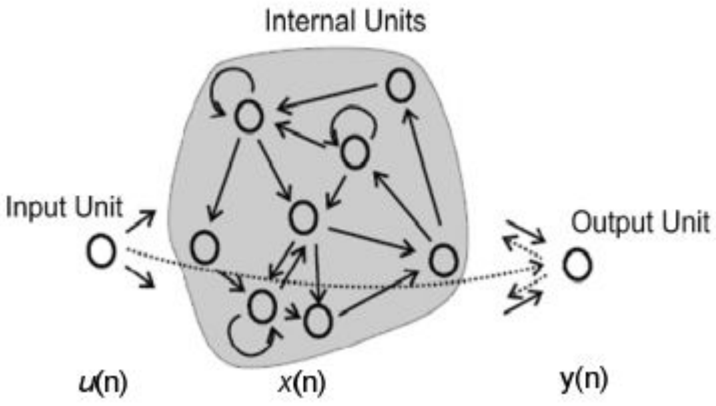
[ ] Deep Echo State Network (DeepESN): A Brief Survey, Gallicchio, 2018.
[ ] An Introduction to the Echo State Network and its Applications in Power System, Dai1, 2009.

# Echo State Network

## Deep ESN vs Shallow ESN



Model of Deep Echo State Network.



Model of Shallow Echo State Network.

[ ] Deep Echo State Network (DeepESN): A Brief Survey, Gallicchio, 2018.
[ ] An Introduction to the Echo State Network and its Applications in Power System, Dai1, 2009.

# Leaky Units and Other Strategies for Multiple Time Scales

## Adding Skip Connections/Residual Connections through Time

Gradients may vanish or explode exponentially with respect to the number of time steps.

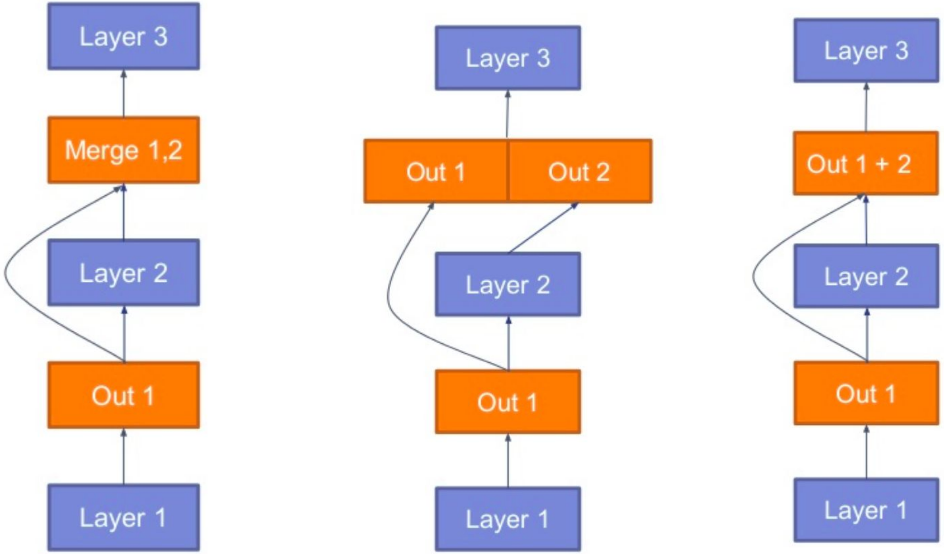
To introduce of recurrent connections with a time-delay of ***d*** to mitigate this problem. Gradients now diminish exponentially as a function of ***T/d*** rather than ***T***.

$$\underset{\text{Out 1 + 2}}{\mathbf{y}} = \overbrace{\underset{\text{Layer 2}}{\mathcal{F}}(\mathbf{x}, \{W_i\})}^{\text{“Residue”}} + \underset{\text{Out 1}}{\mathbf{x}}.$$

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.
[ ] Skip, residual and densely connected RNN Architectures, Frederic Godin, Ghent University.

# Leaky Units and Other Strategies for Multiple Time Scales

## Adding Skip Connections/Residual Connections through Time

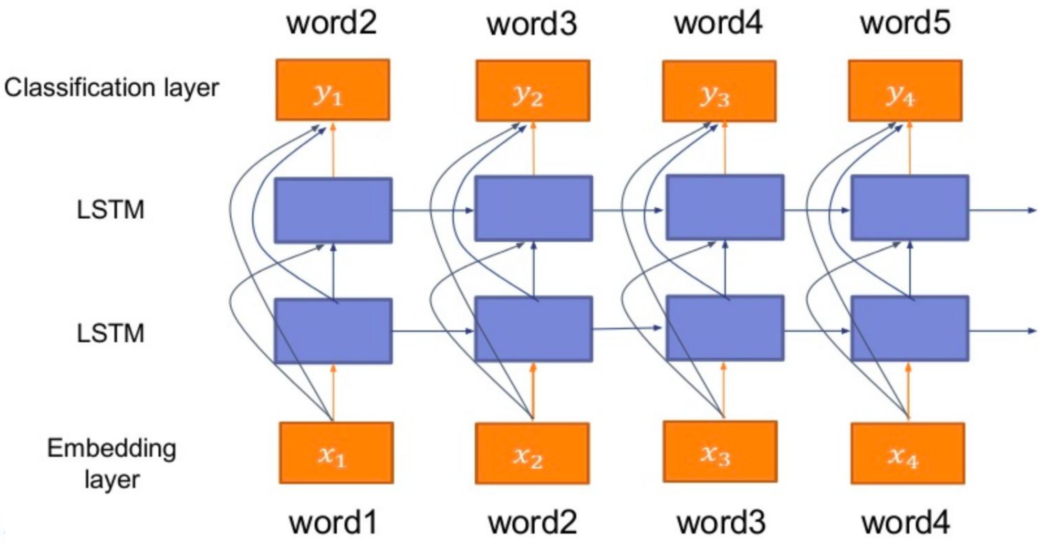


Different models for skip/residual connections.

[ ] Skip, residual and densely connected RNN Architectures, Frederic Godin, Ghent University.

# Leaky Units and Other Strategies for Multiple Time Scales

## Adding Skip Connections/Residual Connections through Time



Language modeling using skip/residual connections.

[ ] Skip, residual and densely connected RNN Architectures, Frederic Godin, Ghent University.

# Leaky Units and Other Strategies for Multiple Time Scales

## Leaky Units and a Spectrum of Different Time Scales

$$\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1-\alpha)v^{(t)}$$

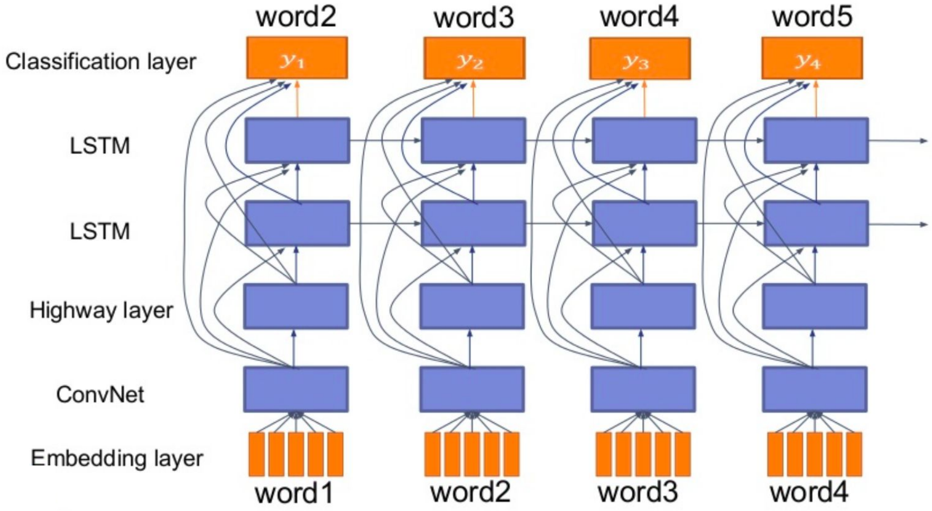
When we accumulate a running average **μ(t)** of some value **v(t)** by applying the update **μ(t)**← **αμ(t−1)+(1−α)v(t)**, the α parameter is an example of a linear self-connection from **μ(t−1)** to **μ(t)**. When **α** is near one, the running average remembers information about the past for a long time, and whenαis near zero, information about the past is rapidly discarded

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

# Leaky Units and Other Strategies for Multiple Time Scales

## Removing Connections

Organizing the state of the RNN at multiple time-scales with information flowing more easily through long distances at the slower time scales



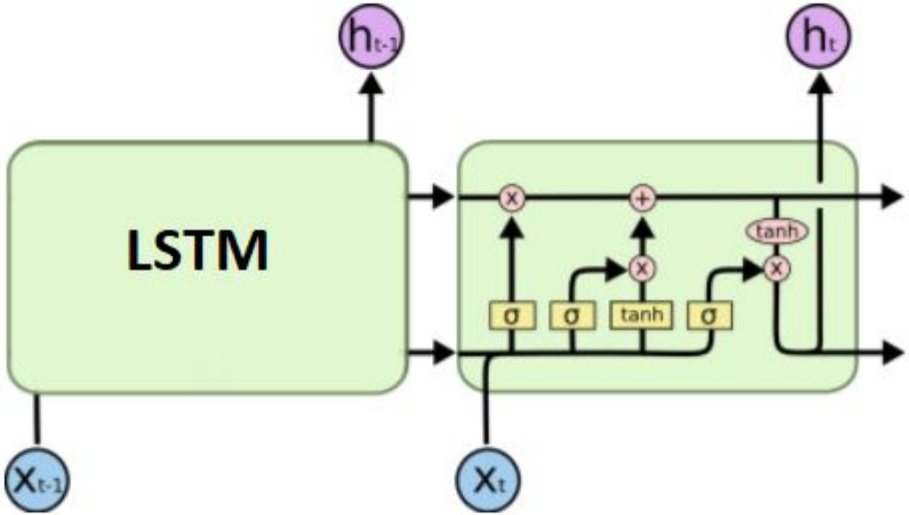
Example of long connections.

[ ] Skip, residual and densely connected RNN Architectures, Frederic Godin, Ghent University.
[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

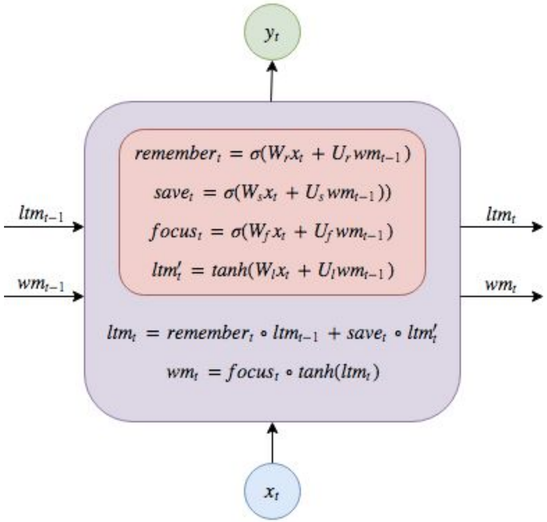
# Gated RNNs

## Long Short Term Memory

Solve partially the problem of Long Term Dependencies



To regulate the gates allows to add or remove information to the "cell state"



$$remember_t = \sigma(W_r x_t + U_r wm_{t-1})$$

$$save_t = \sigma(W_s x_t + U_s wm_{t-1}))$$

$$focus_t = \sigma(W_f x_t + U_f wm_{t-1})$$

$$ltm'_t = tanh(W_l x_t + U_l wm_{t-1})$$

$$ltm_t = remember_t \circ ltm_{t-1} + save_t \circ ltm'_t$$

$$wm_t = focus_t \circ tanh(ltm_t)$$

LSMT architecture.

[ ] Understanding LSTM Networks, Colab Blog, 2015.
[ ] Exploring LSTMs, Edwin Chen, 2016.

# Gated RNNs

## Long Short Term Memory (LSTM)

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i) \qquad (2.4a)$$

El valor de la **compuerta de entrada** $\boldsymbol{i}_t$ es representada por la ecuación (2.4a); la cual usa el estado oculto anterior, para determinar cuáles valores de la entrada actual merecen ser conservados y ser usados en la nueva celda de memoria $\tilde{\boldsymbol{c}}_t$.

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o) \qquad (2.4c)$$

La **compuerta de salida** es representada por la ecuación (2.4c). El resultado de esta compuerta es $\boldsymbol{o}_t$, que determina cuáles valores del nuevo estado oculto $\boldsymbol{h}_t$ (2.4f), merecen ser conservados o retirados.

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f) \qquad (2.4b)$$

La **compuerta de olvido** es representada por la ecuación (2.4b). Es similar a la compuerta de entrada, en vez decidir cuáles valores de entrada merecen ser conversados. El valor de la compuerta de salida ft determina aquellos valores de la anterior final celda de memoria $\boldsymbol{c}_{t-1}$ deben ser descartados

[ ] Long short-term memory, Hochreiter, S. and Schmidhuber, 1997. (seminal paper)

# Gated RNNs

## Long Short Term Memory (LSTM)

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_{\hat{h}}[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_{\hat{h}}) \qquad (2.4\text{d})$$

La **nueva celda de memoria** $\tilde{\boldsymbol{c}}_t$, es representada por la ecuación (2.4d). De acuerdo a la entrada actual $\boldsymbol{x}_t$ y el estado oculto $\boldsymbol{h}_{t-1}$ representa los posibles valores a ser usados en la memoria final.

$$\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t \qquad (2.4\text{e})$$

La ecuación (2.4e), representa la **memoria final** $\boldsymbol{c}_t$, que es la memoria a largo plazo. Usando la compuerta de entrada y la compuerta de olvido, determina qué valores de la nueva celda memoria deben ser usados, y que valores de la memoria final anterior deben ser olvidados.

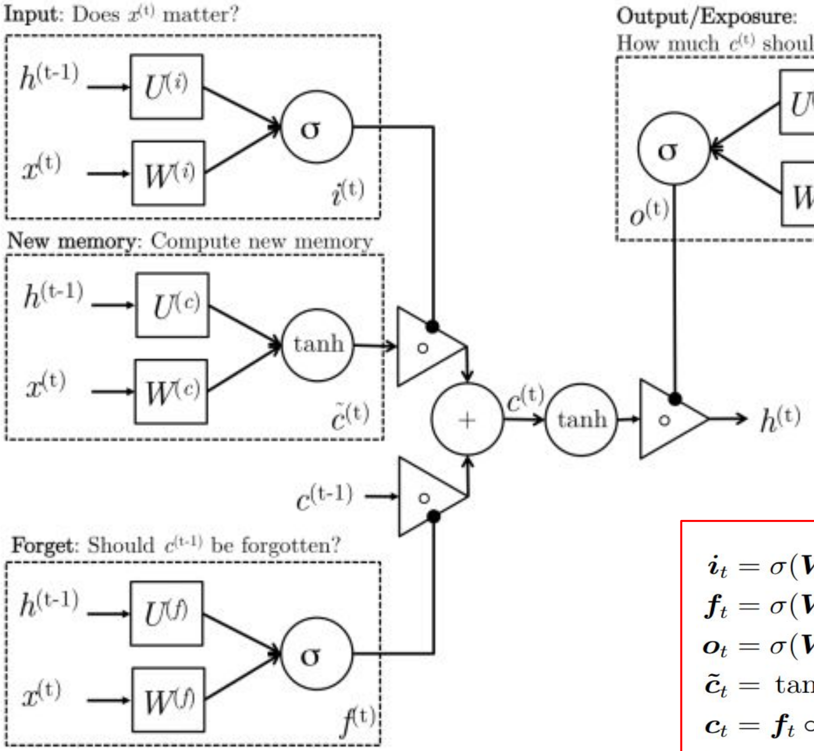
$$\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_{t-1}) \qquad (2.4\text{f})$$

El **nuevo estado oculto** $\boldsymbol{h}_t$ , es representado por la ecuación (2.4f)

[ ] Long short-term memory, Hochreiter, S. and Schmidhuber, 1997. (seminal paper)

# Gated RNNs

## Long Short Term Memory (LSTM)



LSMT architecture.

| | |
|---|---|
| $\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i)$ | Input Gate |
| $\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f)$ | Forget Gate |
| $\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o)$ | Output Gate |
| $\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_{\hat{h}}[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}_{\hat{h}})$ | New Cell Memory |
| $\boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \circ \tilde{\boldsymbol{c}}_t$ | Final Memory |
| $\boldsymbol{h}_t = \boldsymbol{o}_t \circ \tanh(\boldsymbol{c}_{t-1})$ | Hidden State |

[ ] Lecture note 04 in Deep Learning for Natural Language Processing, Mohammadi, 2015.

# Gated RNNs

## Gated Recurrent Unit (GRU)

$$\boldsymbol{u}_t = \sigma(\boldsymbol{W}_u[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_u) \qquad (2.5a)$$

El valor de la **compuerta de reinicio** $r_t$ es representada por la ecuación (2.5b), determina qué tan importante es el estado oculto anterior $\boldsymbol{h}_{t-1}$ para la nueva memoria $\boldsymbol{h}_t$.

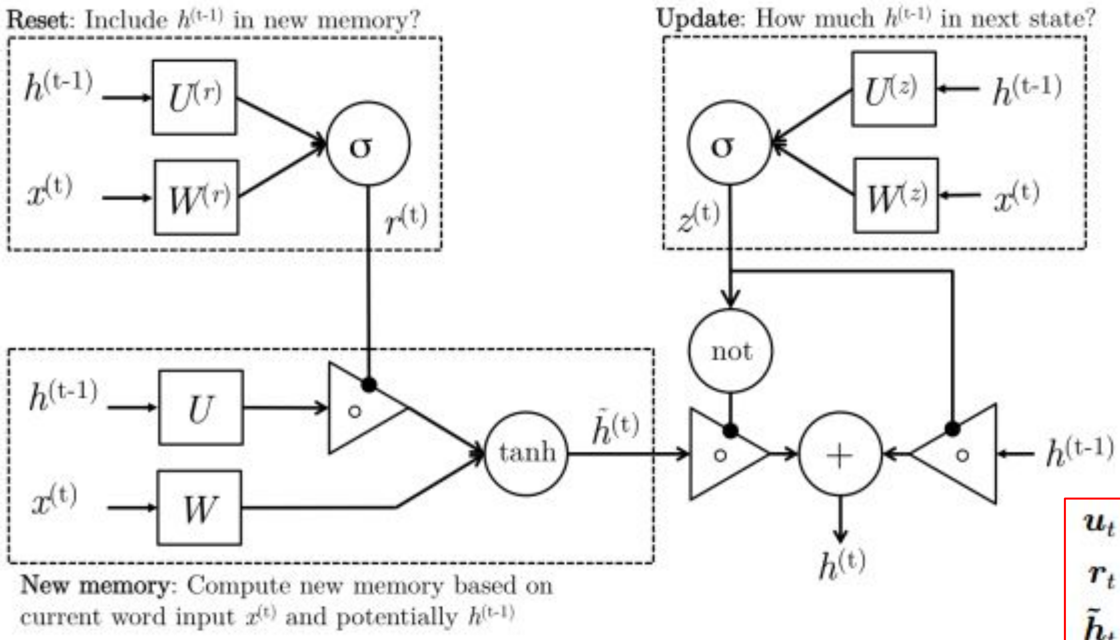
$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) \qquad (2.5b)$$

La **compuerta de actualización** es representada por la ecuación (2.5a), el resultado de esta compuerta $u_t$ determina que tanto se debe conservar del estado oculto anterior $\boldsymbol{h}_{t-1}$ para ser usado en el nuevo estado oculto $\boldsymbol{h}_t$ . Si $\boldsymbol{u}_t \approx 1$, casi complementamente se copia $\boldsymbol{h}_{t-1}$ .De otro lado, si $\boldsymbol{u}_t \approx 0$, la nueva memoria , $\tilde{\boldsymbol{h}}_t$ es usada para el siguiente estado oculto $\boldsymbol{h}_t$ .

[ ] Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho, 2014. (seminal paper)

# Gated RNNs

## Gated Recurrent Unit (GRU)



Reset: Include $h^{(t-1)}$ in new memory?
$h^{(t-1)}$
$x^{(t)}$
$U^{(r)}$
$W^{(r)}$
σ
$r^{(t)}$
Update: How much $h^{(t-1)}$ in next state?
$U^{(z)}$
$W^{(z)}$
$h^{(t-1)}$
$x^{(t)}$
σ
$z^{(t)}$
not
$h^{(t-1)}$
$x^{(t)}$
$U$
$W$
∘
tanh
$\tilde{h}^{(t)}$
∘
+
∘
$h^{(t-1)}$
$h^{(t)}$
New memory: Compute new memory based on current word input $x^{(t)}$ and potentially $h^{(t-1)}$

GRU architecture.

$$\begin{array}{ll}
\boldsymbol{u}_t = \sigma(\boldsymbol{W}_u[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_u) & \text{Update Gate} \\
\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) & \text{Restart Gate} \\
\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h[\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h) & \text{New Memory} \\
\boldsymbol{h}_t = \boldsymbol{u}_t \circ \tilde{\boldsymbol{h}}_t + (\vec{1} - \boldsymbol{u}_t) \circ \boldsymbol{h}_{t-1} & \text{Hidden State}
\end{array}$$

[ ] Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho, 2014.

# Gated RNNs

## Gated Recurrent Units (GRU)

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h[\boldsymbol{r}_t \circ \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] \qquad (2.5\mathrm{c})$$

La ecuación (2.5c), representa la **nueva celda de memoria** $\tilde{\boldsymbol{h}}_t$ De acuerdo a la entrada actual $\boldsymbol{x}_t$ y el estado oculto $\boldsymbol{h}_{t-1}$ representa posibles valores a ser usados.

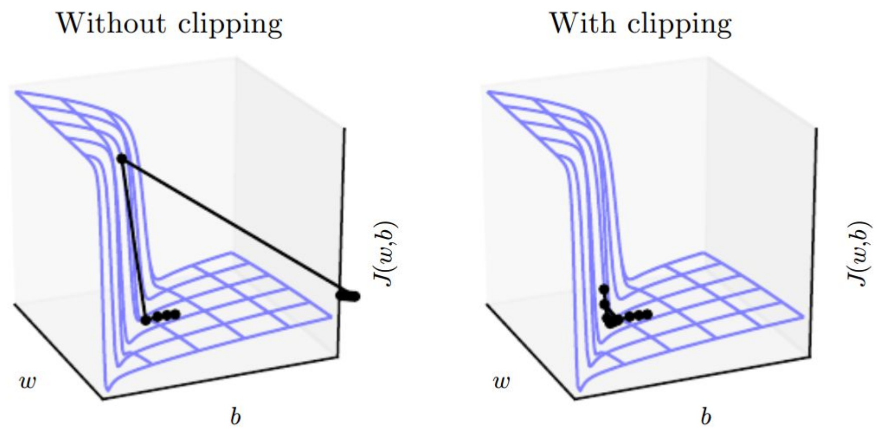
$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) \qquad (2.5\mathrm{b})$$

**Estado oculto** $\boldsymbol{h}_t$ , ver ecuación (2.5d) es generado usando el anterior estado oculto, y aplicando el valor de la compuerta de actualización sobre la nueva memoria.

[ ] Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho, 2014. (seminal paper)

# Optimization for Long-Term Dependencies

## Gradient Clipping



Example of the effect of gradient clipping in a recurrent network with two parameters w and b.

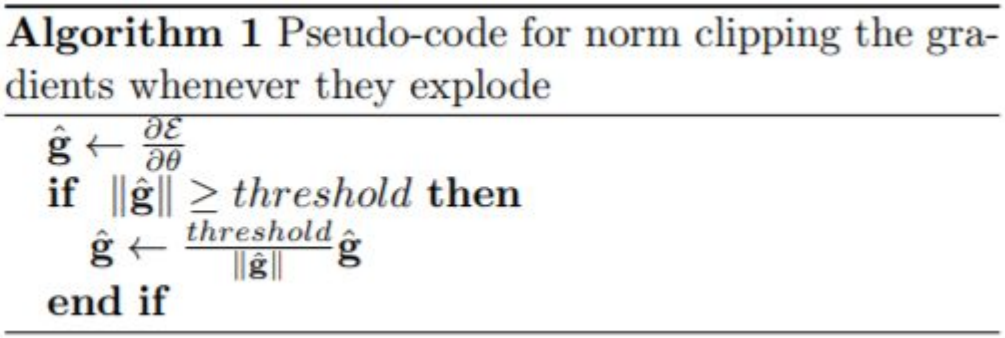
* **(Left)**Gradient descent without gradient clipping overshoots the bottom of this small ravine, then receives a very large gradient from the cliff face. The large gradient catastrophically propels the parameters outside the axes of the plot.

* **(Right)**Gradient descent with gradient clipping has a more moderate reaction to the cliff.

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

# Optimization for Long-Term Dependencies

**Gradient Clipping**



**Algorithm 1** Pseudo-code for norm clipping the gradients whenever they explode

```
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ then
    $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if
```

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.
[ ] Advanced in Optimizing Recurrent Networks, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

# Optimization for Long-Term Dependencies

## Regularizing to Encourage Information Flow

Gradient clipping helps to deal with exploding gradients, but it does not help with vanishing gradients. Another idea is to regularize or constrain the parameters so as to encourage "information flow." In particular, we would like the gradient vector $\nabla_{\boldsymbol{h}^{(t)}} L$ being back-propagated to maintain its magnitude.

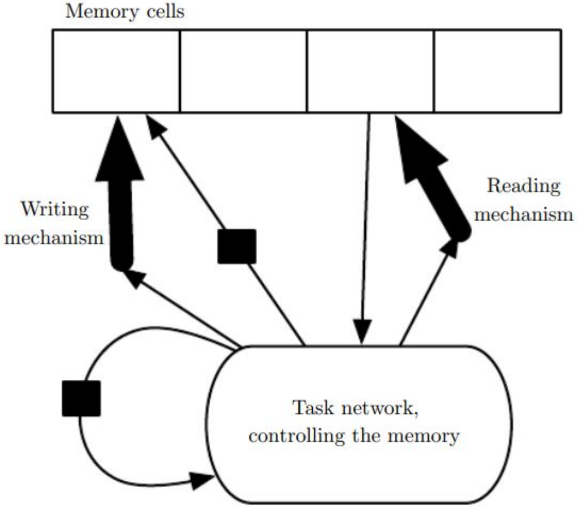
Formally, we want $(\nabla_{\boldsymbol{h}^{(t)}} L) \dfrac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$ to be large as $\nabla_{\boldsymbol{h}^{(t)}} L.$

One solution

$$\Omega = \sum_t \left( \frac{\left|\left| (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} \right|\right|}{||\nabla_{\boldsymbol{h}^{(t)}} L||} - 1 \right)^2$$

[ ] On the difficulty of training recurrent neural networks, Pascanau, 2013.

# Explicit Memory

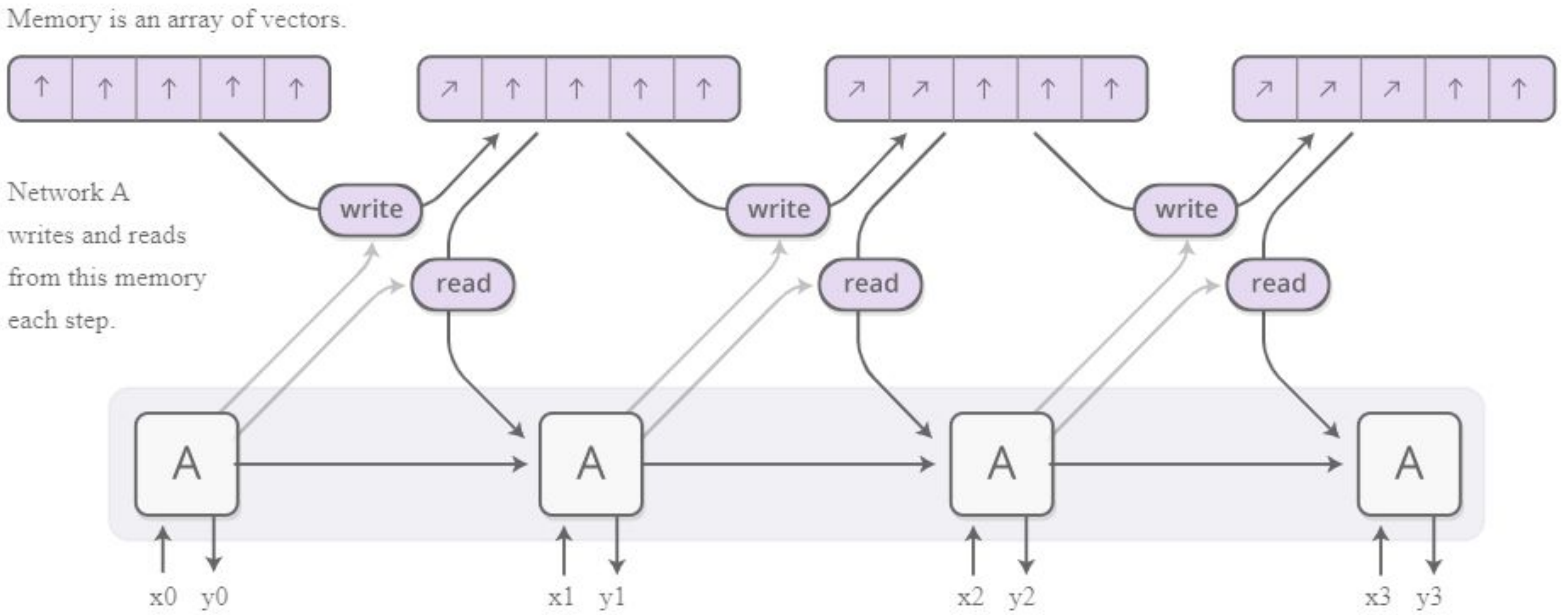


A schematic of an example of a network with an explicit memory, capturing some of the key design elements of the neural Turing machine.

[ ] Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016.

# Explicit Memory

## Neural Turing Machine



Model of Neural Turing Machine.

[ ] Attention and Augmented Recurrent Neural Networks, Chris Olah, 2016.
[ ] Neural Turing Machines, Graves, 2014 (seminal paper)

# Code

- https://github.com/marbramen/100DaysOfMLCode/blob/master/Day%2037%20-%20Simple%20recurrent%20neural%20networks.ipynb
- https://github.com/marbramen/100DaysOfMLCode/blob/master/Day%2038%20-%20Comparasion%20between%20LSTM%2C%20GRU%2C%20BiLSTM%20in%20Keras.ipynb
- https://github.com/marbramen/100DaysOfMLCode/blob/master/Day%2076%20-%20Stack%20LSTM%20-%20%20Sentiment%20Analysis%20in%20Tensorflow.ipynb
- https://github.com/marbramen/100DaysOfMLCode/blob/master/Day%2077%2C78%2C79%20-%20Word%20Level%20Language%20Modeling%20in%20TensorFlow.ipynb

# References

- Deep Learning, Ian Goodfellow, Yoshua Bengio and Aaron Courville, 2016.
- Skip, residual and densely connected RNN Architectures, Frederic Godin, Ghent University.
- Long short-term memory, Hochreiter, S. and Schmidhuber, 1997.
- Lecture note 04 in Deep Learning for Natural Language Processing, Mohammadi, 2015.
- Understanding LSTM Networks, Colab Blog, 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Exploring LSTMs, Edwin Chen, 2016, http://blog.echen.me/2017/05/30/exploring-lstms/
- Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho, 2014.
- Attention and Augmented Recurrent Neural Networks, Chris Olah, 2016, https://distill.pub/2016/augmented-rnns/
- Liquid State Machine (LSM), https://www.techopedia.com/definition/33292/liquid-state-machine-lsm
- The Liquid State Machine (LSM), Hananel Blog.
- Real-time computing without stable states: a new framework for neural computation based on perturbations, Maass, 2002.
- Liquid State Machines: Motivation, Theory, and Applications, Maass, 2010.
- Deep Echo State Network (DeepESN): A Brief Survey, Gallicchio, 2018.
- An Introduction to the Echo State Network and its Applications in Power System, Dai1, 2009.

# Chapter 10 - Part II

## Sequence Modeling: Recurrent and Recursive Nets

March
Brag

cesarbrma91@gmail.com
@MarchBragagnini

Universidad Católica
San Pablo